ProbClean: A Probabilistic Duplicate Detection System

George Beskales  Mohamed A. Soliman  Ihab F. Ilyas  Shai Ben-David  Yubin Kim

School of Computer Science, University of Waterloo
{gbeskale,m2ali,ilyas,shai,yl2kim}@uwaterloo.ca

Abstract—One of the most prominent data quality problems is the existence of duplicate records. Current data cleaning systems usually produce one clean instance (repair) of the input data, by carefully choosing the parameters of the duplicate detection algorithms. Finding the right parameter settings can be hard, and in many cases, perfect settings do not exist.

We propose ProbClean, a system that treats duplicate detection procedures as data processing tasks with uncertain outcomes. We use a novel uncertainty model that compactly encodes the space of possible repairs corresponding to different parameter settings. ProbClean efficiently supports relational queries and allows new types of queries against a set of possible repairs.

I. INTRODUCTION

Many current commercial tools (e.g., Oracle Data Integrator [1]) support the extraction, transformation and loading (ETL) of (possibly unclean) data into a trust-worthy (cleansed) database. In a typical ETL process (illustrated in Figure 1(a)), data are first extracted from the source data stores, which can be legacy database systems, files, web pages, or streaming data sources. After extraction, transformation of data is performed to standardize data formats, remove duplicate records, resolve violations of business rules and integrity constraints, as well as performing schema mapping to ensure that data fit the target data warehouse schema. Finally, data are loaded to the central data warehouse. The transformation process depends on user defined cleaning specifications (e.g., carefully chosen parameter settings of the cleaning algorithms, or fixed rules and business logic) to produce a single clean data instance (i.e., a repair of the unclean database).

In traditional data warehousing settings, the transformation process usually involves uncertainty about the correct cleaning specifications (e.g., the similarity threshold that determines whether two records are duplicates). Generating one clean database instance requires resolving such uncertainty, which can potentially result in loss of information (e.g., merging two records that are incorrectly believed to be duplicates). Moreover, users of the cleansed data may have different cleaning requirements.

In [3], we introduced a new ETL approach that defers the destructive operations of the transformation process to query time as we show in Figure 1(b). Specifically, in our proposal, the transformation process is divided into an offline expensive procedure that produces multiple possible repairs, and an online procedure that can be efficiently executed at query time. We propose the use of probabilistic databases principles and techniques to model and maintain multiple possible repairs of the unclean data. Queries are evaluated against a probabilistic data model that compactly encodes a set of possible repairs. Moreover, online cleaning specifications can be provided by the user to process a specific clean instance.

In this demonstration, we introduce ProbClean, a probabilistic ETL tool that focuses on a specific data transformation task, which is detecting and eliminating duplicate records. Duplicate records are defined as (possibly non-identical) records in the unclean database that refer to the same real world entity. In this case, a repair of an unclean database is a clustering of records where records that refer to the same real world entity are grouped in the same cluster.

Figure 2 shows an example input relation representing sample census data that contains duplicate records. Duplicate detection algorithms generate a clustering of records (represented as sets of record ID’s in Figure 2), where each cluster is a set of duplicates that are eventually merged into one representative record.

The current duplicate detection approaches identify records as either duplicates or non-duplicates based on the given cleaning specifications (e.g., a threshold on records’ similarity).
Hence, the result is a single clustering (repair) of the input relation. On the other hand, in the probabilistic duplicate detection approach, this restriction is relaxed to allow for uncertainty in deciding on the true duplicates (e.g., based on multiple possible similarity thresholds). The result is a set of possible clusterings (repairs) as shown in Figure 2.

II. UNCERTAIN DUPLICATE DETECTION

We generate all possible repairs corresponding to a set of possible parameter settings of any fixed parameterized clustering algorithm (e.g., single-linkage [5]). This approach is motivated by the fact that similarity measures between records are usually noisy, which makes finding the right parameter setting difficult, and in many cases, perfect settings do not exist. We focus on a class of clustering algorithms, namely hierarchical clustering algorithms, to perform uncertain clustering due to two features of such a class: (1) the number of generated repairs is linear in the number of records, which allows for efficiently generating, storing and querying the possible repairs, and (2) hierarchical clustering algorithms can be used to generate multiple repairs through simple modifications to the algorithms.

Hierarchical clustering algorithms cluster records of an input relation $R$ in a hierarchy that represents a series of possible clusterings starting from a clustering containing each record in a separate cluster, to a clustering containing all records in one cluster. The algorithms use specific criteria to determine which clustering to return. For example, Figure 3(a) gives the hierarchy generated by a greedy agglomerative clustering algorithm [5], for the relation $R = \{r_1, \ldots, r_5\}$. The algorithm parameter $\tau$ represents a threshold on inter-cluster distances, which are represented by the $Y$-axis. Setting $\tau$ to the value depicted by the solid horizontal line in Figure 3(a) results in the clustering $\{r_1, r_2\}, \{r_3, r_4, r_5\}$.

In order to capture multiple possible outcomes of a hierarchical clustering algorithm, we view the parameter $\tau$ as a random variable rather than a fixed value. We denote by $f_{\tau}$ the probability density function of $\tau$. $f_{\tau}$ is defined over the interval $[\tau^l, \tau^u]$. Applying a clustering algorithm $\mathcal{A}$ to an unclean relation $R$ using a parameter value $t \in [\tau^l, \tau^u]$, generates a possible clustering (i.e., repair) of $R$, denoted $\mathcal{A}(R, t)$. Note that multiple parameter values may lead to the same clustering.

The set of possible repairs, denoted $\mathcal{X}$, is defined as $\{\mathcal{A}(R, t) : t \in [\tau^l, \tau^u]\}$. The set $\mathcal{X}$ defines a probability space created by drawing random parameter values in $[\tau^l, \tau^u]$, based on the density function $f_{\tau}$, and using the algorithm $\mathcal{A}$ to generate the possible repairs corresponding to these values. The probability of a specific repair $X \in \mathcal{X}$, denoted $Pr(X)$, is equal to the probability of the parameter settings resulting in the repair $X$.

For example, in Figure 3(b), setting the threshold $\tau$ to the range depicted by the shaded area results in four possible repairs that are shown in the same figure.

III. MODELING POSSIBLE REPAIRS

We define an uncertain clean relation (Unclean relation for short) that encodes the possible repairs $\mathcal{X}$ of an unclean relation $R$, which are generated by a parameterized clustering algorithm $\mathcal{A}$. A U-clean relation, denoted $R^c$, is a set of $c$-records where each $c$-record is a representative record of a cluster of records in $R$. Attributes of $R^c$ are all attributes of $R$, in addition to two distinguished attributes: $C$ and $P$. Attribute $C$ of a $c$-record is the set of records identifiers in $R$ that are clustered together to form this $c$-record. The attribute $P$ of a $c$-record represents the parameter settings of the clustering algorithm $\mathcal{A}$ that lead to generating the cluster represented by this $c$-record.

Attribute $P$ is represented as an interval within the range of the parameter $\tau$. Note that it is possible to have overlapping clusters represented by $c$-records of $R^c$ since $R^c$ may encapsulate multiple clusterings of records in $R$.

Figure 4 shows an example U-clean relation, named Person$^c$ that captures possible repairs generated by the algorithm $\mathcal{A}$ using a uniformly distributed parameter $\tau$ in the range $[0, 10]$. For brevity, we omit Attributes Name, ZIP and Birth Date from Relation Person$^c$. We assume that the attribute Income of each cluster is the average value of its member records.

IV. IMPLEMENTATION IN RDBMS

In [3], we show how to construct U-clean relations efficiently and how to store U-clean relations in a relational
A. Implementing U-clean Relations

We implement Attributes $C$ and $P$ in a relational database as abstract data types (ADTs). We implement a set of user defined functions (UDFs) over the attributes $C$ and $P$ to manipulate their contents during query processing. More specifically, we provide the following functions:

- $\text{Conj}(C_1,\ldots,C_n)$ and $\text{Conj}(P_1,\ldots,P_n)$ compute the conjunctions of multiple clusters $C_1,\ldots,C_n$ and multiple parameter settings $P_1,\ldots,P_n$, respectively.
- $\text{Disj}(C_1,\ldots,C_n)$ and $\text{Disj}(P_1,\ldots,P_n)$ compute the disjunctions of multiple clusters $C_1,\ldots,C_n$ and multiple parameter settings $P_1,\ldots,P_n$, respectively.
- $\text{Contains}(P,x)$ returns $\text{True}$ iff the parameter settings $P$ contain a given parameter setting $x$.
- $\text{ContainsBaseRecords}(C,S)$ returns $\text{True}$ iff a set of base records identifiers $S$ is contained in a cluster $C$.
- $\text{Prob}(P,f_{\tau_1},\ldots,f_{\tau_d})$ computes the probability of parameter settings $P$, given the probability distribution functions (PDFs) $f_{\tau_1},\ldots,f_{\tau_d}$ of the parameters $\tau_1,\ldots,\tau_d$ that appear in $P$. PDFs are represented as histograms.
- $\text{MostProbParam}(\text{UR},f_{\tau_1},\ldots,f_{\tau_d})$ computes a parameter setting corresponding to the most probable repair of a U-clean relation $\text{UR}$, given the parameters PDFs $f_{\tau_1},\ldots,f_{\tau_d}$.

Algorithmic details of these functions are described in [3].

B. Supported Query Types

Our system allows evaluation of selection, projection, join, and aggregation over U-clean relations through the UDFs defined in Section IV-A. For example, in order to report the distinct incomes of persons according the possible repairs encoded in Person$^c$ (shown in Figure 4), we submit the following SQL query:

$$
\text{SELECT Income, } \text{Disj}(C), \text{Disj}(P) \\
\text{FROM Person}^c \\
\text{GROUP BY Price}
$$

which returns a U-clean relation that is shown in Figure 5.

Another example query type is to extract the clean instance corresponding to a given parameter setting. For example, the repair of Relation Person (Figure 4) corresponding to the parameter value $\tau = 4.1$ can be retrieved using the following SQL query:

$$
\text{SELECT ID, Name, ZIP, BirthDate, Income} \\
\text{FROM Person}^c \\
\text{WHERE Contains}(P,4.1)
$$

which results in the repair represented by the $c$-records $\{CP2, CP4, CP5\}$.

In [3], we introduce other query types such as reporting the most probable repair, obtaining the probability of cluttering specific records, and retrieving $c$-records with probabilities above a given threshold.

V. DEMONSTRATION OUTLINE

Our system is mainly composed of two components: (1) an application that implements the uncertain cleaning algorithms and provides a GUI for submitting user queries, and (2) a modified version of the PostgreSQL database system [2] that includes the ADTs and UDFs defined in Section IV-A to allow storing U-clean relations and executing user queries. In our demonstration, we illustrate the process of uncertain duplicate detection, and querying the U-clean relations.

A. Constructing U-clean Relations

We demonstrate the process of cleaning an input relation using an uncertain duplicate detection algorithm. The user provides specifications for the cleaning process which are used for generating an output U-clean relation. We describe our demonstration scenario as follows.

1) The user selects the source of the unclean data, and the name and location of the output U-clean relation.
2) The user specifies the method that computes the similarity between pairs of records. A distance function is defined for each attribute in the unclean relation. The overall distance between two records is computed by aggregating the individual attributes distances (e.g., using the functions $\text{sum}$, $\text{max}$ or a linear combination
of attributes distances). Once the similarity graph is constructed, a visualization of the resulting graph is provided to the user to allow viewing and editing of the graph edges (Figure 6). Additionally, a histogram of the pair-wise distances is reported to the user.

3) The user adjusts multiple knobs of the uncertain clustering process such as the used clustering algorithm, the parameter range of the algorithm, and the probability distribution of the parameter. Another optional input is the maximum cluster size as shown in Figure 7.

4) The user defines how attributes of a cluster are derived from attributes of the member records.

After cleaning the input relation, the resulting U-clean relation is presented to the user to view the contained c-records and their corresponding records in the unclean relation. Moreover, the encoded repairs are reported to the user as shown in Figure 8. Statistics about each repair (e.g., the repair probability, the number of records and the corresponding range of parameter) are also reported. We also show the running time of each step of the cleaning process to the user.

B. Queries over U-clean Relations

ProbClean supports several query types against U-clean relations as shown in Figure 9. Queries are created using a GUI, and then translated into SQL statements that are executed by the modified PostgreSQL DBMS. The user is given the opportunity to edit the generated SQL statements as well as providing her own SQL statements.

REFERENCES