Data Cleaning

Xu Chu

Synonyms

Data cleansing.

Definition

Data cleaning is used to refer to all kinds of tasks and activities to detect and repair errors in the data.

Overview

Enterprises have been acquiring large amounts of data from a variety of sources to build their own "Data Lakes", with the goal of enriching their data asset and enabling richer and more informed analytics. Data collection and acquisition often introduce errors in data, for example, missing values, typos, mixed formats, replicated entries for the same real-world entity, outliers, and violations of business rules.

A Kaggle's 2017 survey about the state of data science and machine learning reveals that dirty data is the most common barrier faced by workers dealing with data (Kaggle 2017). Not surprisingly, developing effective and efficient data cleaning solutions is a challenging venue and is rich with deep theoretical and engineering problems.

There are various surveys and books on different aspects of data quality and data cleaning. Rahm and Do (Rahm and Do 2000) give a classification of different types of errors that can happen in an Extract-Transform-Load (ETL) process, and survey the tools available for cleaning data in an ETL process; Bertossi (Bertossi 2011) provides complexity results for repairing inconsistent data, and performing consistent query answering on inconsistent data; Hellerstein (Hellerstein 2008) focuses on cleaning quantitative data, such as integers and floating points, using mainly statistical outlier detection techniques; Fan and Geerts (Fan and Geerts 2012) discuss the use of data quality rules in data consistency, data currency, and data completeness, how different aspects of data quality issues might interact; Dasu and Johnson (Dasu and Johnson 2003) summarize how techniques in exploratory data mining can be integrated with data quality management; Ilyas and Chu (Ilyas et al 2015) provide taxonomies and example algorithms of qualitative error detection and repairing techniques; and Ganti and Das Sarma (Ganti and Sarma 2013) focus on an operator-centric approach for developing a data cleaning solution, which involves the development of customizable operators that could be used as building blocks for developing common solutions.

This chapter covers four commonly encountered data cleaning tasks, namely, outlier detection, rule-based data cleaning, data transformation, and data deduplication. Since there is a very large body of work on these tasks, this chapter only intends to provide an introduction to each data cleaning task and categorize various techniques proposed in the literature to tackle each task. There are also many complementary research efforts to data cleaning that are not covered in this chapter, such as data profiling, provenance and meta data management, data discovery, consistent query answering, meta-data management, schema mapping, and data exchange.

Key Research Findings

Outlier Detection

Outlier detection refers to the activity of detecting outlying values, and is an quantitative error detection task. While an exact definition of an outlier depends on the application, there are some commonly used definitions, such as *"an outlier is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism"* (Hawkins 1980). For example, for a company whose employees' salaries are mostly around $100K$, an employee with a salary of $10K$ can be considered to be an outlying record.

Multiple surveys and tutorials have been published to summarize different definitions of outliers, and algorithms for detecting them (Hodge and Austin 2004; Chawla and Sun 2006; Aggarwal 2013). In general, outlier detection techniques fall into three categories: statistics-based, distance-based, and model-based. Statistics-based outlier detection techniques assume that the normal data points would appear in high probability regions of a stochastic model, while outliers would occur in the low probability regions of a stochastic model (Grubbs 1969). They can often provide a statistical interpretation for discovered outliers, or a score/confidence interval for a data point being an outlier, rather than making a binary decision. Distance-based outlier detection techniques often define a distance between data points, which is used for defining a normal behavior, for example, normal data point should be close to a lot of other data points, and data points that deviate from such normal behavior are declared outliers (Knorr and Ng 1998). A major advantage of distance-based techniques is that they are unsupervised in nature and do not make any assumptions regarding the generative distribution for the data. Instead, they are purely data driven. Model-based outlier detection techniques first learn a classifier model from a set of labeled data points, and then apply the trained classifier to a test data point to determine whether it is an outlier (De Stefano et al 2000). Model-based approaches assume that a classifier can be trained to distinguish between the normal data points and the anomalous data points using the given feature space. They label data points as outliers if none of the learned models classify them as normal points.

Enforcing Data Quality Rules

One common way to detect and rectify errors in databases is through the enforcement of data quality rules, often expressed as integrity constraints (ICs) (Chu et al 2013; Fan and Geerts 2012). In this context, (qualitative) error detection is the process of identifying *violations of ICs*, namely, subsets of records that cannot co-exist together; and error repair is the exercise of modifying the database, such that the violations are resolved, and the new data conforms to the given ICs.

Figure 1 shows a typical workflow of qualitative data cleaning. In order to clean a dirty dataset using qualitative data cleaning techniques, there need to be data quality rules that reflect the semantics of the data. One way to obtain data quality rules is by consulting domain experts, which requires a lot of domain expertise and is usually a time-consuming processing. An alternative approach is to design an algorithm to automatically discover data quality rules. Given a dirty dataset and the associated data quality rules, the error detection step detects violations of the specified rules in the data. A violation is *minimal set of cells in a dataset that cannot coexist together*. Finally, given the violations and the rules that generate those violations, the error repair step produces data updates, which are applied to the dirty dataset. The error detection and the error repair loop goes on until the data conforms to the data quality rules, namely, there is no violations in the data.

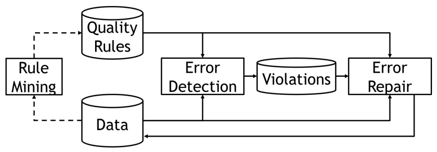


Fig. 1: Rule-based data cleaning workflow with an optional rule mining step, the error detection step, and the error repair step

TID	FN	LN	LVL	ZIP	ST	SAL
t_1	Anne	Nash	5	10001	NM	110,000
t_2	Mark	White	6	87101	NM	80,000
t_3	Jane	Lee	4	10001	NY	75,000

Table 1: An example employee table

Example 1. Consider Table 1 that contains employee records in a company. Every tuple specifies a person in a company with her id (GID), name (FN, LN), level (LVL), zip code (ZIP), state (ST), and salary (SAL). Suppose two data quality rules hold for this table. The first rule states that, if two employees have same zip code, they must have the same state. The second rule states that among employees working in the same state, a higher level employee cannot earn less salary than a lower level employee.

Given these two data quality rules, the error detection step detects two violations. The first violation consists of four cells $\{t_1[ZIP], t_1[ST], t_3[ZIP], t_3[ST]\}$, which together violate the first data quality rules. The second violation consists of six cells $\{t_1[ROLE], t_1[ST], t_1[SAL], t_2[ROLE], t_2[ST], t_2[SAL]\}$, which together violate the second data quality rule. The data repair step takes the violations and produces an update that changes $t_1[ST]$ from “NM” to “NY”, and the new data now has no violation with respect to the two rules.

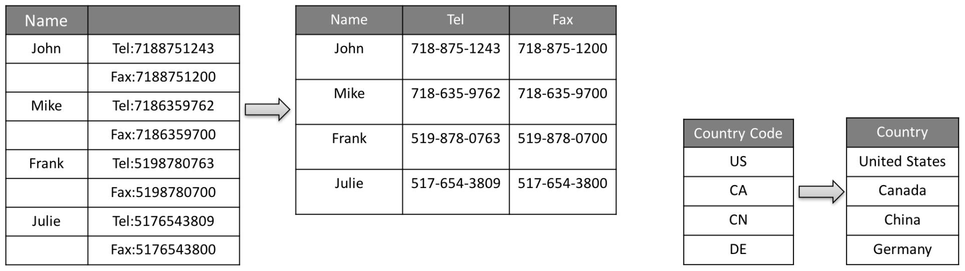
Data Transformation

Data transformation refers to the task of transforming data from one format to another. Data transformation tasks can be classified into two types in general: *syntactic data transformations* and *semantic transformations*. Syntactic transformations aim at transforming a table from one syntactic format to another, often without requiring external knowledge or reference data. On the other hand, semantic transformations usually involve understanding the meaning/semantics, instead of simply as a sequence of characters; hence, they usually require referencing external data sources.

Example 2. Figure 2a shows a syntactic transformation, where a tabular data containing person names, telephone numbers, and fax numbers, originally arranged in a tabular format with phone and fax numbers stored in the same column as a complex data type, are flattened into a normal relational table representation to enable efficient querying of this data. In addition, the telephone numbers and the fax numbers are also transformed by adding two dashes between the nine digits. Both cases do not require external knowledge to perform these transformations.

Figure 2b shows a semantic transformation, where country codes are transformed into country names; obviously, it requires external knowledge, such as an external knowledge base that contains both full and abbreviated country names.

There are generally three major components/dimensions of a syntactic data transformation system: *language*, *authoring*, and *execution*. Since there are many ways to transform a dataset, syntactic data transformation solutions usually adopt a *transformation language* that limits the space of possible transformations. The language could consist of a finite set of operations allowed on a table (Raman and Hellerstein 2001; Kandel et al 2011), such as splitting a column and merging two columns, or it could consist of a set functions for string manipulations (Gulwani 2011), such as extracting substring and concatenating two substrings. There is a balance that needs to be achieved when choosing a particular language: it needs to be expressive enough so that it captures many real-world transformation tasks, and at the same time, restricted enough to allow for effective and easy authoring of the transformations. For a specific



Name	
John	Tel:7188751243
	Fax:7188751200
Mike	Tel:7186359762
	Fax:7186359700
Frank	Tel:5198780763
	Fax:5198780700
Julie	Tel:5176543809
	Fax:5176543800

Name	Tel	Fax
John	718-875-1243	718-875-1200
Mike	718-635-9762	718-635-9700
Frank	519-878-0763	519-878-0700
Julie	517-654-3809	517-654-3800

(a) Syntactic Transformation

Country Code
US
CA
CN
DE

Country
United States
Canada
China
Germany

(b) Semantic Transformation

Fig. 2: Example Transformations

transformation task, syntactic transformation tools have different interaction models with the users to *author a transformation program* using the adopted language. The interaction models could be generally classified as declarative, example-driven, or proactive. In the declarative model, users specify the transformations directly, usually through a visual interface. In the example-driven model, users are required to give a few input-output examples, based on which the tool automatically infer likely transformations that match the provided examples. In the proactive interaction model, the transformation tool provides hints or highlights for which data needs transformation, and sometimes even suggests possible transformations so that users simply need to accept or reject the suggestions. Finally, *transformation execution* applies the specified transformations to the dataset. Since there might be multiple specified transformations from the previous step, transformation tools usually offer assistance to the users in selecting the desired transformation, for example, by displaying the effect of the specified transformation immediately on a sample data, or by providing interpretable statements of the specified transformations. Different transformation tools offer different capabilities along these three dimensions.

Semantic transformations usually involve understanding the meaning/semantics, or the typical use of the data, instead of simply as a sequence of characters. In contrast to syntactic transformations, semantic transformations cannot be computed by solely looking at input values; rather, they usually require external data sources to look up the values that need to be transformed. Example-driven techniques have been mostly used for semantic transformations (Singh and Gulwani 2012; Abedjan et al 2016). Users are usually asked to provide a few input-output examples, and semantic transformation tools then will search the available external data sources to find a “query” that matches the examples. The query can then be used to transform new unseen data.

Data Deduplication

Data deduplication, also known as duplicate detection, record linkage, record matching, or entity resolution, refers to the process of identifying tuples in one or more relations that refer to the same real world entity (Elmagarmid et al 2007). A data deduplication process usually involves many steps and choices, including designing similarity metrics to evaluate the similarity for a pair of records, training classifiers to determine whether a pair of records are duplicates or not, clustering all records to obtain clusters of records that represent the same real-world entity, consolidating clusters of records to unique representations, designing blocking or distributed strategies to scale up the deduplication process, and involving humans to decide whether a record pair are duplicates when machines are uncertain.

Example 3. Figure 3 illustrates a typical example of data deduplication. The similarities between pairs of records are computed, and are shown in the similarity graph (upper right graph in Figure 3). The missing edges between any two records indicate that they are non-duplicates. Records are then clustered together based on the similarity graph. Suppose the user sets the threshold to be 0.5, namely, any record pairs having similarity greater than 0.5 are considered duplicates. Although Record P_1 and P_5 have similarity less than 0.5, they are clustered together due to transitivity; that is, they are both considered duplicates to Record P_2. All records in the same cluster are consolidated into one record in the final clean relation.

Applications of Data Cleaning

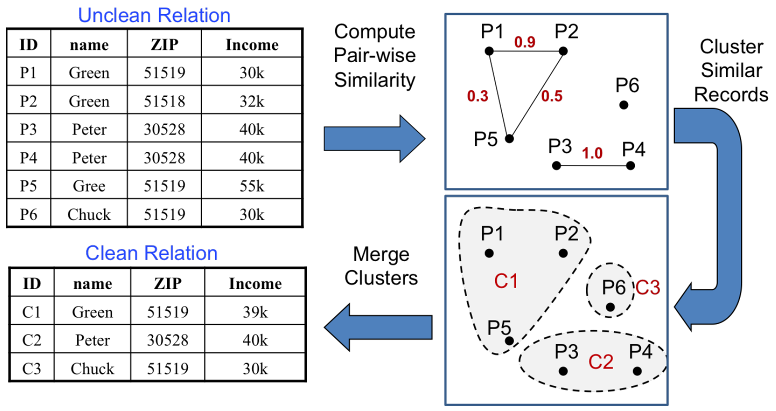
Data cleaning is a necessary step in many data-driven analytics. Different data cleaning tasks target different types of errors.

Applications of outlier detection include network intrusion detection, financial fraud detection, and abnormal medical condition detection. For example, in the context of computer networks, different kinds of data, such as operating system calls and network traffic, are collected in large volumes. Outlier detection can help with detecting possible intrusions and malicious activities in the collected data.

Rule-based data cleaning can help clean any relational databases where data quality rules can be defined. The richer the semantics of the data is, the better rule-based data cleaning techniques are at detecting and repairing violations.

Data transformations are used in a variety of tasks, and at different stages of the ETL life cycle. For example, before running a data integration project, transformations are often used to standardize data formats, to enforce standard patterns, or to trim long strings. Transformations are also used at the end of the ETL process, for example, to merge clusters of duplicate records, to find a unique representation for a cluster of records (aka golden record), or to prepare data to be consumed by analytics tools. Data transformations can also be seen as a tool for data repair in rule-based data cleaning, since it can be used to "transform" erroneous data.

Duplicate records can occur due to many reasons. For example, a customer might be recorded multiple times in a customer database if the customer used different names at the time of purchase; a single item might be represented multiple times in an online shopping Website; and a record might appear multiple time after a data integration project because that record had different repre-



ID	name	ZIP	Income
P1	Green	51519	30k
P2	Green	51518	32k
P3	Peter	30528	40k
P4	Peter	30528	40k
P5	Gree	51519	55k
P6	Chuck	51519	30k

ID	name	ZIP	Income
C1	Green	51519	39k
C2	Peter	30528	40k
C3	Chuck	51519	30k

Fig. 3: A typical deduplication task.

sentations in original data sources. Data deduplication targets specifically duplicate records, and resolves them.

Future Directions for Research

Data cleaning is a complicated process, and an end-to-end data cleaning solution usually involves many different cleaning sub-tasks. We have discussed techniques for tackling four common data cleaning tasks. There are still many challenges and opportunities in building practical data cleaning systems: (1) the scale of data renders many data cleaning techniques insufficient. New cleaning solutions must adapt to growing datasets of the Big Data era, for example, by leveraging sampling techniques or distributed computation. (2) although there are existing research about involving humans to perform data deduplication, for example, through active learning (Tejada et al 2001; Sarawagi and Bhamidipaty 2002; Arasu et al 2010), involving humans in other data cleaning tasks, such as repairing IC violations, and taking user feedback in discovering of data quality rules, is yet to be explored; (3) a significant portion of data is residing in semi-structured formats (e.g., JSON) and un- structured formats (e.g., text documents). Data quality problems for semi-structured and unstructured data remain largely unexplored; and (4) there is significant concerns about data privacy as increasingly more individual data are collected by governments and enterprises. Data cleaning is by nature a task that requires examining and searching through raw data, which may be restricted in some domains including finance and medicine. How to perform most data cleaning tasks, while preserving data privacy, remains an open challenge.

References

Abedjan Z, Morcos J, Ilyas IF, Ouzzani M, Papotti P, Stonebraker M (2016) Dataxformer: A robust transformation discovery system. In: Proc. 32nd Int. Conf. on Data Engineering, pp 1134–1145

Aggarwal CC (2013) Outlier Analysis. Springer

Arasu A, Götz M, Kaushik R (2010) On active learning of record matching packages. In: Proc. ACM SIGMOD Int. Conf. on Management of Data, pp 783–794

Bertossi LE (2011) Database Repairing and Consistent Query Answering. Morgan & Claypool Publishers

Chawla S, Sun P (2006) Outlier detection: Principles, techniques and applications. In: Advances in Knowledge Discovery and Data Mining, 10th Pacific-Asia Conf.

Chu X, Ilyas IF, Papotti P (2013) Holistic data cleaning: Putting violations into context. In: Proc. 29th Int. Conf. on Data Engineering, pp 458–469

Dasu T, Johnson T (2003) Exploratory Data Mining and Data Cleaning. John Wiley & Sons, Inc.

De Stefano C, Sansone C, Vento M (2000) To reject or not to reject: that is the question-an answer in case of neural classifiers. IEEE Trans Systems, Man, and Cybernetics 30(1):84–94

Elmagarmid AK, Ipeirotis PG, Verykios VS (2007) Duplicate record detection: A survey. IEEE Trans Knowl and Data Eng 19(1):1–16

Fan W, Geerts F (2012) Foundations of Data Quality Management. Synthesis Lectures on Data Management

Ganti V, Sarma AD (2013) Data cleaning: A practical perspective. Synthesis Lectures on Data Management 5(3):1–85

Grubbs FE (1969) Procedures for detecting outlying observations in samples. Technometrics 11(1):1–21

Gulwani S (2011) Automating string processing in spreadsheets using input-output examples. In: Proc. 38th ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages, pp 317–330

Hawkins D (1980) Identification of outliers, vol 11. Chapman and Hall

Hellerstein JM (2008) Quantitative data cleaning for large databases. United Nations Economic Commission for Europe (UNECE)

Hodge VJ, Austin J (2004) A survey of outlier detection methodologies. Artificial Intelligence Review 22(2):85–126

Ilyas IF, Chu X, et al (2015) Trends in cleaning relational data: Consistency and deduplication. Foundations and Trends® in Databases 5(4):281–393

Kaggle (2017) URL `https://goo.gl/ZAZGsD`

Kandel S, Paepcke A, Hellerstein J, Heer J (2011) Wrangler: Interactive visual specification of data transformation scripts. In: Proc. SIGCHI Conf. on Human Factors in Computing Systems, pp 3363–3372

Knorr EM, Ng RT (1998) Algorithms for mining distance-based outliers in large datasets. In: Proc. 24th Int. Conf. on Very Large Data Bases, pp 392–403

Rahm E, Do HH (2000) Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin 23:2000

Raman V, Hellerstein JM (2001) Potter's wheel: An interactive data cleaning system. In: Proc. 27th Int. Conf. on Very Large Data Bases, pp 381–390

Sarawagi S, Bhamidipaty A (2002) Interactive deduplication using active learning. In: Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp 269–278

Singh R, Gulwani S (2012) Learning semantic string transformations from examples. Proc VLDB Endowment 5(8):740–751

Tejada S, Knoblock CA, Minton S (2001) Learning object identification rules for information integration. Information Systems 26(8):607–633