Scalable data cleansing based on qualitative attributes

Mika Huttunen
Helsinki University

ABSTRACT

This is an example abstract...

Keywords

data cleansing, big data, scalability

1. INTRODUCTION

Nowadays companies are gathering large amounts of data in different ways such as via user input, and all kinds of sensors. It's also becoming easier and easier for them to use data in all kinds of decision making, analytics, and for example in automating human-involved tasks. Problems arise when the used data is *dirty*, or invalid, and thus can lead into making incorrect decisions. These can sometimes cause serious issues - especially in health care and financing sides [6].

Humans often make data input errors via misspelling, and sensors may grab unwanted noise along the data they're designed to catch. In fact, over 25% of critical data in the world's top companies is flawed [7]. Not to mention that today, the variety of data is also large which leads into collecting data of different formats together. *Data cleansing* is the solution to the above-mentioned problem. For data processors, understanding data cleansing, and the problems it tries to solve is thus naturally important.

Data cleansing is a research field that explores ways to improve *quality* of dirty data. If data isn't of high quality, it means that it has usually both *schema*, and *instance* level problems [8]. Another used definition for dirty data is that it doesn't meets its usage needs. A simple example of that would be such that a front developer of a service can't provide end-users a good UI because the server cannot deliver all kinds of useful data because there are faults in the way the data is stored in the database.

Figure 1 shows how *single-source problems* with data quality can yet be divided into schema and instance level data problems. Single-source problems simply stand for such that can occur in data that is stored in a single site with possible

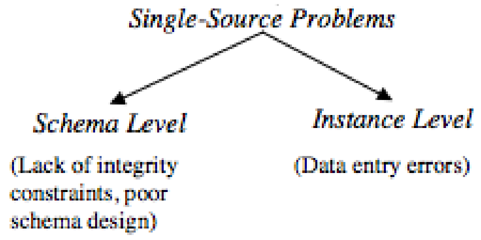

Figure 1: Single-source data quality problems [8]

data replication on the database side. The same problems can also arise in *multi-source* systems where the data may also be scattered across several sites.

Instance level problems in data arise as attributes of data *tuples* being out of their scope, or just wrong. These contain misspellings, and *outliers* by for example possible sensor errors, and processing errors before gathered data is actually stored. These problems can be detected via *quantitative techniques* related to data cleansing that focus on error detection and correction based on numerical attributes of data [5]. If for example some framework can reduce multi-dimensional data in two-dimensional space, it's fairly easy for an expert to detect outliers in that space, and thus possible errors in the data.

Schema level problems on the other hand arise as *violations* of *rules* such as *functional dependencies* (FD) [6] (p. 289 – 291). As an example, functional dependency

$$ZIP \rightarrow STATE \tag{1}$$

defines a constraint between data attributes ZIP and $STATE$ such that ZIP implies the $STATE$. In other words, if we had a dataset with two tuples having the same ZIP attribute, but different values of $STATE$, and we trust rule 1, at least one of them is erroneous.

Duplicate (or partically duplicate) data entries can also be considered both schema level, and instance level data quality problems [6] (p. 283). They usually appear as the same entries with some attributes having different values from each other, and thus should be classified as instance level problems. They can however be detected by similar rules that are used for other detecting schema level problems. Consider us having a person register where each person has its name, phone number, address, zip code (ZIP), and state ($STATE$) defined. Now we could have tuples t_1 and t_2 with the same name, phone number, and address, but different zip codes. An error between t_1 and t_2 could be detected by rule 1, and solved either automatically, or via human interaction.

In traditional ways, the data errors are detected by algorithms that compare data tuples trying to find rule violations. Rules can either be pre-defined by an expert, or automated code, or such that the data cleansing framework itself generates them by processing the data. The found errors can afterwards be fixed either via totally automatic, or (partially) human guided procedures.

This seminar report focuses on *qualitative* data cleansing frameworks that are highly scalable for *big data*. They detect data errors via traditional data quality rules such as FDs, *CFDs* (conditional functional dependencies), *DCs* (denial constraints) which are explained in the paper by Ilyas and Chu [6] (p. 287 - 301). I will leave further investigation of them up to the reader.

Some data cleansing frameworks like *BigDansing* also apply *UDFs* (user-defined functions) as data quality rules [7]. UDFs extend traditional rules by allowing more complex norms using procedural language thus improving the data cleansing accuracy. At the same time some frameworks don't actually require cleansing the whole data at all. *SampleClean* for instance applies sampling for improving answer quality for queries [10].

The rest of the report is divided as follows: in Section 2, I will introduce recent findings, and history related to big data cleansing, and where we stand on the field at the moment. Section 3 discusses recent frameworks introduced in Section 2 in more detail. It does also analysis and comparison with them. Section 4 discusses the open challenges in big data cleansing field, and Section 5 has my own ideas for future work. Section 6 concludes the report.

2. RELATED WORK

Data cleansing has been an interesting topic for decades, and there has been a large amount of research on the field in the recent years, Extensive summary from 2015 by Ilyas and Chu discusses recent techniques on cleansing relational data [6]. The summary introduces algorithms for finding different kinds of data quality rules based on a database instance, and its schema. Error detection is afterwards possible by just doing tuple-wise comparisons between the data records.

In 2013, Chu et al. introduced *Holistic data cleaning* for doing error detection based on multiple rules at the same time [4]. It can be shown that this improves the error detection accuracy versus tracking invalid tuples by doing tuple-wise comparisons one rule at a time.

For error repairing, Ilyas and Chu introduce a classification of different data repairing techniques [6] (p. $330-332$). Based on the classification, they also discuss recent frameworks that apply different techniques. Figure **??** represents data repairing techniques classification.

As we can see from the figure, we can have three different *repair targets*. We can either repair data, rules, or both. So far, we have only been discussing data repairing, and this report will bypass techniques for the other two targets. Repairing rules however stands for trusting that data is correct, and modifying data quality rules to match with the current data. Trusting both data and rules on the other hand is a technique for modifying both data and rules in such way that a new data and a set of rules is found so that the data matches with the rules.

Data repairing can be done

We can repair data either in holistic way by

, or by repairing tuples treated by

The data cleansing involves

finding correct data quality rules, and of course both error detection and repairing of relational data.

Error detection process can be

The errors are generally detected in data as violations of rules such as FDs, CFDs and DCs.

Ilyas and Chu

A big issue in error detection is the actual discovery of the data quality rules.

A big problem in error detection

Given a dataset I, and an empty set of traditional data quality rules R, the main problem in error detection phase is finding

Ilyas and Chu define violation respect to a rule r being the minimal subset of database cells such that at least one of the cells has to be modified to satisfy r. Here cell stands for an attribute value of a tuple, or a row. The same definition can of course be extended to non-relational, but structured data that NoSQL document databases can consist of.

the manual construction of R is time-consuming, and requires lots of domain expertise. Thus having automatic tools for generation of R based on I is essential [2]. Previously implemented $TANE$ and $FASTFD$ that Ilyas and Chu discuss [6] can be used for finding FDs based on a relational data, and its schema, while $FASTDC$ can be used for finding DCs.

3. STATE-OF-ART FRAMEWORKS

This section discusses the recent frameworks in big data cleansing field. I introduce BigDansing [7], SampleClean [10], Dis-Dedup [3], and HoloClean [9]. Although the frameworks are quite different, some deeper analysis between them can be found at the end of this section.

3.1 BigDansing

BigDansing is a highly scalable framework for data cleansing. It does parallelised error detection by data quality rules as UDFs

I introduce BigDansing that is a highly scalable framework for distributed data cleansing, SampleClean which avoids the problem of cleansing the actual whole data, Dis-Dedup, recent framework particularly for distributed duplicate data records detection, and HoloClean that unites a range of data repairing methods in a common framework.

The following part of the report is yet to be written out. But the plan was to especially discuss recent frameworks for error detection with relatively small datasets, and then introduce more recent, scalable ones like BigDansing (distributed rule-based error detection and correcting), SampleClean (probability based fixed data sampling while having dirty database on the background), DeDoop / Dis-Dedup (frameworks specifically designed for duplicate tuples detection), and HoloClean. HoloClean is by far the most interesting of the above-mentioned frameworks. This is because it's designed to be applicable for detecting errors that are untrackable by comparing tuples with each other by using only a single rule. [6].

Abedjan et al. show in their research paper from 2016 that even though frameworks such as BigDansing are highly scalable, and can be applied to cleansing all kinds of datasets, they can track only approximately 40% of all possible er-

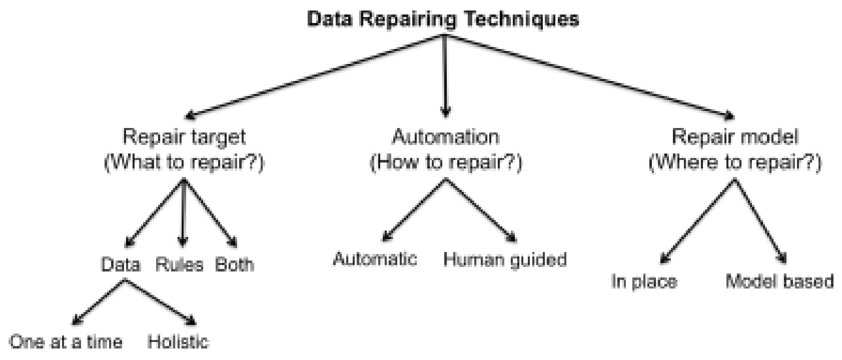
Data Repairing Techniques
Repair target (What to repair?)
Automation (How to repair?)
Repair model (Where to repair?)
Data
Rules
Both
Automatic
Human guided
In place
Model based
One at a time
Holistic

Figure 2: Data repairing techniques [?]

rors - even with fully optimised configuration parameters. If an expert is however used to cleanse data by applying several frameworks in the right order for specific data cleansing tasks, much more than 40% of the possible errors are trackable, but the cleansing accuracy still hardly reaches even 70% [1].

This specific research probably inspired partially the same research team in designing HoloClean of which purpose was to overcome the long lasted challenges with data cleansing tools not achieving good cleansing accuracy. According to the Rekatsinas et al., HoloClean achieves over twice the accuracy of existing state-of-art methods with average precision of 90% [9]. What makes HoloClean so much better than the existing state-of-art methods is that accordingly to its name, it's a *holistic* data cleansing framework. Holisticity means that it can do tuple-based comparisons by not just using one rule at the time, but the set of all defined rules.

Holisticity in data cleansing was no way a new innovation. Chu et al. published a research paper on the topic in 2013 [4]. For now, I haven't done enough research to find out, why HoloClean wasn't invented sooner though. Perhaps the real need for it wasn't really known before 2016 when Abedjan et al. published their work about cleansing accuracy going significantly up when carefully selecting, which data cleansing frameworks to use, for what tasks and in what order [1].

4. REFERENCES

[1] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting data errors: Where are we and what needs to be done? *Proc. VLDB Endow.*, 9(12):993–1004, Aug. 2016.

[2] X. Chu and I. F. Ilyas. Qualitative data cleaning. *Proc. VLDB Endow.*, 9(13):1605–1608, Sept. 2016.

[3] X. Chu, I. F. Ilyas, and P. Koutris. Distributed data deduplication. *Proc. VLDB Endow.*, 9(11):864–875, July 2016.

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

[5] J. M. Hellerstein. Quantitative data cleaning for large databases, 2008.

[6] I. F. Ilyas and X. Chu. Trends in cleaning relational data: Consistency and deduplication. *Found. Trends databases*, 5(4):281–393, Oct. 2015.

[7] Z. Khayyat, I. F. Ilyas, A. Jindal, S. Madden, M. Ouzzani, P. Papotti, J.-A. Quiané-Ruiz, N. Tang, and S. Yin. Bigdansing: A system for big data cleansing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1215–1230, New York, NY, USA, 2015. ACM.

[8] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23:2000, 2000.

[9] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *Proc. VLDB Endow.*, 10(11):1190–1201, Aug. 2017.

[10] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 469–480, New York, NY, USA, 2014. ACM.