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 [4].

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 [5]. 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 [6]. 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

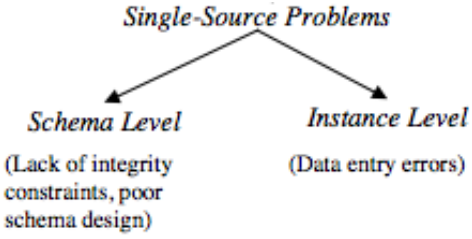

Figure 1: Single-source data quality problems [6]

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. Today, there is no good solution for cleaning huge amounts of data in multi-source systems although some research towards this has been made [4] (p. 379).

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 [3]. 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 *integrity constaints* (IC), and *functional dependencies* (FD) in the data [4] (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 $STATE$, based on trusting 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 [4] (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 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 two tuples with the same name, phone number, and address, but different zip codes. An error between these two tuples found be detected by rule 1, and could be solved either automatically, or via human interaction.

This seminar report focuses on highly scalable *qualitative* data cleansing techniques. They focus on detecting data errors in *big data* via rules such as FDs, *CFDs* (conditional functional dependencies), and *DCs* (denial constraints). Ilays and Chu well-define these in their paper [4] (p. 287 - 301). In traditional ways, the data errors are detected by algorithms that compare data tuples trying to find violations based only on user-defined rules, or also such rules or *patterns* that the data cleansing framework itself generates by processing the data. The found errors can afterwards be fixed either via totally automatic, or (partially) human guided procedures.

The rest of the report is divided as follows: in Section 2, I will introduce recent findings related to big data cleansing, and where we stand on the field at the moment. Section 3 gives background for following sections by introducing different kinds of techniques related to handling big data, but also for getting more accurate data cleansing results. Section 4 discusses recent frameworks for scalable data cleansing big data, and in Section 5, I have a brief conclusion to summarise the report.

2. LITERATURE REVIEW

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 [4] discusses recent techniques on both error detection and repairing on relational data. The errors are generally detected in data as violations of rules such as FDs, CFDs and DCs. They 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 (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 data quality rules such as FDs, CFDs and DCs is time-consuming and requires lots of domain expertise. Thus having automatic tools for their generation is essential [1]. Previously implemented $TANE$ and $FASTFD$ that Ilyas and Chu discuss [4] can be used for finding FDs based on a relational data, and its schema, while $FASTDC$ can be used for finding DCs.

They can be constructed either manually by experts, or by using

The errors are generally detected in data as violations of rules such as FDs, CFDs and DCs. They 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 (row). The same definition can of course be extended to non-relational, but structured data that NoSQL document databases can consist of.

Based on the above definition, they introduce algorithms such as $FASTFD$ for discovering FDs based on a dataset and its schema (p. 294), and its extension for CFD discovery.

Figure 2 represents their classification of qualitative error detection techniques.

The errors are generally detected in data as violations of rules such as FDs, CFDs and DCs. The detection itself can be automatic, or human guided, and the layer where the

They 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 (row). The same definition can of course be extended to non-relational, but structured data that NoSQL document databases can consist of.

Based on the above definition, they introduce algorithms such as $FASTFD$ for discovering FDs based on a dataset and its schema (p. 294), and its extension for CFD discovery.

Qualitative Data Cleaning

- What kind of errors, how and where to detect them - Data repairing – Trusting integrity constraints / rules, data or both? – Automatic / human guided (training ML model, suggesting fixes etc.) – Repairing on place / generating a model for repairing

3. 2015: TRENDS

- Data deduplication can be seen as enforcing a key constraint defined on all the attributes of a relational schema, since two duplicate tuples can be seen as a violation of the key constraint.

3.1 Error detection

- Given a relational database instance I of schema R and a set of integrity constraints ?, we need to find another database instance I' with no violations with respect to ?.

- Given a dirty database instance, the first step toward cleaning the database is to detect and surface anomalies or errors.

- Automation (How to Detect?): We classify proposed approaches according to whether and how humans are involved in the anomaly detection process. Most techniques are fully automatic, for example, detecting violations of functional dependencies, while other techniques involve humans, for example, to identify duplicate records.

- Let ? denote a set of integrity constraints (ICs). We use I |= ? to indicate that database Instance I conforms with the set of integrity constraints ?. We say that I' is a repair of I with respect to ? if I' |= ?, and CIDs(I) = CIDs(I') (i.e., no deletions or insertions are allowed to clean a database instance).

- Constraints (refer to p. 289 - 301) – FD, CFD, DC (subsumes the former so explaining them not useful?)

- Duplicate record detection – Similarity graph based on some similarity metric – Clustering enough similar records together (A <–> B and B <–> C —-> A <–> C) — In other words, finding connected components in the similarity graph – Merging the records into a single records (based on their attributes)

Holistic data cleaning (p. 318): - Automatically detects violations based on multiple ICs

Where to detect (p. 322) - Problematic that data errors are often found in business layers

3.2 Error reparing

- Repair target (What to repair?) – Repairing algorithms make different assumptions about the data and the quality rules: (1) trusting the declared integrity constraints, and hence, only data can be updated to remove errors; (2) trusting the data completely and allowing the relaxation of the constraints, for example, to address schema evolution and

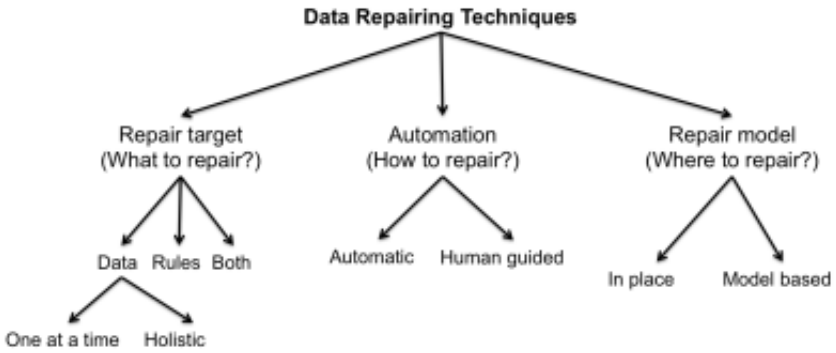

Figure 2: Data repairing techniques [2]

obsolete business rules; and finally (3) exploring the possibility of changing both the data and the constraints.

- -Holistic repairing (p. 336 - 342) — We describe these techniques as 'One at a time techniques'. Most available data repairing solutions are in this category. They address one type of error, either to allow for theoretical quality guarantees, or to allow for a scalable system.

— Multiple data quality problems, such as missing values, typos, the presence of duplicate records, and business rule violations, are often observed in a single data set. These heterogeneous types of errors interplay and conflict on the same dataset, and treating them independently would miss the opportunity to correctly identify the actual errors in the data. We call the proposals that take a more holistic view of the data cleaning process Holistic cleaning approaches.

– Mention also rules-only repairing, and both data and rules repairing (p. 345 - 350) — Continuous data cleansing

- Automation (How to repair?) – Fully automatic (trying to minimize cost between moving from I to I') vs. human involvement (verify / suggest fixes, train ML model)

– Fully automatic: cardinality-minimal and cost-minimal repairs (p. 351) — - Unverified fixes, may introduce new errors during the process

– Automatic data repairing techniques use heuristics, such as minimal repairs to automatically repair the data in situ, and they often generate unverified fixes. Worse still, they may even introduce new errors during the process. It is often difficult, if not impossible, to guarantee the accuracy of any data repairing techniques without external verification via experts and trustworthy data sources. (p. 357)

– Example 3.10. Consider two tuples t1 and t8 in Table 1.1; they both have the same values ?25813? for ZIP attribute, but t1 has ?WA? for ST attribute and t8 has ?WV? for ST attribute. Clearly, at least one of the four cells t1[ZIP], t8[ZIP], t1[ST], t8[ST] has to be incorrect. Lacking other evidence, existing automatic repairing techniques [17, 26] often randomly choose one of the four cells to update. Some of them [17] even limit the allowed changes to be t1[ST], t8[ST], since it is unclear which values t1[ZIP], t8[ZIP] should take if they are to be changed. (p. 357)

– Guided data repair – KATARA (scale-out?) – Data Tamer (scale-out?)

- Repair model (Where to repair?) – Repair database instance in place vs. generate a model for repairing the instance

– Most of the proposed data repairing techniques (all discussed so far) identify errors in the data, and find a unique fix of the data either by minimally modifying the data according to a cost function or by using human guidance (Figure 3.17(a)). (p. 366)

– As follows, we describe a different model-based approach for nondestructive data cleaning. Data repairing techniques in this category do not produce a single repair for a database instance; instead, they produce a space of possible repairs (Figure 3.17(b)). The space of possible repairs is used either to answer queries against the dirty data probabilistically (e.g., using possible worlds semantics) [12], or to sample from the space of all possible clean instances of the database [9, 11].

– Probabilistic Cleaning — Probabilistic deduplication (probability based duplication "removal")

– Sampling the Space of Possible Repairs — Relates closely to SampleClean for big data cleansing

BIG DATA CLEANING - Pyrkimys vähentää tarvittavaa ihmiskommunikaatiota - Mahd. vain datan pienen osajoukon käsittely ja tämän perusteella todennäköisyyksiin perustuvia vastauksia / jatkokäsittelyä

- Deduplicating – Blocking, windowing, canopy clustering

- Sampling (SampleClean)

- Incremental data cleaning – Entity resolution (ER) algorithm

- Distributed data cleaning – MapReduce, Spark, Dedoop (with ER) / Dis-Dedup, – BigDansing (runs on top of a data processing platform like DBMS or MapReduce) – HoloClean? – CleanM ?

- Problems with data partitioned across multiple sites (p. 379) – Objective to minimize data shipment cost

4. BIG DATA CLEANSING

This section discusses problems that arise when handling big data.

- "Scalability. Large volumes of data render most current techniques unusable in real settings. "

This section also discusses recent frameworks designed particularly for handing big data.

I did further investigation with BigDansing, SampleClean, HoloClean, Dis-Dedup, and CleanM frameworks. Ihab F. Ilyas has been part of research team of BigDansing, HoloClean and Dis-Dedup joined by Xu Chu (with the exception of not being part of BigDansing research team) et al. and thus I picked also CleanM for further investigation not to only have content from the same researchers. Ilyas and

Chu's have nevertheless conducted lots of essential research especially related to qualitative data cleansing.

TODO: I also discuss CLAMS which is not actually a big data cleansing framework, but rather one for managing data quality via ICs with limited schema information (?).

BigDansing is a framework that can be run for example on top of MapReduce to distribute error detection work.

HoloClean is

Dis-Dedup is a framework for detecting duplicate data records.

5. WEAK POINTS ON PAPERS

This section discusses weak points of the papers related to the papers current ongoing situation with big data cleansing.

6. OWN IDEAS

Give new ideas/algorithms/experiments on this research problem. This part is very important, because it shows the potential of the author to be an independent innovative researcher. This section can be as long as possible.

Come up with a new idea / algorithm that could be used for dealing with big data and does something better than the existing tools available...

7. CONCLUSION

Summarize the research problem and the main contributions of previous papers. The main weakness of previous works could be also mentioned here. Some future works can be described as well.

8. REFERENCES

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

[2] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 2201–2206, New York, NY, USA, 2016. ACM.

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

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

[5] 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.

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

9. EXAMPLES

THEOREM 1. *Let f be continuous on $[a, b]$. If G is an antiderivative for f on $[a, b]$, then*

$$\int_a^b f(t)dt = G(b) - G(a).$$

Table 1: Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

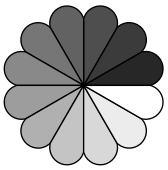

Figure 3: A sample black and white graphic that has been resized with the `includegraphics` command.

Definition 1. If z is irrational, then by e^z we mean the unique number which has logarithm z:

$$\log e^z = z$$

PROOF. Suppose on the contrary there exists a real number L such that

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = L.$$

Then

$$l = \lim_{x\to c} f(x) = \lim_{x\to c}\left[gx \cdot \frac{f(x)}{g(x)}\right] = \lim_{x\to c} g(x) \cdot \lim_{x\to c} \frac{f(x)}{g(x)} = 0 \cdot L = 0,$$

which contradicts our assumption that $l \neq 0$. □