Effective Exploiting of Statistical Relational Learning for Data Cleaning

Larysa Visengeriyeva
TU Berlin
Germany
fn.ln@tu-berlin.de

Alan Akbik
TU Berlin
Germany
fn.ln@tu-berlin.de

Sebastian Schelter
TU Berlin
Germany
fn.ln@tu-berlin.de

Manohar Kaul
TU Berlin
Germany
fn.ln@tu-berlin.de

Volker Markl
TU Berlin
Germany
fn.ln@tu-berlin.de

ABSTRACT

High quality data is important for data science methods. Unfortunately, digitally collected data often suffers from many data quality issues, such as duplicate, incorrect or incomplete data. A common approach for counteracting such issues is to formulate a set of data agnostic cleaning rules intended to identify and repair incorrect, duplicate and missing data. There is also a need for new approaches to overcome the limits of existing heuristic data cleaning solution. In this paper, we address this issue by proposing an approach to data cleaning based on Statistical Relational Learning (SRL) and probabilistic inference. We argue that a well-known formalism - Markov logic - is a natural fit for modeling interacting data quality rules in a flexible and extensible way. We show how data quality rules expressed as first-order formulas can be directly translated into a predictive model in an SRL framework. This approach allows the usage of probabilistic joint inference over interleaved data cleaning rules to improve data quality. In an experimental study we demonstrate the viability of our proposed approach and show that our data cleaning results outperform state-of-the-art data cleaning systems.

1. INTRODUCTION

Having access to high quality data is of great importance in data analysis. However, in the real world, data is often considered *dirty*, meaning that it contains inaccurate, incomplete, inconsistent, duplicated, or stale values [8]. 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* [15]. As previous work has observed, such data quality issues may be detrimental to data analysis [10, 16] and cause huge costs to businesses [13]. Therefore, improving data quality with respect to business and integrity constraints is a crucial component of data management. A common approach to increasing data quality is to formulate a set of *data cleaning rules* that detect semantic errors by utilizing data dependencies [15, 2, 11, 21]. However, previous research identified a number of challenges associated with creating such rule sets:

- Firstly, while each such rule may address one data quality issue individually, previous work in [15] and [19] has observed that such rules may *interact*. For instance, a rule that deletes duplicates might perform better if 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. This means that data quality rules such as deduplication and missing value imputation should be modeled jointly, rather than as separate processes.
- Secondly, rules in such a rule-set may need to be *"soft"* and *"hard"* in order to balance constraints of different importance, especially within a set of interacting rules. This makes the creation of such rule sets challenging from a modeling perspective. Defining probabilistic data quality rules enables us to use statistical inference in order to predict errors.

Our hypothesis: While authomatic data cleaning, it is impossible to specify the optimal order of rules execution [11], therefore we use joint inference for the simultaneous rules execution.

In this paper, we present an approach to data cleaning based on Statistical Relational Learning (SRL) [23] and probabilistic inference. 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 [1] on top of a database schema. We show how such functional dependencies, expressed as first-order logic formulas, can be translated into probabilistic-logical languages, allowing us to reason over inconsistencies, duplicates or missing values in a probabilistic way.

We propose to use *Markov logic* [12] as a framework for improving data quality, for the following reasons:

1. In order to infer the types of errors and their sources, we consider integrity constraints to be probabilistic. We rely on the first approach to specify *"soft"* functional dependencies between columns that was suggested in the CORDS system in

Customer *(CUST)*

C_ID	First Name (FN)	Last Name (LN)	Street	City	Zip Code	Tel
1	Ron	Howard	1 Sun Dr.	Los Angeles	90001	12345
2	Max	Miller	12 Hay St.	Napa	94558	11234

Transactions *(TRANS)*

T_ID	Item	First Name	Last Name	Street	City	Zip Code	Phone
1	iPhone6	R.	Howard	1 Sun Dr.	L.A.	**null**	**null**
2	Galaxy 5	**null**	Miller	12 Hay St.	**null**	94558	11234
3	Nexus 5	Howard	Ron	**null**	**null**	90001	12345

Figure 1: Example database with two tables: A CUSTOMER table that records information on persons and a TRANSACTIONS table that records commercial acquisitions of the customers. The latter is heavily corrupted, containing missing (NULL) and false values. For instance, the customer "Ron Howard" (*C_ID1*) is involved in two transactions, namely *T_ID1* and *T_ID3*. But *T_ID1* contains NULL values and in *T_ID3*, first and last names are reversed ("Howard Ron"). By linking entries in the TRANSACTIONS table to the CUSTOMER table, this could be addressed, but this is not trivial as many fields are either corrupted or use different values ("L.A." vs. "Los Angeles" and "R." vs "Ron").

[25]. We use the notion of *"soft"* FDs, where one attribute determines the value of the another attribute in a probabilistic way, as a basis for modeling Markov Logic Networks. Markov logic supports *soft* and *hard* first-order logic formulas interchangeably. First-order logic formulas as typically used in data quality rules are *hard*. However, within complex rule-sets we may want to assign different importance to different rules. Furthermore, if no perfect resolution of a rule-set is possible, we may be interested in finding data cleaning steps that fulfill as many rules as possible while violating only a few [22]. This means that we require the means to add weights to rules, i.e. define *soft* rules, which the Markov logic framework allows us to do. For example, in our work we denote all key constrains based FDs as *hard* rules [3].

2. One of the challenges while applying machine learning for repairing errorneous data over-fitting. This problem becomes apparent when modeling data quality rules that hold only for a fraction of data, but do not hold globally. Markov logic programs with *"soft"* rules prohibit over-fitting by adding weights to the constaint-based rules.

3. Markov logic facilitates the incorporation of rich domain knowledge for data with relational structure or dependencies between attibutes (also called non-i.i.d. models [38]) and allows us to perform joint inference over several interleaved hidden predicates. By making use of this joint inference and soft rules we show that we can improve data cleaning performance over other data cleaning systems. Furthermore, we show that we are not limited to functional dependencies as data quality rules, but rather are able to define arbitrary rules as long as they are expressible as weighted first-order logic. For instance, within the Markov logic program one can define both positive and negative data cleaning rules;

4. Markov logic as a formalism is independent of the inference engine. This means that we can exchange the method of joint inference used for data repair prediction as suits our needs. For instance, in this work we use an Integer Linear Programming optimization for MAP (maximum a-posteriori) inference [32];

This paper is outlined as follows: First we provide a motivating example in Section 2 in order to demonstrate the definition of data cleaning rules and illustrate the limitations of hard rules for creating rule sets. In Section 3, we discuss our proposed approach; we show how integrity constraints can be translated into an MLN and how probabilistic inference can be used to determine data repair operations. In Section 4, we validate our approach in experiments on a reference data set and show that our proposed approach outperforms previously reported results for data cleaning on this data. We conclude that our proposed approach allows us to elegantly express data cleaning rules within the Markov Logic framework and that it allows to take a holistic view on data cleaning, leading to simpler modeling and effective inference of data repair operations.

2. DATA CLEANING RULES

To motivate the problem, we begin with an example of an incomplete database table that is affected by several data quality issues and formulate data cleaning rules which we formalize as data dependencies and integrity constraints.

2.1 Motivating Example

As an example, consider the tables illustrated in Figure 1, from a company that sells mobile phones to customers and records information in two tables: The first table (CUSTOMER) records address and contact data of each customer, including their first and last names, their street, city and zip codes, as well as their phone number. The second table (TRANSACTIONS) records each item bought and personal data as entered by the customer when buying the data.

Apparently from the example tables, they are affected by numerous data quality issues. The first are missing values in the TRANSACTIONS table, indicated by NULL values. The second are false values: For instance, the customer "Ron Howard" (customer ID 1 in the example table) is involved in transaction 3, but here his name is falsely recorded as "Howard Ron". The third are duplication issues, the city of "Los Angeles" is sometimes entered into the table as "L.A." while the first name "Ron" is once abbreviated as "R.".

These issues can be addressed for each transaction by identifying the corresponding entries in the CUSTOMER table and using this to automatically clean the transaction data. However, as the example shows, enough information remains to make such links possible through manual linking, defining automatic cleaning rules to accomplish this is not trivial.

For instance, while we can define a hard rule that states that CITY field values "L.A." and "Los Angeles" are always the same, we cannot do the same for the FIRSTNAME values "R." and "Ron". Rather, we would need a *soft rule* that indicates that both string *could* refer to the same entity, but that we may require additional evidence.

Consider also the NULL value in the ZIP field of transaction 1. We may want to fill this missing value by linking the transaction to customer 1, but only the fields LASTNAME and STREET fully match, while CITY (and possibly FIRSTNAME) may match using the dedu-

plication rules mentioned above. Even though this provides significant evidence that the transaction should be linked to customer 1, there is still a chance that two different people share the same last name and street address in a large city. Only if we include the PHONE field into the rule can we be reasonably certain that they should be linked, but as the example shows, the values for this field is missing.

This example illustrates the limitations of data cleaning via hard rules: Because the example table is heavily corrupted, nearly no hard data cleaning rules can be defined that may not also create errors. Instead, we could define a number of soft rules that aggregate evidence, such as a first name abbreviation rule ("R." vs. "Ron"), a first and last name switch rule (indicating that there is a small chance that "Ron Howard" and "Howard Ron" are the same person) and rules that indicate that the more fields two tuples in the CUSTOMER and TRANSACTIONS tables share, the more likely it is that they refer to the same person. But, as this example illustrates, defining such rule sets in such a way that inference can be executed over the entire rule set is challenging from a modeling perspective.

2.2 Problem Definition

We consider a database instance D with a relational schema S. Furthermore, we consider relation $R \in S$ that is defined for the set of attributes $\alpha(R)$. Moreover, $\delta(U_i)$ denotes the domain of an i-th attribute $U_i \in \alpha(R)$. For the data cleaning, we consider a set of data quality rules, which are data-agnostic and rely on data integrity constraints [1]. In the following we study data cleaning rules based on (Conditional) Functional Dependencies, Matching Dependencies and Inclusion Dependencies.

Definition 1. A **functional dependency** (FD) on a relational schema S is an expression of the form $\phi : X \rightarrow Y$ where $X \subseteq \alpha(R)$ and $Y \subseteq \alpha(R)$ are subsets of $R's$ attributes $\alpha(R)$. This FD holds if every pair of tuples of R that agree in each of attributes X, also agree in the Y attributes. □

For example, given the database schema explained in Section 2, consider the functional dependency:

$$\mathsf{fd} : \text{TRANS}([\text{CITY}, \text{PHONE}] \rightarrow [\text{STREET}, \text{ZIP}])$$

This rule declares that the two fields CITY, PHONE together uniquely determine the two fields STREET and ZIP. This means that if two entries share the same city and phone number, they must also share the same address. In our example in Figure 1, this rule cannot be applied: Although two instances of the customer "Ron Howard" are recorded, one is missing the PHONE value. Thus, in the example, this rule can only be applied together with additional data cleaning rules.

Such functional dependencies are used by data quality management systems to find and correct dirty data, because these dependencies specify the semantics of the data in a declarative way. The main disadvantage of functional dependencies is that they operate solely on the schema level and are often not able to detect conflicts in real-life data. For this reason we consider *conditional functional dependencies*, which are extensions of the traditional functional dependencies.

Definition 2. A **conditional functional dependency** (CFD) defined on the relational schema S is a pair $\psi(X \rightarrow Y, T_p)$, where

1. $X \rightarrow Y$ is a standard FD and $X \cup Y \in \alpha(R)$;
2. T_p is a tuple pattern, which is a set of constraints holding on the particular subset $X \cup Y$ of tuples. For each $U \in X \cup Y$, the value of the attribute U for T_p, $T_p[U]$ is either a variable value '_', or a constant $u \in \delta(U)$

We assume that CFDs are provided in the 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: $\psi_1(X \rightarrow Y_1, T_p)$, $\psi_2(X \rightarrow Y_2, T_p) \ldots$. Furthermore, we distinguish between *constant* and *variable* CFDs. The former contains tuple pattern T_p where $T_p[Y] \neq$ '_'. Otherwise, ψ is called *variable*. □

As an example, consider this CFD for the tables in Figure 1:

$$\mathsf{cfd}_1 : \text{TRANS}([\text{ZIP}] \rightarrow [\text{CITY}], \mathsf{t}_1 = (90001 \parallel \text{Los Angeles}))$$

This rule states that every tuple in which the value for ZIP equals "90001" should have its CITY attribute set to "LOS ANGELES". In our example in Figure 1, this rule can be used to correct the NULL value in transaction 3.

Data deduplication techniques use FDs to define the matching dependencies. The difference here is that matching dependencies use the notion of the similarity predicate instead of equality predicate.

Definition 3. A **Matching Dependency** (MD) for schemas S_1 and S_2 is syntacticly defined as:

$$\mu : S_1[X_1] \approx S_2[X_2] \rightarrow S_1[Y_1] \rightleftharpoons S_2[Y_2]$$

where $X_1 \cup Y_1$ and $X_2 \cup Y_2$ are pairwise compatible sets of attributes in $\alpha(R_1), R_1 \in S_1$ and $\alpha(R_2), R_2 \in S_2$, respectively; $\approx$ indicates similar attributes and $\rightleftharpoons$ is called the *matching operator*. □

In other words the matching operator $\rightleftharpoons$ means that for each S_1 tuple t_1 and each S_2 tuple t_2: $t_1[Y_1]$ and $t_2[Y_2]$ refers to the same real-world entity. Having dynamic semantics, that is pointing *how* to repair data, MDs forces to update $t_1[Y_1]$ and $t_2[Y_2]$ such that they have the same values. The operator $\rightleftharpoons$ only requires that the $t_1[Y_1]$ and $t_2[Y_2]$ are identified.

Consider the FIRSTNAME field in transactions 1 in Figure 1. As discussed in the previous section, we may define a rule that indicates that the values "R." and "Ron" could refer to the same name, but cannot always be certain whether this is the case. To include additional evidence, we create the following MD to link transaction 1 to customer 1:

$$\begin{aligned}\mathsf{md}_1 : &\ \text{TRANS}[\text{LASTNAME}, \text{CITY}, \text{STREET}] \\ &= \text{CUST}[\text{LASTNAME}, \text{CITY}, \text{STREET}] \\ &\wedge \text{TRANS}[\text{FIRSTNAME}] \approx \text{CUST}[\text{FIRSTNAME}] \\ &\rightarrow \text{TRANS}[\text{FIRSTNAME}] \rightleftharpoons \text{CUST}[\text{FIRSTNAME}]\end{aligned}$$

This MD states that if any two tuples from TRANS and CARD agree on LASTNAME, CITY, STREET, and their FIRSTNAME values are similar, then TRANS[FIRSTNAME] and CARD[FIRSTNAME] should be identified and have the same values.

This is an example of 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 though, we may be able to determine more graded similarity, i.e. some types of similarity that we deem to be more probable ("L.A." and "Los Angeles"), and others that are more uncertain ("R." and "Ron"). Moreover, as we discussed in Section 2.1, we may have different levels of confidence regarding functional or matching dependencies that we define. The above rule for example may be deemed as likely, but not necessarily always true. However, hard first-order logic does not permit us to formalize this intuition.

Detecting inconsistencies between tuples across different relations requires definition of the interrelation dependencies such as (Conditional) Inclusion Dependencies:

Definition 4. A **Conditional Inclusion Dependency** (CIND) for schemas S_1 and S_2 is syntactically defined as a pair $\xi(S_1[X; X_p] \subseteq S_2[Y; Y_p], T_p), where$:

1. disjoint lists of attributes $(X; X_p) \in \alpha(S_1)$ respectively $(Y; Y_p) \in \alpha(S_2)$;
2. $S_1[X] \subseteq S_2[Y]$ is the embedded in ξ IND (inclusion dependency) with a meaning that for each tuple $t \in S_1[X]$ there should be corresponding $t' \in S_2[Y]$ such that t and t' agree on their corresponding attributes.
3. T_p is a tuple pattern with attributes in X_p and Y_p, such that for each pattern tuple $t_p \in T_p$ and each attribute U in X_p (respectively in Y_p), $t_p[U] \in \delta(U)$. □

The examples presented in this section illustrate the limitations of modeling data cleaning rules purely using hard rules. To overcome these limitations, we look beyond first-order logic.

3. PROPOSED APPROACH

Our proposed approach is to model data cleaning rules in a Markov Logic formalism and use probabilistic inference for data cleaning. Figure 2 shows a high-level overview of our system that consists of three main components: *I*) a Rule-based Data Quality Management component; *II*) a Statistical Inference with Markov Logic component; and *III*) the Data Layer. As our system is based on Markov Logic we first give a brief introduction into the main concepts of this formalism and then proceed to give details on the components of our system. We illustrate how we transfer data cleaning rules into Markov Logic and how we used probabilistic inference to determine data repair operations for a given data set.

3.1 Markov Logic

Markov logic [12] is a knowledge representation system that combines first-order logic as a declarative language and probabilistic graphical models (undirected Markov Networks) on top of which probabilistic inference is performed. Markov Networks can be viewed in terms of *cliques* which are subgraphs where all pairs of nodes are connected. Edges between nodes can be parametrized by values or factors, also known as potentials, to encode the strength of a conditional dependency. The joint probability of a Markov Network can then be attained by computing the normalized product over clique potentials. Semantically, a Markov Logic Network (MLN) is a log-linear model (also known as exponential model), which defines the probability distribution over possible worlds, in our case all possible states in the database.We consider a set of formulae $\bar{F} = F_1, F_2 \dots F_N$ with corresponding weights $w_1 \dots w_N$ as an input. These weighted first-order logic formulae define a probability distribution as follows:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{F_i} w_i n_i(x)\right)$$

Where $n_i(x)$ is the number of true groundings of formula F_i in x and w_i is the corresponding formula weight and Z is a normalization constant. Given a formula F_i with variables $(x_1 \dots x_n)$, we create new formula g called *ground formula* as follows: for each x_k and its domain $\delta(U_k)$, each variable x_k is being substituted with a value from its domain $\delta(U_k)$. Let define w to be a function that maps each ground formula g to its assigned weight. If g is false (true) and $w(g) > 0$ (respectively $w(g) < 0$), then the ground formula is violated. Hence, the probability of the particular word is higher the less ground formulae is violated and the lower is the cost of the world W:

$$cost_W = \left(\sum_{g \in V(W)} w(g)\right)$$

Here $V(W)$ is the set of violated ground formulae.

Please note that formula weights are not probabilities itself but the certainty of this formula. This also means that weights can be considered as empirical frequencies where each formula should hold. Weights can be specified in two ways: first, heuristically (e.g using the domain expertise); second, by performing weight learning on data, e.g. by running weight learning algorithms on MLN [27]. In this paper we focus on an inference task as a prediction for the data cleaning operations.

MLNs are typically created by writing a set of first-order logic rules with weights, constructed using variables that range of objects in the domain of interest and predicates that represent relations between these objects. Predicates can be observed or hidden. *Observed predicates* are relations between objects that can be seen in a given data set. For instance, the data set used in our running example may have observed predicates that indicate which customer bought which item. These observations "ground" the MLN, meaning that they provide evidence to the first-order rules that we define. In addition, an MLN may have a number of *hidden predicates*, meaning that they are not observed in the input data, but can be inferred through rules. The goal of probabilistic inference is to infer the likelihood for observed and hidden predicates given a rule set and evidence. As we show in Section 3.2, 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.

3.2 Data Cleaning Rules as Markov Logic

In the Rule-based Data Quality Management component of our approach (see Figure 3 for an overview) we define data cleaning rules in the form of CFDs and MDs as introduced in Section 2. For example, given the following functional dependency $\phi : X \rightarrow Y$ where $X \subseteq \alpha(R)$ and $Y \subseteq \alpha(R)$ are subsets of $R's$ attributes $\alpha(R)$. According to the [14], ϕ can be expressed as *first-order logic* sentence:

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

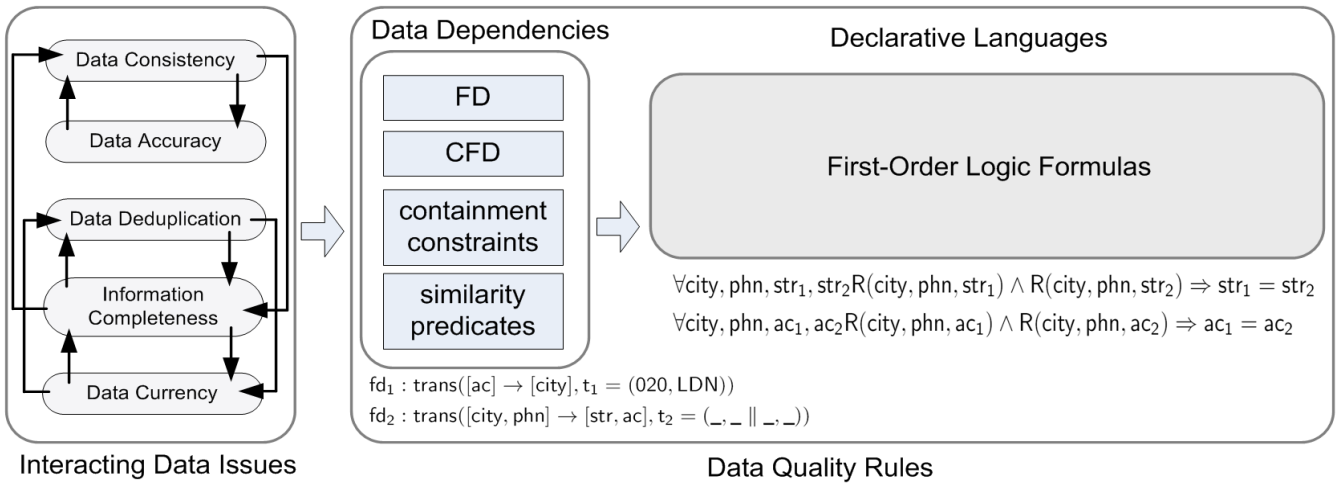

Figure 3: System Overview: Component (I), the Rule-based Data Quality Management component, is based on the data quality rules that address different data quality issues.

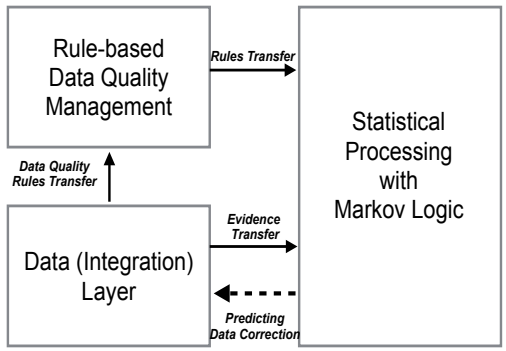

Figure 2: Our proposed method for data cleaning based on Markov logic consists of three main components: (I) A Rule-based Data Quality Management component that is based on the data quality rules to address different data quality issues; (II) A relation prediction component based on probabilistic inference performed by the Markov logic framework and (III) a Data Layer that includes a number of data sources including relational and semi-structured data.

This first-order logic sentence is crucial for compilation of data cleaning rules into predictive models. In order to specify an MLN to solve data cleaning problem we first specify data quality rules and then these rules should be translated into Markov logic. As described previously, the base of the Markov logic is predicate calculus. Therefore, one important step is the generalization of the compilation from the formal constraint based data cleaning rules into the predicate calculus.

We can think about a data quality rule ϕ as a composite component, consisting of subcomponents such as atomic sentences (attribute), logical and quantified sentences (RHS and LHS of the rule). To describe the structure of ϕ in a predicate calculus, we need to choose symbols that designate the elements of our conceptualization. In the following we define the *vocabulary* we use for data quality rules:

Phase	Example
1) Schema definition	t_2(Galaxy 5, NULL, Miller, 12 Hay St., NULL, 818, 11234)
2) Observed predicates MLN declaration	FIRSTNAME(id, value) 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)

Table 1: MLN declaration process and creation of grounded atoms for tuple 2 in the TRANSACTIONS example table.

Universe of Disclosure is specified by the set of all objects from domain $\delta(U)$ that is fixed for the set of attributes $\alpha(R)$.

Terms A *term* is used as a name for an object in the universe of discourse. We define *variables* to denote arguments in atoms and *constants* to denote data constants of particular domain $\delta(U_i)$ of an i-th attribute $U_i \in \alpha(R)$.

Atoms To designate the tuple in relation R: $R(x_1, x_2, \ldots, x_n)$, we use atomic sentences $attrX_1(id, v_1)$, $attrX_2(id, v_2)$ ... $attrX_n(id, v_n)$ where $attrX_i(id, v_i)$ means that v_i is a attribute value of the i-th attribute in $R(x_1, x_2, \ldots, x_n)$ of the id-th tuple in relation R.

Relation Constants which denote relations between several objects:

- Similarity: $similar(x_1, x_2)$ means that x_1 similar to x_2 (e.g by using different similarity measures like cosine or jaccard similarities).

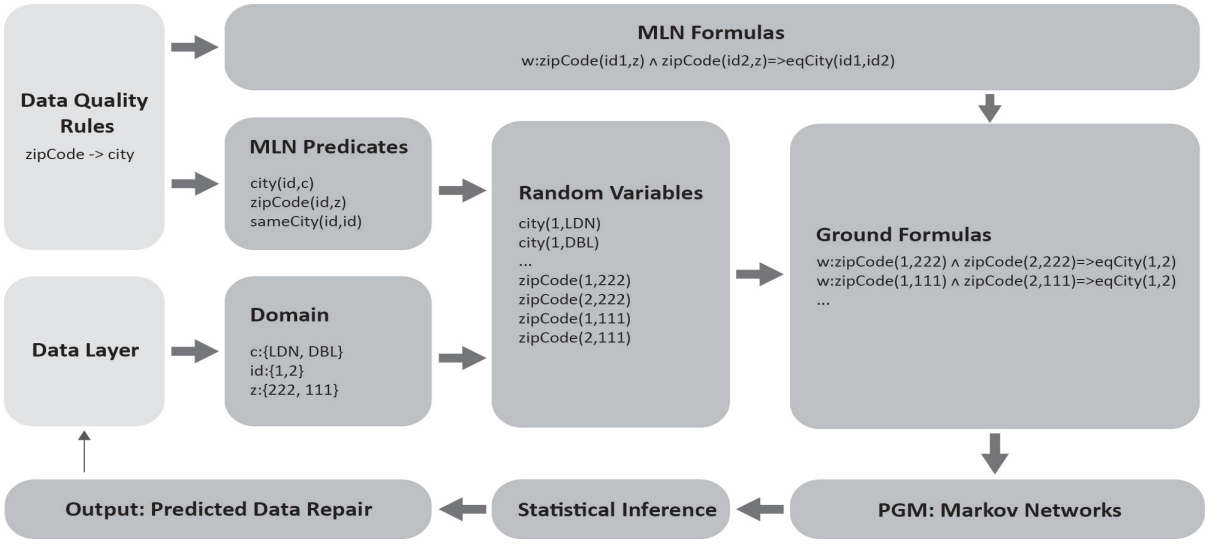

Figure 4: Markov Logic Network grounding process constists of two phases 1) MLN definintion by fixing MLN schema, domain and weighted formulas; 2) MLN instantiation by assigning truth values to the all possible instantiations of the MLN predicates and using these ground atoms in formulas.

- Equality: $equalX(id_1, id_2)$ means that the values of the attribute X of two tuples id_1 and id_2 should be equal.
- Matching: $matchX(id_1, id_2)$ means that values of two tuples id_1 and id_2 of the attribute X are identified to match.

Given this vocabulary, we can describe our conceptualization of the data quality rules with predicate-calculus sentences.

3.2.1 First-Order Logic Formulae

To demonstrate the first-order syntax, consider the following CFD from the motivation example in Section 2, which states that if any two tuples agree on attribute values for CITY and PHONE, then the attribute values on STREET and ZIP should agree as well:

$$\mathsf{fd} : \text{TRANS}([\text{CITY}, \text{PHONE}] \rightarrow [\text{STREET}, \text{ZIP}])$$

Following the normalization rule for functional dependencies where the right hand side (RHS) of the FD consists of a single attribute, we split the fd rule into two:

$$\mathsf{cfd}' : \text{TRANS}([\text{CITY}, \text{PHONE}] \rightarrow [\text{STREET}], \mathsf{t}_1 = (\_, \_ \parallel \_))$$
$$\mathsf{cfd}'' : \text{TRANS}([\text{CITY}, \text{PHONE}] \rightarrow [\text{ZIP}], \mathsf{t}_2 = (\_, \_ \parallel \_))$$

According to [14] these cfd' and cfd'' can be represented as the following two first-order logic formulae:

$$1)\ \forall\ \text{CITY}, \text{PHONE}, \text{STREET}_1, \text{STREET}_2$$
$$\text{TRANS}(\text{CITY}, \text{PHONE}, \text{STREET}_1)\ \wedge\ \text{TRANS}(\text{CITY}, \text{PHONE}, \text{STREET}_2)$$
$$\Rightarrow \text{STREET}_1 = \text{STREET}_2$$
$$2)\ \forall\ \text{CITY}, \text{PHONE}, \text{ZIP}_1, \text{ZIP}_2$$
$$\text{TRANS}(\text{CITY}, \text{PHONE}, \text{ZIP}_1) \wedge\ \text{TRANS}(\text{CITY}, \text{PHONE}, \text{ZIP}_2)$$
$$\Rightarrow \text{ZIP}_1 = \text{ZIP}_2$$

In words, the formulas state that if two tuples of the TRANS relation agree on the CITY and PHONE values, then their STREET and ZIP values should be the same.

Please note that discovering dependencies from data is out of the scope of this paper. We assume that these dependencies already determined by using methods reviewed in [26] or specified by domain experts manually.

3.2.2 Markov Logic Compilation

Once we have formulated our integrity constraints as first-order logic formulae, they can be translated into a Markov Logic syntax. We illustrate this with the first-order logic formulae cfd' and cfd'' from the previous section: Given that every attribute from the schema TRANS is expressed as a predicate, we declare two observed predicates, namely $\text{CITY}(\mathsf{id}, \mathsf{city})$ and $\text{PHONE}(\mathsf{id}, \mathsf{phone})$. They indicate the values for the fields CITY and PHONE for each tuple, given by its id. The full example is given in Table 1, which shows how a tuple from the example database is translated into grounded atoms.

In addition, we define two hidden predicates for our data quality rule, namely $\text{EQSTREET}(\mathsf{id}, \mathsf{id})$ and $\text{EQZIP}(\mathsf{id}, \mathsf{id})$. These predicates constitute our Markov Logic model that reflects fd rule above:

$$1)\text{CITY}(\mathsf{id}_1, \mathsf{city}) \wedge \text{CITY}(\mathsf{id}_2, \mathsf{city}) \wedge \text{PHONE}(\mathsf{id}_1, \mathsf{phone})$$
$$\wedge\text{PHONE}(\mathsf{id}_2, \mathsf{phone}) \Rightarrow \text{EQSTREET}(\mathsf{id}_1, \mathsf{id}_2)$$
$$2)\text{CITY}(\mathsf{id}_1, \mathsf{city}) \wedge \text{CITY}(\mathsf{id}_2, \mathsf{city}) \wedge \text{PHONE}(\mathsf{id}_1, \mathsf{phone})$$
$$\wedge\text{PHONE}(\mathsf{id}_2, \mathsf{phone}) \Rightarrow \text{EQZIP}(\mathsf{id}_1, \mathsf{id}_2)$$

Please note that using the same argurments in predicates $\text{CITY}(\mathsf{id}, \mathsf{city})$ and $\text{PHONE}(\mathsf{id}, \mathsf{phone})$ encodes equality of the corresponding values in Markov logic program.

Applying the normalization for functional dependencies, we model the fd consistency rule as two formulas in Markov Logic. Note that two different tuples are distinguished by id_1 and id_2. The data

quality rules are the basis for MLN construction. The MLN itself is a template for a probabilistic graphical model (Markov Networks or factor graphs). It constructs the probability distribution over all possible predicates in the MLN. A particular advantage of Markov Logic is its *modularity* while modeling. This means that one can create "complex" statistical models by utilizing various "atomic" models. We consider each declared data quality rule as an "atomic" probabilistic model (function). Together these small models form a "compound" model. Having declared the data quality rules by capturing all correlations and constraints, we are able to write the MLN and perform the inference for the potential predicates. In other words, we are able to predict the possible data quality issues such as inconsistency, missing values, values duplicates etc.

In the example above, we also declared two hidden predicates, one of which is EQSTREET(id, id). This predicate holds the information about possible repairs on the attribute STREET in the TRANSACTIONS tables in our running example from Section 2. Reasoning about such hidden predicates allows the system to decide what attribute gets a particular repair and what records are potential matches. Handling inclusion dependencies (IND) was studied in [5], although we have not conducted any experiments on the INDs, we claim to be able to model INDs in a similar manner as they can be represented in first-order logic: $\forall \mathsf{x}, \mathsf{y} \mathsf{R}(\mathsf{x}, \mathsf{y}) \Rightarrow \exists \mathsf{z}\, \mathsf{S}(\mathsf{y}, \mathsf{z})$ and therefore being directly represented in Markov Logic.

So far we have shown how integrity constraints can be expressed in terms of observed and hidden predicates. These rules are data agnostic [15]. In order to clean a data set with these rules, we need to "ground" these predicates with the available evidence, i.e. the database to be processed. The Data Layer component of our system (see Figure 2 III) performs this task: It takes the content of the database, which is usually presented as a set of tuples, and produces set of observed predicates. Each attribute value of the tuple is converted into an grounded atom. Assuming that n is the number of schema attributes, each tuple from the database is converted into n different Markov Logic evidence atoms.

For example, transaction 2 in our running example is translated into the following 6 grounded atoms: ITEM(2, Galaxy 5), FIRSTNAME(2, Max), LASTNAME(2, Miller), STREET(2, 12 Hay St.), ZIP(2, 818) and PHONE(11234). These are examples of observed predicates for the data set.

3.3 Data Repair as Joint Inference

The main advantage of Markov logic is the ability to reason about complex and interacting relationships, such as different data quality issues. The query computation is also known as *inference*. In addition, through the cleaning rules that we have defined in the Rule-based Data Quality Management component of our system, there are a number of hidden predicates we use for data repair. The Statistical Processing component of our system (see Figure 4 for an overview) executes probabilistic inference to determine probable groundings for the hidden predicates. This component takes as input both the Markov Logic Networks defined through component I, as well as the grounded atoms produced by component III from the available data. It then constructs a probabilistic graphical model using these inputs on top of which it performs probabilistic inference. Figure 4 visualizes the MLN grounding process.

Given an MLN that models data cleaning rules, assuming q as a query (hidden) predicate and x - evidence predicate, our data cleaning problem is to maximize the conditional probability of data repair operations (encoded as hidden predicates) given data in the database:

$$P\left(q|x\right) = \frac{1}{Z} \exp\left(\sum_{F_i} w_i n_i\left(q \cup x\right)\right)$$

Where $n_i(q \cup x)$ is the number of true groundings of formula F_i and equals to 1 if the grounded formula is true, 0 otherwise. It is notable that each formula can consist of hidden predicates q and observed predicates x. The corresponding formula weight is denoted as w_i and Z is a normalization constant. Therefore, to determine the assignment q that maximizes the $P\left(q|x\right)$, we need to maximize the sum of weighted formulas:

$$\arg\max_q P\left(q|x\right) \text{ where } P\left(q|x\right) \propto \exp\left(\sum_{F_i} w_i n_i\left(q \cup x\right)\right)$$

The output of the inference are data repair operations; for example, the hidden predicate EQSTREET we defined in the previous section may be determined by the system to have among other groundings the following likely value: EQSTREET$(1, 3)$, 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.". This output should not be regarded as the direct result of one data quality rule. As we noted in the discussion of the running example, there is an interplay of different rules that affect the probability of the hidden predicates. Therefore, 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 data set. Our data repair prediction problem is to find the most likely grounding of the hidden predicates and this reduces to computing maximum a-posteriori (MAP) inference on the MLN model. In fact, the inference problem instantiated as weighted MAX-SAT problem, which is known to be NP-hard [34]. To overcome this, we use an Integer Linear Programming optimization for MAP inference [32] in our experiments, where each ground clause is compiled as a single linear constraint.

4. EXPERIMENTAL STUDY

We now evaluate our proposed method through an experimental study in which we execute our method on a well-known data sets used for evaluating data cleaning systems. To evaluate the effectiveness of our method we asses the accuracy and the scalability of proposed data cleaning with Markov logic. We provide a quality assessment and runtime evaluation for each of the datasets.

4.1 Experimental Setting

We conducted our experiments on a real-life and synthetic data sets, which were also used by previous work to investigate data cleaning approaches.

Dirty Data. The dirty data we use in our experiments is produced by introducing noise. There are several arts of noise our method is capable to deal with: missing values, errors from the active domain and typos. The initial data set is considered as being clean and therefore acting as a ground truth. We also manually assessed that the ground truth data set is consistent regarding the CFDs and MDs. Afterwards, we inserted noise to the data sets. We conducted our experiments on two data sets with a different noise rate ranging

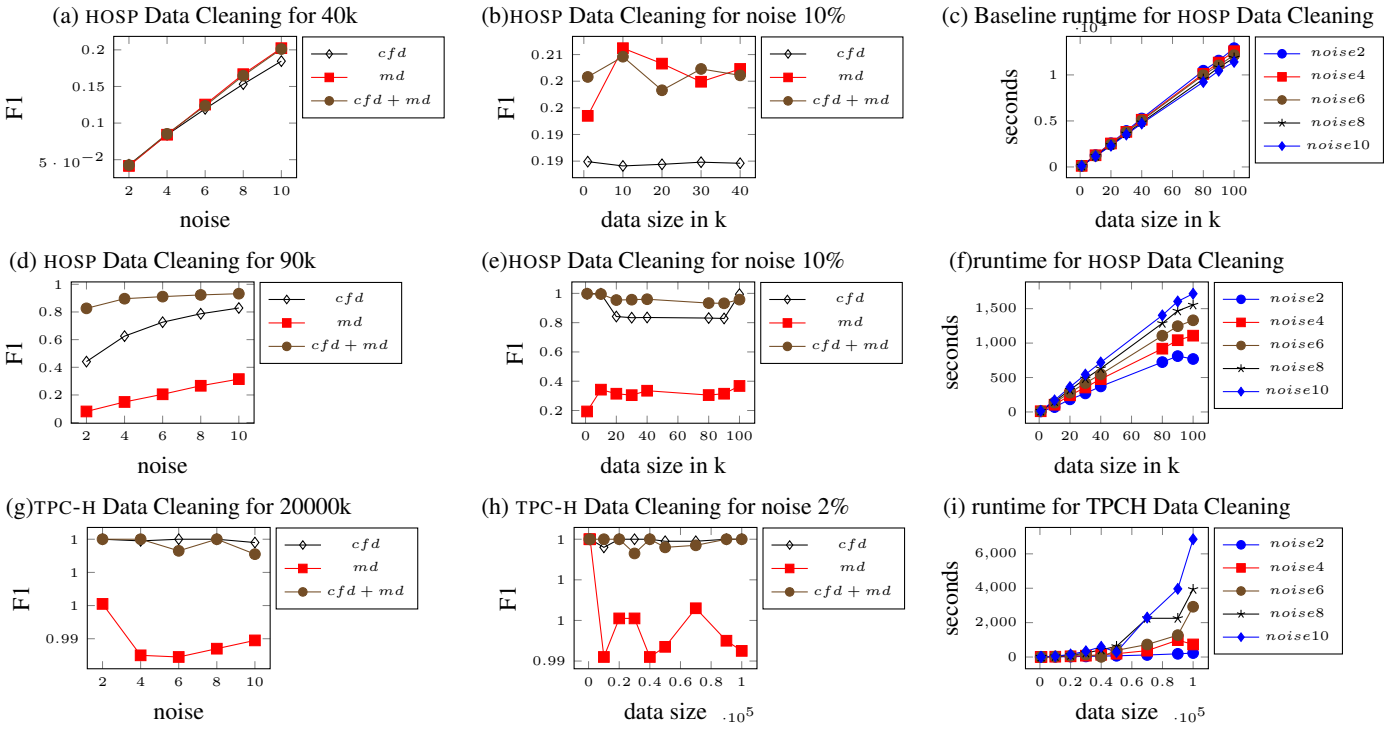

Table 2: Evaluation of the data repair method based on Markov logic applied on the HOSP and TPC-H data sets. (a)-(c) Baseline data cleaning approach for the HOSP data set. (d)-(f) Data repair on HOSP with an extended Markov logic method. (g)-(i) Experimantal study of the Markov logic data cleaning on synthetic data set TPC-H.

from noi%=2% to noi%=10%, which were introduced into different data sets sizes ranging from 1k to 100k data points. The noise rate is a ratio of the number of erroneous values to the total number of values in the data set. Furthermore, we only introduced noise to the attributes which are involved in data quality rules.

HOSP. The HOSP data set is taken from US Department of Health & Human Services[1]. This data set comprises 9 attributes: ADDR, CITY, COND, COUNTRY, HOSPNAME, MEASURE, PHONE, STATE, ZIP. We used 6 CDFs and one MD, which were manually designed. These data quality rules were generously provided to us by the researchers Dallachiesa, M. et al. from [11]. One example of their rules is a CFD that states that if two tuples of HOSP agree on attribute values for ZIP, then they should also agree on COUNTRY and STATE attributes values. They define one MD that makes use of another table, namely US ZIP codes: ZIPCode[2]. This additional data set contains $43K$ tuples with two attributes: ZIP and STATE. The MD defines that if two tuples from HOSP and ZIPCode respectively 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] is well-known data set used in decision support benchmarks for databases. For our experiments we used two relations *Customer* and *Orders*, wich we joined in order to introduce duplications on the *Customer* relations data. The resulted data set 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.

We use the state-of-the-art inference engine for Markov logic *RockIt*, developed by Noessner J. et al. [32] for Markov logic modeling and performing statistical inference. All results in our experiments were achieved by using a Maximum A Posteriori (MAP) inference method for Statistical Relational Learning.

4.2 Evaluation metrics

To evaluate the effectiveness of our data cleaning method we consider two aspects: the accuracy and the scalability of the method.

Accuracy. To assess the quality of the data cleaning framework on relational data, we use *Precision* (P), *Recall* (R) and *F-measure* (F_1). We also acquired master data (also referred to as *"gold standard"*), which is clean and correct. We use this master data to determine $true\ positives(tp)$ - correctly selected by the method attribute values; $false\ positives(fp)$- is a degree of false assignment of the attribute values that are selected to be corrected and $false\ negatives(fn)$- correct attribute value, which were not found by our framework.

$$P = \frac{tp}{tp+fp} \qquad R = \frac{tp}{tp+fn} \qquad F_1 = \frac{2PR}{P+R}$$

Therefore, precision is the ratio of true positives to all values found and recall is the ratio of true positives to the all correct values. We computed the F_1 measure, which is the harmonic mean of precision and recall.

[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/

experiment	numer of tuples	evidence atoms	number of formulas	runtime (sec)
BASE	63K	266K	21	2551
	83K	446K	21	5128
	143K	1286K	21	12576
HOSP	63K	266K	21	242
	83K	446K	21	477
	143K	1286K	21	1107
TPCH	20K	210K	15	41
	40K	419K	15	131
	100K	1055K	15	732

Table 3: Scalability of the data cleaning for TPC-H and HOSP data sets (with fixed noise 4%).

Scalability. The efficiency of our method we asses by running our experiments on data sets of different size ranging from 1k to 100k tuples each.

4.3 Experimental Results

We conducted three series of experiments: two series on HOSP data set with different noise percentage and one experiment set was conducted on the synthetic data set TPC-H:

Baseline: Data Quality Rules Modeling. In this part of experiments we show the feasibility of translating the data quality rules into Markov logic applied on HOSP data. In order to demonstrate the advantage of Markov logic for data cleaning, we performed a set of baseline experiments where we translated *"as it is"* data quality rules into first order logic sentences. In our data cleaning method for the HOSP data we use 6 manually designed CDFs and one MD, which resulted in 15 normalized CFDs. One MD rule is transformed into 2 formulas. All data cleaning rules are positive. In the current experiments we have not modeled negative data cleaning rules, although such rules can be trivially defined within the Markov logic framework. Finally, all interleaved rules were translated into 21 Markov logic formulas. In the following we provide an example of the above specified data quality rules (not normalized CFD), which were defined on pair of tuples. The MD rule is specified on a single relation:

cfd_1 : HOSP([ZIP] $\rightarrow$ [STATE, CITY], $\mathsf{t1} = (\_ \parallel \_, \_)$)
cfd_2 : HOSP([PHONE] $\rightarrow$ [ADDR, ZIP,STATE, CITY],
$\mathsf{t2} = (\_ \parallel \_, \_, \_, \_)$)
.....
md_1 : HOSP[ZIP]=ZIPCODE[ZIP]
$\wedge$HOSP[STATE] $\neq$ ZIPCODE[STATE]
$\rightarrow$ HOSP[STATE] $\rightleftharpoons$ ZIPCODE[STATE]

Markov logic predicates used for data quality formulae are shown in Table 4. After transforming the 100k HOSP tuples into Markov logic grounded atoms, the resulting data comprises 1.3Mio evidence atoms, which are then used for the inference.

Exp-1: Extended Data Quality Rules Modeling. In this part of experiments we demonstrate that by introducing additional knowledge about data set we are able to improve overall errors detection. This part of experiments we conducted on both, real-world and synthetic data.

	HOSP	TPC-H
observed predicates	providerNH(hid, pn) hospitalNameH(hid, n) addressH(hid, add) cityH(hid, c) stateH(hid, st) zipCodeH(hid, code) countryNameH(hid, country) phoneNumberH(hid, numb) conditionH(hid, cond) measureCodeH(hid, mcode) measureNameH(hid, mname) scoreH(hid, score) zipZ(zid, code) stateZ(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) orderStatus(id, st) totalPrice(id, p) orderDate(id, d) orderPriority(id, pr) clerk(id, c)
hidden predicates	eqHospitalNameH(hid, n, hid, n) eqAddressH(hid, add, hid, add) eqCityH(hid, c, hid, c) eqStateH(hid, st, hid, st) eqZipCodeH(hid, code, hid, code) eqCtrNameH(hid, ctr, hid, ctr) eqPhoneNumberH(hid, nr, hid, nr) eqMNameH(hid, mn, hid, mn) eqCondH(hid, cond, hid, cond) matchState(hid, zid) matchZipCode(hid, code)	eqNames(id, n, id, n) eqAddr(id, add, id, add) eqNatkey(id, nkey, id, nkey eqPhone(id, ph, id, ph) eqAcc(id, a, id, a) eqMrkt(id, m, id, m) matchPhone(id, id) matchAddr(id, id)

Table 4: Markov logic predicates used in data quality rules.

HOSP Quality Rules. The data quality rules set for HOSP then has been extended for additional conditions like reducing the search space. Each first order logic formula, which represent the RHS of the normalized data quality rule
$\text{ATTR}(\mathsf{id1}, \mathsf{v1}) \wedge \text{ATTR}(\mathsf{id2}, \mathsf{v2})$
becomes an inverse part
$!\text{ATTR}(\mathsf{id1}, \mathsf{v2}) \wedge !\text{ATTR}(\mathsf{id2}, \mathsf{v1})$.
This additional part denotes that we consider only tuples with different values. Here the normalized cfd_1 rule is being compiled as:

w_1 : $\text{ZIP}(\mathsf{id1}, \mathsf{code}) \wedge \text{ZIP}(\mathsf{id2}, \mathsf{code}) \wedge$
$\text{STATE}(\mathsf{id1}, \mathsf{s1}) \wedge \text{STATE}(\mathsf{id2}, \mathsf{s2}) \wedge$
$!\text{STATE}(\mathsf{id1}, \mathsf{s2}) \wedge !\text{STATE}(\mathsf{id2}, \mathsf{s1}) \Rightarrow \text{EQSTATE}(\mathsf{id1}, \mathsf{s1}, \mathsf{id2}, \mathsf{s2})$

TPC-H Quality Rules. For this data set 9 CFDs and 3 MDs have been written. One example of the rules is a CFD that states if two tuples agree on C_CUSTKEY, then they should agree on C_NAME and C_ADDRESS attributes. For this part of experiments, MDs are designed on the same schema TPC-H (T, T). These MDs state that for any pair of tuples (t_1, t_2) if the LHS is similar, then the attribute values on the RHS should be identified. In the following we provide an except

of the data quality rules we created for TPC-H:

cfd_1 : T([C_CUSTKEY] $\rightarrow$ [C_NAME,C_ADDRESS],
$\mathsf{t1} = (\_ \parallel \_, \_)$)
. . . . . .
md_1 : T[C_ADDRESS]=T[C_ADDRESS] $\rightarrow$
T[C_PHONE] $\rightleftharpoons$ T[C_PHONE]
md_2 : T[C_NAME]=T[C_NAME] $\rightarrow$
T[C_ADDRESS] $\rightleftharpoons$ T[C_ADDRESS]

Markov logic predicates used for data quality rules are shown in Table 4. After transforming the 100k TPC-H tuples into Markov logic grounded atoms, the resulting data comprises 1 Mio evidence atoms, which are then used for the inference.

Exp-2: Effectiveness. We evaluated the accuracy of our method applied on different noise rate and different data set size. For this part of experiment we introduced noise by adding either typos or replacing values of attributes (active domain errors). In our experiments we do not distinguish these two art of errors. As shown in Figure 2 (a)-(c) the Baseline on HOSP experiments reveal moderate F_1-score. The reason for such low F_1 values are very low precision values. Results of using extended Markov logic programs for data cleaning is demonstrated by (d)-(f) for HOSP and (g)-(i) for TPC-H. Here we could achieve excellent results in errors identifying. We compare results of separate execution of CDFs and MDs. After that we ran our experiments by combining CFDs and MDs. Because CFDs and MDs are modeled together in a single model for identifying dirty data, they are treated jointly and therefore we do not have any influence on order of the data quality rules execution. The provided results clearly demonstrate that joint execution improves the overall result. In particular, the low F_1 score of MDs can be explained by the low precision and high recall. Although, by adding CFDs to MDs the overall F_1 score improves. This experiment tells us that the holistic execution of data cleaning rules achieves better results.

Exp-3: Efficiency. To assess the efficiency of our method we studied the runtime of our method for different data size and noise rate. This is shown in Figure 2 (c) for Baseline, (f) for HOSP and (i) for TPC-H and in Table 3. This experiment tells us that our method is robust against noise, but adding more noise leads to higher runtime. By reducing the search space in Markov logic we were able to reduce the runtime by factor 10. Furthermore, the predicate-arity used in Markov logic formulas influences the runtime of data cleaning. Empirically, the optimal predicate-arity for this method is two. All provided experiments were performed on predicates with two to four arguments.

4.4 Summary

These results indicate that multiple types of data cleaning rules should be considered *holistically*, which confirms the same statement made in the previous research in [11], [19], and [18]. Adding domain or structural knowledge about data into the Markov logic program improves overall data cleaning. Furthermore, by using joint inference we are able to achieve very good results without defining the order of the data cleaning rules execution. The results also show that our method, which jointly models data quality rules by using Markov logic, is competitive to the state-of-the-art data cleaning systems. One of the critical points in employing the Markov logic is the runtime of the inference engine. Although, we use the, at the moment, fastest Markov logic inference engine - RockIt, the runtime varies depending on the noise percentage in the data. Furthermore, the runtime depends on the data size and the design of the hidden predicates and therefore MLN as whole. To summarize: *1*) The joint modeling of data quality rules results in better data correction. As demonstrated in our experiments, by combining matching and repairing processes, we achieved better results than by performing these processes separately. Please note that the prediction of errors is a result of the joint inference, therefore there is no order preservation for rules execution; *2*) When no matches are found we are still able to repair data with CDFs; *3*) By using the probabilistic-logical framework Markov logic we can benefit from its flexibility in constraints definition and joint inference over different data repair and match rules. Therefore it is a great fit into the data quality management field; *4*) Our data cleaning solution is as scalable as the Markov logic inference engine is, however we are able to demonstrate results on data sets of reasonable sizes.

5. RELATED WORK

Our research builds on previous work from two areas: *1*) Data Quality Management and *2*) Statistical Relational Learning, namely probabilistic-logical languages.

Data Quality Management: A rich theoretical and practical investigation into data cleaning was presented in [19] and [20]. Fan W. et al. [18] first proposed to unify the record matching and data repairing processes. They also proved that this unification greatly contributes to improvement of the data quality. Based on this fundamental research we propose to use probabilistic-logical frameworks to model the interaction of data quality rules and show that Markov logic enables us to simultaneously use MDs and CFDs. This enables the incorporation of integrity constraints, such as inclusion constraints. Holistic data cleaning methods based on integrity constraints and denial constraints with ad-hoc predicates has been also studied in [9]. In this approach, authors consider the generalization of the integrity constraints by translating them into denial constraints. However, denial constraints can not express the inclusion dependencies, hence in our work we use clausal form of the first-order predicate logic to express all kind of integrity constraints.

A generalization of dependencies was proposed by the Llunatic system in [21]. They proposed a new language based on equality generated dependencies to standardize the way to express intra- and inter-dependencies. A clustering-based declarative approach for deduplication by considering data constraints was suggested in the Dedupalog language in [2].

A reference system to ours is UNICLEAN, proposed in [19]. Being a repairing system, UNICLEAN also creates fixes by declaring cleaning rules and unifying matching and repairing processes. In general, UNICLEAN achieves high accuracy by executing three algorithms for generating fixes in consecutive manner. In contrast, our data cleaning solution is a one-step process for executing the inference on MLN modeled according to data cleaning rules. Furthermore, as our experiments show, our approach does not require the availability of master data.

The NADEEFF system from [11] is the closest system to ours regarding to the data cleaning systems requirements coverage. Similar to our system, they treat data quality rules holistically. In contrast to NADEEFF we *1*) Use first-order logic to define all the kind

of data quality rules and therefore in our system we do not have black-boxes in form of UDFs. Furthermore, we claim better usability of the system through the declarative rules formulation; *2*) Perform data cleaning as an inference process on Markov Networks. Through the formulation of data quality rules as Markov logic, the inference is fully hidden from the user. This achieves a great deal of complexity reduction regarding such data cleaning systems;

Relevant work in statistical inference and data cleaning was conducted by Mayfield Ch. and his team in [28]. Their system ERACER was 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 apply a MAP inference to infer the types of errors and their sources. We also assume that the data quality rules based on FDs, CFDs and MDs are already defined. Profiling algorithms to discover CFDs and MDs automatically are presented in the work of Song and Chen in [37] and Fan W. et al. in [17]. Applying machine learning for entity deduplication was demonstrated in [24]. The work in [4] uses a probabilistic model for duplicate detection with uncertain outcomes. Throughout our work we use SRL, which is an extension of machine learning w.r.t. existing relations in the data.

The most recent work in data curation that subsumes the data cleaning is the Data Timer System [39]. The researchers presented 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 emphasize on joint execution of data cleaning rules, which can be declared as *"soft"* or *"hard"*. Furthermore, we do not use 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 was introduced in [40]. Their data cleaning system SCARE does not focus on database constraints due to databases contains non-trivial errors and accurately predicts replacement for the noisy values. In contrast to SCARE, our solution is capable to model and predict not only missing values imputation and data consistency but also data deduplication.

Statistical Relational Learning: An advantage of probabilistic modeling for improving data quality was investigated in [30] and [7]. Markov logic as a formalism for joint inference has been successfully used in a number of tasks, including natural language processing [6], [35], [29], ontology alignment and data integration [31] and coreference resolution [33], [36]. These research results demonstrated the clear advantage of the joint modeling vs. pipeline execution. We are the first to apply this formalism to data cleaning and show the benefits of a joint data cleaning approach that uses MLNs as a framework for interacting data quality rules.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a declarative data-cleaning approach based on Statistical Relational Learning and probabilistic inference. Generally, data quality rules represent relationships between attributes in the database schema and these rules are mainly based on functional dependencies on top of a database schema. We demonstrated how such functional dependencies, expressed as first-order logic formulas, can be translated into probabilistic logical languages, allowing us to reason over inconsistencies or duplicates in a probabilistic way. Our proposed approach allows the usage of probabilistic joint inference over interleaved data cleaning rules to improve data quality. In an experimental study on a noisy real-life data set we demonstrated the viability of our proposed approach and showed that our method outperforms state-of-the-art systems. By using a declarative probabilistic-logical formalism such as Markov Logic, we are potentially able to incorporate more semantic constraints (latent semantic factors) and therefore extend traditional data quality rules.

We are currently extending the algorithm to massive data set settings and preliminary results are encouraging. With regards to experimenting with modeling additional semantic constraints, larger and more heterogeneous data sets, present and future research will focus on the following topics: *a*) Extending our data cleaning solution to the semi-structured data such as XML, JSON, RDF and linguistic data in general to transfer existing data cleaning technologies to the new use-cases. *b*) Improving data quality for distributed data. Much research on data cleaning is conducted on data, which is located on a single machine. Recent development in massive data storage systems (e.g. cloud systems) also generated the need to create cleaning framework that could operate in the cloud. The results that we have presented in 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.

7. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] A. Arasu, C. Ré, and D. Suciu. Large-scale deduplication with constraints using dedupalog. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE '09, pages 952–963, Washington, DC, USA, 2009. IEEE Computer Society.

[3] L. Bertossi. *Database Repairs and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[4] G. Beskales, M. A. Soliman, I. F. Ilyas, S. Ben-David, and Y. Kim. Probclean: A probabilistic duplicate detection system. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 1193–1196. IEEE, 2010.

[5] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 143–154. ACM, 2005.

[6] W. Che and T. Liu. Jointly modeling wsd and srl with markov logic. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 161–169. Association for Computational Linguistics, 2010.

[7] K. Chen, H. Chen, N. Conway, J. M. Hellerstein, and T. S. Parikh. Usher: Improving data quality with dynamic forms. *Knowledge and Data Engineering, IEEE Transactions on*, 23(8):1138–1153, 2011.

[8] M. Chu. *Blissful Data: Wisdom and Strategies for Providing Meaningful, Useful, and Accessible Data for All Employees*. Amacom, 2004.

[9] X. Chu, I. F. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 458–469. IEEE, 2013.

[10] U. N. R. Council. *Frontiers in Massive Data Analysis*. The

National Academies Press, 2013.

[11] M. Dallachiesa, A. Ebaid, A. Eldawy, A. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. Nadeef: A commodity data cleaning system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 541–552, New York, NY, USA, 2013. ACM.

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

[13] W. W. Eckerson. Data warehousing special report: Data quality and the bottom line. 2002. Online; accessed 12-July-2014.

[14] R. Fagin. Horn clauses and database dependencies. *J. ACM*, 29(4):952–985, Oct. 1982.

[15] W. Fan and F. Geerts. Foundations of data quality management. *Synthesis Lectures on Data Management*, 4(5):1–217, 2012.

[16] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33(2):6:1–6:48, June 2008.

[17] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *IEEE Trans. on Knowl. and Data Eng.*, 23(5):683–698, May 2011.

[18] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Interaction between record matching and data repairing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 469–480, New York, NY, USA, 2011. ACM.

[19] W. Fan, S. Ma, N. Tang, and W. Yu. Interaction between record matching and data repairing. *J. Data and Information Quality*, 4(4):16:1–16:38, May 2014.

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

[21] F. Geerts, M. Giansalvatore, P. Papotti, and D. Santore. The llunatic data cleaning framework. *PVLDB*, 6(9):625–636, 2013.

[22] M. R. Genesereth and N. J. Nilsson. Logical foundations of artificial. *Intelligence. Morgan Kaufmann*, 1987.

[23] L. Getoor and B. Taskar. *Introduction to statistical relational learning*. MIT press, 2007.

[24] S. Guo, X. L. Dong, D. Srivastava, and R. Zajac. Record linkage with uniqueness constraints and erroneous values. *Proceedings of the VLDB Endowment*, 3(1-2):417–428, 2010.

[25] I. F. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga. Cords: Automatic discovery of correlations and soft functional dependencies. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 647–658, New York, NY, USA, 2004. ACM.

[26] J. Liu, J. Li, C. Liu, Y. Chen, et al. Discover dependencies from dataâĂŤa review. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):251–264, 2012.

[27] D. Lowd and P. Domingos. Efficient weight learning for markov logic networks. In *Knowledge Discovery in Databases: PKDD 2007*, pages 200–211. Springer, 2007.

[28] C. Mayfield, J. Neville, and S. Prabhakar. Eracer: A database approach for statistical inference and data cleaning. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 75–86, New York, NY, USA, 2010. ACM.

[29] I. Meza-Ruiz and S. Riedel. Jointly identifying predicates, arguments and senses using markov logic. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '09)*, 2009.

[30] J. I. Naus, T. G. Johnson, and R. Montalvo. A probabilistic model for identifying errors in data editing. *Journal of the American Statistical Association*, 67(340):943–950, 1972.

[31] M. Niepert, J. Noessner, C. Meilicke, and H. Stuckenschmidt. Probabilistic-logical web data integration. In *Reasoning Web. Semantic Technologies for the Web of Data*, pages 504–533. Springer, 2011.

[32] J. Noessner, M. Niepert, and H. Stuckenschmidt. Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. In *AAAI*, 2013.

[33] H. Poon and P. Domingos. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the conference on empirical methods in natural language processing*, pages 650–659. Association for Computational Linguistics, 2008.

[34] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[35] S. Riedel and I. Meza-Ruiz. Collective semantic role labelling with markov logic. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL' 08)*, pages 193–197, 2008.

[36] P. Singla and P. Domingos. Entity resolution with markov logic. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 572–582. IEEE, 2006.

[37] S. Song and L. Chen. Discovering matching dependencies. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1421–1424. ACM, 2009.

[38] M. Spies. Knowledge discovery from constrained relational data: A tutorial on markov logic networks. In *Business Intelligence*, pages 78–102. Springer, 2013.

[39] M. Stonebraker, G. Beskales, A. Pagan, D. Bruckner, M. Cherniack, S. Xu, V. Analytics, I. F. Ilyas, and S. Zdonik. Data curation at scale: The data tamer system. In *In CIDR 2013*, 2013.

[40] M. Yakout, L. Berti-Équille, and A. K. Elmagarmid. Don't be scared: Use scalable automatic repairing with maximal likelihood and bounded changes. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 553–564, New York, NY, USA, 2013. ACM.