Bulletin of the Technical Committee on

Data Engineering

June 2016 Vol. 39 No. 2

IEEE Computer Society

Letters

Special Issue on Data Quality

Conference and Journal Notices

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TCDE web page is http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

The Current Issue

Data is the "new oil". Or is it the "new bacon"? It hardly matters. Data is everywhere, being accumulated everywhere, being exploited everywhere, and expanding in usefulness, in an ever widening collection of applications. The results are dramatic, from the mundane ad placement application to the socially envaluable use in fighting sex trafficing. We are awash in data. And in an ideal world, finding the right clever machine learning and/or mining techniques would be our main focus.

Life is not so simple. While clever techniques are surely important, many data scientists find themselves spending more time dealing with the "rat's snarl" of data that is dumped in their laps. Data is all too frequently awash in inconsistencies, spelling errors, ambiguities, noise, and any number of other contaminants. It is this "data quality" that is the focus of the current issue.

Editors Shazia Sadiq and Divesh Srivastava, who were recruited by Bulletin editor Xiaofang Zhou, have assembled a very comprehensive set of papers that span large portions of the "data quality landscape". This landscape is too large to be fully encompassed by any single issue. Nonetheless, this June issue does a great job of surveying both data quality in many of its variations, and technology that can clean it. This is a topic that will not be going away. This issue can help you prepare for the task and lead you to some promising approaches to dealing with quality issues. Thanks to Shazia and Divesh for their work in preparing the issue and to the very successful result of their efforts.

"No" on Proposed IEEE Constitutional Amendment

The Computer Society is part of the IEEE. As such it has a role to play in the governance of the IEEE, as do all societies within the IEEE. A proposed amendment to the IEEE constitution would change the way that the IEEE is governed. As of this moment, the Computer Society is joined with nine other societies who have had formal votes in opposition to the amendment. And no society has voted in favor of the amendment.

The thrust of the IEEE amendment is to move governance partially out of the constitution and make it subject to the more easily changed by-laws. Prior versions of the amendment have suggested that the purpose may be to reduce the role of societies in IEEE governance. It also removes IEEE members from some governance votes that are currently required. The proposal in detail can be seen at
`https://www.ieee.org/documents/constitution_approved_amendment_changes_election.pdf`
(login is required using your IEEE account).

As a member of the Computer Society Board of Governors (BOG), I have participated in BOG discussions on the proposed IEEE amendment, and agree with the position taken by the Computer Society in opposition. I believe the amendment is misguided. Professional meetings can be contentious, as there are sometimes conflicting interests that need to be resolved. But the change proposed seems to me to not improve the way we deal with these conflicts but rather seeks to bury them. That is not the way democracy is supposed to work.

Part of my "agenda" at the Computer Society BOG has been to empower constituent organizational parts to do more, e.g. by enabling Technical Committees to retain a fund balance from year to year. The IEEE amendment moves things in the opposite direction, toward concentrating power at the top of the organization. So I urge you to vote "no" on this amendment.

IEEE TCDE Award Winners

The Technical Committee on Data Engineering (TCDE) initiated awards for the data engineering community in 2014. This is the third year for the awards. The winners of the awards this year are listed on the TCDE web site
`http://tab.computer.org/tcde/tcdeawardsrecipients.html`

This year, award winners have the opportunity to publish a short communication about their thoughts as a result of the award in the Bulletin. The current issue contains short communications from Early Career winner Arnab Nandi and Contributions Award winner Gio Wiederhold. Impact Award winner Michael Carey's communication is scheduled for the September issue.

Award winners have made truly distinguished contributions. I want to congratulate them on their awards, and I would encourage you to see what they have to say in the current issue.

Changing Editors

Every two years, I am faced with the most important part of my job as Editor-in-Chief of the Bulletin. That is, I need to appoint new editors. One of the big reasons for the Bulletin's continuing success is that great computer scientists have been willing to serve as editors for its special issues.

The current set of editors, Chris Jermaine, Bettina Kemme, David Maier and Xiaofang Zhou have continued the high standards set by prior editors and have produced great issues, from declarative systems to data quality. I try very hard to appoint terrific editors who span a wide crosssection of the engineering community. I count these most recent editors as a very successful continuation of this story, and want to thank them for the outstanding issues that they have produced.

So... you are asking, who are the new editors? I am happy to be able to announce that my success in attracting great editors continues. The new editors will, starting with the September issue, be Tim Kraska of Brown University, Tova Milo of Tel Aviv University, Chris Ré of Stanford University, and Haixun Wang of Facebook. I am absolutely delighted that Tim, Tov, Chris, and Haixun have agreed to serve as editors and I look forward to working with them over the next two years to continue to bring our readers terrific issues with papers from many of the premier database researchers and practitioners in a wide array of data engineering areas.

David Lomet
Microsoft Corporation

Letter from the 2016 IEEE TCDE Contributions Award Winner

Why Data Engineering revisited

First of all I can't take all the credit for the establishment of the initial Data Engineering conferences. Richard Shuey from GE and Mas Tsuchiya from TRW were equally effective, and we received organizational support fromChittoor V. Ramamoorthy. These computing experts cannot be here now.Ram died this March; at the time he was working on his Handbook for Software Engineering, leading us to a new term, Data Engineering.

I provided some international and commercial background within this diverse group. I had learned programming in 1958 in the Netherlands, working for NATO. During 1964-1965 I taught at a IIT Kanpur in India. In 1973 I led a too-early startup on remote access to remote business data over 300bps AT&T modems. To save on phone costs we sent an employee weekly between Los Angeles and New York with the updated date files on a 2400 tape.

Several IEEE members had encountered a dissemination problem. In the late seventies there was much industrial innovation in the database arena within companies as GE, who had built a network DBMS following CODASYL standards, and TRW, who had worked with IBM on the database for the Shuttle program, which became IMS. Such work was hard to disseminate directly, in part because these efforts were components of larger applications. Engineering issues became more broadly significant as people were starting to put new theoretical database concepts into practice, as grown from the seeds sown by Ted Codd's 1970 relational model paper.

Getting started was hard and not always clean. ORACLE, one of the earliest commercially viable relational products, was created by providing relational access to a hierarchical database written for the US Air Force, by automatically inserting dummy parent nodes when relational operations required it.

But such technical innovations related to database engineering had to be presented in broad general computer science venues, as AFIPS or IFIP (the American and international Federations of Information Processing Societies) conferences, in their broad application tracks. For instance, during 1967-1968 Gio's ACME project developed a column-store database system for on-line use by medical researchers. The database was built and enabled within a time-sharing system which incorporated an incremental compiler. Such a compiler incorporates the advantages in terms of being able to make changes in the code during execution. There were no hypotheses, only feasibility, being proven, and being first, no comparisons could be made, hence it was not research. So its description was published in the AFIPS proceedings, and hidden as such from the later Internet. Later column stores were again considered novel.

Of current significance is that ACME also checked if references were within array bounds, at about a 10% overhead, well worth it in terms of code reliability.

Other technologies were equally hard to disseminate, as Smalltalk, an object-oriented compiling system for the Alto system. It was used by Allan Borning, an early student of mine, now at the University of Washington, then at XEROX PARC. I learned that it also provided secure execution. I believe that if those user-facing technological concepts had been more widely shared, wed now have safer code and fewer opportunities for hacking. Smalltalk was not made freely available because XEROX lawyers wanted to protect its commercial property value. Now valuing IP and modeling the surrounding Intellectual Capital is my research focus.

Subsequently, free-standing products, as C, and its successor, C++ were made liberally available to a general audience by Bells research arm. That openess enabled further broad development, outside of coherent system environments. Being developed for telecommunication experts, performance was more important than code reliability. Most hacking attacks today are enabled by intruders gaining access to cells by using indexes outside of array bounds.

Sang Cha, another student of mine, here with us at ICDE, was able to use the performance demands of the telecommunication industry to develop in-memory technology, now available as HANA from SAP.

Today, entire capable database systems, as MySQL, are distributed as freeware. They are sufficiently capable that startups, as Twitter, can depend on them for their initial growth. But soon, engineering issues, often associated with reliability, arise. With mechanical storage becoming a backup for semiconductor storage, and, with ubiquitous high-speed communication, moving much data into the cloud, the technological foundations keep changing.

It is great to see so many early colleagues, postdocs, students, and grand-students here in a corner of the world that was not known to the IT community on the seventies and eighties.

I expect that the Data Engineering conference will continue to play a role in this world that now covers all sciences and entrepreneurial fields.

A final anecdote relevant to this ICDE meeting being in the Baltic and my background.

I named an early computer on the ARPAnet, now the Internet, funded for electronic commerce-work, "Haring". Colleagues would be able to send email to Gio@Haring (that was before 3-field IP-addressing was introduced.)

The reason for the name is a Dutch saying: The fishing of haring is the basis for all commerce.

The Dutch needed salt to preserve the Herring they caught. The sun allowed the Portuguese to make salt, but they were not interested in the Dutch herrings. But they needed wood for ships, they had deforested much of their country in the preceding centuries. So the Dutch had to take their herring in the Baltic regions, trading it for wood to get the salt. Later Dutch went to Indies to get spices and really got rich. Marten Kersten here is well aware of that history.

Gio Wiederhold
Stanford University

Letter from the 2016 IEEE TCDE Early Career Award Winner

Chasing Interactivity: Query, *Feedback*, and Result

As we take to data-driven approaches across all disciplines, the ability for practitioners to effectively interact with and make use of the data has become ever more important. While decades of research have gone into making data infrastructure more performant, the focus has typically been on throughput and efficiency of large-scale pipelines, and not the experiences of end users. At the same time, end user data consumption has become pervasive: with over two billion smartphones and tablets sold in the past few years, the number of humans performing complex data tasks is now a significant fraction of the entire human population. We are now at a critical transition point of how the world consumes and interacts with data.

Enabling better *interaction* with the data often solves crucial challenges in search, analysis, and management of data. Consider the example of an exploratory query intent, such as *"I am looking for a particular senior employee's record in the company directory... I'll know it when I see it."* Even when there is a valid result, this query challenges some popular assumptions: that a user is capable of unambiguously expressing their needs, or that there exists a well-formed query to issue to the database. In such cases, providing interactive query interfaces that enable users to quickly sift through and play with the data (e.g., structured autocompletion [1]) can significantly expedite the querying process. In contrast to the traditional *Query→Result* paradigm, we are better served with a new *Query↻Feedback→Result* paradigm. The purpose of *Feedback* is to guide the user to the right query, and is provided in a tightly coupled loop during the query specification step.

While at first glance this seems like a user interface or application-level issue, it is important to note that such a paradigm poses questions to multiple layers of the database stack. At the query language layer, how do we design abstractions that allow iterative composition of queries? From an aggregation and ranking standpoint, how do we provide the right feedback so as to best guide the user? At the execution layer, how do we provide near-instant response times for user feedback? How does the architecture of the overall system change when latency becomes the constraint, as opposed to correctness? We have made attempts to answer some of these questions in the *GestureDB* project [3], which investigates the querying and exploration of databases using touch or motion capture-based gestures. For example, directly mapping gestures to queries would not provide any guidance to the user *during* the articulation of the gesture itself. Thus, we built a gesture recognizer to classify gestures to queries in real time, while using the data and the schema of the database to improve classification quality [4]. To provide perceptibly instantaneous feedback, we used cyclic table scans [2] to surface the first-k tuples of the most likely query result to the end user. While we have been able to flesh out an end-to-end vision for gesture-driven queries, these attempts merely scratch the surface of research in interactive database systems. Answers to these questions will require not only building upon a rich body of existing work in database systems such as iterative query refinement, approximate query processing, and online aggregation, but will additionally require us to draw from a wide variety of complementary disciplines such as visualization, data mining, human-computer interaction, and cognitive psychology.

References

[1] A Nandi, HV Jagadish. *Assisted Querying using Instant-response Interfaces.* SIGMOD 2007.

[2] R Ebenstein, N Kamat, A Nandi. *FluxQuery: An Execution Framework for Highly Interactive Query Workloads.* SIGMOD 2016.

[3] A Nandi. *Querying Without Keyboards.* CIDR 2013.

[4] A Nandi, L Jiang, M Mandel. *Gestural Query Specification.* VLDB 2014.

Arnab Nandi
The Ohio State University

Letter from the Special Issue Editors

The prevalence of large volumes and varieties of accessible data is profoundly changing the way business, government and individuals approach decision making. Organizational big data investment strategies regarding what data to collect, clean, integrate, and analyze are typically driven by some notion of perceived value. However, the value of the data is inescapably tied to the underlying quality of the data. Although for big data, value and quality may be correlated, they are conceptually different. For example, a complete and accurate list of the books read on April 1, 2016 by the special editors of this issue may not have much value to anyone else. Whereas even partially complete and somewhat noisy GPS data from public transport vehicles may have a high perceived value for transport engineers and urban planners. In spite of significant advances in storage and compute capabilities, the time to value in big data projects often remains unacceptable due to the quality of the underlying data. Poor data quality is being termed as the dark side of big data, inhibiting the effective use of data to discover trusted insights and foresights.

Finding the nexus of use and quality is a multifaceted problem encompassing organizational and computational challenges. These challenges are often specific to the type of data (e.g. structured/relational, text, spatial, time series, social/graph, multimedia, RDF/web), the dimension of data quality (e.g. completeness, consistency, timeliness), and the preparatory processes (e.g. data acquisition, profiling, curation, integration) that precede the actual use of the data. Designing a practical strategy for tackling quality issues in big data requires data scientists to bring together these multiple aspects of data type, quality dimension and process within the context of their application setting.

In this special issue we have endeavoured to present recent research of some of the leading experts in the field of data quality with the aim of informing the design of such practical strategies. Out of the eight papers, four are on relational/structured data while the remaining four are on time series data, spatio-temporal data, micro-blog data and web data. The papers have targeted a number of data quality dimensions through a range of innovative approaches as outlined below.

The first two papers tackle data quality dimensions of **meta-data compliance** and **schema quality**. Sebastian Kruse, Thorsten Papenbrock, Hazar Harmouch, and Felix Naumann present data anamnesis as a means of meta-data discovery with an aim to assess the quality and utility of the underlying relational datasets. In the second paper, Henning Kohler, Sebastian Link and Xiaofang Zhou present a method for discovering meaningful certain keys, in the presence of **incomplete** and **inconsistent** data with an aim to tackle **redundancy** and maintain the integrity constraints of the underlying relational data.

The next two papers discuss data cleaning in the context of associated data transformation and curation activities. These works are instrumental in evaluating the effectiveness of data cleaning algorithms. A number of data quality dimensions are covered by these papers including **value**, **format and semantic consistency**, and **business rule compliance**. The paper by Ihab Ilyas proposes a decoupling between detecting data errors and the repairing of these errors within a continuous data cleaning life-cycle with humans in the loop. The paper by Patricia C. Arocena, Boris Glavic, Giansalvatore Mecca, Renee J. Miller, Paolo Papotti and Donatello Santoro, outlines the challenges and solutions for benchmarking data curation systems.

Spatio-temporal and time-series data are known to suffer from a variety of data quality problems, due to the correlated nature of the data. The paper by Juliana Freire, Aline Bessa, Fernando Chirigati, Huy Vo and Kai Zhao uses exploratory techniques and powerful visualizations to differentiate between error and feature in spatio-temporal data. Tamraparni Dasu, Rong Duan, and Divesh Srivastava in their paper on data quality for temporal streams use statistical distortion as a means of measuring data quality in a near-real time fashion. The papers address a range of data quality dimensions including **incompleteness**, **redundancy**, **inaccuracy** (e.g. GPS noise), **format consistency**, **dependency constraint violation**, **uniqueness** issues and handling **duplicates**.

The final two papers target the elusive data quality dimensions of **trustworthiness** and **understandability**. The paper by Wen Hua, Kai Zheng and Xiaofang Zhou focuses on improving the understandability of short-text as found in microblogs towards resolving entity ambiguity. The paper by Xin Luna Dong, Evgeniy Gabrilovich,

Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang presents a knowledge-based approach to estimating the trustworthiness of web sources.

Given the contextual nature of data quality research and the need to have reproducible results, we have compiled a list of the datasets used by the papers in the special issue. The list and links are available below.

We would like to thank all the authors for their insightful contributions and for also playing the dual role of reviewers of other papers, resulting in a synergized special issue. We hope you enjoy reading the papers as much as we enjoyed putting them together.

Link to datasets used in the papers

Data Anamnesis: Admitting Raw Data into an Organization
Datasets: MusicBrainz
Public Links: `https://musicbrainz.org`
Discovering Meaningful Certain Keys from Incomplete and Inconsistent Relations
Datasets: GO-termdb (Gene Ontology)
Public Links: `www.geneontology.org`
Datasets: IPI (International Protein Index)
Public Links: `www.ebi.ac.uk/IPI`
Datasets: LMRP (Local Medical Review Policy)
Public Links: `www.cms.gov/medicare-coverage-database`
Datasets: Naumann (benchmarks for FD mining)
Public Links: `https://hpi.de/naumann/projects/repeatability/data-profiling/fd.html`
Datasets: PFAM (protein families)
Public Links: `pfam.sanger.ac.uk`
Datasets: RFAM (RNA families)
Public Links: `rfam.sanger.ac.uk`
Datasets: UCI (Machine Learning Repository)
Public Links: `https://archive.ics.uci.edu/ml/datasets.html`
Benchmarking Data Curation Systems
Datasets: Real and synthetic dirty datasets from BART project
Public Links: `http://www.db.unibas.it/projects/bart`
Datasets: Real and synthetic integration scenarios from iBench project
Public Links: `http://dblab.cs.toronto.edu/project/iBench`
Exploring What not to Clean in Urban Data: A Study Using New York City Taxi Trips
Datasets: TaxiVis
Public Links: `https://github.com/ViDA-NYU/TaxiVis`
Datasets: TLC Trip Record Data
Public Links: `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`
Quality-Aware Entity-Level Semantic Representations for Short Texts
Datasets: Wikipedia
Public Links: `https://dumps.wikimedia.org/enwiki`
Datasets: Probase
Public Links: `http://probase.msra.cn/dataset.aspx`

Shazia Sadiq, Divesh Srivastava
The University of Queensland (Sadiq), AT&T Labs-Research (Srivastava)

Data Anamnesis: Admitting Raw Data into an Organization

Sebastian Kruse, Thorsten Papenbrock, Hazar Harmouch, and Felix Naumann
Hasso Plattner Institute, 14482 Potsdam, Germany
firstname.lastname@hpi.de

Abstract

Today's internet offers a plethora of openly available datasets, bearing great potential for novel applications and research. Likewise, rich datasets slumber within organizations. However, all too often those datasets are available only as raw dumps and lack proper documentation or even a schema. Data anamnesis is the first step of any effort to work with such datasets: It determines fundamental properties regarding the datasets' content, structure, and quality to assess their utility and to put them to use appropriately. Detecting such properties is a key concern of the research area of data profiling, which has developed several viable instruments, such as data type recognition and foreign key discovery.

In this article, we perform an anamnesis of the MusicBrainz dataset, an openly available and complex discographic database. In particular, we employ data profiling methods to create data summaries and then further analyze those summaries to reverse-engineer the database schema, to understand the data semantics, and to point out tangible schema quality issues. We propose two bottom-up schema quality dimensions, namely conciseness and normality, that measure the fit of the schema with its data, in contrast to a top-down approach that compares a schema with its application requirements.

1 Engaging in Raw Data

Data are a key driver in nowadays businesses as they are involved in most operational and strategic processes, such as decision-making and market predictions. For various companies, such as news agencies, booking platforms, and map providers, the data themselves are the main requisite to offer their services. Gartner also argues that companies should value their data as an asset [17]. Consequently, more and more data are being collected in today's world. Many datasets are published on the internet, with the Linked Open Data Cloud and open government data being well-known representatives. Companies have also started to gather their raw datasets in so-called *data lakes*. However, the mere availability of datasets is not sufficient to employ them for any project or application. Before any use, it is crucial to *understand* what content datasets actually contain, how to query them, and what implicit properties they have. Without these metadata, the actual data are basically worthless. Unfortunately, proper documentation of datasets is scarce. In fact, many raw datasets do not even provide a definition of their schema. Moreover, even when documentation is available, it might be imprecise, incomplete, or outdated. We claim that practically all datasets obtained by a company, data scientist, or anyone else, must undergo an adoption process – *data anamnesis* – before being put to use.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

In medicine, the concept of anamnesis, which is Greek and literally means reminiscence, has been practiced for several centuries. When treating a patient for the first time, the doctor systematically asks that patient directly for the medical history to obtain a comprehensive view and determine any information relevant to the patient's treatment, such as current conditions, blood pressure, etc. A similar proceeding can be applied for data management. Here, a user ($\approx$ the doctor) faces an unknown dataset ($\approx$ the patient) that he or she wants to employ, e.g., for data integration or data mining. However, assume that the data do not come with a schema or other documentation or that one simply cannot trust the given metadata. The dataset, therefore, appears as an unusable black box. To cope with this situation, the user needs to determine relevant dataset properties and establish an understanding ($\approx$ the anamnesis) directly from the data.

Definition 1 (Data anamnesis): Data anamnesis describes the process and result of discovering and assessing the structure of a raw (semi-)structured dataset. This discovery process is data-driven, i.e., all insights of the data anamnesis should come directly from the dataset so as to ensure that they accurately describe the data.

In medicine, there are standardized proceedings and questionnaires to obtain a patient's anamnesis. Analogously, frameworks for database reverse engineering have been conceived, such as [11]. Such frameworks rather focus on expert interviews and associated application code analysis, which might yield imprecise, incomplete, or incorrect insights for several reasons. For instance, an expert's knowledge about data might be outdated, and application code might not cover all relevant aspects of a dataset. Furthermore, the data themselves might be erroneous or incomplete and therefore do not adhere to alleged characteristics. These examples demonstrate the necessity to inspect the *actual* dataset at hand, instead of (or in addition to) deliberating on a *supposed* dataset.

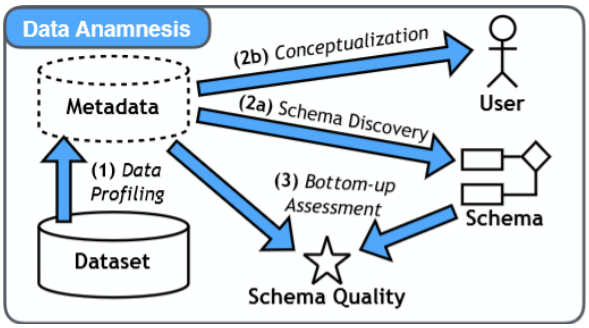


Figure 1: Data anamnesis enables the usage of raw datasets by means of data profiling and analyzing the acquired metadata.

To this end, we propose a data anamnesis process as shown in Figure 1. At first, the dataset in question should be analyzed for relevant metadata (Step (1)). Metadata discovery is a computationally complex task that has been tackled in the area of *data profiling* for at least two decades and still has many open questions [1]. The produced metadata can then be used to derive a technical schema definition for the data (Step (2a)) and should be explored to obtain an understanding of this schema (Step (2b)). Both are prerequisite to work with the data. Data anamnesis is also important in cases where documentation or a schema definition is given to diagnose *schema quality issues* (Step (3)): Do schema and data match? Are the data types and formats suitable? Are the constraints sound and complete? When Coelho et al. analyzed 512 open-source database schemata, they found these questions insufficiently answered by the provided metadata and attested overall poor schema quality [8]. So apart from data quality, which is out of the scope of this paper, we observe schema quality to be an important building block to assess the fitness for use of a given dataset. Given only the data and a schema, specified or discovered, data anamnesis judges schema quality from the viewpoint of the data. As the counterpart of data quality, which describes to what extent data fulfill (schema) constraints, this data-driven schema quality is the degree to which a schema fits its data. We refer to this approach as *bottom-up* schema quality assessment contrary to the traditional top-down assessment that judges schemata from the viewpoint of users, applications, and requirements. In particular, we propose two schema quality dimensions in this paper, namely conciseness and normality, and use metadata to detect suspicious parts of schemata that violate these.

To make the notion of data anamnesis tangible, we carry out an anamnesis for the openly available dataset MusicBrainz [18]. This user-maintained music encyclopedia collects a wide range of information from songs over relationships of album releases to acoustic fingerprints. Moreover, it is not only used in practice by multiple applications but also is the subject of various research projects. In addition, MusicBrainz' complexity is

challenging: Its published export[1] consists of 324 headerless CSV files with a data volume of 35.1 GB, but no schema definition. Fortunately, in this case, the dataset's schema can be extracted from the MusicBrainz Server application source code[2], which allows us to evaluate the quality of our data anamnesis. Besides data types, the schema specifies various constraints, e.g., string formats, number ranges, keys, and foreign keys. Simultaneously, we shall diagnose the quality of this schema – i.e., determine how well it fits its data.

The remainder of this paper is organized as follows: In Section 2, we present the research area of *data profiling*, which addresses the problem of extracting metadata from given datasets, and apply state-of-the-art data profiling algorithms to the MusicBrainz dataset. In Section 3, we employ the discovered metadata to derive the dataset's schema. This step effectively turns the raw data into a queryable dataset. Section 4 then assesses the two aforementioned schema quality aspects, conciseness, and normality, for the dataset. Finally in Section 5, we use the gained insights to discuss further challenges in the area of data profiling to improve data anamnesis.

2 Data Profiling for Data Anamnesis

Descriptive metadata, such as integrity constraints, reveal inherent properties of a dataset and are useful for the preparation of various data-oriented tasks, e.g., query processing, data mining, and data integration [10]. As proposed in the previous section, metadata furthermore help to reconstruct database schemata and diagnose schema quality issues and are, thus, the key enabler for data anamnesis. Data profiling is the act of determining such descriptive metadata for a given dataset, including *single-column statistics*, such as data types and value distributions of columns, and *multi-column dependencies*, such as inclusion dependencies and functional dependencies. While the former can be obtained with most commercial data profiling tools, the latter are a focus in research [1]. As such, data profiling is the first step of data anamnesis. To this end, we describe two data profiling platforms that bundle important state-of-the-art profiling techniques and their produced metadata. Then, we apply these to the MusicBrainz dataset and report on the results, which form the input for the next data anamnesis steps.

2.1 Metanome and the Metadata Management System

For the analysis of the MusicBrainz dataset, we utilized two data profiling frameworks: Our *Metanome* tool [19] to compute the metadata and our *Metadata Management System* to harvest interesting insights from the results. Both frameworks are available online[3].

The Metanome tool is a hub between datasets and profiling algorithms. Research in the area of data profiling has devised many different metadata discovery algorithms that automatically detect *all* statistics or dependencies of a certain type in a dataset. This is a notoriously difficult task, even for machines. Indeed, most commercial profiling tools lack real *discovery* features and allow only the examination of a few user-chosen dependency candidates. In contrast, Metanome bundles state-of-the-art metadata discovery algorithms with a unified interface and provides them to data scientists. In our data anamnesis process, we feed the MusicBrainz dataset into Metanome to extract several single-column statistics (e.g., number of distinct values) and several multi-column dependencies, namely *unique column combinations (UCCs)* [13] (combinations of columns that do not contain duplicate values), *inclusion dependencies (INDs)* [15] (dependencies stating that all values in one column combination are also contained in another column combination), and *functional dependencies (FDs)* [21] (dependencies stating that the values in a particular column combination determine the values in a particular another column). These four types of metadata can be used to determine data types (statistics), key constraints (UCCs), foreign-key constraints (INDs), and normal forms (FDs), which are the most important schema properties for relational databases and, hence, the most relevant metadata types for data anamnesis. Note that many other types

[1] `http://ftp.musicbrainz.org/pub/musicbrainz/data/fullexport/`

[2] `https://github.com/metabrainz/musicbrainz-server/tree/master/admin/sql`

[3] `http://www.metanome.de/` and `https://github.com/stratosphere/metadata-ms`

of metadata exist, such as order dependencies to reason about table sortation criteria and matching dependencies to spot data inconsistencies, but for this article they are of lower relevance.

Because data are not always correct, each type of metadata has several relaxation degrees, e.g., *approximate* metadata (valid with a certain confidence only), *conditional* metadata (only valid for records fulfilling a certain condition), or *partial* metadata (only valid for some percentage of records). However, data quality issues are not the focus of this article: We do not intend to clean, but to describe the data. Therefore, we focus on *exact* metadata. Nonetheless, one might also perform a data anamnesis with relaxed metadata. Another interesting extension of above mentioned metadata types addresses null values. In contrast to regular domain values, null values can be interpreted in various ways, such as *not applicable*, *no value*, or *unknown value*. Specialized metadata types accommodate such semantics. Amongst others, certain keys [14] and strong FDs [16], respectively, describe UCCs/FDs that are always obtained from a relation by replacing nulls with domain values, no matter which values are chosen. In this article, we take a rather pragmatic approach: In our applications of UCCs and FDs, the SQL standard prohibits null values, so we just treat null as another domain value; and for INDs, we ignore tuples with null values in accordance with the default semantics of SQL foreign keys.

Efficiently extracting statistics and dependencies from a dataset is a crucial, but only the first step towards a data anamnesis. So far, these metadata are a loose set of facts about the original dataset and still far from the desired reconstructed schema or an assessment of the same. For instance, a discovered dependency might reflect an actual database constraint or merely occur by chance. To obtain practical insights on the original dataset, it is necessary to interlink the different types of metadata into a cohesive "metadata knowledge base" and then thoroughly comb through it. This step is challenging, because metadata are highly heterogeneous and often huge. The Metadata Management System (MDMS for short) approaches this challenge by mapping the diverse types of metadata onto a unified vocabulary and saving them in a scalable storage. This storage layer is further complemented with an analytical layer on top that offers ad-hoc queries and visualizations on the metadata for interactive explorations. We store the metadata produced by Metanome in the MDMS and, from there, we manually explore the metadata but also devise specialized metadata refinement operators (e.g., to designate the primary keys among given UCCs), both for the purpose of schema reconstruction and diagnosing schema quality.

2.2 Raw data profiling results

The MusicBrainz dataset comprises 324 headerless CSV files, each of which constitutes a table, with a total volume of 35.1 GB. This size renders a manual inspection for statistics and dependencies virtually impossible, particularly facing the exponential complexity of dependency discovery. However, we were able to profile MusicBrainz with Metanome on a computer with 128 GB of RAM (statistics, FD, and UCC discovery) and on a commodity hardware cluster of 10 desktop machines, respectively (IND discovery). Table 1 summarizes for each analyzed metadata type, the number of elements that Metanome has discovered and the time it took to discover them.

Metadata type	Results	Discovery time
Column/table statistics	1,835	4.4 h
Minimal functional dependencies	4,193	0.5 h
Minimal unique column combinations	675	0.5 h
Maximal inclusion dependencies	93,502	2.0 h

Table 1: Basic profiling results for the MusicBrainz dataset.

At first, we remark that for 73 of the 324 tables ($\approx 23\,\%$) are completely empty, so that we could not produce any profiling results. In fact, the profiling lacks any evidence to state even how many columns these tables are

supposed to have. Apart from that, we observe that the different types of metadata yield vastly different numbers of results, with INDs being the most frequent metadata type – on average 288 INDs for each table tables. INDs are different from the other dependencies, because they reach across tables, while the other dependencies regard tables in isolation of one another. Nonetheless, there are also on average more than two UCCs (28 at most) and almost 17 FDs (295 at most) per table. These numbers demonstrate the amount of metadata that a data anamnesis typically entails and a purely manual evaluation of these metadata would require a substantial amount of time. In addition, for all types of metadata we observe a trend: With increasing numbers of columns in a table, there are more dependencies to be found. In particular FDs exhibit a superlinear, somewhat exponential growth. It is therefore notable that MusicBrainz has at most 20 columns in a single table, while datasets with tables of several hundred columns are not unusual [20].

3 Schema Discovery

Schema discovery constitutes the data-driven reconstruction of a dataset's schema. It is at the core of any data anamnesis. Even if a dataset is accompanied by a schema, it is still advisable to verify the correctness of that schema. And indeed, in our experiments, we found that our version of the MusicBrainz dataset slightly differs from the schema obtained from GitHub (see Section 1). We advocate that schema discovery should not directly inspect the bare dataset but operate on its previously discovered metadata, which not only summarize the actual data but also expose their latent properties, such as INDs, FDs, that are essential to data anamnesis but need substantial effort to gather from the data. However, as described in the previous section, the output of metadata discovery algorithms can be fairly extensive itself. In consequence, these raw metadata are not yet useful to be directly presented to users and leave them alone with the task of schema discovery. To this end, we identify two different schema discovery phases, similar to [11], and show for both of them, which techniques support the user in carrying them out. The first phase, the *constraint reconstruction*, recovers technical schema details. The second one, *conceptualization*, aims at helping the user to build a mental model of the schema.

3.1 Constraint reconstruction

The goal of the constraint reconstruction is to detect meaningful database constraints for a given dataset, such as primary keys and value ranges. In contrast to any profiled metadata, which is *descriptive*, database constraints are *prescriptive*. Consequently, the constraint reconstruction is a heuristic process: We can exactly tell which constraints a given dataset satisfies, but we cannot guarantee that these constraints are meaningful. For instance, for an attribute track_number, we might propose the constraint track_number ≤ 33, assuming 33 is the maximum value for that attribute. Intuitively, however, this is not a reasonable constraint.

In our case study, we focus on reconstructing four core constraint types, namely data types, nullability, primary keys (PKs), and foreign keys (FKs), using automated algorithms that exclusively rely on previously extracted metadata. Further constraint types, such as inferring `CHECK` clauses from denial constraints [6], are out of the scope of this paper. To evaluate the quality of the applied reconstruction algorithms, we derived a constraint gold standard from the existing schema definition from GitHub. We discovered, however, that this schema definition contains 54 tables that are not in our dataset export and also lacks one exported table. This is an unintentional example for incorrect metadata and endorses the idea of data-driven anamnesis. Nevertheless, we use the constraints of the 323 tables that are both in the dataset export and the existing schema definition as our gold standard. In addition, we use a second gold standard that further neglects the 73 empty tables (see Section 2.2) for two reasons: First, it is plausible that in many scenarios, the user has no interest in reconstructing the schema for empty tables. Second, any data-driven approach to reconstruct the schema for empty tables is inherently condemned to fail due to the lack of data. Thus, the empty tables bloat the error of our reconstruction algorithms.

Constraint	Recon-structed	with empty tables				without empty tables			
		Gold	Accuracy	Precision	Recall	Gold	Accuracy	Precision	Recall
Data types	1,314	1,880	0.68	–	–	1,511	0.84	–	–
`NOT NULL`	1,256	1,454	–	0.89	0.77	1,112	–	0.89	1.00
PKs	250	315	–	0.77	0.62	242	–	0.77	0.80
FKs	310	554	–	0.77	0.44	421	–	0.77	0.57

Table 2: Qualitative results of the constraint reconstruction with the two different gold standards. *Reconstructed* and *Gold* show the number of reconstructed constraints and gold standard constraints, respectively.

At first, we reconstruct the data type and nullability of each column. We propose the most specific SQL data type (e.g., `SMALLINT` is more specific than `INT`, which is more specific than `FLOAT`) that accommodates all values of a column. If a column does not contain null values, we additionally propose accordingly a `NOT NULL` constraint. Then, we determine for each table a primary key (PK), which has to be a unique column combination (UCC) that consists exclusively of `NOT NULL` columns. If a table comprises more than one such UCC, we designate a PK heuristically: (i) PKs are typically among the first columns of a table, (ii) their columns are typically contiguous, and (iii) PKs usually involve mostly short values.

Finally, we propose foreign keys (FKs) from the INDs in three steps: We first consider only those INDs whose right-hand side is a UCC, because FKs can only reference either PKs or `UNIQUE` columns. Then we apply four FK heuristics (inspired by [22, 25]) to quantify the eligibility of each IND as an FK: (i) the referencing and referenced columns of an FK should have similar value distributions and (ii) a similar number of distinct values; further, (iii) the names of the referencing and referenced table are usually similar, where (iv) referenced table names are supposedly rather short. Finally, in a greedy approach, we iteratively promote the most eligible INDs to FKs, thereby ensuring that each attribute appears as a referencing attribute in at most one FK. It is notable that neither of the previous works were directly applicable in our case study. Not all of the features proposed in [22] were available (e.g., column names) and moreover this approach requires training data. Also, relying only on value distributions as in [25] was not sufficient, as we explain later in this section. Instead, we found it to be crucial to leverage the peculiarities of the given data. In our case, all features pertain almost equally to the ranking of the FK candidates and are configured rather conservatively, e.g., an FK should cover at least 90 % of the referenced columns' distinct values (Criterion (i)).

All above reconstruction algorithms finished within seconds, because we use light-weight, greedy heuristics, but also because these algorithms process only metadata, not the data. Table 2 displays the quality of our reconstructed constraints. Our simple heuristics did not deliver perfect results, yet, they were able to reconstruct the majority of constraints correctly. The errors in the data type reconstruction arise, because 6 % of all columns contain only null values, but also due to unexpected exotic data types, such as `BYTEA`, and incorrectly predicted `(VAR)CHAR` lengths. The detection of `NOT NULL` constraints discovered all true `NOT NULL` constraints, but also reveals that roughly 10 % of all null-free columns would actually permit null values.

In the reconstruction of the PKs, we identify two major problems: Firstly, eight tables have no specified PK, violating Codd's entity integrity. Nevertheless, we propose a PK for each of them. Secondly, in 43 tables the PKs are superkeys, i.e., a subset of their attributes already uniquely identifies each tuple. For these 43 tables, we propose exactly these subsets as PKs. In general, smaller keys are preferable, because they need smaller indices, can be enforced more efficiently, and allow for more efficient join processing. In these cases, a domain expert should determine whether our proposed PKs are indeed semantically correct.

The reconstruction of FKs achieved the lowest recall, but also was the most tricky: Of the 93,502 INDs, 43,290 reference an alleged PK and are thus FK candidates; so on average, there is only one FK among 102 FK candidates. It naturally ensues that among the many non-FKs there are some that look like FKs. The reason is surrogate keys: In MusicBrainz, tuples in a table are identified by an auto-incremented integer. Consequently,

many tables have similar sets of PK values and, hence, FK values; and these form the pseudo-FKs among each other. Surrogate keys are popular for efficiency reasons and due to their robustness w.r.t. schema changes, so that the above described problem is likely faced in other datasets, too.

In general, our results indicate that automatic constraint reconstruction methods are a viable aid and our simple methods can most likely be further improved. Nevertheless, they should be employed in a semi-automatic reconstruction process, in which they support a human who decides upon the eventual set of constraints.

3.2 Conceptualization

While the constraint reconstruction reveals technical properties of a dataset, it does not yet tell users how to work with that dataset, e.g., *what* information they can query and *how*. Therefore, schema discovery should contain a complementary conceptualization phase, in which users familiarize themselves with the structure of the dataset. Indeed, various conceptualization approaches exist. One line of research conceived frameworks to derive logical and/or conceptual schemata from a database schema, usually as (extended) entity-relationship diagrams [5, 11]. Such higher-order schemata support human understanding by scrapping technical concerns from the database schema and providing comprehensible abstractions. However, for the same reasons, they are rather suited to understand the domain of a dataset while concealing potentially important details of the database schema. A second line of research aims at enriching database schemata using data mining methods [10, 23, 24]. These works particularly create informative summaries *directly on* the database schemata, e.g., clustering tables or detecting composite fields. Such summarization techniques are a very useful schema exploration tool. To exploit these techniques in an interactive fashion in practical scenarios, we propose to embed them in a metadata query language, complemented with rich visualizations. Note that this approach fits well with notebook applications, such as IPython[4], which are recently gaining in popularity in the data science community. In the following, we exemplify this idea with a conceptualization workflow for the MusicBrainz dataset.

Assuming that not all of the 324 tables in the dataset are equally important, we want to find the "points of interest", that is, the most relevant tables, to get an intuition of the dataset's contents. The simple heuristic that voluminous tables are important is a sufficient starting point in our case study, but note that more sophisticated relevance criteria exist [23]. We query against the MDMS: *Retrieve all tables, sorted descendingly by their number of columns multiplied by their number of tuples.* Among the top 20 results are the tables track, release, and artist. Intuitively, the concepts of these tables are easily grasped, so they form a good starting point to fan out the further exploration with follow-up queries, e.g., by looking for similar tables [10, 23].

As we proceed, we are also interested in grasping the extent of the MusicBrainz dataset. We query: *Group the tables by their number of rows and columns and count the number of tables in each group. Visualize the result in a bubble chart.* Figure 2 displays the result and reveals that the majority of tables have few columns and few or even no values. For the sake of visual clarity, we alter the query to group the tables only by their number of columns and choose a bar chart visualization. This yields Figure 3. It is in a way intuitive that the majority of tables have only very few columns. However, the number of tables with exactly 5, 9 and 16 columns are surprisingly high given their neighbor groups. Accordingly, we pose queries to drill down into these groups. The result reveals regularities: Tables with 5 columns are named ****_type*, tables with 9 columns are named *1_***_****, and tables with 16 columns are called ****_alias*. Apparently, these 104 tables implement only three different table templates, i.e., type-, link-, and alias-tables. This insight can accelerate the familiarization process, e.g., when names for the columns of these tables should be devised.

This simple example demonstrates the utility of integrated metadata in conjunction with data mining methods and interactive tool support when conceptualizing database schemata. Obtaining the above insights by manually browsing through the dataset would have been very tedious and less exact. Nevertheless, we believe that this approach still bears much potential to be unlocked by further research. At first, more data mining methods

[4] https://ipython.org/

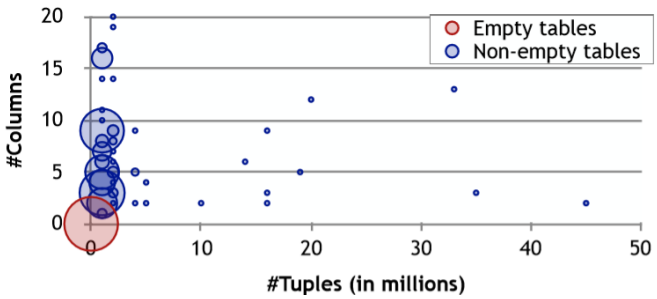


Figure 2: Number of tables (= circle area) grouped by their number of columns and rows.

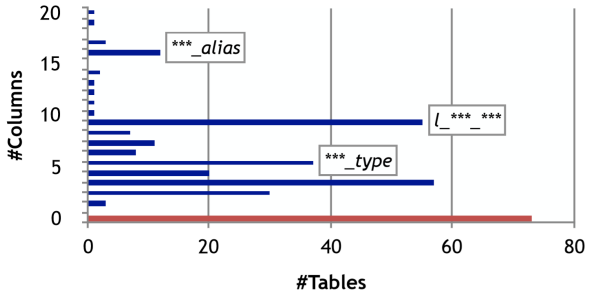


Figure 3: Number of tables grouped by their number of columns.

should be applied to metadata to answer novel questions, e.g., detecting schema design patterns using frequent itemset mining. Data mining and summarization could also be a viable tool to handle unwieldy data profiling results (cf. Table 1). Second, a query language should be devised to integrate the various data mining methods. A major challenge of such a query language is that it has to accommodate multiple different facets. For instance, database schemata are typically regarded as a set of tables, but they can also be interpreted as graphs, where the tables are the vertices and foreign key relationships form the edges. Finally, the visualization of the query results should be carefully addressed. Devising elaborate and interactive visualizations can drastically improve the understandability of those results. Also, picking and configuring the right visualizations automatically can enhance the schema exploration considerably.

4 Schema Quality Diagnosis

Regardless of whether a schema was available with the dataset or whether it (or parts of it) were discovered, it is a worthwhile endeavor to assess its quality. Not only do schemata model data and are in consequence an important factor to assure data quality aspects, such as completeness and consistency. Schemata also mediate between applications and data, thereby influencing the application, e.g., in terms of complexity. The description and assessment of schema quality is a well-known topic in the literature, to which we add a new perspective. Previous work has proposed a *top-down* assessment [4]: Given a specification, a list of requirements, an application, or an ER-model, determine the quality of a database schema along various dimensions, so as to predict whether a database with this schema can manage all incurring data of the intended use case. This proceeding is suitable when engineering new schemata and applications. However, in the context of data anamnesis, we face a reverse setting: Instead of applications or requirements, we are given only the data. In this scenario, the traditional, anticipatory schema quality assessment cannot be applied and a retrospective assessment is needed instead. To this end, we propose a *bottom-up* view of the schema quality problem: Given a database instance and a corresponding schema (specified or discovered), determine the quality of that database schema along various dimensions. Some of these dimensions are the same as for the top-down analysis, others are new. The two approaches can be interpreted as psychic (soul) and somatic (body) anamnesis and complement each other.

Next, we briefly outline existing work on top-down schema quality. Then, and in more detail, we discuss how to assess the quality of a schema using only metadata derived from an instance.

4.1 Top-down schema quality dimensions

We briefly summarize related work on assessing the quality of a schema. According to our classification, the following approaches take a top-down view, i.e, they are independent of any instance data.

Arguably, Heath, Codd, Bernstein and others were the first to address schema quality by introducing various normal forms, i.e., constraints to reduce redundancy while maintaining same semantics [12, 7, 3]. Burkett provides a comprehensive survey of frameworks to explicitly assess schema quality [4]. Among them, a typical work is by Batini et al. about the quality of a conceptual (ER-based) schema [2]. The authors define a set of schema quality dimensions, namely *completeness* (schema represents all relevant features of the application domain), *syntactic and semantic correctness* (proper use of concepts of ER-model), *minimality* (every aspect of requirements appears only once), *expressiveness* (schema represents requirements in a "natural way and can be easily understood), *readability* (graceful diagram), *self-explanation* (prefer model constructs over natural language explanations), *extensibility* (easily adaptable to changing requirements), and *normality* (adherence to normal forms). Clearly, most dimensions relate the schema to a concrete application and can only be assessed with respect to the application's requirements. The normality dimension is an exception, and we address it in our data-driven schema quality assessment.

Another typical opportunity to assess schema quality is for integrated schemata, because here the reference point is not a possibly vague set of requirements but the source schemata that contributed to the integration effort. For instance, Batista and Salgado examine minimality, consistency, and completeness of integrated schemata, and offer transformations to improve each of them [9].

4.2 Bottom-up schema quality assessment

While traditional top-down schema quality assessment judges schemata from the viewpoint of users, applications, and requirements, our proposed bottom-up assessment judges schemata from the viewpoint of the data. Instead of measuring the fit of a conceptual domain and a technical schema, this approach measures the fit of the schema and its data. In the following, we want to show that it is generally possible to assess schema quality based on the data and its metadata. For this purpose, we devise assessment methods for two schema quality dimensions, namely *conciseness* and *normality*, and exemplify them using the MusicBrainz dataset. The purpose of these methods is not to certify low or high schema quality, but to point out tangible schema quality issues.

4.2.1 Conciseness

We refer to conciseness as the schema counterpart of data completeness: If a schema is concise, it allows to exactly express the data. In contrast, a schema that is too general is also more complex, making it harder to understand that schema and increasing the complexity of queries and applications working on top of it.

An obvious indicator for a lack of conciseness are unused schema elements, i.e., empty tables and columns with all or mostly null values. As a counter-measure, such schema elements might be pruned. In the MusicBrainz dataset, 22.5 % of the tables do not contain any tuples. We regard the remaining tables in more detail and analyze the *fill level* of their columns, i.e., the percentage of non-null values. This yields a fill profile for each table, as exemplified in Figure 4, which reveals the inner structure of selected tables: Each series corresponds to a table and each data point in the series represents a column and its fill level. For visual clarity, the columns of each table are ordered by this percentage. We find that 5 % of all columns are completely empty and another 6 % have a fill level of less than 20 %. Apparently, the MusicBrainz schema models some properties that apply to only few or even no actual data instances. Those findings do not necessarily entail low quality, however, they show that the MusicBrainz schema is not as concise as it could be.

Another indicator for conciseness are relationships among tables. In the relational model, 1-to-1 relationships can be satisfied using a single table, 1-to-n relationships are realized by foreign keys, and m-to-n relationship need a link table. More general relationships require more complex modeling, so a concise schema should model the most specific possible type for each relationship. To check this, we proceed as follows: At first, we search for FKs "A references B", where B is a key candidate and $B \subseteq A$ is a valid IND. This reveals 1-to-1 relationships that are modeled as 1-to-n. Second, we look for link tables $R(A, B)$, where A and/or B are

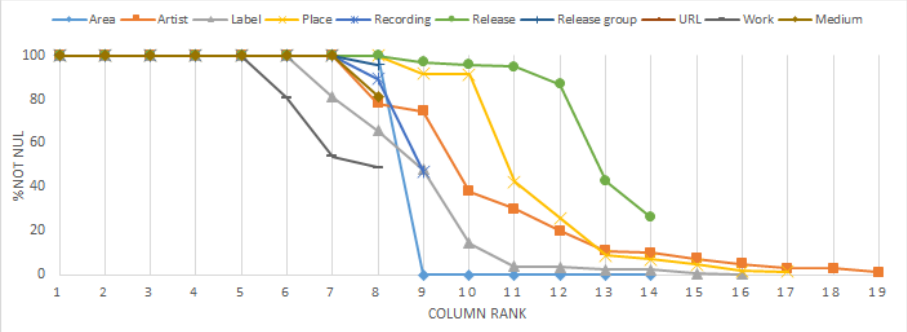


Figure 4: Visualization of the fill level profile for 10 core tables in MusicBrainz.

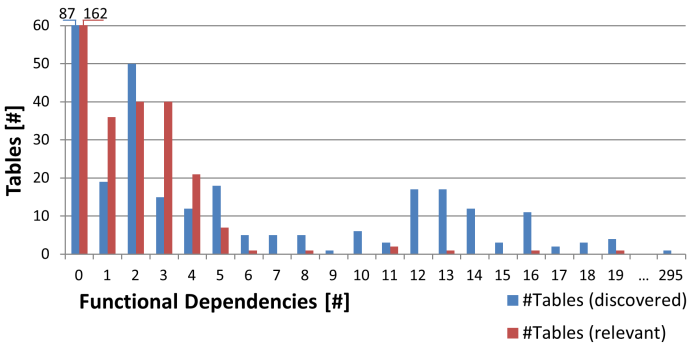
key candidates, thereby revealing modeled m-to-n relationships that actually accommodate only 1-to-n or even 1-to-1 relationships. Indeed, we find examples for both in MusicBrainz: The tables release and release_meta are linked via an FK but are de facto in a 1-to-1 relationship and the tables artist and annotation are interlinked by the table artist_annotation into an m-to-n relationship, although each annotation belongs to exactly one artist. It might well be that a deliberate design decision stands behind this mismatch, but it might also indicate a lack of schema conciseness.

The two above diagnoses are by far not exhaustive. For instance, one might check if the schema always uses the most specific datatype for its data. The choice of the datatype can influence the disk requirements and computation speed of databases (e.g., `SMALLINT` and `INT`). Furthermore, INDs could be used to detect duplicate data, allowing to further reduce the schema.

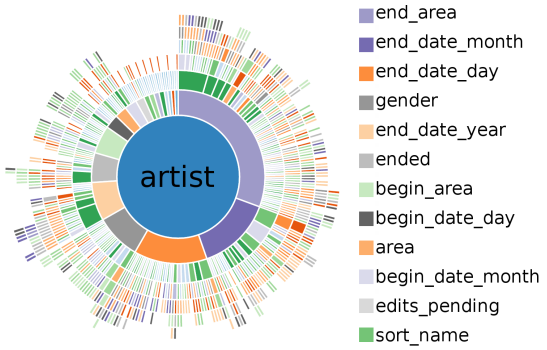
4.2.2 Normality

Ideally, a schema should not impose redundancy to its data, i.e., the schema should ensure that any fact in a database is stored only once. Redundancy typically entails various problems, such as increased data volume and the potential for inconsistencies. In the context of relational data, a major cause of redundancy are denormalized schemata. To diagnose denormalization, FDs are a most valuable asset. Basically, an FD reveals implicit 1-to-n relationships between columns in a single table. As discussed in Section 2.2, we discovered 4,193 FDs in the 250 non-empty tables of the MusicBrainz dataset. Compared to typical evaluation datasets for FD algorithms, where many tables with 20 attributes alone contain more than 8,000 FDs [20], this is a low number. Moreover, combining the FDs with the UCCs shows that 61 % of the FDs have a left-hand side that is a key or superkey. Such FDs are not relevant for the typical normalization goal of Boyce-Codd normal form [7]. However, there are still approximately ten relevant FDs per table on average, which is not negligible. Thus, we investigate the distribution of the discovered and relevant FDs over the MusicBrainz tables in Figure 5a.

Apparently, 162 tables (including the 73 empty tables) contain no relevant FDs at all. For the other 162 tables, it must be determined manually, which FDs reflect actual denormalizations and which hold only by incident. Here, we can exploit our observations that many tables are created from templates (see Section 3.2) and reduce the amount of manual work by grouping FDs that appear multiple times among tables with the same template. Then, there are a few tables that accommodate relatively many FDs. With a number of 245, the artist table is the one with the most relevant FDs. Here, visualization techniques can help to understand and classify those FDs, as shown for the artist table in a sunburst diagram in Figure 5b: The circle in the center of this diagram represents the table itself, the inner-most ring represents the dependent attributes, and all further rings represent the determining attributes, so that each sector forms an FD. With this visualization, we directly see that most FDs determine end_area, followed by end_date_month and end_date_day. This is, because these attributes are all very sparse, i.e., they contain 98%, 97%, and 96% null values. If they would be entirely empty



(a) Number of tables with certain numbers of FDs.



(b) FDs in artist table as sunburst diagram: First ring represents RHS attributes and outer circles LHS attributes.

Figure 5: Visualizations of the "denormalizing" FDs in MusicBrainz.

or more populated, fewer FDs would exist, because each attribute alone determines an empty attribute and it is much more difficult to determine an attribute with many distinct values. For this reason, most of these FDs are accidental and simply a consequence of sparsely populated attributes.

In our manual review, we could identify no indisputable violation of schema normality. Note, however, that types of redundancies other than denormalization exist. It is therefore interesting to inspect how other integrity constraints, particulary INDs and multi-valued dependencies, can be employed for that purpose as future work.

5 Conclusion & Outlook

Data quality is an important topic whenever data integration, data analytics, or big data in general are mentioned. However, before data quality can be measured, the data's schema must be captured and understood. Data anamnesis is the process of automatically determining the schema from a given dataset and then supporting the expert in understanding the schema and its quality, thereby exclusively resorting to the data themselves to avoid any error or bias from external sources. We substantiated and demonstrated this notion of data anamnesis with the help of the MusicBrainz dataset. At first, we used data profiling techniques to extract relevant metadata and then reconstructed a database schema from the metadata using classification algorithms and data exploration techniques. Finally, we used the metadata to assess the quality of that schema. For this purpose, we proposed a novel bottom-up approach that relates schema quality to a schema's actual data instance (and its metadata, respectively) and not to its fitness for an application. Thus, our approach complements existing work while putting to use the recent advances in data profiling. We illustrated this approach with a new and an existing schema quality dimension, thereby learning that the MusicBrainz schema is highly normalized but overly extensive.

Throughout the paper, we pointed out that much work remains to be done. In particular, we demonstrated the great potentials of data profiling results for database-reverse engineering and quality assessment. However, research is still in its infancy regarding the interpretation of these results: How can we make the leap from measured metadata to semantics? Most discovered functional dependencies, inclusion dependencies, data types, and other constraints are likely to be spurious. How can we separate the wheat from the chaff? Any existing data can be viewed as a (possibly skewed) sample of the data yet to come. Understanding data and its schema is a topic not likely to vanish from the research agenda.

Acknowledgements This research was partially funded by the German Research Society (DFG grant no. FOR 1306) and the German Academic Exchange Service (DAAD program no. 57169181).

References

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. Profiling relational data: A survey. *VLDB Journal*, 24(4):557–581, 2015.

[2] Carlo Batini, Stefano Ceri, and Shamkant B. Navathe. *Conceptual Database Design*, chapter 6. Benjamin/Cummings Publishing Company, Redwood City, 1992.

[3] Philip A. Bernstein. Synthesizing third normal form relations from functional dependencies. *ACM Transactions on Database Systems (TODS)*, 1(4):277–298, 1976.

[4] William C. Burkett. Database schema design quality principles, 1997.

[5] Roger HL. Chiang, Terence M. Barron, and Veda C. Storey. Reverse engineering of relational databases: Extraction of an EER model from a relational database. *Data and Knowledge Engineering (DKE)*, 12(2):107–142, 1994.

[6] Xu Chu, Ihab Ilyas, and Paolo Papotti. Discovering denial constraints. *Proceedings of the VLDB Endowment*, 6(13):1498–1509, 2013.

[7] Edgar F. Codd. Recent investigations into relational data base systems. Technical Report RJ1385, IBM, 1974.

[8] Fabien Coelho, Alexandre Aillos, Samuel Pilot, and Shamil Valeev. On the quality of relational database schemas in open-source software. *International Journal on Advances in Software*, 4(3 and 4):378–388, 2011.

[9] Maria da Conceição Moraes Batista and Ana Carolina Salgado. Information quality measurement in data integration schemas. In *Proceedings of the International Workshop on Quality in Databases (QDB)*, pages 61–72, 2007.

[10] Tamraparni Dasu, Theodore Johnson, Shanmugauelayut Muthukrishnan, and Vladislav Shkapenyuk. Mining database structure; or, how to build a data quality browser. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 240–251, 2002.

[11] Jean-Luc Hainaut, Jean Henrard, Vincent Englebert, Didier Roland, and Jean-Marc Hick. Database reverse engineering. In *Encyclopedia of Database Systems*, pages 723–728. 2009.

[12] I. J. Heath. Unacceptable file operations in a relational data base. In *Proceedings of the ACM SIGFIDET Workshop on Data Description, Access, and Control*, pages 19–33, 1971.

[13] Arvid Heise, Jorge-Arnulfo Quiané-Ruiz, Ziawasch Abedjan, Anja Jentzsch, and Felix Naumann. Scalable discovery of unique column combinations. *Proceedings of the VLDB Endowment*, 7(4):301–312, 2013.

[14] Henning Köhler, Sebastian Link, and Xiaofang Zhou. Possible and certain SQL keys. *Proceedings of the VLDB Endowment*, 8(11):1118–1129, 2015.

[15] Sebastian Kruse, Thorsten Papenbrock, and Felix Naumann. Scaling out the discovery of inclusion dependencies. In *Proceedings of the Conference Datenbanksysteme in Business, Technologie und Web (BTW)*, pages 445–454, 2015.

[16] Mark Levene and George Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theoretical Computer Science*, 206(12):283 – 300, 1998.

[17] Heather Levy. Why and how to value your information as an asset. `http://www.gartner.com/smarterwithgartner/why-and-how-to-value-your-information-as-an-asset/`, 2015. Accessed: 11-25-2015.

[18] MusicBrainz. `https://musicbrainz.org/`. Accessed: 11-25-2015.

[19] Thorsten Papenbrock, Tanja Bergmann, Moritz Finke, Jakob Zwiener, and Felix Naumann. Data profiling with Metanome. *Proceedings of the VLDB Endowment*, 8(12):1860–1863, 2015.

[20] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment*, 8(10):1082–1093, 2015.

[21] Thorsten Papenbrock and Felix Naumann. A hybrid approach to functional dependency discovery. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2016.

[22] Alexandra Rostin, Oliver Albrecht, Jana Bauckmann, Felix Naumann, and Ulf Leser. A machine learning approach to foreign key discovery. In *Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB)*, 2009.

[23] Xiaoyan Yang, Cecilia M. Procopiuc, and Divesh Srivastava. Summarizing relational databases. *Proceedings of the VLDB Endowment*, 2(1):634–645, 2009.

[24] Xiaoyan Yang, Cecilia M. Procopiuc, and Divesh Srivastava. Summary graphs for relational database schemas. *Proceedings of the VLDB Endowment*, 4(11):899–910, 2011.

[25] Meihui Zhang, Marios Hadjieleftheriou, Beng Chin Ooi, Cecilia M. Procopiuc, and Divesh Srivastava. On multi-column foreign key discovery. *Proceedings of the VLDB Endowment*, 3(1):805–814, 2010.

Discovering Meaningful Certain Keys from Incomplete and Inconsistent Relations

Henning Köhler
Massey University, Palmerston North, New Zealand
h.koehler@massey.ac.nz

Sebastian Link
The University of Auckland, New Zealand
s.link@auckland.ac.nz

Xiaofang Zhou
The University of Queensland, Brisbane, Australia
Soochow University, Suzhou, China
zfx@itee.uq.edu.au

Abstract

Completeness and consistency are two important dimensions for the quality of data, in particular relational data. This is true because most data sets found in practice are both incomplete and inconsistent. The simplest yet arguably most important integrity constraint are keys. Recently, certain keys were introduced for incomplete relations. Certain keys can efficiently manage the integrity of entities while still permitting incompleteness in columns of the key. It is therefore an important task to discover the set of certain keys that hold in a given incomplete relation. However, if the given incomplete relation is also inconsistent with respect to some meaningful certain keys, algorithms that discover keys cannot succeed. As meaningful keys are likely to have a small number of violations, we propose an algorithm that discovers certain keys that do not exceed a given number of violations. We illustrate the effectiveness and efficiency of our algorithm in discovering meaningful certain keys from publicly available data sets.

1 Introduction

Data-driven decision making is a competitive imperative today, determining, for example, how businesses acquire, retain, and sell to their customers. Poor data quality is a main inhibitor to data-driven decision making [22, 23]. In fact, the better the quality of the data, the better the insight and value will be that we can derive from the data. This intuitive causal relationship has resulted in a lot of academic research and data quality tools [5]. In this article, we will address the integrity and completeness dimensions of data sets conforming to the relational model of data with missing information. More specifically, we are interested in the discovery of meaningful keys from inconsistent and incomplete data sets. Here, a key is considered to be meaningful when it models semantic properties of the underlying application domain. Furthermore, inconsistency refers to the violation of meaningful keys, and incompleteness refers to the presence of null marker occurrences in the given data set.

A very basic data quality principle is that of entity integrity: To represent each entity of an application domain uniquely in a data set. Entity integrity is one of Codd's three inherent integrity rules [3], and has been

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

enforced in database systems by primary keys over the last 40 years or so. Codd's rule of entity integrity states that every table must have a primary key and that the columns which form the primary key must be unique and not null [3]. Primary keys therefore address the consistency and completeness dimensions for the quality of data in relations. The benefits of enforcing entity integrity by keys are manifold: We can i) uniquely reference entities across data repositories, ii) minimize data redundancy at schema design time to process updates efficiently at run time, iii) provide better selectivity estimates in cost-based query optimization, iv) provide a query optimizer with new access paths that can lead to substantial speedups in query processing, v) allow the database administrator (DBA) to improve the efficiency of data access via physical design techniques such as data partitioning or the creation of indexes and materialized views, and vi) provide new insights into application data [24]. Modern applications raise the importance of keys even further. They can facilitate the data integration process and prune schema matches. Keys can further help with the detection of duplicates and anomalies [20], provide guidance in repairing and cleaning data [25], and provide consistent answers to queries over dirty data [13]. The discovery of keys, and other classes of dependencies, from data is one of the core activities in data profiling [1, 21, 19].

Consider the snippet of the PFAM (protein families) data set in Table 1. The whole data set is available at `http://pfam.sanger.ac.uk/`. The snippet fails to comply with the entity integrity rule as every potential primary key over the schema is violated. In particular, column *taxonomy* carries the null marker $\perp$. Nevertheless, every tuple in I can be uniquely identified by combining the two columns species and taxonomy. This is not possible with SQL `UNIQUE` constraints which cannot uniquely identify tuples in which null markers occur in the columns involved. The inability to insert important data into the database may force organizations to abandon key validation altogether, exposing their future database to less data quality, inefficiencies in data processing, waste of resources, and poor data-driven decision making.

species	*taxonomy*
cellular organisms	$\perp$
environmental samples	Bacteria;Proteobacteria;Gammaproteobacteria;Xanthomonadales;Xanthomonadaceae;Xylella;
environmental samples	Bacteria;Cyanobacteria;Prochlorales;Prochlorococcaceae;Prochlorococcus;

Table 1: Snippet of the ncbi_taxonomy data set

These observations had motivated us to investigate keys over relations from a well-founded semantic point of view. For this purpose, we interpret occurrences of $\perp$ as *no information* [15]. We use a possible world semantics, in which a possible world results from a table by replacing independently each occurrence of $\perp$, by a value from the corresponding domain, and where missing information is represented by the distinguished domain 'value' N/A for *not applicable*. In each possible world, occurrences of N/A are handled just as any domain value. In particular, the equality relation between two domain values extends to N/A. That is, N/A equals the domain value v if and only if $v = N/A$. In conclusion, each possible world can be handled as a relation without $\perp$ values in which duplicate tuples may occur. As usual, a possible world w satisfies a key X if and only if there are no two tuples in w that have distinct tuple identities and matching values on all the attributes in X. For example, the possible world W_1 in Table 2 satisfies the key $\{$*taxonomy*$\}$, and the possible world W_2 in Table 3 satisfies the key $\{$*species,taxonomy*$\}$ but violates the keys $\{$*taxonomy*$\}$ and $\{$*species*$\}$.

species	*taxonomy*
cellular organisms	N/A
environmental samples	Bacteria;Proteobacteria;Gammaproteobacteria;Xanthomonadales;Xanthomonadaceae;Xylella;
environmental samples	Bacteria;Cyanobacteria;Prochlorales;Prochlorococcaceae;Prochlorococcus;

Table 2: Possible World W_1 of snippet from Table 1

species	*taxonomy*
cellular organisms	Bacteria;Cyanobacteria;Prochlorales;Prochlorococcaceae;Prochlorococcus;
environmental samples	Bacteria;Proteobacteria;Gammaproteobacteria;Xanthomonadales;Xanthomonadaceae;Xylella;
environmental samples	Bacteria;Cyanobacteria;Prochlorales;Prochlorococcaceae;Prochlorococcus;

Table 3: Possible World W_2 of snippet from Table 1

In [12] we defined that a *certain key* $c\langle X\rangle$ is satisfied by a table I if and only if the key X is satisfied in every possible world of I. In particular, the semantics of certain keys does not prevent the entry of incomplete tuples that can still be identified uniquely, independently of which information any of the null marker occurrences represent. For example, the snippet in Table 1 satisfies the certain key $c\langle species, taxonomy\rangle$, but violates the certain key $c\langle taxonomy\rangle$ as witnessed by world W_2 in Table 3.

The concept of certain keys was not available yet to the designers of the PFAM database. In an effort to meet Codd's rule of entity integrity, they introduced the surrogate primary key *ncbi_taxid*. We will briefly discuss the consequences of this decision. For this purpose, consider the extension I of the snippet from Table 1 which is shown in Table 4.

ncbi_taxid	*species*	*taxonomy*
39378	Catenula sp.	Eukaryota;Metazoa;Platyhelminthes;Turbellaria;Catenulida;Catenulidae;Catenula;
66404	Catenula sp.	Eukaryota;Metazoa;Platyhelminthes;Turbellaria;Catenulida;Catenulidae;Catenula;
131567	cellular organisms	$\perp$
329529	environmental samples	Bacteria;Proteobacteria;Gammaproteobacteria;Xanthomonadales;Xanthomonadaceae;Xylella;
379362	environmental samples	Bacteria;Cyanobacteria;Prochlorales;Prochlorococcaceae;Prochlorococcus;

Table 4: Snippet I of the ncbi_taxonomy data set

The first two rows of snippet I illustrate that some entities, a combination of a species and taxonomy, have been duplicated. The presence of the surrogate primary key does not help at all with preventing this. In particular, the relation violates the meaningful certain key $c\langle species, taxonomy\rangle$ and is therefore inconsistent. Unfortunately, such cases are not an exception in practice. In New Zealand, hospital patients are identified by their National Health Index (NHI) number. In a 12-month intensive duplicate-resolution programme in 2003, over 125,000 duplicate NHI numbers were identified and resolved[1]. This means there are some people who have more than one NHI number. The consequences can be severe, as the NHI numbers identify the health history of a patient. Being unaware of different NHI numbers for the same patient means that the health history is incomplete, which may lead to incorrect decisions.

In [12] we introduced a transversal-based method for the discovery of certain keys from incomplete relations. This method is agnostic to inconsistencies in the given incomplete relation, making it impossible to discover meaningful certain keys that are incorrectly violated. For example, an application of this method to the snippet I from Table 4 will only return the certain key $c\langle ncbi\_taxid\rangle$, giving the wrong impression that $c\langle species, taxonomy\rangle$ cannot uniquely identify entities.

Contribution and Organization. In this article we ask the challenging question how meaningful certain keys can be discovered from incomplete and inconsistent relations. After distinguishing our approach from previous work in Section 2, we briefly summarize our previous findings on certain keys and their discovery in Section 3. In Section 4, we will introduce certain near-keys of order n, stipulating that every possible world of a table must not have $n+1$ tuples with distinct tuple identities that all have matching values on all the attributes of the

[1] http://www.health.govt.nz/our-work/health-identity/national-health-index/nhi-information-health-consumers/national-health-index-questions-and-answers#duplicates

near-key. In particular, a certain near-key of order 1 is a certain key introduced in [12]. We will characterize this semantic definition syntactically, allowing us to decide satisfaction of a given near-key of order n just by looking at the given relation. The computational complexity of deciding satisfaction of certain near-keys is much more involved than deciding satisfaction of certain keys. While the latter can be done efficiently by checking weak agreement of tuple pairs, deciding satisfaction of near-keys is coNP-complete to decide, and even W[1]-hard in the order of near-keys. This can be interpreted as the overhead that inconsistency incurs on the problem of identifying meaningful certain key candidates. In Section 6, we will describe our algorithm for the discovery of certain near-keys of a given order, and illustrate the computation on our running example. As certain near-keys discovered can be accidental, we introduce an additional conflict-rate measure in Section 7 and discuss how it can be employed to quickly identify certain near-keys that are unlikely to constitute certain keys. In Section 8 we present some quantitative and qualitative results on applying our algorithm to a variety of publicly available data sets. The results suggest that our approach discovers good candidates for meaningful certain keys from inconsistent and incomplete tables. In practice, certain near-key mining does not seem to incur the large time penalties in comparison to certain key mining that may be expected due to the hardness of deciding certain near-key satisfaction. Finally, we conclude in Section 9.

2 Related Work

We briefly distinguish our approach from related work.

Certain keys were introduced recently in [12]. In this paper we introduce certain near-keys of order n, which subsume certain keys as the special case where $n = 1$. Apart from the discovery problem of certain keys, [12] addressed a variety of other problems which are beyond the scope of our contribution to certain near-keys. The discovery algorithm in [12] can only find certain keys that are satisfied by the given data set. This excludes any certain keys that are meaningful in the application domain but violated by the given data set.

The discovery of data dependencies from given data sets was introduced as the *dependency inference problem* in [17], and has received dedicated attention ever since. Recent surveys are given in [1, 21, 16]. In contrast to our work, most algorithms i) focus on complete data or treat null marker occurrences as any other domain value, ii) only return data dependencies that are satisfied by the given data set, and iii) do not attempt to distinguish between dependencies that are meaningful and those that are accidentally satisfied.

In the approximate dependency inference problem [8] not all the tuples of the given relation are considered. This is done in an effort to trade-in efficiency for the soundness of the results. In contrast, our algorithms report the exact degree of violation for the discovered certain keys, which is reported as the order of a near-key. Approximate functional dependencies [7] are those functional dependencies whose margin of violation in the given data set does not exceed a given threshold. The threshold is a global measure of approximation as it refers to a minimal number of tuples that need to be removed from the given data set to satisfy the functional dependency. In contrast, the order n of a certain near-key is a local measure of approximation as it refers to the maximum number of permissable duplicates (i.e. tuples that pairwise agree weakly with one another) for each combination of values on the attributes of the key.

The discovery of meaningful data dependencies from inconsistent tables has not received much attention. An exception is [27] in which good foreign key candidates are derived from inconsistent tables. In this work, null marker occurrences do not receive a special treatment. In contrast, the work in [18] studies the problem of foreign key discovery under different semantics of null marker occurrences recommended by SQL.

3 Certain Keys

We repeat the definition of certain keys from [12]. Beginning with basic terminology, let $\mathfrak{A} = \{A_1, A_2, \ldots\}$ be a (countably) infinite set of distinct symbols, called *attributes*. Attributes represent column names of tables. A

table schema is a finite non-empty subset T of $\mathfrak{A}$. Each attribute A of a table schema T is associated with an infinite domain *dom*(A) which represents the possible values that can occur in column A. In addition, we assume that each domain *dom*(A) contains the distinguished symbol 'N/A', which is short for *not applicable*. We include 'N/A' in each domain out of convenience, but stress that 'N/A' is not an actual domain value, but rather a marker that indicates that information does not exist, thereby representing Codd's null marker *inapplicable*. The marker 'N/A' will not feature in tables, but only in their possible worlds. This is in contrast to the *no information* null marker $\perp$ [15], representative of SQL's NULL value and also included in domains for syntactic convenience, which only features in tables but is replaced in possible worlds – either by 'N/A' or some other domain value.

Example 1: Consider a person's mobile phone number. This value may be missing, indicated by $\perp$, which can represent two different scenarios: (1) the person has a phone but its number in unknown, or (2) the person has no mobile phone, so no number exists. The latter case is represented by the value 'N/A'.

For attribute sets X and Y we may write XY for their set union $X \cup Y$. If $X = \{A_1, \ldots, A_m\}$, then we may write $A_1 \cdots A_m$ for X. A *tuple* over T is a function $t : T \to \bigcup_{A \in T} dom(A)$ with $t(A) \in dom(A)$ for all $A \in X$. For $X \subseteq T$ let $t[X]$ denote the restriction of the tuple t over T to X. Let t, t' be tuples over T. We define *weak similarity* of t, t' on $X \subseteq T$ as follows:

$$t[X] \sim_w t'[X] :\Leftrightarrow \forall A \in X.\, \big(t[A] = t'[A] \vee t[A] = \perp \vee t'[A] = \perp\big)$$

We will use the phrase t, t' *agree* interchangeably for t, t' *are similar*.

We say that $X \subseteq T$ is a *key* for the $\perp$-free table I over T, denoted by $I \vdash X$, if there are no two tuples $t, t' \in I$ that have distinct tuple identities and agree on X.

3.1 Definition

We will now define the notion of a certain key over general tables, using a possible world semantics. Given a table I on T, a *possible world* of I is obtained by independently replacing every occurrence of $\perp$ in I with a domain value different from $\perp$. We say that $X \subseteq T$ is a *certain key* for I, denoted by $c\langle X \rangle$, if the following holds:

$$I \vdash c\langle X \rangle :\Leftrightarrow X \text{ is a key for every possible world of } I\ .$$

We illustrate this semantics on our running example.

Example 2: The snippet I' in Table 1 violates the certain key $c\langle species \rangle$ because the value *environmental samples* is a duplicate in every possible world. The certain key $c\langle taxonomy \rangle$ does also not hold in I' as witnessed by the possible world W_2 in Table 3. However, the certain key $c\langle species, taxonomy \rangle$ does hold in I' because copies of the second and third tuples have different values on *taxonomy*, and the value *cellular organisms* is unique in *species*. The snippet I in Table 4 satisfies the certain key $c\langle ncbi\_taxid \rangle$, but violates the certain key $c\langle species, taxonomy \rangle$.

3.2 Syntactic Characterization

While the semantics of certain keys is well-founded in our possible worlds model, it is infeasible to explore infinitely many possible worlds in order to verify whether a given certain key holds on a given table. The following result from [12] characterizes the semantics of certain keys syntactically, using our notion of weak similarity. This means that we can directly use the given table to verify whether a given certain key holds, without having to look into any of its possible worlds. Theorem 1 provides a foundation for developing efficient algorithms that effectively exploit our keys in data processing.

Theorem 1: $X \subseteq T$ is a certain key for I iff no two tuples in I with distinct tuple identities are weakly similar on X.

Note that the validity of Theorem 1 does not rely on the availability of infinite domains, but also holds for finite domains of at least size two.

4 Certain Near-Keys and Their Satisfiability Problem

In incomplete tables the notion of *certain key* ensures entity integrity while permitting null values in key columns [12]. This helps us to deal with uncertainty with respect to unknown or missing values. We now introduce the notion of a certain near-key that handles inconsistent data in incomplete tables.

Definition 2 (certain near-key): Let T be a table schema and I an instance over T. A *certain near-key of order* n is an expression of the form $C_n\langle X\rangle$ where n is a positive integer and $X \subseteq T$ is a set of attributes. We say that $C_n\langle X\rangle$ holds on I iff for every possible world W of I there do not exist $n+1$ tuples $t_1, \ldots, t_{n+1} \in W$ with pairwise distinct tuple identities such that $t_1[X] = \ldots = t_{n+1}[X]$. We will also refer to X as a near-key of I, if $C_n\langle X\rangle$ holds on I, or as an *anti-key* otherwise.

According to this definition the semantics of a certain near-key forbids in all possible worlds the occurrence of too many tuples with distinct identities that have matching values on all of the key columns.

Example 3: We have seen before that the snippet I in Table 4 satisfies the certain key $c\langle ncbi\_taxid\rangle$, but violates the certain key $c\langle species, taxonomy\rangle$. In addition, I satisfies the certain near-key $C_1\langle ncbi\_taxid\rangle$ of order 1 and the certain near-key $C_2\langle species, taxonomy\rangle$.

As we experienced with certain keys already, the question arises how it can be decided whether a given certain near-key of order n is satisfied by a given table. We will see now that deciding satisfaction is a lot more complex for certain near-keys.

4.1 Satisfaction of Certain Near-keys

The first issue is how to deal with the "all possible worlds" requirement. Violation of a certain key can be tested by comparing tuples with respect to weak similiarity. This can be done efficiently (at least when the number of key-columns that permit null marker occurrences is small) by creating an index with respect to every null-columns combination [12].

For certain keys $c\langle X\rangle$ we could then iterate over each tuple t and check (using the index) whether t is weakly similar (on X) to any other tuple. For certain near-keys $C_n\langle X\rangle$ one may try to continue this check until n tuples weakly similar to t are found. If no such n tuples exist, the certain near-key $C_n\langle X\rangle$ is indeed satisfied. However, even if we find n tuples all weakly similar (on X) to our reference tuple t, there is no guarantee that a single possible world exists in which they are all identical (on X).

Example 4: Consider the following table:

$I =$

A	B	C
1	1	$\perp$
2	2	$\perp$
3	$\perp$	1
4	$\perp$	2

Here, each tuple is weakly similar on BC to two other tuples in I, but the certain near-key $C_2\langle BC\rangle$ still holds on I as no subset of 3 tuples displays pair-wise similarity.

Note that in the example the certain key holds on every possible world since the third and fourth tuple in I are *not* weakly similar (on BC), and thus the first tuple cannot be identical (on BC) to both of them in any possible world. Thus we require at least pairwise weak similarity (which is not transitive!) between tuples to ensure the existence of a possible world in which they are all identical. As it turns out, this necessary criterium is also sufficient.

Lemma 3: Given a set $\mathcal{S}$ of tuples, there exists a possible world in which all tuples in $\mathcal{S}$ have matching values on all the attributes in X iff the tuples in $\mathcal{S}$ are pairwise weakly similar on X.

Lemma 3 lets us decide whether a given certain near-key holds on a given instance.

4.2 Intractability of Deciding Satisfaction

Unfortunately, the problem of deciding satisfaction for certain near-keys is likely to be computationally hard.

Theorem 4: Given an instance I over table schema T and a certain near-key $C_n\langle X\rangle$ over T, deciding whether I satisfies $C_n\langle X\rangle$ is coNP-complete. □

The NP-hardness in Theorem 4 can be established by a reduction from the Maximum Clique problem, which is known to be NP-hard: Given an undirected graph G and integer n, does there exist a subset of n vertices which are pairwise adjacent (a clique) in G?

The close relationship to Maximum Clique becomes evident in Section 5.1. In fact, maximum clique is even W[1]-hard [4], and the reduction in our proof preserves the parameter n. Therefore, fixed parameter-tractability with respect to the order n also appears to be impossible for certain near-key satisfaction.

Theorem 5: Certain near-key satisfaction, parameterized by the order of the certain near-key, is W[1]-hard. □

However, it seems likely that for genuine (near-)key candidates weak similarity of tuples is relatively rare, and hence the corresponding graph for which we need to solve the maximum clique problem is sparse, thus allowing effective kernelization.

5 Identifying Certain Near-keys

In [12] a *row-based* approach for mining certain keys was presented. However, we find that it requires pairwise comparison of tuples to identify agree sets, which becomes expensive for tables with a large number of rows, but scales well for tables with a large number of columns.

An alternative approach is to consider subsets of attributes of a table, and to test for each of the corresponding certain keys whether it holds on the relation. This requires traversal of (part of) the lattice of attribute subsets in some fashion, and while it scales well in the number of rows, the number of attribute subsets grows exponentially with the number of table columns. We shall refer to such an approach as *column-based*. For a comparison of row-based and column-based approaches in the related context of functional dependency mining see e.g. [1, 19].

While both row-based and column-based approaches can work well for the discovery of certain keys, depending on the characteristics of the dataset, we find that discovery of certain *near*-keys strongly favours column-based approaches. Since a (certain) near-key of order n is only violated if $n + 1$ tuples (weakly) agree on it, a row-based approach requires examination of all subsets of rows of size $n + 1$, of which there are $O(|I|^{n+1})$ many. As this makes row-based approaches quickly infeasible even for fairly small data-sets, we will consider a column-based approach instead.

Here we are faced with the NP-hard problem of deciding whether a given certain near-key holds on I, but it turns out that this can be resolved fairly efficiently in practice. In the following, we will present an algorithm which solves the certain key satisfaction problem by finding cliques on the corresponding conflict graph, while exploiting the way it is constructed to achieve an effective decomposition into subgraphs. Afterwards we present a column-based approach which utilizes this near-key satisfaction test.

5.1 Conflict Graph

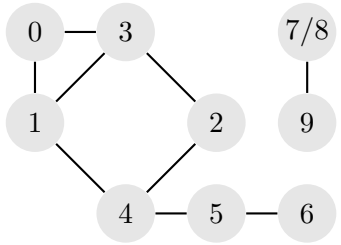
A *conflict graph* $\mathcal{G}$ corresponding to a certain near-key candidate $C_n\langle X\rangle$ consist of a vertex for every row in I and an edge between two vertices iff the corresponding rows are weakly similar on X. By Lemma 3 we have that $C_n\langle X\rangle$ is violated iff $\mathcal{G}$ contains a clique of size $n+1$.

When constructing the conflict graph, we only maintain tuples with conflicts, and combine tuples with identical key-values. Conflicts are identified by constructing indices similar to those used for certain keys [12]. We note that conflict graphs constructed in this fashion tend to be small when compared to the number of rows in I, and contain few edges. This holds true even for candidate sets X with a large number of conflicts, as clusters of weak-similarity tend to occur where multiple tuples are identical on X (rather than just weakly similar) and are thus merged into a single vertex.

Example 5: Consider the certain near-key $C_2\langle BCD\rangle$ on the table I below. Vertices in the corresponding conflict graph are labeled with the value of the matching tuple in column A.

A	B	C	D
0	1	1	1
1	1	1	$\perp$
2	1	2	$\perp$
3	1	$\perp$	1
4	$\perp$	$\perp$	2
5	2	1	$\perp$
6	2	1	1
7	3	1	1
8	3	1	1
9	3	$\perp$	1
10	4	1	1

$\Rightarrow$

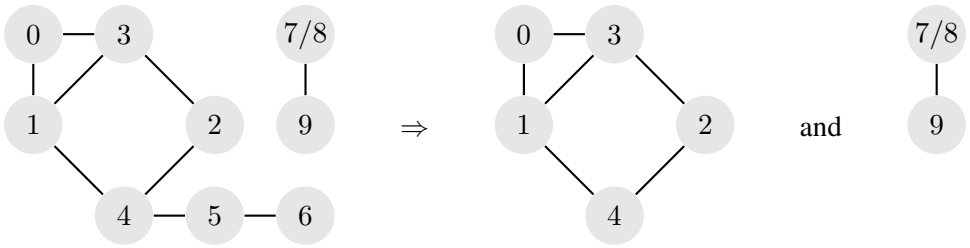


Note that tuple 10 is not weakly similar to any other tuple and thus omitted, while tuples 7 and 8 are merged.

5.2 Kernelization

Our kernelization phase is fairly standard for the maximum clique problem. We first keep removing vertices of degree $< n$, where n is the order of the near-key we want to check. Afterwards we partition the graph into connected components and process each component one-by-one. As conflicts between tuples are often isolated, this initial kernelization phase tends to produce many small components.

Example 6: Consider again the conflict-graph from Example 5. Initially the only vertex of degree < 2 is 6, as the vertices $7/8$ and 9 both have an "effective" degree of 2, as $7/8$ represents two tuples. After pruning 6 vertex 5 has degree 1, causing it to be pruned as well. Afterwards the remaining graph is partitioned into its connected components with vertices 0–4 and 7–9.



5.3 Decomposition

To check a connected component for cliques of size $n+1$, we pick one of the key columns (call it A). As edges are induced by weak similarity of tuples, no clique can contain tuples with distinct not-null values on A. This leads to two cases:

1. If the tuples associated with the vertices in the component are identical on A we simply drop the column.
2. Otherwise we partition the tuples with not-null values on A by their value on A, and add all remaining tuples (with $\perp$ on A) to each partition. We then drop columns A, and any vertices whose associated tuples have identical values on the remaining columns are merged (within each partition).

Note that in the second case, tuples within a component are weakly similar on the key iff they are weakly similar on the remaining columns, and any clique must fully lie within (at least) one component.

We continue this process until no columns remain. If at any point we encounter a vertex of multiplicity $> n$, we can abort[2] and report existence of a clique of size $n+1$.

Example 7: Consider the reduced components from Example 6. As the following steps are based on the values of associated tuples and edges are not considered directly, we shall revert to the underlying tuples, projected onto the near-key columns BCD. This gives us the following components to investigate:

B	C	D
1	1	1
1	1	$\perp$
1	2	$\perp$
1	$\perp$	1
$\perp$	$\perp$	2

and

B	C	D	
3	1	1	$\times 2$
3	$\perp$	1	

We will begin with the component on the left. Starting with column B we find the values are not identical, so we partition by not-null values of which there is only one, and end up with a single component and no merges:

B	C	D
1	1	1
1	1	$\perp$
1	2	$\perp$
1	$\perp$	1
$\perp$	$\perp$	2

$\Rightarrow$

C	D
1	1
1	$\perp$
2	$\perp$
$\perp$	1
$\perp$	2

Proceeding with column C we find distinct not-null values, and thus obtain two partitions, where we merge tuples $(1,1)$ and $(\perp,1)$ which become identical after projection onto D:

[2] In fact multiplicity n and degree ≥ 1 is sufficient, so we could have aborted after encountering vertex $7/8$.

C	D
1	1
1	⊥
2	⊥
⊥	1
⊥	2

$\Rightarrow$

D
1
⊥
2

$\times 2$ and

D
⊥
1
2

In a last step we partition each of the two components again, this time merging tuples $(1) \times 2$ and $(\perp)$ which gives us a 3-clique, so we can abort and report that the certain near-key $C_2\langle BCD\rangle$ does not hold.

While it is possible to apply a kernelization step after each column-projection, we found that this actually increased computation time.

6 Algorithm for Discovering Certain Near-keys Up to a Given Order

We shall now employ our certain near-key check in a column-based algorithm for finding all minimal near-keys up to a given order. In this context we say that a column set $X \subset T$ is a *certain anti-near-key of order* n iff $C_n\langle X\rangle$ does not hold on I. Note that anti-near-keys correspond to the agree sets in the row-based algorithm of [12], and that their complements are the transversals of all certain near-keys of the same order n.

6.1 The Algorithm

The principle idea is to compute all minimal near-keys of each order in sequence, starting with near-keys of order 1 (i.e. keys), and utilizing the near-keys of the previous order subsequently. Consider two sets:

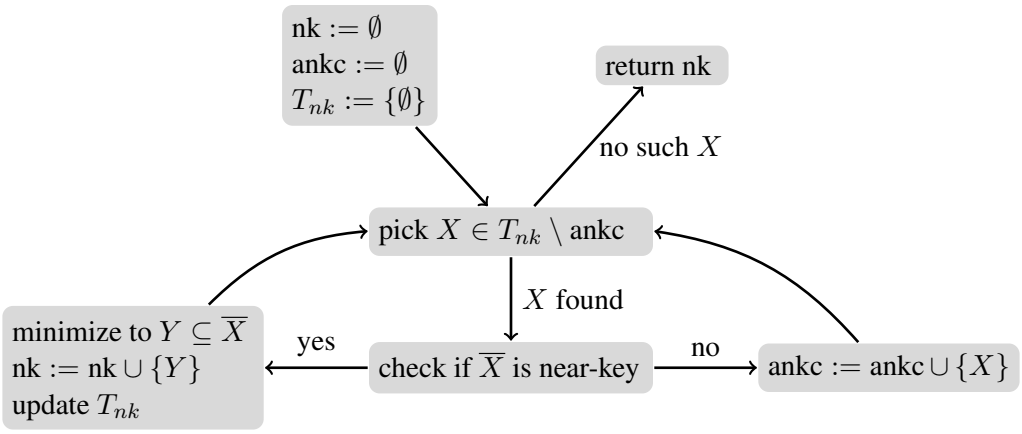
1. the set of all minimal near-keys (of a particular order), and
2. the set of all complements of maximal anti-near-keys (of the same order).

These two sets are transversal sets of each other, and we will compute them simultaneously. During computation, we maintain sets of all *currently known* minimal keys and complements of maximal anti-near-keys. Initially these sets are empty, and we extend at least one of them in each iteration until they become transversal sets of each other, at which point we must have found all of them.

To find a new minimal near-key or maximal anti-near-key, we track all minimal transversals T_{nk} of known near-keys, and pick one of these, say X, which is not the complement of a currently known anti-near-key. We then check whether its complement $\overline{X}$ is a near-key using the Algorithm sketched in Section 5, which can have two possible outcomes:

1. If $\overline{X}$ is not a near-key it must be a maximal anti-near-key which was not previously known.
2. If however $\overline{X}$ is a near-key then we minimize it to obtain a new minimal near-key. For this we attempt to remove each attribute from it in turn and check whether the resulting subset is still a near key.

The control flow of this algorithm is illustrated below.



Remarks. We can either search for near-keys and find anti-near-keys along the way, or vice versa. The final result is the same. The difference is that we minimize sets of the type we focus on, and track their transversals. Maximizing anti-near-keys may be slightly faster, as sometimes multiple attributes can be added with a single check using the counter-example $t_1, \ldots, t_{n+1}$ found, but the overall difference in speed turns out to be rather small. On the other hand, focusing on near-keys is not just more intuitive, but also allows us to 'carry over' results from the previous order, as discussed next. Hence we take the latter approach.

6.2 Utilizing near-keys of previous order

In the Algorithm described so far, near-keys of any particular order are computed independently of each other. The next step is to exploit the relationship between near-keys of different orders: A near-key of order n is also a near-key for any higher order.

This relationship can be utilized as follows: Instead of starting from scratch, we track all previous near-keys as known non-minimal near-keys. Note that using earlier orders other than the immediate predecessor would be redundant. In a first step we then minimize all known non-minimal near keys to convert them into minimal near-keys. Afterwards we compute the transversal set of known minimal near-keys and proceed as before.

6.3 An Example

We will now illustrate the computation for certain near-keys on the following real-world sample I from the `ncbi_taxonomy` data set of the pfam database:

ncbi_taxid	*species*	*taxonomy*
44482	Anopheles	Eukaryota;Metazoa;Arthropoda;Hexapoda;Insecta;Pterygota;Neoptera;Endopterygota;Diptera; Nematocera;Culicoidea;Culicidae;Anophelinae;Anopheles;
44484	Anopheles	Eukaryota;Metazoa;Arthropoda;Hexapoda;Insecta;Pterygota;Neoptera;Endopterygota;Diptera; Nematocera;Culicoidea;Culicidae;Anophelinae;Anopheles;
44534	Cellia	Eukaryota;Metazoa;Arthropoda;Hexapoda;Insecta;Pterygota;Neoptera;Endopterygota;Diptera; Nematocera;Culicoidea;Culicidae;Anophelinae;Anopheles;
59139	Cellia	Eukaryota;Metazoa;Arthropoda;Hexapoda;Insecta;Pterygota;Neoptera;Endopterygota;Diptera; Nematocera;Culicoidea;Culicidae;Anophelinae;Anopheles;
131567	cellular organisms	$\perp$
329529	environmental samples	Bacteria;Proteobacteria;Gammaproteobacteria;Xanthomonadales;Xanthomonadaceae;Xylella;
379362	environmental samples	Bacteria;Cyanobacteria;Prochlorales;Prochlorococcaceae;Prochlorococcus;
511133	environmental samples	Bacteria;Proteobacteria;Gammaproteobacteria;

We compute all certain near-keys that hold on I up to order 2. The three attribute names are abbreviated by the first letters, i.e., n, s, and t. At the start we do not know any certain near-keys or complements of anti-near-keys, i.e., $minKeys = \emptyset$ and $\overline{maxAnti} = \emptyset$. To find a minimal near-key or maximal anti-near-key, we

pick the maximum set $X = \{n, s, t\}$ and check, using the algorithm from the previous section, whether $C_2\langle X\rangle$ holds on I. For the minimization of this certain near-key of order 2 we determine that $C_2\langle s, t\rangle$ also holds on I, but further removal of attributes are not possible as neither $C_2\langle s\rangle$ nor $C_2\langle t\rangle$ hold on I. Consequently, we have $minKeys = \{\{s, t\}\}$, whose transversal set is $TR = \{\{s\}, \{t\}\}$. The complements of the elements in TR do not contain any known certain near-keys of order 2. A candidate for another certain near-key is therefore $X = \{n, t\}$, the complement of $\{s\} \in TR$. Indeed, our algorithm from the previous section confirms that $C_2\langle n, t\rangle$ holds on I. During minimization we can see that $C_2\langle t\rangle$ does not hold, but $C_2\langle n\rangle$ does hold on I. Consequently, we update $minKeys = \{\{s, t\}, \{n\}\}$, whose transversal set is $TR = \{\{n, s\}, \{n, t\}\}$. The complements of the elements in TR do not contain any known certain near-keys of order 2. A candidate for another certain near-key is therefore $X = \{t\}$, the complement of $\{n, s\} \in TR$. However, $C_2\langle t\rangle$ does not hold on I, which means that $\{t\}$ is a maximal anti-near-key and its complement $\{n, s\}$ in $\overline{maxAnti}$. The only remaining candidate for a certain near-key is $X = \{s\}$, the complement of $\{n, t\} \in TR$. However, $C_2\langle s\rangle$ does not hold on I, which means that $\{s\}$ is a maximal anti-near-key and its complement $\{n, t\}$ in $\overline{maxAnti}$. Hence, $\overline{maxAnti} = \{\{n, s\}, \{n, t\}\} = TR$, which means that $\overline{maxAnti}$ and $minKeys$ are transversal sets of each other. Consequently, the set of minimal certain near-keys of order 2 is

$$C_2\langle species, taxonomy\rangle, \text{ and } C_2\langle ncbi\_taxid\rangle \ .$$

Other near-keys that hold on this data set are $C_1\langle ncbi\_taxid\rangle$, $C_3\langle species\rangle$, and $C_4\langle taxonomy\rangle$.

7 Identifying Accidental Near-Keys

The notion of certain near-key was introduced to deal with dirty data issues which cause certain keys to be violated even though they *should* hold. However the opposite can happen as well, that is, certain keys should not hold in general but *happen* to hold for the instance examined. This occurs frequently for data sets with few rows, and likely more so for certain near-keys.

While the final decision on whether a mined certain near-key is sensible will typically rest with a domain expert, one may consider additional measures beyond the order of a certain near-key, to help decide whether it requires closer manual examination or can be discarded as accidental.

Two such measure quickly spring to mind: the number of conflicts (i.e., edges in the conflict graph) or the number of tuples participating in conflicts. While these tend to be closely related, the latter allows for a more sensible and intuitive normalization to the unit-interval $[0, 1]$. The point here is that the number of *potential* conflicts grows quadratically with the number of rows, while the number of actual conflicts hardly ever does, making a percentage-based measure behave very differently depending on the size of the table considered.

Example 8: Consider the following table *contacts*:

Name	Type	Phone
Ann Rosen	daytime	187629
Ann Rosen	mobile	762354
Bob Smith	daytime	655139
Bob Smith	mobile	334932

Here the certain near-key $C_2\langle$Name$\rangle$ holds, but the certain key $c\,\langle$Name$\rangle$ is not sensible. However, as the relation grows, the ratio of actual over potential conflicts converges towards 0. On the other hand, the percentage of tuples participating in conflicts remains at (or close to) 100% regardless of relation size.

This motivates the following.

Definition 6 (conflict rate): Let T be a table schema, I and instance over T and $C_n\langle X\rangle$ a certain near-key over T. The *conflict rate* of I w.r.t. $C_n\langle X\rangle$ is the percentage of rows in I that are weakly similar on X to at least one other row in I. We say that a *certain near key with conflict rate* r, written as $C_n^r\langle X\rangle$, holds on I if $C_n\langle X\rangle$ hold on I and the conflict rate of I w.r.t. $C_n\langle X\rangle$ is at most r.

Conflict rates can be used in at least two different ways:

1. After mining minimal certain near-keys, annotate them with their conflict rate. Optionally filter those that exceed a given threshold.

2. Reject attributes set $\overline{X}$ as near-key during the near-key test of the mining algorithm if the table's conflict rate w.r.t. $\overline{X}$ exceeds a given threshold.

Note that these options lead to different results, as in the latter case super-sets of $\overline{X}$ will be considered as minimal certain near-keys. This ensures that valid certain keys are less likely to be missed, but also increases the number of false positives and requires a threshold to be set a priori.

Finally we remark that searching for all certain near-keys that are minimal w.r.t. attribute set, order and conflict rate is not a promising approach. The problem here is that supersets of a certain near-key or order 2 or greater (but not including a certain key) are likely to meet this criterium of minimality, as any additional attribute is likely to reduce the conflict rate. The result is an explosion in the number of "minimal" near-keys.

8 Experiments with Real-World Data

We conducted experiments to get some insights about certain near-keys that occur in real-world data sets. For this purpose we mined a total of 145 tables from the following publicly available data sets:

- GO-termdb (Gene Ontology): `www.geneontology.org/`
- IPI (International Protein Index): `www.ebi.ac.uk/IPI`
- LMRP (Local Medical Review Policy): `www.cms.gov/medicare-coverage-database/`
- Naumann (benchmarks for FD mining): `https://hpi.de/naumann/projects/repeatability/data-profiling/fd.html`
- PFAM (protein families): `pfam.sanger.ac.uk/`
- RFAM (RNA families): `rfam.sanger.ac.uk/`
- UCI (Machine Learning Repository): `https://archive.ics.uci.edu/ml/datasets.html`

8.1 Quantitative Insights

As we are primarily interested in the discovery of meaningful certain keys, we restricted our attention to certain near-keys of order up to 5. In total, we found 642 minimal certain near-keys containing columns with null values, distributed across 22 tables. The following table shows how many certain near-keys of a particular order up to 5 we found and their distribution by table. In each column the number does not include certain near-keys that have already been found for a previous order. Numbers in brackets indicate how many of the near-keys found have a conflict rate of 5% or less. For order 1, conflict rates are always 0 by definition.

Data Source	Table	#rows	Order:	1	2	3	4	5
go_termdb	db	265		-	2(1)	-	2(0)	1(0)
ipi	protein_xref	156143		1(1)	-	-	-	-
lmrp	contractor	173		1(1)	-	-	-	-
lmrp	draft_contact_lookup	124		-	-	1(0)	-	7(0)
lmrp	lcd_related_documents	2311		-	1(1)	-	1(0)	1(0)
lmrp	lcd_x_advisory_committee	72		2(2)	1(0)	-	-	1(0)
lmrp	lcd_x_hcpc_code_group	2578		-	2(2)	-	-	-
lmrp	lcd_x_icd9_support_group	3429		1(1)	-	-	1(1)	1(1)
naumann	bridges	108		4(4)	30(0)	33(0)	16(0)	14(0)
naumann	echocardiogram	132		9(9)	36(1)	33(0)	32(0)	24(0)
pfam	clans	515		16(16)	8(5)	3(0)	1(0)	1(0)
pfam	dead_clans	38		-	1(1)	-	-	-
pfam	edits	8671		-	2(2)	-	-	-
pfam	literature_references	13784		-	-	1(0)	-	-
pfam	ncbi_taxonomy	822451		-	1(1)	-	-	-
pfam	pdb_pfamA_reg	246658		-	3(3)	2(2)	-	-
pfam	pdb_pfamB_reg	9624		-	6(6)	-	1(0)	-
rfam	features	21439852		2(2)	3(3)	-	-	1(1)
rfam	literature_references	1033		2(2)	1(0)	-	-	-
rfam	rfam	2208		20(20)	90(90)	50(37)	41(10)	45(8)
uci	adult	32561		-	-	37(37)	20(6)	13(1)
uci	auto_mpg	398		1(1)	4(3)	5(1)	3(0)	-
			Total:	59(59)	191(119)	165(77)	118(17)	109(11)

We find that a small number of tables tends to be responsible for the majority of near-keys. Such a skewed distribution suggests that many of the near-keys identified in tables with larger numbers of them are accidental. This is supported by the observation that, with the exception of table *adult*, such tables contain relatively few rows. However, the same skewed distribution occurs for both near-keys without columns containing nulls, and in particular ordinary keys, so a different behaviour for certain near-keys would have been surprising.

It is interesting to note though that there is no clear correlation between the number of minimal certain near-keys found in a table, and their conflict rates. We can however observer that (a) tables with few rows tend to have high conflict rates, and (b) conflict rates tend to increase with the order of certain near-keys. The distribution of conflict rates over all minimal certain keys with null columns, of orders 2-5, is shown on the left of Figure 1, where order 1 is omitted as conflict rates are necessarily 0. On the right, Figure 1 shows the relationship between conflict rates and "null-conflict rates", defined as the number of rows containing one or more null values in key columns over the total number of rows on the right (dot size indicates frequency).

We observe that for many (224 out of 583) of the minimal certain near-keys identified, the conflict rate lies at 5% or less, confirming their status as potential certain keys. However, it also highlights that many near-keys display a rather high conflict rate (218 have a conflict rate of 20% or higher), making it unlikely that they represent actual certain keys. A fair number of these (45) even have conflict rates of 100%, indicating cases similar to Example 8. For these cases, conflict rates can serve as an effective filter when searching for certain keys that may have been obscured by dirty data.

On the other hand, the relationship between conflict rates and null-conflict rates remains inconclusive. While null-conflict rates are by definition limited by conflict rates, they range anywhere from 0% to the conflict rate, with no discernable pattern. So far it appears that the role of null values in violating certain keys is highly specific to the data sets considered, and no general claims can be made.

In regards to the efficiency of our algorithms, the mining of all certain keys from all 145 tables took a total of 270 seconds, while the mining of all certain near-keys of up to order 5 from all 145 tables took a total of 1961 seconds. Hence, our algorithm required less than 7.3 times as much time to mine certain near-keys of up to order 5 in comparison to just certain keys. Note that these numbers report the averages over ten runs.

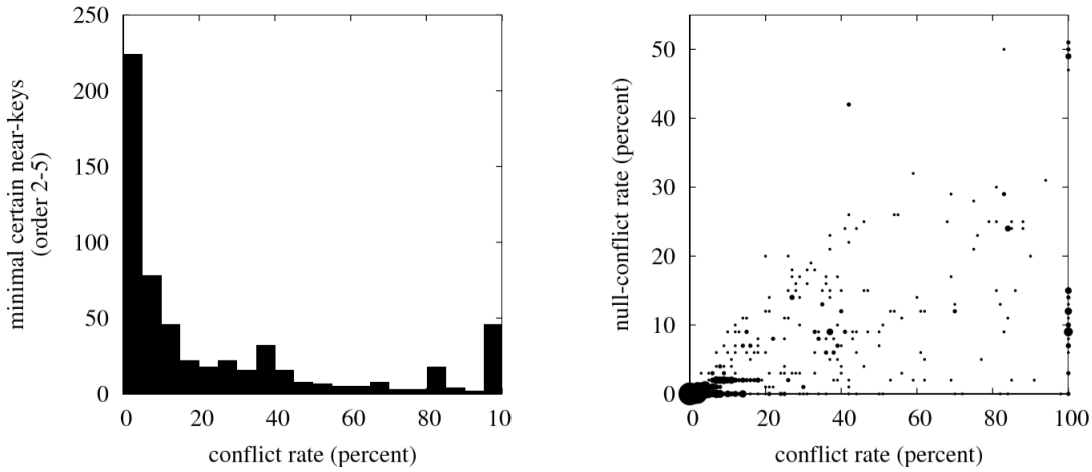


Figure 1: Distribution of conflict rates, and relationship between conflict and null-conflict rates

8.2 Qualitative Insights

Our running example for this paper was motivated by the `ncbi_taxonomy` table from the PFAM data set. The table contains three columns: *ncbi_taxid*, *species*, and *taxonomy*. Here, *ncbi_taxid* forms a surrogate primary key, and *taxonomy* features null marker occurrences. The table consists of 822,451 rows. Our algorithm found all certain keys (i.e. $c\langle ncbi\_taxid\rangle$) in an average 1.7 seconds over ten runs, and all certain near-keys up to order 5 (i.e. $C_1\langle ncbi\_taxid\rangle$ and $C_3\langle species, taxonomy\rangle$) in an average 13.9 seconds over ten runs.

As a second example we take a look at the `features` table from the RFAM (RNA families) data set. The table contains nine columns: *auto_features*, *auto_rfamseq*, *database_id*, *primary_id*, *secondary_id*, *feat_orient*, *feat_start*, *feat_end*, and *quaternary_id*. The columns *primary_id*, *secondary_id*, and *quaternary_id* each contain null marker occurrences. The table contains 21,439,852 rows, of which we mined the first million only. The following table shows the certain near-keys along with their order, number of conflicts and conflict rate.

certain near-key	*order*	*#conflicts*	*conflict rate*
auto_features	1	-	-
auto_rfamseq,database_id,feat_end,secondary_id	1	-	-
auto_rfamseq,database_id,feat_end,primary_id	1	-	-
auto_rfamseq,database_id,feat_end,feat_start	1	-	-
auto_rfamseq,feat_start,primary_id	2	10	0.002%
auto_rfamseq,feat_start,secondary_id	2	12	0.0024%
auto_rfamseq,feat_end	2	25	0.005%
auto_rfamseq,feat_start,quaternary_id	2	28	0.0056%
auto_rfamseq,feat_start	3	48	0.0093%
auto_rfamseq,database_id,primary_id	5	1473	0.2221%

Our algorithms found all certain keys in an average 31.6 seconds over ten runs, and all certain near-keys up to order 5 in an average 244 seconds over ten runs. Notably, the attribute *auto_features* serves as surrogate primary key, and all the certain near-keys of order 2 look very much like meaningful certain keys due to their small conflict rate. For example, the conflict rate for the certain key $c\langle auto\_rfamseq, feat\_start, primary\_id\rangle$ is only 0.002%, justifying the name *near*-key. Samples such as

auto_features	*auto_rfamseq*	*database_id*	*primary_id*	*secondary_id*	*feat_orient*	*feat_start*	*feat_end*	*quaternary_id*
102842	101318	EMBL-tRNA	tRNA-Ile	$\perp$	1	37601	37669	$\perp$
102841	101318	EMBL-tRNA	tRNA-Ile	$\perp$	1	37601	37678	$\perp$

indicate similar problems caused by the primary key as described for the `ncbi_taxonomy` table.

9 Conclusion and Future Work

We developed the first approach towards the discovery of meaningful keys from incomplete and inconsistent data sets. The new concept of a certain near-key of order n requires every possible world of a given data set to contain no more than n tuples that have matching values on all key attributes. Based on hypergraph transversals we developed an algorithm that discovers all certain near-keys up to a given order. The problem of deciding certain near-key satisfaction in a given data set is coNP-complete and W[1]-hard in the order of the certain near-key, i.e., it is unlikely that satisfaction can be decided in $O(P(|r|) \cdot f(n))$ time for some polynomial function P. Experiments with real-world data strongly indicate that our algorithm can discover meaningful certain keys which are violated in the given data set. Our conflict-rate measure can act as another filter to remove certain near-keys that are unlikely to constitute certain keys. Despite the theoretical hardness of certain near-key satisfaction, their discovery in practice appears to be not much more time-consuming than the discovery of certain keys.

There are many directions that warrant future research. It would be interesting to extend the work on certain keys and certain near-keys to conditional variants [5], which would also address the accuracy dimension of data quality. More generally, it is interesting to investigate the trade-offs between the expressivity of data dependencies and the efficiency of associated decision problems, such as deciding their satisfaction in given tables or discovering them from tables. One may think of suitable extensions to certain functional dependencies [11], differential dependencies [26], or inclusion dependencies [10], and conditional variants thereof, over relations or more sophisticated data formats [2, 6, 9]. A challenging and important problem is to determine to which degree affordable automation can help with establishing the meaningfulness of the discovered constraints [14]. The problems of discovering constraints from given data sets may become more effective or efficient when the application domain is more defined. In general we think that small numbers of violations are beneficial to the discovery of meaningful constraints. After all, exceptions prove the rule.

Acknowledgements

This research is supported by the Marsden Fund Council from New Zealand Government funding, by the Natural Science Foundation of China (Grant 61472263) and the Australian Research Council (Grants DP140103171).

References

[1] Z. Abedjan, L. Golab, and F. Naumann. Profiling relational data: a survey. *VLDB J.*, 24(4):557–581, 2015.

[2] P. Brown and S. Link. Probabilistic keys for data quality management. In *CAiSE*, pages 118–132, 2015.

[3] E. F. Codd. *The Relational Model for Database Management, Version 2*. Addison-Wesley, 1990.

[4] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

[5] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

[6] S. Hartmann and S. Link. Efficient reasoning about a robust XML key fragment. *ACM Trans. Database Syst.*, 34(2), 2009.

[7] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. TANE: an efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111, 1999.

[8] J. Kivinen and H. Mannila. Approximate inference of functional dependencies from relations. *Theor. Comput. Sci.*, 149(1):129–149, 1995.

[9] H. Köhler, U. Leck, S. Link, and H. Prade. Logical foundations of possibilistic keys. In *JELIA*, pages 181–195, 2014.

[10] H. Köhler and S. Link. Inclusion dependencies reloaded. In *CIKM*, pages 1361–1370, 2015.

[11] H. Köhler and S. Link. SQL schema design: Foundations, normal forms, and normalization. In *SIGMOD*. http://dx.doi.org/10.1145/2882903.2915239, 2016.

[12] H. Köhler, S. Link, and X. Zhou. Possible and certain SQL keys. *PVLDB*, 8(11):1118–1129, 2015.

[13] P. Koutris and J. Wijsen. The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints. In *PODS*, pages 17–29, 2015.

[14] S. Kruse, T. Papenbrock, H. Harmouch, and F. Naumann. Data anamnesis: Admitting raw data into an organization. *Data Engineering Bulletin*, 39(2), 2016.

[15] Y. E. Lien. On the equivalence of database models. *J. ACM*, 29(2):333–362, 1982.

[16] J. Liu, J. Li, C. Liu, and Y. Chen. Discover dependencies from data - A review. *IEEE Trans. Knowl. Data Eng.*, 24(2):251–264, 2012.

[17] H. Mannila and K. Räihä. Dependency inference. In *VLDB*, pages 155–158, 1987.

[18] M. Memari, S. Link, and G. Dobbie. SQL data profiling of foreign keys. In *ER*, pages 229–243, 2015.

[19] F. Naumann. Data profiling revisited. *SIGMOD Record*, 42(4):40–49, 2013.

[20] F. Naumann and M. Herschel. *An Introduction to Duplicate Detection*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[21] T. Papenbrock and F. Naumann. A hybrid approach to functional dependency discovery. In *SIGMOD*. http://dx.doi.org/10.1145/2882903.2915203, 2016.

[22] S. Sadiq. *Handbook of Data Quality*. Springer, 2013.

[23] B. Saha and D. Srivastava. Data quality: The other face of big data. In *ICDE*, pages 1294–1297, 2014.

[24] Y. Sismanis, P. Brown, P. J. Haas, and B. Reinwald. GORDIAN: Efficient and scalable discovery of composite keys. In *VLDB*, pages 691–702, 2006.

[25] S. Song, L. Chen, and H. Cheng. On concise set of relative candidate keys. *PVLDB*, 7(12):1179–1190, 2014.

[26] M. W. Vincent, J. Liu, H. Liu, and S. Link. "Differential dependencies: Reasoning and discovery" revisited. *ACM Trans. Database Syst.*, 40(2):14, 2015.

[27] M. Zhang, M. Hadjieleftheriou, B. C. Ooi, C. M. Procopiuc, and D. Srivastava. On multi-column foreign key discovery. *PVLDB*, 3(1):805–814, 2010.

Effective Data Cleaning with Continuous Evaluation

Ihab F. Ilyas
University of Waterloo
ilyas@uwaterloo.ca

Abstract

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. The pace of the acquisition and the variety of the data sources make it impossible to clean this data as it arrives. This new reality has made data cleaning a continuous process and a part of day-to-day data processing activities. The large body of data cleaning algorithms and techniques is strong evidence of how complex the problem is, yet, it has had little success in being adopted in real-world data cleaning applications. In this article we examine how the community has been evaluating the effectiveness of data cleaning algorithms, and if current data cleaning proposals are solving the right problems to enable the development of deployable and effective solutions.

1 Introduction

Data collection and acquisition often introduce errors in data, for example, missing values, typos, mixed formats, replicated entries of the same real-world entity, and even violations of business rules. Figure 1 shows examples of data errors in a relational table, where multiple types of errors co-exist in the same data set. Even when data is machine-generated (e.g., data center's logs and sensor readings) errors still occur due to device malfunctions, interrupted connections, or due to aggregating data from various data sources.

With the recent surge in the availability of a variety of big data technologies and tools that allow fast ingestion of a large amount of data in various formats, dirty data has become the norm rather than the exception. Not surprisingly, developing data quality solutions is a challenging topic and is rich with deep theoretical and engineering problems. A large body of work has addressed some of the well-defined problems; for example, refer to the survey [11] on cleaning relational data with violations of integrity constraints, and to the multiple surveys on record linkage and de-duplication [14, 9, 8, 15, 10], and also on data profiling [1]. Most of this research focuses primarily on algorithms and techniques to detect certain types of errors, and a smaller subset of the solutions provide automatic or semi-automatic techniques to clean the detected errors.

However, most of the available solutions suffer from fundamental pragmatic problems when applied to real-world enterprise dirty data. To list a few, (1) *scale* renders algorithms with quadratic complexity (or worse) infeasible in big data settings—for example, those algorithms that enumerate all pairs of records to assess if a violation of a functional dependency exists among pairs of records, or to detect duplicate pairs; (2) *involving humans* in the data cleaning cycle is often mandated by business owners and is necessary when dealing with

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

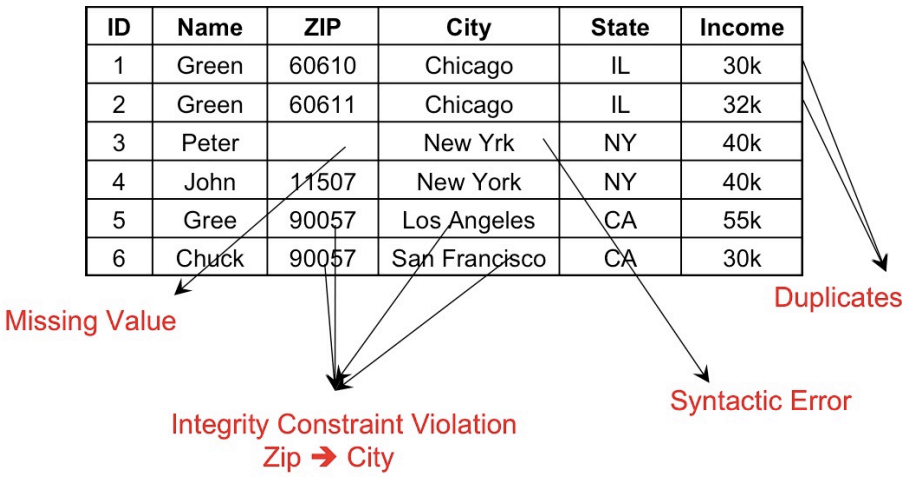


Figure 1: Example Data Errors

critical data, where ad-hoc automatic cleaning criteria are neither trusted nor accepted (we elaborate more on this in the rest of the paper); (3) *data variety* is often as serious as data volume in today's "data lakes": dealing with multiple formats, lack of unified schema, and the need for a rich set of transformations becomes crucial to carry out the cleaning process; (4) *holistic cleaning* of the data with respect to multiple quality problems is an effective way to spot the erroneous portion of the data; trying to clean each of the problems in isolation often results in wrong guesses of where errors are due to the limited context of these algorithms; and (5) *expressing business rules and sanity checks* is not often performed in terms of data quality and integrity constraints on the data sources, rather, these rules are usually expressed on top of aggregates and analytics (e.g., SQL views) computed from the raw source data; this decoupling often makes it very difficult to assess the effectiveness of cleaning tools and their impact on the business value of this data.

The scale challenge is often addressed using a variety of techniques including the use of sampling [17], approximate cleaning by partitioning the data into blocks as often done in data deduplication [3], and by adopting scale-out computing infrastructure such as Spark [19]. On the other hand, the variety in data syntax is often tackled by developing a large number of "connectors" that allow for accessing many external data sources with proprietary formats. It is less obvious how the rest of the challenges will be handled when developing an industry-strength data cleaning solutions.

In this paper, we will focus on a couple of these challenges, mainly, the problem of the decoupling in space and time between detecting data errors and the repairing of these errors, which is also tied to the problem of involving humans in guiding and evaluating data cleaning solutions. Furthermore, we argue in Section 3 that a continuous data cleaning life-cycle, with humans in the loop, is necessary to address the decoupling of error detection and error correction.

The rest of this article is organized as follows: in Section 2, we present some of the commonly used cleaning objective functions or evaluation criteria in most of the current solutions, and we discuss the challenges with these objectives. In Section 3, we give an example of a pragmatic solution to convey the impact of effectively involving data analysts and business users in guiding and evaluating the cleaning solution. In Section 4, we describe a real world deployment of a commercial data curation solution, and we conclude in Section 5 with a summary and final remarks.

2 Data Cleaning Objective Functions

Data cleaning primarily involves primarily two main exercises: (1) *Error Detection*, where errors (such as duplicates or violations of business rules) are spotted and surfaced to be examined either by a human or by a repairing mechanism; and (2) *Data Repairing*, which involves updating the data sources to remove these errors, for example, deleting erroneous records, merging duplicate records, or updating specific values that are believed to cause the violations.

Error detection can be performed in a variety of ways, ranging from "eye-balling" the data by experts to automatic tools and algorithms to highlight possible errors in the data with respect to some objective function. For example, to automatically detect errors with respect to declared integrity constraints (such as functional dependencies or denial constraints [6]), error detection algorithms often enumerate possible combinations of records to create context for checking a rule: to detect possible duplicates, all pairs of records might need to be generated to apply a classifier or a similarity function to detect duplicates. For other types of errors, for example, outliers, statistical properties of the underlying data are often calculated to decide on "unusual" values.

Unfortunately, *data repairing* is often more challenging; asking humans and experts to correct all possible errors spotted by the error detection tools is obviously prohibitively expensive in large data settings. On the other hand, master or reference data sets that can be used to guide the repairing process might not be available and often suffer from poor coverage of the data [7]. Much work adopt other guidelines to suggest data repairs based on less-justified principles such as *minimality of repairs* [5, 13]. The general intuition is that since there are more clean data than errors, then if algorithm A changes more data than Algorithm B to remove data errors (with respect to some chosen error detection algorithm), Algorithm B is more preferred. While the concept makes sense, it is not obvious if it can be translated into an operational objective function, where the minimum number of cell updates to a dirty database produces "the clean" version of this data.

In previous work [2], we conducted an experiment to measure the precision and the recall of multiple clean instances produced by an automatic repairing algorithm. All these instances enjoy some notion of minimality of updates (in this case cardinality-set minimal). Figure 2 plots the precision and recall for each of these minimal repairs. Besides the low values for precision and recall, the figure further shows a spread in the values of precision and recall of these repairs. These results suggest that there is little to no correlation between minimality-based repairing and accuracy of the repairs. We argue that the reason for these results is the way we used minimality as operational semantics of data repairing as opposed to a guiding principle. In other words, believing that "there is more good than bad in the data" does not directly translate into favouring cleaning operations with fewer changed cells, especially when performed in local contexts and without fully understanding the sources of theses errors or how these errors interact. We show in Section 3 that minimality can be a very useful concept in ranking and evaluating the likelihood of errors as long as humans have the final decision in the repairing process.

From these observations, and since data cleaning is a continuous process, evaluating the effectiveness of data cleaning activity is essential in guiding error detection and repairing of erroneous data. When a "ground truth" is available or can be obtained, the accuracy of the data cleaning (detection and repairing) can be measured via standard measures such as precision and recall. However, in the absence of such ground truth, it is not always obvious how to measure the effectiveness of the proposed cleaning technique. We briefly discuss two main approaches:

- *Sample and verify by experts:* to address the scale problem of involving humans in the verification of data repairs, one can present a sample of the repairs (data updates) to be examined by human experts. Several challenges need to be addressed when using samples for verification, for example, (1) how to provide enough context to humans to judge the correctness of the repairs; while this might not be crucial for certain types of errors such as the check constraint $Salary > 45k$, other errors like outliers require more "global" context to surface for the expert; (2) the normal challenges involved in using human experts

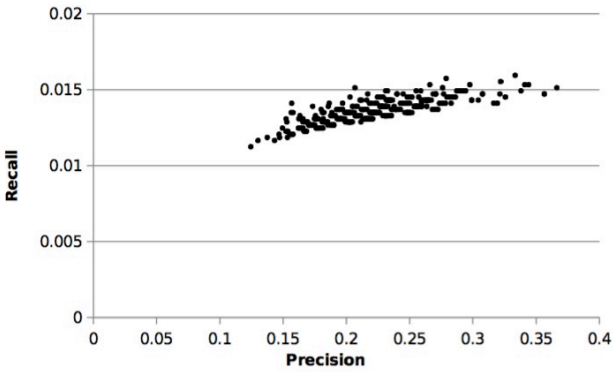


Figure 2: Accuracy of Minimal Repairs

such as expertise registration, handling conflicting opinions, and dealing with access controls; and (3) how to extrapolate and scale the effectiveness measure from measures calculated from the sample, which can vary in complexity depending on the type of errors addressed.

- *Assessing the impact on reports and analytics:* Often, the raw data, where data is generated or acquired, might not have enough semantics in the form of declared integrity constraints to even detect errors, and errors can only be spotted by analysts and business users at the reporting or analytics layer. This complicates both error detection and evaluating the impact of the repairing strategy employed. Several challenges arise, but mainly (1) how to push spotted errors in reports down to where errors can be corrected and repaired? and (2) how to effectively propagate the effect of the repairs up to the reporting layer to guide and to evaluate the impact of the repairing process? are of utmost importance to address.

While discussing human involvement and sampling for data cleaning is outside the scope of this article, in Section 3, we focus on pragmatic approaches to assess the impact of the data cleaning tools on reports and analytics.

3 Measuring The Effectiveness of Cleaning Solutions

Through our experience with deploying Tamr [1] (the commercial descendant of the Data Tamer System [16]) in multiple large enterprises (e.g., an international automotive manufacturer) with large-scale data cleaning projects, it was evident that one of the biggest challenges in adopting a data cleaning approach is measuring the impact of this solution on the main line of business (aka Return on Investment), which decides (1) the amount of resources to be dedicated to the cleaning project, (2) the incentives to involve domain experts in the curation cycle, and (3) the prioritization of the type of anomalies to fix. For the rest of this section, we assume data errors that can be detected by either human involvement, or by automatic techniques that spot conflicts with other database values (e.g., as in the case of duplicates and violations of integrity constraints).

To address the challenge of assessing the effectiveness of a proposed data cleaning approach, we focus on the decoupling in space (the data set) and time (the data processing phase) between error detection and data repairing efforts [4]: most errors are born during data generation or data acquisition, due to a variety of error generation models (e.g., human errors, using the wrong transformations, or as a result of integrating data from multiple data sources). However, data is almost never consumed in its raw format; instead, layers of transformations, aggregation, and analytics are often in place before data is consumed by business analysts or data scientists

[1] www.tamr.com

in the form of reports, or in rich visualizations layers. Business analysts and data scientists can easily spot inconsistencies, fishy results, violations of business rules, and various types of errors. In other words, while detection makes sense much later in data processing pipelines on views built on top of the source data (with the availability of rich semantics and domain experts), errors need to be corrected much earlier in the data processing life cycle, directly on the raw data sources.

Example 1: To illustrate the problem, we use a simple example, where data sources are relational tables and the "analytics" is a simple aggregation query to compute a target view on this data: consider the report T in Figure 3 about shops for an international franchise. T is computed according to the following query on two sources $Shops$ and $Emps$:

Q:
```
SELECT SId as Shop, Size, Grd, AVG(Sal) as
  AvgSal, COUNT(EId) as #Emps,'US' as Region
  FROM US.Emps JOIN US.Shops ON SId
  GROUP BY SId, Size, Grd
```

The HR business analysts enforce a set of policies in the franchise workforce and identify two problems in T. The first violation (t_a and t_b, in bold) comes from the rule "in the same shop, the average salary of the managers (GRD=2) should be higher than that of the staff (GRD=1)". The second violation (t_b and t_d, in italic) comes from the rule " a bigger shop cannot have a smaller number of staff". Note that no business rules are defined on the sources $Shops$ and $Emps$.

T	Shop	Size	Grd	AvgSal	#Emps	Region
t_a	**NY1**	46 ft^2	**2**	**99** $	1	US
t_b	**NY1**	*46* ft^2	***1***	**100** $	*3*	US
t_c	NY2	62 ft^2	2	96 $	2	US
t_d	NY2	*62* ft^2	*1*	90 $	*2*	US
t_e	LA1	35 ft^2	2	105 $	2	US

Emps	EId	Name	Dept	Sal	Grd	SId	JoinYr
t_1	e4	John	S	91	1	NY1	2012
t_2	e5	Anne	D	99	2	NY1	2012
t_3	e7	Mark	S	93	1	NY1	2012
t_4	e8	Claire	S	116	1	NY1	2012
t_5	e11	Ian	R	89	1	NY2	2012
t_6	e13	Laure	R	94	2	NY2	2012
t_7	e14	Mary	E	91	1	NY2	2012
t_8	e18	Bill	D	98	2	NY2	2012
t_9	e14	Mike	R	94	2	LA1	2011
t_{10}	e18	Claire	E	116	2	LA1	2011

Shops	SId	City	State	Size	Started
t_{11}	NY1	New York	NY	46	2011
t_{12}	NY2	New York	NY	62	2012
t_{13}	LA1	Los Angeles	CA	35	2011

Figure 3: A report T on data sources Emps & Shops.

Example 1 clearly highlights the following:

1. Errors can be easily spotted when richer semantics are available to detect violations of integrity constraints, which can be done either automatically or by involving human experts who are familiar with a given report.

2. Data repairing cannot be done at the report level, since these are usually read-only views that can be arbitrarily complex. Data curation tools and solutions will only be applicable to raw data sources.

3. Repairing algorithms on the data sources will be evaluated based on their ability to correct the errors spotted in the computed reports and not solely on the number of cells they change in the source data. Note, however, that minimality of repairs can be used as a guiding principle and a ranking mechanism of more probable repairs to surface for experts.

We argue that any effective cleaning solution has to address two main challenges: (1) closing the gap between detection and evaluation in the reporting layer, and data repairing at the raw data sources; this necessarily requires reasoning about provenance and data lineage; and (2) providing experts and evaluators with error explanations and prescriptions to clean data at the sources in a way that is relevant to and in the context of the analytics and reports of interest.

Figure 4 depicts an *evaluate-clean* loop that adopts a pragmatic cleaning evaluation (via business analysts and domain experts) and leverages provenance, automatic error detection and repairing techniques to suggest possible errors, to mine possible explanations of these errors, and to translate them into cleaning actions. The life cycle is comprised of four main phases:

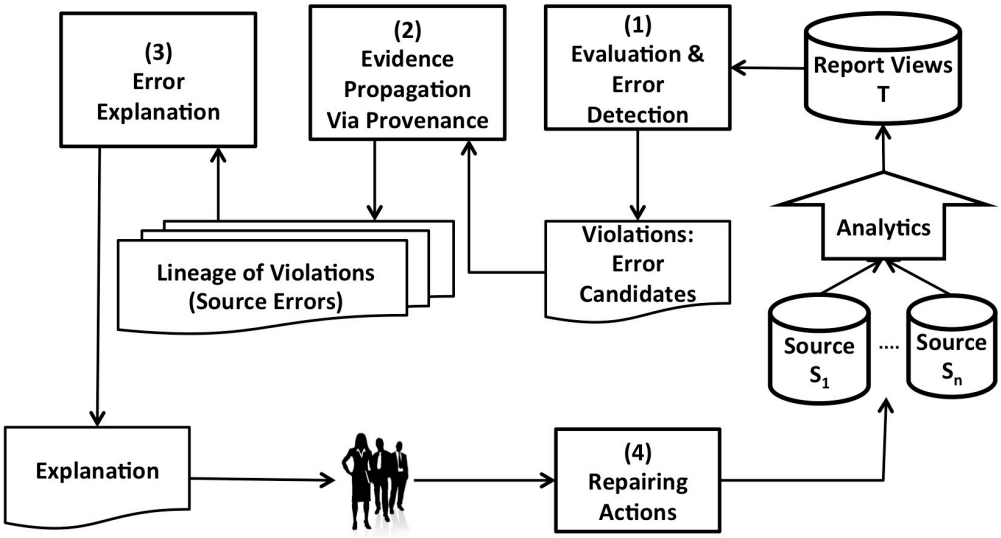


Figure 4: Clean-Evaluate Loop

1. *Evaluation and Error Detection:* Errors are detected on the reports level via normal error detection techniques such as running automatic error detection scripts, checking for violations of business rules, or visually spotting outliers [18]. This is also the phase where the reports are observed to evaluate the effect of cleaning or updating data sources (e.g., via A/B or split testing).

2. *Error Propagation:* Since violations detected in the report are actually caused by errors that crept in at an earlier stage, i.e., from the sources, propagating these errors from a higher level in the transformation to the underlying sources can help in identifying the source of the errors and in prescribing actions to correct them. Scorpion [18] and DBRx [4] are example solutions that attempt to trace back the tuples that contributed to the problems in the target. For example, Tuples t_a, t_b are in violation in T and their lineage is $\{t_1 - t_4\}$ and $\{t_{11} - t_{12}\}$ over Tables $Emps$ and $Shops$. By removing or fixing these source tuples, the violation is removed.

As the view definition gets more complex, for example, to include aggregation and user-defined functions, the provenance of the reported data gets more complex and hence, very challenging to reason about. This complicates error propagation and relating errors in data sources to errors that can be easily spotted at the analytics layer.

Error propagation is further complicated by the fact that the source values that belong to the lineage of an erroneous report value might not be all wrong, as errors may not be deterministically inferred from data lineage. For example, an aggregate query would clump together several source tuples with few containing actual errors. Source tuples do not contribute to violations in equal measure. Hence, we need a mechanism to accumulate evidence on tuples across multiple violations of business rules to identify the most likely tuples with errors. The work in [18, 4] provides mechanisms to compute the likelihood of errors in the source data, given the lineage and the type of errors detected at the reports level.

3. *Error Explanation:* After identifying the likely errors at the sources, mining concise and accurate explanations of these errors becomes an important task to allow experts to perform cleaning actions on the sources that have a direct effect on the output results.

   Error explanation is done in [4] as building a view on the data sources that covers (only) the errors propagated from reports to sources via error propagation. Among all possible views that can be built, the ones that adhere to predefined plausible properties (e.g., precision, recall, and conciseness) are chosen.

   In Example 1, two possible explanations of the problems are: $[Emps.JoinYr = 2012]$ on Table $Emps$, and $[Shops.State = NY]$ on Table $Shops$. As we can see, t_b is the only tuple that is involved in both violations and, hence, is identified by the repairing algorithm as erroneous. Its lineage is $\{t_1, t_3, t_4\}$ and $\{t_{11}\}$ over Tables $Emps$ and $Shops$, respectively. By focusing on this tuple, we can compute more precise explanations on the sources, such as $[Emps.Dept = S]$.

4. *Repairing Actions:* Based on the explanation, experts might invoke multiple actions to clean the data sources, for example, running transformation scripts for units alignment, performing targeted updates on the records identified as erroneous, or even deleting sources with low quality or redundant information. Once these actions take place, running analytics again and moving to Phase 1 of the process will allow for measuring the impact of these actions in a usage-based manner and in the context of the given analytics.

4 Real-World Experience

The aforementioned Clean-Evaluate loop in Figure 4 has been adopted by many current commercial data cleaning and curation platforms such as Tamr [2] (based on the Data Tamer system and Trifacta [3] (based on the Wrangler system [12]). For example, Tamr employs a *continuous* curation process, where user's engagement is used at multiple levels, for example: (1) to assess the quality of analytics and translate user feedback into feedback on the underlying data sources; (2) to generate training data for the back-end machine learning models, (e.g., deduplication models); and (3) to administer repairing actions on the underlying data sources given the feedback on analytics and the learned models.

Deploying this curation life-cycle in a single curation project at a Fortune 500 manufacturer involved integrating data from more than 50 ERP (Enterprise Resource Planning) systems with millions of records in a variety of formats. The curation project employed more than 20 experts for validating the detected error (e.g., duplicate records) and the suggested repairs. The result was a cleaner data set that uncovered a saving of more than $100M for the company. This exercise won't be possible without an iterative process, where error detection and repairing are driven by the analytics that can be examined by domain experts and business analysts, and

[2] www.tamr.com

[3] www.trifacta.com

anomalies are tied back to the errors in the raw data sources (in this case the ERP systems). In this example, various traditional automatic data cleaning algorithms have been re-purposed to serve as ranking mechanisms to show the most probable repairs to the human-in-the-loop, which maximize the benefit of the budget allocated to expert involvement in the project.

5 Conclusion

A key aspect of data cleaning is the ability to evaluate the accuracy and the effectiveness of the algorithms and tools used in the process. The evaluation is almost straightforward when humans examine the whole set of data cleaning results, or when ground truth can be obtained via, for example, reference data sources. However, with the large amounts of dirty data currently acquired in big data infrastructures, neither humans nor complete master data sources are available for evaluation. We highlighted multiple challenges with widely adopted cleaning objective functions such as minimality of repairs. We also showed an example workflow to make the evaluation process an integral part of the data cleaning process itself, where experts are involved in evaluating the effectiveness of the underlying cleaning tools at the report and analytics level, while automatic techniques are re-purposed to suggest ranked updates. Numerous challenges remain to be addressed in developing adoptable data cleaning platforms, including effective ways to engage users and experts in guiding and evaluating the cleaning algorithms, connecting to the continuously growing number of data sources with a variety of data formats, and developing scalable repairing algorithms that can deal with large amounts of data to suggest the most relevant repairs.

References

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. Profiling relational data: a survey. *VLDB J.*, 24(4):557–581, 2015.

[2] George Beskales, Ihab F. Ilyas, and Lukasz Golab. Sampling the repairs of functional dependency violations under hard constraints. *PVLDB*, 3(1):197–207, 2010.

[3] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. Adaptive blocking: Learning to scale up record linkage. In *6th International Conference on Data Mining*, pages 87–96, 2006.

[4] Anup Chalamalla, Ihab F. Ilyas, Mourad Ouzzani, and Paolo Papotti. Descriptive and prescriptive data cleaning. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 445–456, 2014.

[5] Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Information and Computation*, 197(1/2):90–121, 2005.

[6] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. Holistic data cleaning: Putting violations into context. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 458–469, 2013.

[7] Xu Chu, Mourad Ouzzani, John Morcos, Ihab F. Ilyas, Paolo Papotti, Nan Tang, and Yin Ye. KATARA: reliable data cleaning with knowledge bases and crowdsourcing. *PVLDB*, 8(12):1952–1963, 2015.

[8] Xin Luna Dong and Felix Naumann. Data fusion: resolving data conflicts for integration. *Proceedings of the VLDB Endowment*, 2(2):1654–1655, 2009.

[9] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[10] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.

[11] Ihab F. Ilyas and Xu Chu. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends in Databases*, 5(4):281–393, 2015.

[12] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, pages 3363–3372, 2011.

[13] Solmaz Kolahi and Laks V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, 2009.

[14] Nick Koudas, Sunita Sarawagi, and Divesh Srivastava. Record linkage: similarity measures and algorithms. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 802–803, 2006.

[15] Felix Naumann and Melanie Herschel. *An Introduction to Duplicate Detection*. Synthesis Lectures on Data Management. 2010.

[16] Michael Stonebraker, Daniel Bruckner, Ihab F. Ilyas, George Beskales, Mitch Cherniack, Stanley B. Zdonik, Alexander Pagan, and Shan Xu. Data curation at scale: The data tamer system. In *CIDR 2013, Sixth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings*, 2013.

[17] Jiannan Wang, Sanjay Krishnan, Michael J Franklin, Ken Goldberg, Tim Kraska, and Tova 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*, pages 469–480, 2014.

[18] Eugene Wu and Samuel Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8):553–564, 2013.

[19] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, volume 10, page 10, 2010.

Benchmarking Data Curation Systems

Patricia C. Arocena*
University of Toronto

Boris Glavic
Illinois Institute of Technology

Giansalvatore Mecca
University of Basilicata

Renée J. Miller*
University of Toronto

Paolo Papotti
Arizona State University

Donatello Santoro
University of Basilicata

Abstract

Data curation includes the many tasks needed to ensure data maintains its value over time. Given the maturity of many data curation tasks, including data transformation and data cleaning, it is surprising that rigorous empirical evaluations of research ideas are so scarce. In this work, we argue that thorough evaluation of data curation systems imposes several major obstacles that need to be overcome. First, we consider the outputs generated by a data curation system (for example, an integrated or cleaned database or a set of constraints produced by a schema discovery system). To compare the results of different systems, measures of output quality should be agreed upon by the community and, since such measures can be quite complex, publicly available implementations of these measures should be developed, shared, and optimized. Second, we consider the inputs to the data curation system. New techniques are needed to generate and control the metadata and data that are the input to curation systems. For a thorough evaluation, it must be possible to control (and systematically vary) input characteristics such as the number of errors in data cleaning or the complexity of a schema mapping in data transformation. Finally, we consider benchmarks. Data and metadata generators must support the creation of reasonable gold-standard outputs for different curation tasks and must promote productivity by enabling the creation of a large number of inputs with little manual effort. In this work, we overview some recent advances in addressing these important obstacles. We argue that evaluation of curation systems is itself a fascinating and important research area and challenges the curation community to tackle some of the remaining open research problems.

1 Introduction

A curated database is a valuable asset that has been created and maintained with a great deal of human effort [13]. The term data curation has been used as an umbrella term to encompass the activities required to maintain and add value to data over its lifetime, and more specifically the tools and algorithms that attempt to reduce human curation effort by automating some of these important activities. Some data curation tasks that have received significant attention in the database literature include data cleaning (identifying and repairing errors in data), entity resolution (identifying and resolving duplicates in data), data transformation (exchanging or translating

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Supported by NSERC

data), data integration (including data federation), data provenance (understanding the origin of data), metadata or schema discovery, data and metadata profiling (including statistical profiling of data and data usage), and data archiving. Some of these tasks, such as data profiling [1], are also important to operational data management, for example, statistical profiles can be used to improve query performance. In this paper, we focus on the use of these tasks to curate data by improving the value or quality of information. In contrast to basic data management problems like query or transaction processing, data curation has not benefitted from the availability of commonly accepted benchmarks which can be used to compare systems, resolve discrepancies, and advance the field. As a result, evaluation and comparison of systems have relied on a few real data and metadata scenarios (for example, the Illinois Semantic Integration Archive[1], the Sherlock@UCI[2] data cleaning and entity resolution data sets or the more recent *annealling standard* [30]). Large scale sharing of real scenarios is simply not feasible due to the inherent value and proprietary nature of many data resources. And importantly, real scenarios do not provide control over many of the input characteristics that may impact the performance or efficacy of a system. As a result, researchers rely on *ad hoc* scenario generators with knobs to control a few selected data or metadata characteristics.

At first sight it may be surprising that evaluations of data curation tasks are not up to par with evaluations of query performance in database management systems. However, as we will explain in the following, the different nature of data curation problems imposes unique challenges for evaluations which are not faced when evaluating query performance. From a high-level view, any empirical evaluation of an algorithm roughly follows this pattern: 1) identify relevant input parameters, reasonable values of these parameters, and output measures; 2) select an input parameter to be varied systematically; 3) vary the selected parameter while keeping all other parameters fixed; 4) run the system on input generated for each parameter value; and 5) evaluate the system output using a selected set of output measures. This process is necessarily iterative with the results of one iteration in the evaluation influencing the parameters considered in the next.

The papers in this special issue of the IEEE Data Engineering Bulletin consider the problem of assessing or improving data quality, often this may be done in the context of a specific curation task. In contrast, in our work, we are seeking to understand the principles behind evaluating a specific curation system that seeks to improve data value or quality in a specific, quantifiable way. We consider the problem of assessing whether a system has achieved a specific goal of improving data quality or whether it has reduced the human effort needed to achieve a specific curation goal. To illustrate some of the challenges, in this paper we will concentrate on two specific data curation tasks: data exchange and constraint-based data repair.

1.1 Evaluating Data Exchange Systems

Data exchange systems take as input a pair of schemas (typically named *source* and *target*), an instance of the source schema, and a mapping specifying a relationship between the schemas [15]. They output a transformation of the source instance into a target instance that must satisfy the mapping. However, different systems may create target instances that differ, for example, on how much redundancy they contain [16] and the transformation code they produce may differ in performance [24].

Consider the simple example schemas in Figure 1 that are related by mappings m_1, m_2 and m_3. Assume we want to compare two data exchange systems E_1 and E_2. In this example, system E_1 creates a target instance J_1 and system E_2 creates a target instance J_2. Both instances satisfy the mapping specification and in fact both are *universal solutions* [15]. To evaluate and compare these, we need to select an appropriate *output measure*. As with traditional data management problems, output measures may be performance-based, for example, the response time to perform data exchange. Unlike traditional data management, the accuracy of the system is also an important output measure. For data exchange, an output measure may compare the quality of two exchanged instances, as suggested by Alexe et al. [2]. This measure considers one output superior to another

[1] http://pages.cs.wisc.edu/~anhai/wisc-si-archive/

[2] http://sherlock.ics.uci.edu/data.html

SOURCE INSTANCE

PStat	Name	Season	Team	G
t_1 :	Giovinco	2012-13	Juventus	3
t_2 :	Giovinco	2014-15	Toronto	23
t_3 :	Pirlo	2015-16	N.Y.City	0

Stdm	Team	Stadium
t_1 :	Toronto	BMO Field
t_2 :	N.Y.City	Yankee St.

JuveDB	Name	Season
t_1 :	Giovinco	2012-13
t_2 :	Pirlo	2014-15

MAPPINGS

m_1. PStat(name, seas, team, goals) $\rightarrow \exists N$Player(name, seas, team, goals), Team(team, N)
m_2. Stdm(team, stdm) $\rightarrow$ Team(team, stdm)
m_3. JuveDB(name, seas) $\rightarrow \exists N$Player(name, seas, '$Juv.'$, N), Team('$Juv.'$, '$Juv.Stadium'$)

SOLUTION J_1

Player	Name	Season	Team	Goals
t_1 :	Giovinco	2012-13	Juventus	3
t_2 :	Giovinco	2014-15	Toronto	23
t_3 :	Pirlo	2015-16	N.Y.City	0
t_4 :	Pirlo	2014-15	Juventus	N_1

Team	Name	Stadium
t_1 :	Toronto	BMO Field
t_2 :	N.Y.City	Yankee St.
t_3 :	Juventus	Juventus St.

SOLUTION J_2

Player	Name	Season	Team	Goals
t_1 :	Giovinco	2012-13	Juventus	3
t_2 :	Giovinco	2014-15	Toronto	23
t_3 :	Pirlo	2015-16	N.Y.City	0
t_4 :	Giovinco	2012-13	Juventus	N_1
t_5 :	Pirlo	2014-15	Juventus	N_2

Team	Name	Stadium
t_1 :	Juventus	N_3
t_2 :	Toronto	N_4
t_3 :	N.Y.City	N_5
t_4 :	Toronto	BMO Field
t_5 :	N.Y.City	Yankee St.
t_6 :	Juventus	Juventus St.

Figure 1: Example Data Exchange Scenario With Two Data Exchange Solutions

if it contains less redundancy (in this example J_1 would be preferred over J_2). Alternatively, if we know what the expected output is, then we can measure the difference between an output produced by a system and the expected one, often referred to as a *gold standard*. For data exchange, the output measure typically involves comparing (potentially large) database instances and these measures can be complex [2, 25]. Since a systematic comparison involves many different input scenarios, we must have efficient (and shared) implementations of these measures.

Of course, a real comparison of systems E_1 and E_2 must consider not just a few example scenarios, but should include sets of input scenarios that differ on specific *input parameters*. For data exchange, these input parameters might include the size of the schemas, the size of the mapping, or the complexity of the mappings and schemas. They could also include parameters that measure metadata quality, such as how normalized a schema is (for example, a fully normalized schema may guarantee minimal redundancy in the data [22]). Agreement on what are the right input parameters (for example, how mapping complexity is characterized and controlled) and what are reasonable values for these parameters, is an important part of benchmarking. In addition to the data generators that are commonly available for benchmarking query processing, we need metadata generators that can efficiently generate schemas, schema constraints, and mappings in such a way as to provide control over chosen input parameters.

1.2 Evaluating Constraint-based Data Repairing

In data cleaning, business rules or constraints are often used to express expectations or rules that data should satisfy [20]. Data that is inconsistent with a set of constraints can be cleaned (or repaired) by using evidence in the data. A common example is to select a repair that has the minimal number of changes (or the minimal cost changes) to be consistent with the constraints. Consider the player relation in Figure 2 and assume we are given the following constraints:

(i) A functional dependency (FD) stating that Name and Season are a key for the table:
d_1 : Player : Name, Season $\rightarrow$ Team, Stadium, Goals.

(ii) And, a second FD stating that Team implies Stadium: d_2 : Player : Team $\rightarrow$ Stadium.

Player	Name	Season	Team	Stadium	Goals
t_1 :	Giovinco	2012-13	Juventus	**BMO Field**	3
t_2 :	Giovinco	2014-15	Toronto	BMO Field	23
t_3 :	Pirlo	2014-15	Juventus	Juventus Stadium	5
t_4 :	Pirlo	2015-16	New York City	Yankee Stadium	0
t_5 :	Vidal	2014-15	Juventus	Juventus Stadium	8
t_6 :	Vidal	2015-16	Bayern	Allianz Arena	3

Figure 2: Example Dirty Database

The database in Figure 2 is dirty with respect to d_2. A possible repair, that may be created by a data repair system is to change $t_1[Stadium]$ to the value "Juventus Stadium". There are, however, many possible alternative repairs, like, for example, changing $t_1[Team]$ to the value "Toronto", or changing both $t_3[Stadium]$ and $t_5[Stadium]$ to the value "BMO Field".

To evaluate a data repair algorithm, we must know the expected "gold-standard" output. Then we can use a simple recall and precision measure to evaluate the accuracy of the system (or a weighted recall-precision measure where credit is given for selecting a repair that is close to the desired value).

In addition, to evaluate repair systems, we must have a dirty data generator, that is a data generator that can introduce errors into the data in a systematic way. Again, we must have agreement as to what are the important input parameters for error generation. Certainly we should be able to control the number of errors, but also the quality of the errors – what makes an error hard or easy to repair? The generator should be able to introduce errors in data of different sizes and having different constraints. Ideally, the generator would handle a large class of constraints as a wide variety of constraint languages have been used in cleaning. Error generation could also be coupled with constraint generation to generate different number or types of constraints. And again, it must be possible to generate dirty data efficiently, giving the user (the system evaluator) as much control over the chosen input parameters as possible.

1.3 Evaluating Data Curation

The two examples of data exchange and data repair illustrate the following requirements for large-scale evaluation of data curation tasks.

(a) Quality of the Outputs. Unlike query evaluation where there is one fixed expected output of a system (the unique query result) and a small number of performance related measures to be considered (e.g., throughput, mean latency, or memory consumption), in data curation there may be multiple acceptable outputs for a task that exhibit different quality. This is complicated by the fact that the quality of an output may be a multi-faceted combination of potentially conflicting quality measures. For instance, if a database that violates a set of constraints should be cleaned by removing tuples causing violations, then two important measures of the quality of a cleaned database are the number of remaining constraint violations (how clean is the solution) and the number of tuples that got removed (how much information is lost). Obviously, both measures cannot be fully optimized simultaneously. Many established quality measures are computationally hard or non-trivial to implement, such as measures for data exchange quality that require comparing potentially large database instances on criteria that include how well they preserve source information [2, 25]. A rigorous comparison of solutions for a data curation task has to take multiple quality measures into account to complement performance

metrics.[3] When comparing two data curation algorithms we are interested in understanding their performance as well as the quality of the produced results. For example, if we are faced with the task of cleaning a very large dataset and have to decide which cleaning algorithm to apply to the problem, then to make an informed decision we need to not only know what is the quality of solutions produced by the algorithms, but also how well the performance scales over the input size.

Challenge. The community needs to agree on common measures of output quality to use in evaluating data curation tasks and share implementations of these measures. When exact computation is infeasible on large or complex output, research is needed on developing approximate measures that are still effective measures of output quality.

(b) Input Parameters. As our examples illustrate, the input to data curation tasks can be quite complex. To evaluate query performance the input parameters that are typically considered are hardware and software of the machine used for the experiments, size of the dataset, data distribution, and complexity of the queries. Input parameters for data curation tasks are more diverse and each task may have its own set of relevant parameters. For example, in data cleaning one may evaluate an algorithm varying the number of errors in the input dataset and the data size for a fixed schema, while for data exchange the schema size is an important input parameter. For constraint or schema discovery, the amount of redundancy in the data may be an important parameter [4]. For data profiling, the relative independence (or dependence) of attribute distributions may be important [1]. Furthermore, creating an input that conforms to a specific input parameter setting may be non-trivial for some parameters. For example, an important input parameter for constraint-based data cleaning is the number of constraint violations in the input dataset. However, introducing a given number of errors into a clean database is known to be a computationally hard problem [6].

Challenge. The community must agree on the important input parameters (which may include parameters that vary the quality of the input) for different curation tasks. We must develop and share data and metadata generators that are able to efficiently generate input scenarios for data curation systems. Furthermore, these generators must be able to systematically vary the value of input parameters independently. Providing this fine-grained control of input characteristics while still efficiently generating large numbers of input scenarios is an important research challenge.

(c) Benchmarks. It is important that researchers be able to rely on common benchmarks to test their systems and compare results. In the context of data curation, benchmarks need to provide inputs to the system and establish which measures need to be used to assess the quality of outputs. Notice that some quality measures compare the quality of a solution against a **gold standard**, that is, a solution that is considered to be the correct or canonical solution for an input. Producing such a gold standard for some curation tasks can be highly complicated.

Challenge. We should use data and metadata generators to create new, community-accepted benchmarks for different curation tasks. In addition to generating varied input scenarios, these benchmarks should include - when possible - a gold standard output for each input scenario.

Given these challenges, it is more understandable that the standard of experimental evaluations of data curation systems is still quite low. Most researchers do not have the time and resources to implement complex quality metrics and solve the sometimes complex problems that arise when creating realistic inputs that conform with specific input parameter settings. It is often simply not feasible to spend this amount of effort "just" to evaluate a single system. A reasonable approach to improve the situation is 1) to make implementations of community-approved quality metrics publicly available and 2) to develop benchmarking solutions which enable the generation of synthetic, but realistic, metadata and data that conform to specific input parameters. This

[3] To avoid confusion, in this work we use the term **performance metric** to refer to measures of the runtime performance of an algorithm (e.g., average runtime or memory consumption) and **quality metric** to refer to a measure of the quality of a solution produced by a curation algorithm.

Domain	Example Input Parameters	Example Output Measures
Data Exchange	Schema and relation size Degree of schema normalization Mapping size and complexity Amount and type of mapping incompleteness	Preservation of source information Size of target data Redundancy in target data Similarity to gold-standard target
Constraint-based Data Repair	Data size No. of errors "Hardness" of errors	No. of repairs No. of errors remaining in repair Similarity of repair to gold standard
Discovery of Constraints or Data Quality Rules	No. of constraints Redundancy in the data Data size	Precision and Recall compared to a gold standard

Figure 3: Exemplary Input Parameters and Output Quality Measures for Data Curation

generation should require little user effort and it should be easy to recreate an input scenario produced by a benchmarking solution to make evaluations repeatable, e.g., by sharing the small configuration file that was used as an input for a benchmark generator instead of sharing the potentially large scenario itself.

In this paper, we consider the state-of-the-art in addressing these three evaluation challenges for two data curation tasks: data exchange and constraint-based data repair. Note that the purpose of this work is not to give a comprehensive overview of all quality metrics and benchmark generators that have been proposed, but rather to outline what are the major roadblocks for experimental evaluation of data curation solutions as well as discuss a few exemplary quality measures (as introduced in IQ Meter and others [25]) and systems that generate input scenarios for data curation tasks (BART [6] and iBench [5]). Figure 3 shows some exemplary relevant input parameters that we may want to vary for an evaluation and some meaningful output quality measures.

The remainder of this paper is organized as follows. We discuss output quality measures in Section 2, input parameters in Section 3, and benchmarking systems in Section 4. In Section 5, we briefly describe some new evaluations of curation systems that use these benchmarking systems. We conclude and discuss future work in Section 5.

2 Quality of the Outputs

In many fields, there are widely accepted quality measures that allow for the evaluation of solutions. Some, like precision and recall are pervasive and easy to compute, but require the existence of a gold standard output. Others, including metrics for clustering quality [26] that may be used to evaluate curation tasks like schema discovery [4], are computationally expensive. For data curation, the study of quality metrics is in its infancy. A (non-comprehensive) set of notable exceptions include metrics for evaluating matchings [8] and recent approaches for measuring the quality of data exchange solutions [2, 25]. Given that data curation tools often try to reduce human curation effort, some of these output measures try to quantify how well a tool succeeds in automating curation [3, 23, 25]. An important role of benchmarking is to enable the sharing of a standard suite of performance and quality metrics. For many curation tasks, these metrics may be computationally expensive and non-trivial to implement. In this section, we delve deeper into output measures for data exchange and data repair focusing on a few exemplary measures that illustrate the complexity of measuring the quality of a data curation system's output.

2.1 Data Repair Accuracy Measures

Let us start by discussing accuracy measures for data repairing. Here, there is a natural measure in terms of (weighted) precision and recall. To see this, consider our example in Figure 2. Assume a repair algorithm is executed over the dirty database, that we denote by I_d. The algorithm will apply a number of changes to attribute values, which we will call *cell changes*, of the form $t_1[Stadium] := \text{“}JuventusStadium''$, to I_d in order to obtain a repaired instance, denoted by I_{rep}. We call $\mathcal{C}h_{rep}$ the set of cell updates applied by the system to repair the database.

There are multiple – in fact, exponentially many – repairs that can be generated using this strategy. Typically, a notion of *minimality* of the repairs [10] has been used to discriminate among them (where minimality applies to the set of cell changes that create a repair). However, there are usually many repairs that minimally change the database, not necessarily all of the same quality.

Many researchers have in fact adopted a different approach to their evaluations. We can assume that the dirty database, I_d, has been generated starting from a clean database, I, and by injecting errors for the purpose of the evaluation. We know, therefore, the *gold standard* for this experiment, i.e., the original, clean version of the database. We also know the set of changes $\mathcal{C}h_d$ that are needed to restore the dirty instance I_d to its original clean version. Finally, since the only modification primitive is that of cell updates, the three instances I, I_d and I_{rep} all have the same set of tuple ids.

In this framework, a natural strategy to measure the performance of the algorithm over this database instance is to count the differences between the repaired instance and the correct one. In fact, the *quality* of the repair can be defined as the *F-Measure* of the set $\mathcal{C}h_{rep}$, measured with respect to $\mathcal{C}h_d$. That is, compute the precision and recall of the algorithm in restoring the dirty instance to its clean state. The higher the *F-measure*, the closer I_{rep} is to the original clean instance I. An *F-measure* of 1 indicates that the algorithm has changed the instance to its clean state by fixing all errors within the database, i.e., $I_{rep} = I$.

Notice that many data repairing algorithms not only use constant values to repair the database, but also *variables* [25]. A cell may be assigned a variable when the data-repairing algorithm has detected that the cell is dirty, but it cannot establish the original constant value. Data-repairing algorithms have used different metrics to measure the quality of repairs with constants and variables. Precision and recall may be computed using the following measures:

(i) **Value**: count the number of cells that have been restored to their original values;

(ii) **Cell-Var**: in addition to cells restored to their original values, count (with, for example, 0.5 score) the cells that have been correctly identified as erroneous and marked with a variable (in this case, changing a dirty cell to a variable counts 0.5);

(iii) **Cell**: count with a score of 1 all of the cells that have been identified as erroneous, both those that have been restored to their original value, and those that have been marked with a variable (in this case, changing a dirty cell to a variable counts 1).

2.2 Quality Measures For Data Exchange

The notion of quality for data exchange systems is more elaborate and there are many proposals in the literature for comparing the output of two data exchange algorithms [12, 2, 25].

IQ Meter [25] proposed two main building blocks for evaluating different data exchange systems: a measure of the quality of the solution with respect to a gold-standard solution, and a measure of the user-effort in designing the transformation with a given tool. Both measures were designed for a nested-relational data model for the source and target databases, capable of handling both relational data and XML trees.

Measuring Output Quality with IQ-Meter. Given a data exchange scenario, this measure assumes that a

gold standard has been given in terms of the output instance expected from the data exchange. An algorithm then determines the similarity of the output instance of a transformation tool with respect to this expected instance. Given the nested-relational data model, instances can be compared with traditional metrics such as tree or graph edit distance, but none of these can be used with large datasets because of their high complexity. Moreover, typical tree and graph-comparison techniques would not work in this setting. It is common in data transformations to generate synthetic values in the output – called *labeled nulls* in data-exchange and *surrogate keys* in Extract-Transform-Load (ETL) systems. These are placeholders used to join tuples, and their actual values do not have any business meaning. Therefore the measure needs to check if two instances are identical up to the renaming of their synthetic values. We may say that we are looking for a technique to check *tree or graph isomorphisms*, rather than actual similarities. Unfortunately, techniques for tree and graph isomorphisms are extremely expensive over the size of the instances.

A more efficient quality measure relies on the following key-idea: the instances to be compared are not arbitrary trees, but rather the result of data exchange. Since they are instances of a fixed known schema, this means that we know in advance how tuples are structured, how they should be nested into one another; and in which ways they join via key-foreign key relationships.

The quality measure abstracts these features in a set-oriented fashion, and then compares them by using precision, recall, and ultimately F-measures to derive the overall similarity. More specifically, for each instance, it computes: (i) a set of tuple identifiers, also called *local identifiers*, one for each tuple in the instance; (ii) a set of nested tuple identifiers, called *global identifiers*, which capture the nesting relationships among tuples; (iii) a set of pairs of tuple identifiers, called *join pairs*, one for each tuple t_1 that joins a tuple t_2 via a foreign key. It then compares the respective sets to compute precision, recall, and the overall F-Measure that gives the level of similarity. In addition to the measure of similarity, this metric provides feedback about errors in terms of missing and extra tuples in the generated output.

Other measures of accuracy for the data exchange setting have also been proposed [2, 12]. In particular, two similarity measures have been used to quantify the preservation of data associations from a source database to a target database [2]. Notice that in these approaches a gold-standard target instance is not provided for evaluation, therefore the authors focus their effort in measuring similarity between the original dataset and the result of the transformation. The first measure materializes data associations (joins of two or more relations) using the given referential constraints such as foreign keys or constraints logically implied by these constraints. Once all the associations are computed on the source and on the target, the similarity can be measured. Despite the different use case, this measure is reminiscent of the features discussed above in the IQ-Meter set-oriented measure. However, the second type of association in [2] pushes the idea further by considering *all* the natural joins that exist among the tuples of two relations. This measure, inspired by the notion of full disjunction, captures more associations than the one based only on the given constraints, and ultimately leads to a more precise measure of the similarity of two instances.

The IQ-Meter User Effort Measure. There are several possible ways to estimate user effort [3, 23, 25]. Basic techniques rely on the time needed to completely specify a correct transformation, or on the number of user interactions, such as clicks in the GUI [3]. A more sophisticated measure computes the complexity of the mapping specification provided through a data transformation tool GUI [25]. The IQ-Meter measure models the specification as an *input-graph* with labeled nodes and labeled edges. Every element in the schemas is a node in the graph. Arrows among elements are edges among nodes in the graph. If the tool provides a library of graphical elements – for example to introduce system functions – these are modeled as additional nodes. Extra information entered by the user (e.g., manually typed text) is represented as labels over nodes and edges. The measure evaluates the size of such graphs by encoding their elements according to a minimum description length technique, and then by counting the size in bits of such description. Experience shows that this model is general enough to cover every data transformation, spanning from schema mapping transformations to ETL ones, and provides more accurate results with respect to previous metrics based on point-and-click counts.

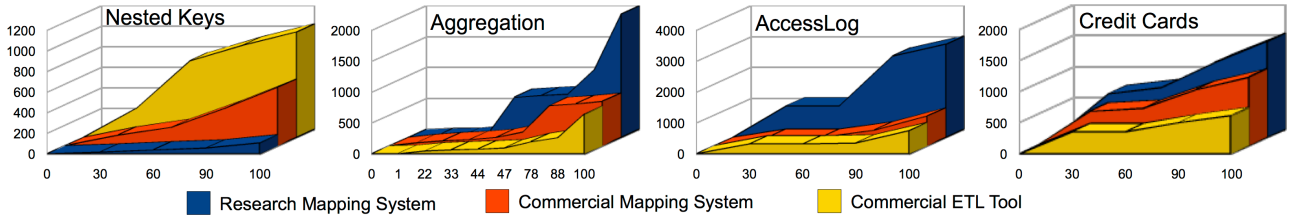


Figure 4: IQ Comparison of Three Transformation Systems over Four Scenarios

The most prominent feature of the two measures is that they enable us to plot *quality-effort* graphs to compare different systems over the same transformation scenario, as shown in Figure 4. Each plot shows how the quality achieved by a data transformation system varies with different levels of efforts. Intuitively, the size of the area below the curve in the plot for a given system is proportional to the amount of effort that is required to achieve high quality outputs with this system. Higher quality with lesser effort means higher effectiveness for the given task, and ultimately *"more intelligence"*.

3 Input Parameters

In data curation, often the goal is to ensure the curated output is of higher quality than the input data (and/or metadata). Hence, in evaluating a system, it is important to understand what are the important characteristics of this input data or metadata that can influence the performance of the system (as judged by both traditional performance metrics and also by data or metadata quality metrics). In this section, we review recent work on identifying important input parameters. In the next section, we consider how modern data and metadata generators provide flexible control over the values of some of these parameters so each can be used as an independent variable in system evaluations.

3.1 Input Parameters for Data Exchange

A data exchange scenario is a source schema (optionally with a set of constraints), an instance of the source schema, a target schema (optionally with constraints), and a mapping from the source to the target. Important input parameters include the size of the metadata, the size of the source data, and also the size and complexity of the mapping. We detail below how the complexity or quality of the input metadata has been characterized. In contrast to data repair, where the characteristics of the data to be cleaned play a major role, in data exchange, the characteristics of the metadata typically are the deciding factor for a system's performance.

Metadata Parameters. An important input characteristic for data exchange and mapping operators like mapping composition or mapping adaptation is the relationship between the source and target schema. In evaluating a mapping composition system, Bernstein et al. [9] use target schemas that were created from a source using a set of schema evolution primitives (for example, an `add-attribute` primitive). Yu and Popa [31] used a similar set of primitives to evaluate a mapping adaptation system. STBenchmark [3] generalized this idea to use a set of mapping primitives. Each primitive describes a specific relationship between a source and target schema (for example, vertically partitioning one relation into two fragments).

Two additional mapping quality dimensions identified in the iBench system are (1) the degree to which mappings reuse or share source and target relations (*metadata sharing*) and (2) the amount of incompleteness in a mapping. *Metadata sharing* directly influences how intertwined the produced mappings are. This in turn determines the degree to which data from the source must be merged in the target (if target metadata is shared) or the data from the source has to be copied to different places in the target (if source metadata is shared). As metadata sharing is increased, some data exchange systems can produce redundant target data or data with too

much incompleteness (labelled nulls) thereby decreasing the accuracy of these methods compared to a gold-standard output or decreasing the quality of the output when using an output measure based on the amount of redundancy and incompleteness in the target instance.

The number and complexity of mappings is another input parameter that can influence the performance of a data exchange or mapping system (such as a mapping inversion system or mapping composition system). The complexity of a mapping includes the language of the mapping which may be global-as-view (GAV), local-as-view (LAV), source-to-target (ST) tuple-generating-dependencies (TGDS), full ST TGDS, or other mapping languages [29]. In addition, the number of joins (in the source or target) used in a mapping may influence the performance of a data exchange system. For mapping languages that permit the modeling of incompleteness (that is, existentials in the target expression), the amount of incompleteness is also an important parameter that may influence system performance.

In addition to mapping characteristics, schema characteristics can influence system performance. These include the number and type of constraints in the schemas (for example, keys only vs. general FDs). By changing the constraints, a user can control whether a schema is normalized. The number and type of constraints may influence the performance or quality of the output of a data curation system. For example, the amount (and type) of target equality-generating-constraints (such as keys) may influence how hard it is for a data exchange system to create a consistent target instance.

3.2 Input Parameters for Data Repair

In constraint-based cleaning, data dependencies are used to detect data quality problems [20]. In quantitative data cleaning, distributional models are used and values that deviate from a distribution (outliers) are considered to be errors [19]. Data cleaning, or repair, is typically done by minimizing the number or cost of changes needed to create a consistent database [11, 21], or by finding a repair whose distribution is statistically close to the original data [28]. There are two main aspects that must be taken into consideration when evaluating constraint-based data repairing systems: the role of constraints and the role of the data.

Constraint Parameters. Different repair algorithms have been developed for different fragments of first-order logic. While the most popular are FDs and conditional functional dependencies (CFDs), lately there have been some proposals to handle also denial constraints [14]. Of course, constraint languages differ in expressive power which leads to different sets of rules and ultimately to different repairs.

Another parameter to consider is the number of constraints, or rules, in a cleaning scenario. A larger number of constraints usually leads to a better repair, as more external information, expressed in the rules, is enforced over the dirty data. However, a larger number of constraints also leads to a higher execution time in the detection and repair process.

Data Parameters. While the role of constraints is quite evident for data cleaning, a more subtle but equally important role is played by the features of the data, and especially of the errors. Recently, BART[4] [6] identified two important properties of errors: *detectability* and *repairability*.

When evaluating a constraint-based repair algorithm, we want to make sure that the dirty input database used in the evaluation only contains errors that are detectable by the system in question. After all, an error that cannot be detected, cannot be repaired. To reason about detectability, we need a notion for determining whether a cell change is involved in a constraint violation. This notion assumes the existence of a clean gold standard database and a cell change is assumed to describe a difference between the gold standard and the dirty database.

Consider our database in Figure 2. Assume now that the dirty cell (in bold) has been restored to its original, clean value ("Juventus Stadium"), i.e., we have a clean database and want to introduce errors in it. To start,

[4]Bart: Benchmarking Algorithms for data Repairing and Translation

consider the following cell change: $\mathsf{ch}_1 = \langle t_1.\mathsf{Season} := \mathsf{2012\text{-}13} \rangle$ that updates tuple t_1 as follows:

Player	Name	Season	Team	Stadium	Goals
t_1 :	Giovinco	**2012-13**	Juventus	Juventus Stadium	3

This change does not introduce a violation to any of the constraints in our example. Therefore, any data-repairing tool that relies on the constraints to detect dirtiness in the database will not be able to identify it. We call this an *undetectable change*.

When we introduce errors into clean data for the purpose of benchmarking, it is important to control the number and the behavior of the errors, but it is hard to control the exact number of errors that are guaranteed to be detectable using a given set of constraints. In fact, this requirement turns the complexity of the error-generation process into an NP-complete problem [6].

Once an error has been detected, the second step in the cleaning process is to repair it. Of course, some errors are easier to repair than other. Back to our example, a change that indeed introduces a detectable error is the following: $\mathsf{ch}_2 = \langle t_1.\mathsf{Season} := \mathsf{2014\text{-}15} \rangle$. After this update, tuples t_1 and t_2 violate FD d_1, which states that Name and Season are a key for the table:

Player	Name	Season	Team	Stadium	Goals
t_1 :	Giovinco	**2014-15**	Juventus	Juventus Stadium	3
t_2 :	Giovinco	2014-15	Toronto	BMO Field	23

This change is easily detected using the constraints. Still, it is quite difficult for an automatic data-repairing algorithm to restore the database to its clean state. Notice, in fact, that after this change, the original value 2013-14 has been removed from the active domain of the dirty database. There is no evidence in the dataset to guide an algorithm to guess the correct value for a repair. Therefore, a correct repair cannot be found by any repair algorithm that uses the values in the database as the candidates for repair.

BART uses the notion of *repairability* of an error to characterize this aspect. In the case above, it would assign repairability 0 to change ch_2. Different detectable changes may have quite different repairability values. As an example, consider now change $\mathsf{ch}_3 = \langle t_1.\mathsf{Stadium} := \mathsf{Delle\ Alpi} \rangle$. The change is detectable using FD d_2. The redundancy in the example dirty database may be used to repair it:

Player	Name	Season	Team	Stadium	Goals
t_1 :	Giovinco	2013-14	Juventus	**Delle Alpi**	3
t_3 :	Pirlo	2014-15	Juventus	Juventus Stadium	5
t_5 :	Vidal	2014-15	Juventus	Juventus Stadium	8

The new dirty tuple t_1 is involved in two violations of d_2, one with t_3, another with t_5. In both cases, the new stadium value Delle Alpi is in conflict with value Juventus Stadium. By a straightforward probabilistic argument, BART would calculate a $2/3$ repairability for this error, and rank it as a medium-repairability error.

In other cases, errors may have higher repairability, even 1 in some cases. Consider, for example, the case in which an additional CFD states that every person with age 40 is named Anne. Since this knowledge is part of the constraint, any tool would easily restore a dirty age value ($\neq 40$) for a person named Anne to its gold standard, clean state.

4 Benchmarking

We consider how modern data and metadata generators provide flexible control over the values of some of the input parameters we have identified so each can be used as an independent variable in system evaluations. We focus on the BART error generator that can be used to evaluate data repair systems and the iBench metadata generator that can be used to evaluate data exchange and mapping operators.

4.1 Data Repair: BART

BART is an open-source system that introduces algorithms that guarantee a compromise between control over the nature of errors and scalability [6]. BART introduces a new computational framework based on *violation-generation queries* for finding candidate cells (tuple, attribute pairs) into which detectable errors can be introduced. While these queries can be answered efficiently, determining if detectable errors can be introduced is computationally hard. To overcome this issue, optimizations for violation-generation queries are introduced. In particular, extracting tuple samples, along with computing cross-products and joins in main memory, brings considerable benefits in terms of scalability. Moreover, the authors identify a fragment of denial constraints (DCs) called *symmetric constraints* that considerably extend previous fragments for which scalable detection techniques have been studied. The main benefit of this subclass it that algorithms for detecting and generating errors with symmetric constraints have significantly better performance than the ones based on joins and allow the introduction of controllable errors over large datasets.

The Error-Generation Problem. BART permits users to declarative specify how to introduce errors into a clean dataset for benchmarking purposes. The input of the tool is an *error-generation task* $\mathbf{E}$, which is composed of four key elements: (i) a database schema S; (ii) a set Σ of DCs encoding data quality rules over S; (iii) an instance I of S that is clean with respect to Σ; and (iv) a set of configuration parameters to control the error-generation process. These parameters specify, among other things, which relations can be changed, how many errors should be introduced, and how many of these errors should be detectable. They also let the user control the degree of repairability of the errors.

Use Cases. BART supports several uses cases. The main one consists of generating a desired degree of detectable errors for each constraint. In addition, users may specify a range of repairability values for each constraint. In this case, BART estimates the repairability of changes, and only generates errors with estimated repairability within that range. In addition to detectable errors, BART can generate random errors of several kinds: *typos* (e.g., 'databse'), *duplicated values*, *bogus* or *null values* (e.g., '999', '***'). Random errors may be freely mixed with constraint-induced ones. Finally, BART can introduce outliers in numerical attributes.

4.2 Data Transformation: iBench

iBench [5] is an open-source system that supports the creation of metadata for a variety of integration tasks including but not limited to data exchange, data integration, and mapping operators. As a metadata generator, iBench can be used to generate independent schemas with an arbitrary or controlled set of mappings between them. The user has at his disposal, the ability to control over thirty distinct metadata dimensions. We now overview how iBench advances the state-of-the-art in benchmarking data-transformation systems.

The Metadata-Generation Problem. Intuitively, iBench takes a *metadata-generation task* Γ and produces an *integration scenario* that fulfills the task. Here an integration scenario is a tuple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \Sigma, I, J, \mathcal{T})$, where $\mathbf{S}$ and $\mathbf{T}$ are schemas, $\Sigma_{\mathbf{S}}$ and $\Sigma_{\mathbf{T}}$ are source and target constraints, Σ is a mapping between $\mathbf{S}$ and $\mathbf{T}$, I is an instance of $\mathbf{S}$ satisfying $\Sigma_{\mathbf{S}}$, J is an instance of $\mathbf{T}$ satisfying $\Sigma_{\mathbf{T}}$, and $\mathcal{T}$ is a program that implements the mapping Σ. A user writes a metadata-generation task (or configuration) Γ by specifying minimum and maximum values for a set Π of input parameters. Note that iBench users do not need to specify every input parameter, rather only the ones they want to control.

For example, a user may request an integration scenario with independent schemas of up to five attributes per relation and with only LAV mappings.[5] To do this, he creates a simple metadata-generation task, specifying that the input parameters $\pi_{SourceRelSize}$ (number of attributes per source relation) and $\pi_{TargetRelSize}$

[5]Recall that LAV mappings have a single relation atom in the source [29].

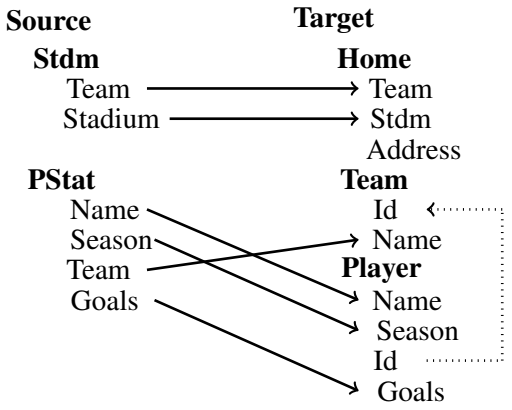


Figure 5: A Sample Integration Scenario Output by iBench

(number of attributes per target relation) be between two and five, and that the input parameter $\pi_{SourceMapSize}$ (number of source atoms per mapping) be exactly one. We show in Figure 5 an integration scenario fulfilling these requirements. Note that both the source and target relations have up to five attributes. The black solid lines in the figure represent mapping correspondences (variables that are shared between the source and the target), and the two LAV mappings being output here are as follows: one takes a source relation `Stdm(Team,Stadium)` and copies it to a target relation `Home(Team,Stdm,Address)`, and another mapping takes a source relation `PStat(Name,Season,Team,Goals)` and vertically partitions it into two target relations, `Team(Id,Name)` and `Player(Name,Season,Id,Goals)`.

iBench supports two kinds of input parameters, *scenario parameters* that help control the characteristics of integration scenarios to generate arbitrary independent metadata, and *primitive parameters* that help control the precise relationship between the source and target schemas. As shown in our example above, typical scenario parameters include controlling the schema size and complexity, the number of mappings and the complexity of the mapping language (from using ST TGDS to richer second-order TGDS [17] which are useful in evaluating integration tasks like mapping composition), and the amount and type of constraints per relation. Notice for example that in Figure 5, the second mapping creates two target keys for `Team` and `Player`, and a foreign key between them. Primitive parameters, on the contrary, act over individual mapping primitives where each mapping primitive is a parameterized integration scenario encoding a common transformation pattern (e.g., vertical or horizontal partitioning). Using primitive parameters, for instance, a user can constrain these scenarios to use a particular type of join (i.e., Star or Chain), or use a different number of partitions when constructing the mapping expressions.

Mapping generation, with the level of control provided by iBench, is computationally hard [5], and moreover there may be metadata tasks for which no solution exists. iBench employs a holistic approach in examining the input requirements, and may choose to relax some scenario parameters to produce a solution. Suppose a user requests an integration scenario with source relations with exactly two attributes (scenario parameter) and requests the use of a vertical-partitioning primitive that partitions a source relation into three fragments (primitive parameter). This task has conflicting requirements, as in order to create a target mapping expression with three fragments we need to have a source relation with at least three attributes. In this case, iBench's best-effort algorithm chooses to obey the restriction on number of partitions and violate the restriction on source relation size, that is, in the presence of conflicts, primitive parameters have precedence over scenario parameters. Still any output generated by iBench is guaranteed to be a correct solution with respect to the relaxed constraints.

Use Cases. iBench supports several use cases. The first, main use case deals with generating integration scenarios that have independent schemas with random mappings and constraints. iBench can generate arbitrary

constraints such as FDs (including keys) and inclusion dependencies (including foreign keys), and the user can easily specify the percentage of constraints per relation, as well as the size of keys, for example. In the second use case, a user can request the generation of primitive-based parameterized scenarios. Notice that these scenarios can be used as a gold standard. Also, using a combination of scenario and primitive input parameters, a user can easily ask for a mix of independent and primitive-based scenarios. This allows for the creation of scenarios with some redundancy. By using metadata sharing, the third use case, we can use iBench to create even more realistic and complex scenarios where the degree of source and target sharing among mappings can be also controlled. Notice that in practice, most integration scenarios exhibit mappings that reuse source or target relations. A fourth use case allows users to import real-world integration scenarios (i.e., schema, mappings, data) into iBench, and systematically vary and scale them along with any other requested metadata. This feature is crucial for evaluating systems that exploit very specific transformation patterns [5]. The main innovation here has been to view the characteristics of metadata as independent variables which can be systematically controlled (via the input parameters of iBench) to generate flexible, diverse metadata in a fast, scalable way.

5 Success Stories

We now discuss some successful evaluations using the iBench and BART systems and the quality measures introduced in Section 2, focusing specifically on how these approaches enabled these evaluations.

5.1 Measuring Success Rates of Mapping Translation in Data Exchange

Arocena et al. [7] proposed an approach for rewriting second-order TGDS into equivalent first-order mappings (ST TGDS or nested mappings). Testing whether a second-order TGD is equivalent to an ST TGD or nested mapping is undecidable [18], hence this approach is correct, but not complete, meaning it may not be able to rewrite the second-order input mapping even if an equivalent first-order mapping exists. Given this incompleteness result it was important to evaluate the algorithm's success rate over a broad range of realistic mappings and compare this rewriting algorithm to alternative approaches (such as an earlier rewriting technique by Nash et al. [27]). The evaluation goal here was to answer the question: "How often do these algorithms succeed in practice?". iBench was used to generate schemas, schema constraints (including keys and FDs), and schema mappings expressed as second-order TGDS. The iBench system enabled the systematic (and efficient) generation of a large number of diverse mappings (over 12 million), in which the degree and complexity of incompleteness (i.e., the Skolem functions used for modeling value invention) could be controlled.

This comprehensive evaluation would not have been possible without using an efficient metadata generator like iBench that provides control over both mapping and schema characteristics. A small collection of real-world scenarios would have not been representative enough to show the differences in algorithms. Relying on a few synthetic scenarios would have not been realistic enough. These experiments depended on the following specific features of iBench: efficient schema and mapping generation (both in terms of computational resources and in terms of user effort where the user only needs to set a few configuration parameters), the ability to generate second-order TGD mappings, support for varying the type and amount of incompleteness in mappings, and varying the type and amount of schema constraints.

5.2 Evaluating the Quality per Effort Ratio of Data Transformation Systems

The IQ metric for transformation quality and user effort was used to answer the question "how much user effort is it required with a particular system to reach a certain data-transformation quality?". In this evaluation [25], the IQ metric was essential because it enabled two very important results: 1) a level comparison of user effort among systems that use very diverse means of user interaction to create a data transformation task (e.g., a mapping-based system may have a GUI that focuses on the task of schema matching and mapping generation while the

actual data transformation step is automatic once a mapping has been designed, while an ETL tool focuses on building a workflow out of basic data transformation steps such as surrogate key generation); and 2) a fair comparison of output quality by comparing the transformed data to a gold standard. An important property of the IQ quality measure is that it measures the quality of the final output (target instance) instead of the generated mappings. This makes the measure applicable for comparing data-transformation systems as diverse as ETL tools and data exchange systems.

5.3 Evaluating the Effect of Repairability on Data Cleaning Quality

BART was used to evaluate how the repairability of a dirty instance affects the quality of the repairs produced by data-cleaning systems [6]. Though preliminary, this evaluation demonstrated that different algorithms show different trends with respect to how repairability affects quality. BART was essential to this evaluation because it enabled the generation of dirty instances with a guaranteed number of errors that are detectable by the constraints, while at the same time controlling how hard these errors are to repair (repairability). BART helped greatly reduce the amount of effort needed to generate the multitude of dirty versions of a dataset. Creating several dirty versions of a clean dataset amounted to just changing a few configuration parameters. Importantly, BART was designed with performance in mind. The actual error-generation process is highly efficient and, thus, it is feasible to generate many large dirty instances for an experiment within very reasonable time.

6 Conclusion and Future Work

We have discussed some of the important challenges in evaluating data curation tasks where both the performance of a system and the accuracy (or quality) of the curation result must be considered. We have presented some of the important input parameters (the independent variables in an evaluation) that have been identified, along with accuracy measures. We motivated the need for data and metadata generators that can efficiently produce suites of input data or metadata for curation systems conforming to specific settings of the input parameters. We focused our discussion on two curation tasks, data exchange and data repair, and discussed the state-of-the-art in evaluating these important tasks. Much remains to be done in understanding the research challenges inherent in evaluating these and other data curation tasks. We feel this is an area that is ripe for innovation. As data curation lies at the heart of data science, we need evaluation standards and tools that inspire confidence in our solutions and drive the field forward.

References

[1] Z. Abedjan, L. Golab, and F. Naumann. Profiling Relational Data: A Survey. *The VLDB Journal*, 24(4):557–581, 2015.

[2] B. Alexe, M. A. Hernández, L. Popa, and W.-C. Tan. MapMerge: Correlating Independent Schema Mappings. *The VLDB Journal*, 21(2):191–211, 2012.

[3] B. Alexe, W. Tan, and Y. Velegrakis. Comparing and Evaluating Mapping Systems with STBenchmark. *PVLDB*, 1(2):1468–1471, 2008.

[4] P. Andritsos, R. J. Miller, and P. Tsaparas. Information-Theoretic Tools for Mining Database Structure from Large Data Sets. In *SIGMOD*, pages 731–742, 2004.

[5] P. C. Arocena, B. Glavic, R. Ciucanu, and R. J. Miller. The iBench Integration Metadata Generator. *PVLDB*, 9(3):108–119, 2015.

[6] P. C. Arocena, B. Glavic, G. Mecca, R. J. Miller, P. Papotti, and D. Santoro. Messing-Up with BART: Error Generation for Evaluating Data Cleaning Algorithms. *PVLDB*, 9(2):36–47, 2015.

[7] P. C. Arocena, B. Glavic, and R. J. Miller. Value Invention in Data Exchange. In *SIGMOD*, pages 157–168, 2013.

[8] Z. Bellahsene, A. Bonifati, F. Duchateau, and Y. Velegrakis. On Evaluating Schema Matching and Mapping. In *Schema Matching and Mapping*, pages 253–291. Springer, 2011.

[9] P. A. Bernstein, T. J. Green, S. Melnik, and A. Nash. Implementing Mapping Composition. *The VLDB Journal*, 17(2):333–353, 2008.

[10] L. Bertossi. *Database Repairing and Consistent Query Answering*. Morgan & Claypool, 2011.

[11] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification. In *SIGMOD*, pages 143–154, 2005.

[12] A. Bonifati, G. Mecca, A. Pappalardo, S. Raunich, and G. Summa. The Spicy System: Towards a Notion of Mapping Quality. In *SIGMOD Conference*, pages 1289–1294, 2008.

[13] P. Buneman, J. Cheney, W.-C. Tan, and S. Vansummeren. Curated Databases. In *PODS*, pages 1–12, 2008.

[14] X. Chu, I. F. Ilyas, and P. Papotti. Discovering Denial Constraints. *PVLDB*, 6(13):1498–1509, 2013.

[15] R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *TCS*, 336(1):89–124, 2005.

[16] R. Fagin, P. Kolaitis, and L. Popa. Data Exchange: Getting to the Core. *ACM Transactions on Database Systems*, 30(1):174–210, 2005.

[17] R. Fagin, P. Kolaitis, L. Popa, and W. Tan. Composing Schema Mappings: Second-Order Dependencies to the Rescue. *ACM Transactions on Database Systems*, 30(4):994–1055, 2005.

[18] I. Feinerer, R. Pichler, E. Sallinger, and V. Savenkov. On the Undecidability of the Equivalence of Second-Order Tuple Generating Dependencies. *Information Systems*, 48:113–129, 2015.

[19] J. Hellerstein. Quantitative Data Cleaning for Large Databases. In *Technical report, UC Berkeley*, Feb 2008.

[20] I. F. Ilyas and X. Chu. Trends in Cleaning Relational Data: Consistency and Deduplication. *Foundations and Trends in Databases*, 5(4):281–393, 2015.

[21] S. Kolahi and L. V. S. Lakshmanan. On Approximating Optimum Repairs for Functional Dependency Violations. In *ICDT*, pages 53–62, 2009.

[22] S. Kolahi and L. Libkin. An Information-Theoretic Analysis of Worst-Case Redundancy in Database Design. *ACM Transactions on Database Systems*, 35(1), 2010.

[23] S. Kruse, P. Papotti, and F. Naumann. Estimating Data Integration and Cleaning Effort. In *EDBT*, pages 61–72, 2015.

[24] B. Marnette, G. Mecca, and P. Papotti. Scalable Data Exchange with Functional Dependencies. *PVLDB*, 3(1):105–116, 2010.

[25] G. Mecca, P. Papotti, S. Raunich, and D. Santoro. What is the IQ of your Data Transformation System? In *CIKM*, pages 872–881, 2012.

[26] M. Meila. Comparing Clusterings: an Axiomatic View. In *ICML*, pages 577–584, 2005.

[27] A. Nash, P. A. Bernstein, and S. Melnik. Composition of Mappings Given by Embedded Dependencies. *TODS*, 32(1):4, 2007.

[28] N. Prokoshyna, J. Szlichta, F. Chiang, R. J. Miller, and D. Srivastava. Combining Quantitative and Logical Data Cleaning. *PVLDB*, 9(4):300–311, 2015.

[29] B. ten Cate and P. G. Kolaitis. Structural Characterizations of Schema-Mapping Languages. In *ICDT*, pages 63–72, 2009.

[30] T. Vogel, A. Heise, U. Draisbach, D. Lange, and F. Naumann. Reach for Gold: An Annealing Standard to Evaluate Duplicate Detection results. *J. Data and Information Quality*, 5(1-2):5:1–5:25, 2014.

[31] C. Yu and L. Popa. Semantic Adaptation of Schema Mappings when Schemas Evolve. In *VLDB Conference*, pages 1006–1017, 2005.

Exploring What not to Clean in Urban Data: A Study Using New York City Taxi Trips

Juliana Freire Aline Bessa Fernando Chirigati Huy Vo Kai Zhao
New York University

Abstract

Traditionally, data cleaning has been performed as a pre-processing task: after all data are selected for a study (or application), they are cleaned and loaded into a database or data warehouse. In this paper, we argue that data cleaning should be an integral part of data exploration. Especially for complex, spatio-temporal data, it is only by exploring a dataset that one can discover which constraints should be checked. In addition, in many instances, seemingly erroneous data may actually reflect interesting features. Distinguishing a feature from a data quality issue requires detailed analyses which often includes bringing in new datasets. We present a series of case studies using the NYC taxi data that illustrate data cleaning challenges that arise for spatial-temporal urban data and suggest methodologies to address these challenges.

1 Introduction

Cities are the loci of resource consumption, of economic activity, and of innovation; they are also the cause of our looming sustainability problems and where those problems must be solved. Our increasing ability to collect, transmit, and store data, coupled with the growing trend towards openness [4, 14, 17, 23, 24, 34], creates an opportunity to leverage these data and make cities more productive, livable, equitable, and resilient. *Urban data* is unique in that it captures the behavior of the different components of a city, namely its citizens, existing infrastructure (physical and policies), the environment (e.g., weather), and interactions between these elements [18]. To understand a city and how its multiple elements interact, intricate analyses are necessary. As in any data analysis process, data cleaning is of crucial importance to bring data from a "messy" to a "tidy" state [40].

Data cleaning may be achieved through a multitude of methods, including filtering operations, statistical analysis, outlier detection, and missing value imputation. Traditionally, this is performed as a pre-processing step: a function $DirtyData \rightarrow CleanData$. We argue that data cleaning must be an integral part of the inherently iterative data analysis cycle: data cleaning must be applied on the fly. While exploring a new dataset, constraints that should be checked in the cleaning function, and which might not be evident at first, are naturally discovered. Consider Figure 8, which shows visualizations of NYC taxi trips on a map. These elicit the fact that the data contain pickups and dropoffs inside the rivers and in the ocean. Since there are no amphibious taxis, these represent erroneous data. This finding suggests the creation of a rule that checks whether the GPS coordinates for the trips are within polygons that lie on land.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

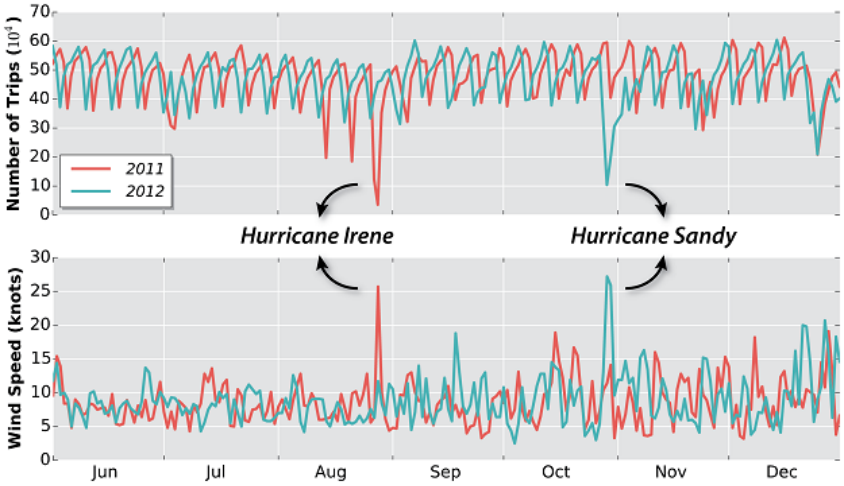


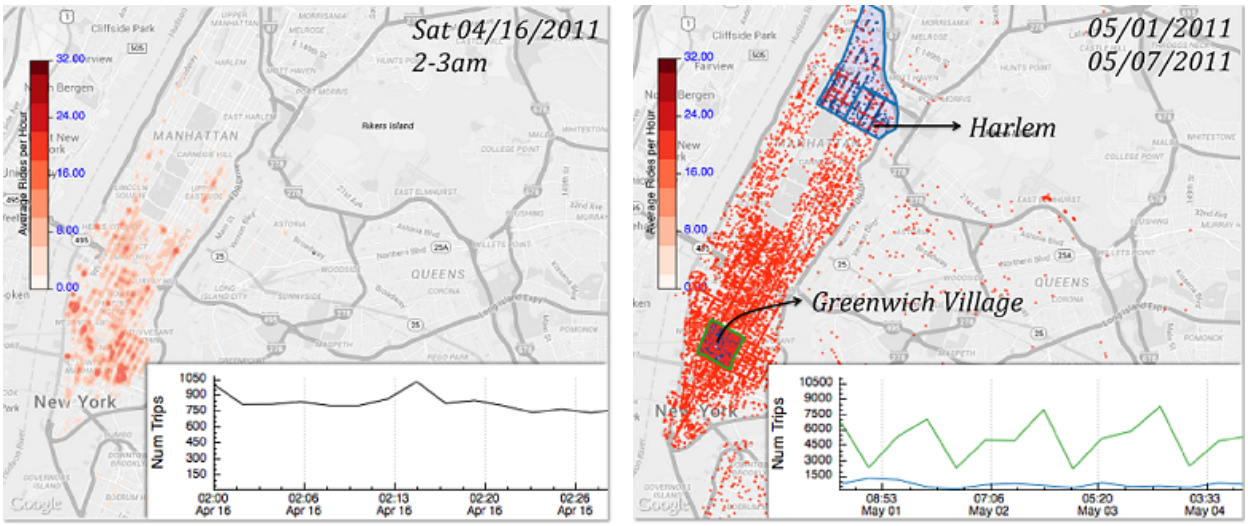
Figure 1: The plot on the top shows how the number of trips varies over 2011 and 2012. While the variation is similar for the two years, there are clear outliers, including large drops in August 2011 and in October 2012. However, examining the variation in wind speed during the same period (plot on the bottom), we can observe an inverse correlation: the large drops in the number of trips happened when the wind speeds were abnormally high. In fact, these correspond to two hurricanes: Irene and Sandy.

Besides the need to refine a cleaning function as the user gets more familiar with a dataset, different questions that arise during exploration may require different cleaning strategies. Thus, we need a function $DirtyData \times UserTask \rightarrow (CleanData, Explanation)$. For example, to create machine learning prediction models, cleaning steps are often applied to remove outliers from historical data. In contrast, to understand specific behaviors (or events), outliers are actually the central objects of the study.

As domain experts explore urban data corpora and consider different datasets, hypotheses are formulated and tested, and interactions among the different components of a city are untangled. In this process, when new datasets are brought into the investigation, seemingly erroneous data points identified when a dataset is analyzed in isolation may actually uncover features that explain important phenomena. Consider the top plot in Figure 1, which shows the number of daily taxi trips in New York City (NYC) during 2011 and 2012. While the distribution of trips over time is very similar across the two years, we observe large drops in August 2011 and October 2012. Standard cleaning techniques are likely to classify these drastic reductions as outliers that represent corrupted or incorrect data. However, by integrating taxi trips with the wind speed data (bottom plot in Figure 1), we discover that the drops occur on days with abnormally high wind speeds, suggesting a causal relation: the effect of extreme weather on the number of taxi trips in NYC. Removing such outliers would hide an important phenomenon.

Answering the question *"Is it dirt or a feature?"* is challenging because experts often need to go beyond a single dataset to seek explanations. Given the plethora of components interacting in urban environments, the integration possibilities for the data exhaust produced by these components are endless. As a point of reference, in the past two years, NYC has published over 1,300 datasets [24] and the city of Chicago has made available around 1,000 datasets [23], and these datasets represent just a small fraction of the data being collected by these cities [4].

In addition to the complex interactions among different datasets, the nature of urban data poses further challenges. First, metadata is limited or inexistent. Many of the datasets are derived from spreadsheets and contain incomplete schema information. Integrity constraints are not provided and type information often needs to be inferred [7]. Second, because urban data is often *spatio-temporal* [4, 7], there are many data slices to consider. For instance, NYC taxi trips are associated with GPS coordinates with time precision in seconds.



(a) (b)

Figure 2: (a) Heatmap of pickups on a Saturday uncovers popular nightspots. (b) Comparison between pickups in Harlem (blue time series) and Greenwich Village (green time series) shows that Harlem is underserved by taxis.

Consequently, such data can be aggregated into different spatial resolutions (e.g., neighborhoods, zip codes, and boroughs) and temporal resolutions (e.g., hourly, daily, weekly, and monthly) during the analysis process. Depending on the resolution, dirty data may become easily identifiable or completely hidden. For instance, missing data along an avenue over the course of an hour can be easily detected when looking at a finer scale (e.g., hourly and zip codes), whereas coarser resolutions (e.g., daily and boroughs) may hide these issues depending on the aggregation function used (see Figure 3). Spatio-temporal patterns present additional challenges to cleaning: data that may be detected as dirty in a specific time period or spatial region may in fact constitute a pattern in space and time. For example, as shown in Figure 1, there are significant drops in the number of trips on Christmas and New Year's day: these are not dirty data but recurring, yearly patterns.

The complexity of urban data coupled with the sheer number of available datasets and their numerous possible interactions make it hard to pinpoint what is an error and what is a feature. In this paper, we discuss challenges involved in cleaning spatio-temporal urban data. We use the New York City Taxi data obtained through a FOIL request[1] and present a series of case studies that illustrate different problems that arise. We also describe techniques that can aid in exploratory data cleaning and outline directions for future research.

2 The New York City Taxi Data

In New York City, every day there are over 500,000 taxi trips serving roughly 600,000 people [6]. Through the meters installed in each vehicle, the Taxi & Limousine Commission (TLC) captures detailed information about trips. Each trip consists of two spatial attributes (GPS readings for pickup and dropoff locations), two temporal attributes (pickup and dropoff times), and additional attributes including taxi identifier, distance traveled, fare, medallion code, and tip amount.

Taxis are unsuspecting sensors of urban life and their data exhaust can help uncover characteristics of the city that are of economic and social importance [12]. For example, by examining patterns involving taxi pickups and dropoffs in NYC at different times, we can discover popular destinations (e.g., popular night spots) and neighborhoods that are underserved by taxis (see Figure 2). In addition, we can discover various events (or

[1] A new version of this dataset was recently released by the Taxi & Limousine Commission [37].

exceptions), such as road closures and hurricanes, and their effect on traffic [10]. It is also possible to identify functional regions, such as tourist attractions, shopping centers, workplaces, and residential places, based on urban human mobility patterns [27, 41].

Taxi data are also valuable in that they can be used to derive other types of data. For example, traffic speed and direction information can be derived from taxi data, helping deal with the sparsity of speed sensors which cover only a limited number of road segments in Manhattan [26]. Similarly, the concentration of $PM_{2.5}$ in a city, which is a metric for air quality, can be inferred from traffic flow in locations where there is a limited number of air-quality monitor stations [43]. Another important use for these data is in the analysis of what-if scenarios. Savage and Vo [31] showed that, even though NYC residents are dissatisfied with taxi availability during rush hour, increasing the number of taxis in this period would lead to congestion and a significant reduction in traffic speed. Ota et al. [25] developed a real-time, data-driven simulation framework that uses historical trip data to supports the efficient analysis of taxi ride sharing scenarios.

2.1 A Quick Look into the Taxi Data

Given the wide range of applications that are enabled by the taxi data, understanding its quality is of utmost importance. Here, we focus on the taxi datasets collected from 2008 to 2012. Table 1 summarizes a few statistics associated with these datasets. Note that, while fare information is available for trips paid either by cash or by credit card, tip information is only available for trips paid by credit card and, as a consequence, reported tip statistics ("Tip Amount" column in Table 1) refer exclusively to credit card trips.

The averages reported show that taxi trips often last less than 17 minutes, cover less than 7 miles, and cost about US$10.00. But there are exceptions which represent potential data quality issues. We computed the average fare values for trips that were exclusively paid by credit cards: US$12.56 for 2008, US$10.81 for 2009, US$11.12 for 2010, US$11.20 for 2011, and US$11.84 for 2012. By crossing these values with tip values on Table 1, we observe that, for most years, passengers who paid with credit cards gave tips of about 20% of the corresponding fare amounts, which is consistent with the customary tip values for other services in New York City. In contrast, in 2008 and 2009, the average tip amount is much lower: 0.7% and 3.5%, respectively. This may have been an error in the way the data were reported. The issue seems to have been resolved by the end of 2009, when the average credit card tip increased to US$2.04.

If we turn our attention to trip duration, distance, and fares, there are clear discrepancies: average trip durations were significantly smaller for 2009 and 2010, even though their corresponding average distances were roughly twice as long as those of 2011 and 2012; the average trip duration for 2008 was significantly higher than those recorded for 2011 and 2012; the average fare in 2009 is much lower than in other years; and the average fare for 2008, US$0.09, is too low.[2] These suggest quality issues in the data that require further investigation.

The table also shows the presence of invalid, negative values in 2010. These are clearly wrong and should be removed during cleaning: the negative values do not carry useful semantics—there is no such thing as negative miles. On the other hand, the decision is not so clear cut for other, positive values. An example is the tip of US$938.02 (maximum credit card tip value for the 2010 dataset). While this could be an error in data acquisition or in the credit card information, it could also be the case that a wealthy passenger overtipped the taxi driver (e.g., a financier that had just made a big profit). Given that negative values are only found in 2010, it may be the case that data for the other years were cleaned and had negative values removed, or that some improvement was made in the way data was transferred from taxis to servers. This underscores the importance of including provenance information [13], detailing the cleaning operations applied to released datasets. Such provenance is essential to ensure that analyses across the different datasets are consistent.

[2] Since the average fare for credit card trips only in 2008 is US$12.56, the low overall average may indicate that cash payments were not properly reported by drivers.

Dataset	Statistic	Trip Duration (min)	Trip Distance (mi)	Fare Amount (US$)	Tip Amount (US$)
2008	Min	0.00	0.00	0.00	0.00
	Avg	16.74	2.71	0.09	0.10
	Max	1440.00	50.00	10.00	8.75
2009	Min	0.00	0.00	2.50	0.00
	Avg	7.75	6.22	6.04	0.38
	Max	180.00	180.00	200.00	200.00
2010	Min	-1,760.00	-21,474,834.00	-21,474,808.00	-1,677,720.10
	Avg	6.76	5.89	9.84	2.11
	Max	1,322.00	16,201,631.40	93,960.07	938.02
2011	Min	0.00	0.00	2.50	0.00
	Avg	12.35	2.80	10.25	2.22
	Max	180.00	100.00	500.00	200.00
2012	Min	0.00	0.00	2.50	0.00
	Avg	12.32	2.88	10.96	2.32
	Max	180.00	100.00	500.00	200.00

Table 1: Statistics for the taxi datasets. Tip amount is available for trips paid by credit card only.

2.2 Exploring Quality Issues in Spatio-Temporal Data

Computing simple statistics over attributes can help uncover potential issues in a dataset. However, in the case of taxi trips, substantial complexity is added to the cleaning process due to the spatio-temporal nature of the data. Manual (exhaustive) exploration is time-consuming and, for large datasets such as the taxi data, it is impractical. For example, temporal aggregation of a years worth of data into a discrete set of hourly intervals results in over 8,000 data slices to be explored.

Recently, techniques and systems have been proposed to streamline and better support exploratory analyses of spatio-temporal data. These include visualization and interaction techniques that allow users to freely explore the data at various levels of aggregation [2, 12, 35, 39] as well as indexing strategies that speed up the computationally expensive point-in-polygon queries required for this type of data [11]. However, effective interaction with spatio-temporal visualizations remains a challenge [15, 28] and, even by using these techniques, domain experts may still need to examine a prohibitively large number of spatio-temporal slices to discover interesting patterns and irregular behaviors, including potential errors in the data. As a step towards addressing this problem, we proposed a scalable technique to automatically discover spatio-temporal events and guide users towards potentially interesting data slices [10] (see Section 3.1 for details). Note that mining for exceptions at different levels of aggregations for relational data has been studied before in the context of OLAP data cubes [29, 30].

While automatic event detection can help steer users to interesting data slices, the user is still faced with the challenge of understanding the events and determining whether they correspond to data quality issues or important features. In [8], we presented the *Data Polygamy* framework, which enables the discovery of relationships between spatio-temporal datasets through their respective events. These relationships provide hints that can help explain the events. The relationship between the number of taxi trips over time and wind speed shown in Figure 1 is one example of a relationship discovered by the Data Polygamy framework.

Techniques that enable users to interactive explore spatio-temporal data, support automatic event detection, and aid in the discovery of relationships among disparate datasets are essential in the discovery (and resolution) of potential data quality issues in spatio-temporal data. In what follows, we present a series of case studies that show how these techniques can help users identify and reason about quality issues in spatio-temporal data.

Case Study	Possible Methodologies
Unusual Spatio-Temporal Behavior	Exploring Different Data Slices Exploring Different Data Resolutions Visualization Event Detection
Taxi Trips and Weather	Combining Multiple Datasets Visualization
Missing Data or Sparsity	Exploring Different Data Slices Exploring Different Data Resolutions Domain Knowledge
Taxis as Sensors	Exploring Different Data Slices Exploring Different Data Resolutions
Speed Computation	Exploring Different Data Slices Exploring Different Data Resolutions Outlier Detection Domain Knowledge
GPS Inaccuracy	Visualization Clustering
Ghost Trips	Domain Knowledge

Table 2: Case studies and possible methodologies for data cleaning.

3 Is It Dirt or a Feature?

In this section, we present case studies that showcase challenges involved in identifying potential quality issues in the NYC taxi data. These case studies demonstrate the importance of exploration and event detection in order to clean data. Table 2 summarizes possible methodologies to address the challenges that arise in each case study. It is worth noting that the decision of whether and how to clean the data depends on the application; such decision is outside the scope of this paper.

3.1 Identifying Unusual Behavior in Spatio-Temporal Data Slices

Consider Figure 3(a), which shows visualizations of the taxi data over four hourly intervals from 7am to 11am, on May 1st, 2011. Note that between 8 and 10am, there are virtually no taxis on 6th avenue between Midtown and Downtown. This anomaly was originally discovered by a user while visually exploring the data using the open-source TaxiVis system [12, 36]. In this case, the anomaly can be easily explained: 6th avenue was closed for the annual NYC 5 Boro Bike Tour.[3] In general, finding explanations for such events is challenging and may require the integration of multiple datasets [5, 8, 9].

To reduce the number of data slices the user has to consider, the usual approach is to apply different types of aggregation and produce visual summaries [1, 20]. These lead to a trade-off between the level of aggregation and the number of data slices to be explored. The use of a coarse (spatial or temporal) aggregation reduces the number of data slices, but it may result in loss of information. If we aggregate the data over time or spatially, as illustrated in Figures 3(b) and 3(c), the Bike Tour event would go unnoticed. Therefore, to detect events and dirty data in urban datasets, data must be analyzed at different granularities. Nevertheless, this is a hard and time-consuming task since it requires the examination of a prohibitively large number of spatio-temporal data slices. As a consequence, methods that automatically discover such events and guide users towards data slices that display interesting events are crucial for data cleaning.

As discussed in Section 2.2, we proposed an approach for automatically detecting events in spatio-temporal data [10]. Event detection is accomplished through the application of topological analysis on a time-varying scalar function derived from the urban data. We use the minima and maxima of a given function to represent

[3] `http://www.nycbikemaps.com/spokes/five-boro-bike-tour-sunday-may-1st-2011`

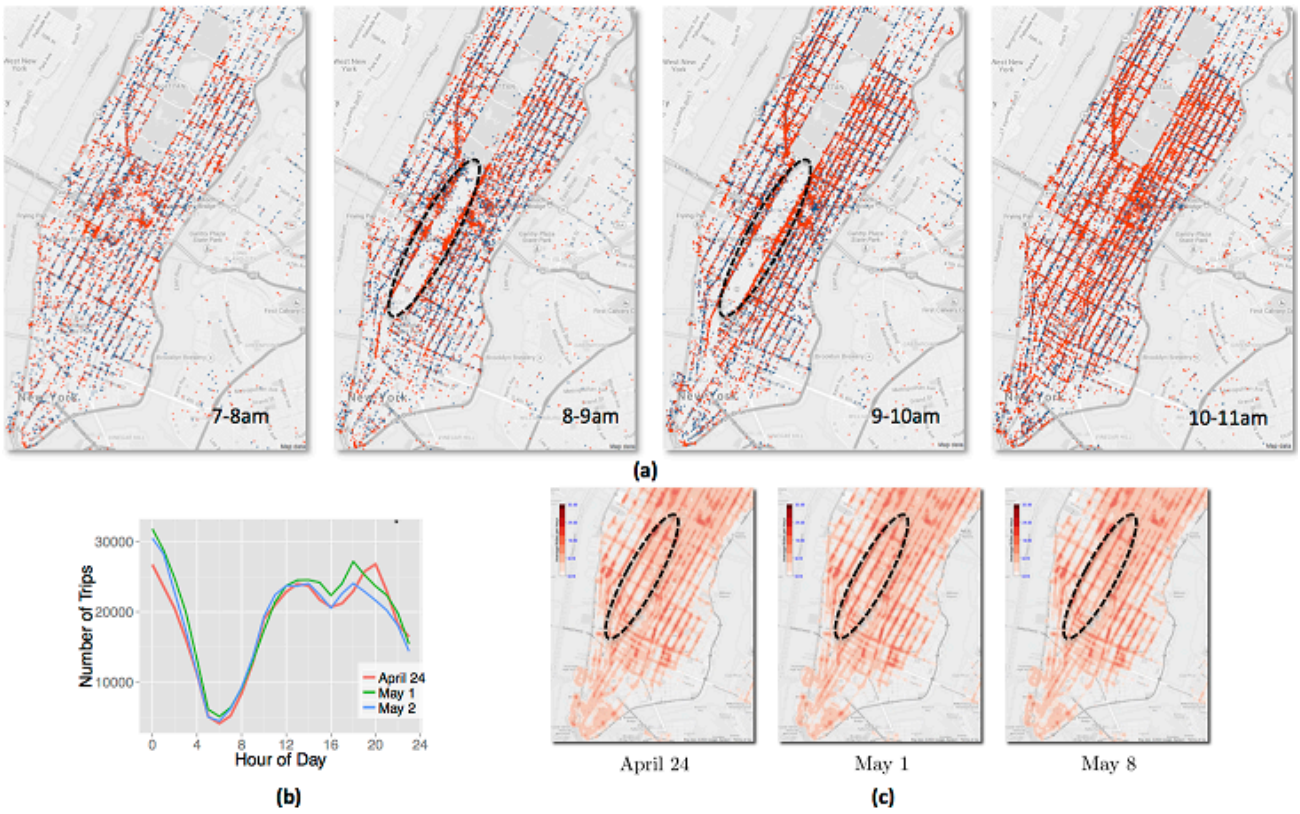


Figure 3: (a) Pickups (blue) and dropoffs (red) in Manhattan on May 1st from 7am to 11am. Notice that from 8am to 10am, there are virtually no trips along 6th Avenue. This avenue was closed to traffic at this time for the annual NYC 5 Boro Bike Tour. (b) The time series plots compare the number of trips that occurred in Manhattan on three Sundays in 2011: 24 April, 1 May, 8 May. It is difficult to distinguish between the three Sundays by using just the number of trips, even though an entire stretch of streets are blocked to traffic on May 1st. (c) The trips are aggregated over time and displayed as a heat map for the three Sundays. Note that the path of the bike tour (highlighted) looks similar in all heat maps.

the events in the data. Intuitively, a minimum (maximum) captures a feature corresponding to a valley (peak) of the data. For example, the lack of taxis along 6th avenue during the bike tour event forms a local minimum and is therefore captured using our technique. The use of topology also allows the detection of events that have an arbitrary spatial structure. In order to support a potentially large number of events, we designed an indexing scheme that groups similar patterns across time slices, thereby allowing for identification of not only periodic events (hourly, daily, and weekly events), but also of events with varying frequency (regular and irregular). Thus, unlike previous approaches that impose a rigid definition of what constitutes an event [3], our technique is flexible and able to capture a wide range of spatio-temporal events. Compared to techniques based on statistical analysis that support different kinds of events, our approach is computationally efficient and scales to large datasets. We also implemented a visual interface designed to aid in event-guided exploration of urban data that integrates event detection and indexing techniques. The interface allows users to interactively query and visualize interesting patterns in the data. We showed that experts applying this framework were able to quickly explain some of the events we found, but they were surprised by others which indicated potential problems they had to investigate. This suggests that our approach, as well as other approaches for event detection over spatio-temporal data, can be very useful for cleaning tasks.

The problem of event detection has been studied by the statistics and machine learning communities [16, 21, 22, 38]. However, the majority of the literature has focused on either purely spatial data or has accounted

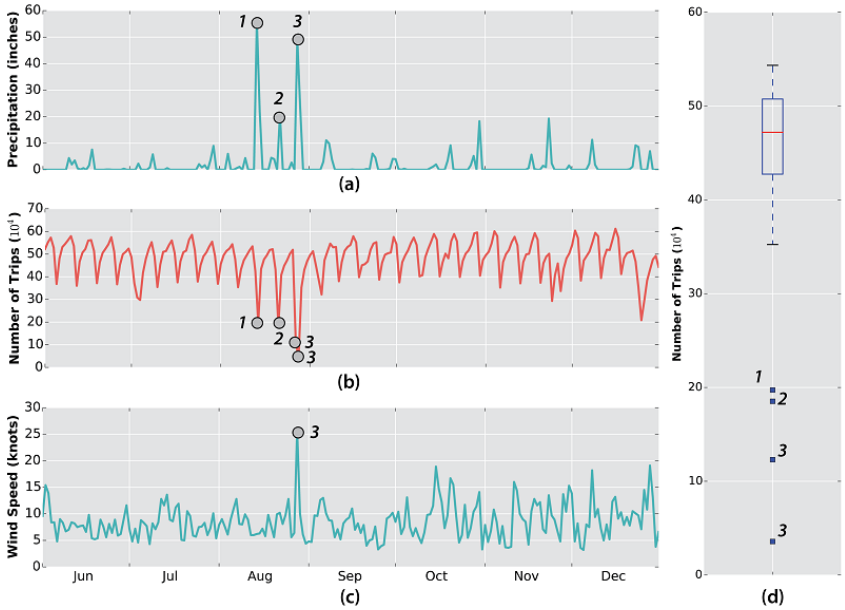


Figure 4: (a) Precipitation, (b) number of taxi trips, and (c) wind speed in a daily basis over the course of 2011. Interesting anomalies, which are detected when using a box plot with taxi data from August (d), are highlighted above, including heavy rainfalls (marked as 1 and 2) and hurricane Irene (marked as 3).

for temporal variations and effects via simplistic approaches such as exponentially weighted linear regression or data partitioning based on day-of-week or season. Furthermore, the time complexity for these approaches is exponential $O(2^N)$ in the number of *pre-defined* space-time partitions. In contrast, the topology-based method has polynomial time complexity [10].

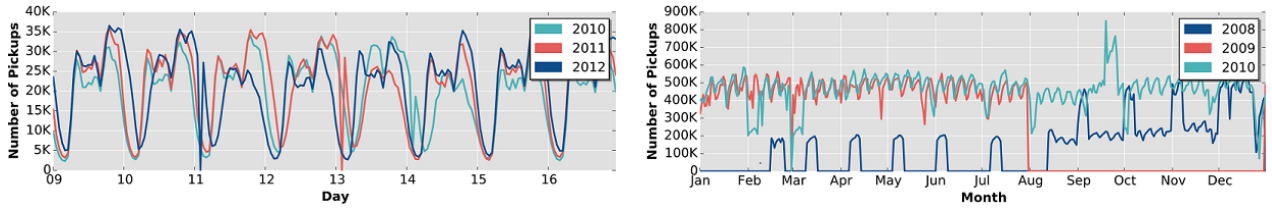
3.2 Combining Multiple Datasets: Taxi Trips and Weather

Consider the plot depicted in Figure 4(b), which shows how the number of taxi trips per day varies over 2011. Note the regularity in the trip distribution over time: the number of trips peaks on Fridays, and bottoms out on Sundays. There are a few exceptions when large drops are observed. Some of these drops can be easily explained, for example, on New Year's Eve and Christmas, which happen every year (see Figure 1). Others, such as the drops in August (labeled 1, 2, and 3), are not clear. By using standard statistical methods such as box plots (Figure 4(d)), these are detected as outliers. Since these points are anomalies, scientists could hypothesize that the data were corrupted on those days (e.g., loss of data when transferring information to servers) and classify these as dirt, removing them from the dataset.

However, further analysis of precipitation (Figure 4(a)) and wind data (Figure 4(c)) shows that these points correspond to different weather events that affected traffic in NYC, including heavy rains (anomalies 1 and 2) and hurricane Irene (anomaly 3). Therefore, anomalies in taxi data are not necessarily a product of dirty data. In these cases, they actually reveal interesting phenomena that demand further exploration.

The question of whether anomalies correspond to dirty data or features, as this example illustrates, may require one to look outside the data and bring additional datasets to the data exploration process. This is a challenging task, especially in the urban data context where a plethora of datasets is available: identifying meaningful connections among thousands of possible datasets is difficult and time-consuming.

Data Polygamy [8] is a scalable topology-based framework that allows users to query for statistically significant relationships and connections across thousands of spatio-temporal datasets. This is accomplished in three steps: (1) each attribute of the two datasets is transformed into a *scalar function*; (2) a topological data



(a) (b)

Figure 5: (a) Comparison between the number of trips in March for different years: on two days, no trips are recorded at 2am. (b) Missing trips observed between 2008 and 2010, and an unusually number of trips at midnight in October 2010.

structure is computed for every scalar function which provides an abstract representation of the peaks and valleys and serves as an index to efficiently identify events; and (3) possible relationships are identified based on the similarity of the events from different scalar functions, and relationships that are statistically significant are returned. The framework not only drastically reduces the number of relationships one needs to analyze, but also uncovers surprising relationships that aids the data analysis process. While Data Polygamy provides insights into which data points are erroneous and which represent important features (or events), other methodologies can also help with this task. For instance, visualizations of the data may elicit connections across datasets (e.g., plots as depicted in Figure 4).

3.3 Missing Data or Sparsity?

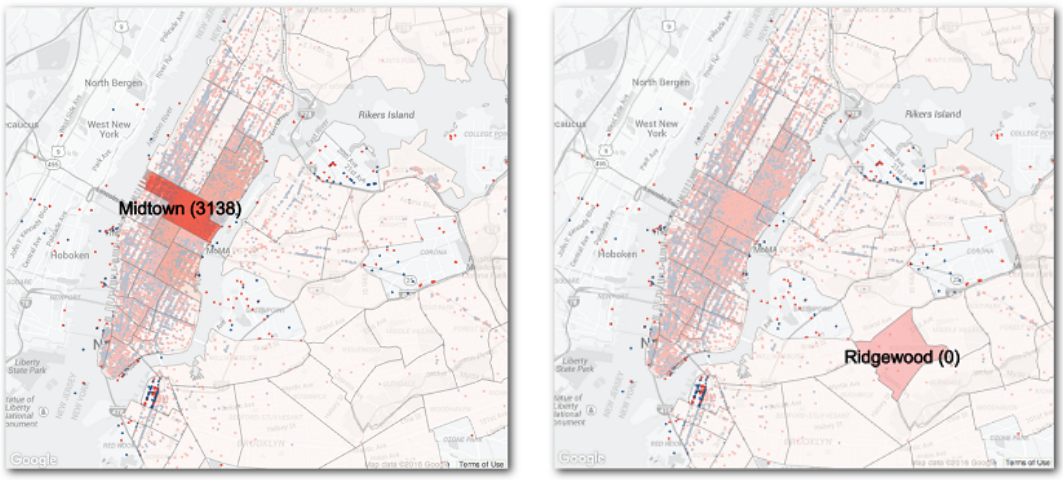
By examining the month of March for different years at an hourly resolution (Figure 5(a)), we observe that there are no trips at 2am on March 13th, 2011 and March 11th, 2012. Also, consider Figure 5(b), which depicts the trip distribution over years between 2008 and 2010. In 2008, we see several periods of missing data, and in 2009, there are no trips between August and December. In 2010, there was a week when the taxi pickups spiked up abnormally: there were 50,000 pickups during one hour at midnight on September 19th, 2010, whereas under normal conditions there are around 10,000 trips per hour. The challenge in this case is to identify whether these situations are due to missing data, data sparsity, or special events.

In Section 3.2, the abnormally small number of trips corresponded to days with extreme weather events. In contrast, the anomalies in Figure 5 are instances of dirty data. The drops in Figure 5(a) are likely due to inconsistencies in how the TLC dealt with Day Light Savings Time, while further examination of the data in Figure 5(b) showed that there is an unusually large number of consecutive and extremely short trips (lasting less than a minute), which cannot happen in practice, indicating that there was an error in data acquisition.

To uncover such issues, we had to first explore different data slices at different granularities: for instance, if data were aggregated by day, the missing data at 2am could not be identified. In addition, the large number of taxi trips observed in Figure 5(b) could be explained by using domain knowledge: we know that taxi trips cannot last less than a minute. This is a data cleaning rule, or a data constraint, that was uncovered *during* data exploration. This shows that the importance of having data cleaning be an integral part of data exploration.

3.4 Coverage: Taxis as Sensors

As discussed in Section 2, many applications have used taxis as sensors to infer new data, including air quality [43] and traffic speed and direction [26]. The quality of the derived data is highly dependent on how much of the city is covered by taxis. The *coverage* of taxis in a city represents the percentage of roads, neighborhoods, or boroughs that is visited by at least one taxi during an hour, a day, or a month. Data for a region is only recorded if that region is visited by a taxi.



(a) (b)

Figure 6: The number of trips in (a) Midtown (Manhattan) and (b) Ridgewood (Brooklyn) on May 2nd (Monday) from 8am to 9am in 2011.

Taxi coverage in a city is often biased. Figures 6(a) and 6(b) depict the number of trips starting in Midtown (Manhattan) and Ridgewood (Brooklyn), respectively, on May 2nd from 8am to 9am in 2011. During that hour (peak time), around 20,690 trips are recorded by over 13,000 taxis in the entire city: while there are 3,138 trips with pickups in Midtown (Figure 6(a)), there are no trips leaving Ridgewood (Figure 6(b)). Out of the 132 neighborhoods in NYC, only 68 neighborhoods are covered by at least one taxi during that hour. The coverage of yellow taxis for NYC is around 51.50%, which means that nearly half of the city is not visited by any taxi. The coverage analysis reveals an instance of *data sparsity*: there is no data for several spatial regions, while there is too much data for others.

Depending on the task, the lack of coverage may severely limit the analysis. For instance, if a domain expert wants to build a human mobility model based on this dataset, a very detailed model of how people move in Midtown can be constructed. However, there will be little information on human mobility patterns for the residents in Ridgewood based on only the yellow taxi dataset, which makes the design of such model challenging. A possible approach to deal with the sparse coverage is to fill the gaps using other datasets, such as green taxis (local taxi service in Brooklyn) or data from Uber.

This scenario also shows the need to examine different data slices at different resolutions to help determine the quality of the data: by analyzing the data at the neighborhood level and on an hourly basis, the sparsity issue can be identified, while a coarser resolution may hide this matter. It is thus crucial to have usable tools that enable users to easily and interactively explore these data at multiple granularities.

3.5 Inferring Speed from Trip Duration and Distance

The taxi data can be used as a proxy to understand how traffic flows in New York City. Given the attributes *trip duration* t and *trip distance* d, the average speed associated to each trip can be computed as d/t. Given the average speeds for all trips, along with their spatio-temporal attributes, it is possible to derive, for example, analyses about how traffic jams are distributed in New York City [26]. In this scenario, it is important to discard outliers for trip duration and trip distance as they will negatively affect the computation of average speeds. While entries with incorrect values (negative or zero values) are easy to identify, detecting which positive valued outliers should be removed is challenging.

Figure 7 shows how average speeds are distributed in the 2011 taxi dataset. The speed limit in New York

City was 30 miles per hour in 2011,[4] but there are taxi trips in the dataset that surpassed such limit. Deciding which of them correspond to data inconsistencies, and which simply correspond to drivers traveling over the speed limit, is a difficult task. In Figure 7, while most results look valid, as speeds between 30 and 50 miles per hour probably correspond to real occurrences, values above 100 miles per hour are likely to correspond to errors in the dataset.

Before deciding which trips should be removed, it is necessary to remove trips that are inconsistent, i.e., trips having attributes d or t equal to zero. Poco et al. [26] showed that these trips carry a significant negative impact on speed computations and general traffic flow analysis. After removing these trips, one can address the problem by using a combination of traditional outlier detection techniques and domain knowledge. For outlier detection, it is possible to define a standard distribution that should fit the average speed distribution (e.g., a Gaussian distribution), and remove all trips that are a few standard deviations (say 1 or 2) away from the mean. Domain experts can also help uncover behaviors that can be normal, even if they seem to be outliers. It is possible, for instance, that drivers reach high speeds in certain parts of uptown Manhattan when moving to upstate New York roads. As in other cases, slicing the data into spatial regions and temporal ranges, alongside the aid of a domain expert, can be useful to uncover specific speed patterns in New York City.

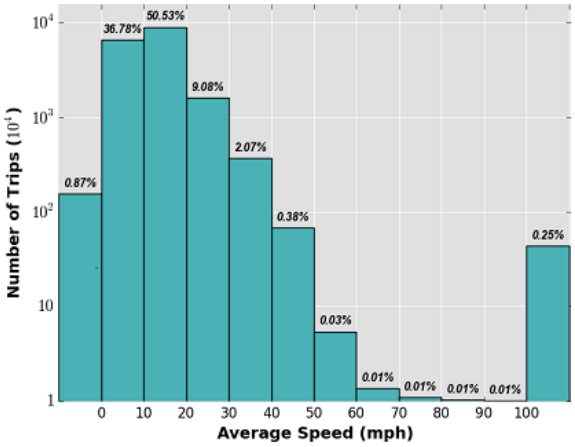


Figure 7: Distribution of taxi average speeds in miles per hour (mph) for the 2011 taxi dataset.

3.6 Inaccurate GPS Readings

GPS readings are not always accurate, especially in cities with a large number of tall buildings. GPS signals are also heavily influenced by the number of GPS satellites: the more satellites are used, the more accurate are the positions. When a taxi passes by a tall building or other obstructions, the set of satellites to which its GPS is associated will likely change. This signal switch between different sets of satellites negatively impacts the position accuracy. The quality of the GPS receiver algorithm for processing the satellite signals might also lead to an inaccurate position.

Figure 8 shows many such errors: taxis in the rivers, in the ocean, and outside North America. Inaccurate GPS points can lead to misleading results. If one wants to detect trendy areas where residents and tourists often go to in NYC, for example by using an algorithm such as k-means, the inaccurate GPS points will lead to meaningless clusters—outside NYC and over the water.

Visualization is an effective mechanism to identify these inconsistencies. By looking at the maps in Figure 8, one can easily see the incorrect locations. To remove GPS inconsistencies, clustering methods can be used. If the geographical boundaries are known in advance, it is possible to check whether they are inside valid polygons. For the NYC taxi data, we can check whether pickups fall within a neighborhood (or zip code) within the city bounds.

3.7 Ghost Trips

While analyzing the taxi data, we discovered a large number of overlapping trips for the same taxi, i.e., for a given taxi, a new trip starts before the previous trip has ended. We call these trips *ghost trips*. The reason behind this data inconsistency is unclear: some trips may overlap due to a device error, or simply because the taxi driver

[4] http://cityroom.blogs.nytimes.com/2011/05/12/a-spooky-reminder-to-obey-the-speed-limit/

Figure 8: Inaccurate GPS points (a) in rivers, (b) in the ocean, and (c) outside North America.

forgot to log the end of a trip after dropping off passengers. Nevertheless, they certainly affect further analysis on the data, such as data-based human mobility models [42].

In the 2010 taxi dataset, for the month of May, there were 7.1 million ghost trips. Given the 154 million trips that took place that month, this corresponds to an error rate of about 4.60%. To better understand which of the overlapping trips are defective, we would need domain knowledge from expert users and TLC to perform data cleaning: all the trips or just a subset may be erroneous. The number of ghost trips is much smaller for the 2011 dataset: the error rate is only 0.20%. Since the taxi dataset for 2011 has considerably fewer invalid values compared to 2010, as described in Section 2.1, one possible explanation is that different cleaning procedures were used for these two years, and inconsistencies such as ghost trips were removed before the release of the 2011 dataset.

4 Discussion

In this paper, we discussed some of the challenges involved in cleaning spatio-temporal urban data. We presented a series of case studies using the NYC taxi data that illustrate data cleaning challenges and suggested potential methodologies to address these challenges. These methodologies form the basis for integrating cleaning with data exploration. Data cleaning is necessary for data exploration, and through data exploration, users can attain a better understanding of the data which can lead to the discovery of cleaning constraints and enable them to discern between errors and features. Data exploration, however, requires a complex trial-and-error process. Thus, usable tools are needed to guide and assist users in the cleaning process. As the case studies we discussed illustrate, this is particularly true for spatio-temporal data, where visual analytics and event detection techniques at different resolutions are essential to identify quality issues.

The case studies presented in Section 3 show that some cleaning decisions are not clear cut. Often, multiple datasets are required to help an expert decide whether a data point is erroneous or represents an important feature. While there has been preliminary work on the discovery of relationships across datasets [8], there are still many open problems in identifying relevant data that can be used to explain events within a large collection of datasets and in a systematic fashion.

Lack of sufficient knowledge is another issue that hampers data cleaning. Even though experts can (and should) be involved in most of the process, they may be unavailable, or it may be expensive to hire them for cleaning large datasets. Crowdsourcing systems could help the data analyst clean data more efficiently: user

feedback can be used to learn features and "separate the wheat from the chaff."

Different questions that arise during exploration may require different cleaning strategies. While visualization helps in identifying potential unusual behavior, other techniques are also necessary in for data cleaning, including automatic event detection and clustering. The fact that these different techniques are applied in trial-and-error fashion underscores the importance of maintaining *provenance* of the cleaning process. Provenance not only enables reproducibility, but it also helps in exploration. As we have shown in previous work, provenance information can be used to support reflective reasoning, to create and refine analysis pipelines by example, to guide the user by providing recommendations for next steps to try, and to perform exploration collaboratively [19, 32, 33]. In addition, provenance provides detailed documentation of all cleaning steps applied to a dataset, and this knowledge is crucial during analyses. For instance, by examining the taxi data, we believe that the years of 2011 and 2012 were better cleaned than, say, the 2010 dataset. This could be confirmed if we had access to the provenance of the cleaning process. In this case, provenance would also allow users to identify which cleaning techniques—applied to newer datasets (e.g., 2012) by the TLC—could be re-used to clean older datasets (e.g., 2010). By applying the same cleaning process to the different datasets, analyses over them would likely be more consistent.

Acknowledgments We thank the New York City TLC for providing the data used in this paper. This work was supported in part by a Google Faculty Award, IBM Faculty Awards, the Moore-Sloan Data Science Environment at NYU, the NYU Tandon School of Engineering, the NYU Center for Urban Science and Progress, and NSF awards CNS-1229185 and CI-EN 1405927. Freire is partially supported by the DARPA MEMEX program.

References

[1] G. Andrienko and N. Andrienko. Spatio-temporal Aggregation for Visual Analysis of Movements. In *Procedings of IEEE Visual Analytics Science and Technology*, pages 51–58, 2008.

[2] G. Andrienko, N. Andrienko, P. Bak, D. Keim, and S. Wrobel. Visual Analytics Focusing on Spatial Events. In *Visual Analytics of Movement*, pages 209–251. Springer Berlin Heidelberg, 2013.

[3] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel. From Movement Tracks through Events to Places: Extracting and Characterizing Significant Places from Mobility Data. In *Proceedings of IEEE Visual Analytics Science and Technology*, pages 161–170. IEEE, 2011.

[4] L. Barbosa, K. Pham, C. Silva, M. Vieira, and J. Freire. Structured Open Urban Data: Understanding the Landscape. *Big Data*, 2(3), 2014.

[5] L. Berti-Equille, T. Dasu, and D. Srivastava. Discovery of complex glitch patterns: A novel approach to quantitative data cleaning. In *Proceedings of the International Conference on Data Engineering*, pages 733–744, 2011.

[6] M. R. Bloomberg and D. Yassky. 2014 Taxicab Fact Book. `http://www.nyc.gov/html/tlc/downloads/pdf/2014_taxicab_fact_book.pdf`, 2014.

[7] D. Castellani Ribeiro, H. T. Vo, J. Freire, and C. T. Silva. An Urban Data Profiler. In *Proceedings of the International Conference on World Wide Web*, WWW '15 Companion, pages 1389–1394, 2015.

[8] F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire. Data polygamy: The many-many relationships among urban spatio-temporal data sets. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2016. To appear.

[9] T. Dasu, J. M. Loh, and D. Srivastava. Empirical glitch explanations. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 572–581, 2014.

[10] H. Doraiswamy, N. Ferreira, T. Damoulas, J. Freire, and C. Silva. Using topological analysis to support event-guided exploration in urban data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2634–2643, 2014.

[11] H. Doraiswamy, H. T. Vo, C. T. Silva, and J. Freire. A GPU-based index to support interactive spatio-temporal queries over historical data. In *IEEE International Conference on Data Engineering*, 2016. To appear.

[12] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva. Visual exploration of big spatio-temporal urban data: A study of New York City taxi trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2149–2158, 2013.

[13] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science and Engineering*, 10(3):11–21, 2008.

[14] B. Goldstein and L. Dyson. *Beyond Transparency: Open Data and the Future of Civic Innovation*. Code for America Press, San Francisco, USA, 2013.

[15] Y. Gu and C. Wang. itree: Exploring Time-Varying Data Using Indexable Tree. In *IEEE Pacific Visualization Symposium*, pages 137–144, 2013.

[16] M. Hoai and F. De la Torre. Max-Margin Early Event Detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2863–2870, 2012.

[17] J. Höchtl and P. Reichstädter. Linked Open Data - A Means for Public Sector Information Management. In *Electronic Government and the Information Systems Perspective*, volume 6866 of *Lecture Notes in Computer Science*, pages 330–343. Springer, Berlin Heidelberg, 2011.

[18] B. Katz and J. Bradley. *The Metropolitan Revolution: How Cities and Metros Are Fixing Our Broken Politics and Fragile Economy*. Brookings Focus Book. Brookings Institution Press, 2013.

[19] D. Koop, C. E. Scheidegger, S. P. Callahan, J. Freire, and C. T. Silva. Viscomplete: Automating suggestions for visualization pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1691–1698, 2008.

[20] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for Real-Time Exploration of Spatiotemporal Datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, 2013.

[21] E. McFowland III, S. Speakman, and D. B. Neill. Fast Generalized Subset Scan for Anomalous Pattern Detection. *Journal of Machine Learning Research*, 14:1533–1561, 2013.

[22] D. B. Neill and G. F. Cooper. A Multivariate Bayesian Scan Statistic for Early Event Detection and Characterization. *Machine learning*, 79(3):261–282, 2010.

[23] City of Chicago Data Portal. `https://data.cityofchicago.org`.

[24] NYC OpenData. `https://nycopendata.socrata.com`.

[25] M. Ota, H. Vo, C. Silva, and J. Freire. A scalable approach for data-driven taxi ride-sharing simulation. In *IEEE International Conference on Big Data*, pages 888–897. IEEE, 2015.

[26] J. Poco, H. Doraiswamy, H. Vo, J. L. Comba, J. Freire, C. Silva, et al. Exploring traffic dynamics in urban environments using vector-valued functions. *Computer Graphics Forum*, 34(3):161–170, 2015.

[27] W. Rao, K. Zhao, Y. Zhang, P. Hui, and S. Tarkoma. Towards maximizing timely content delivery in delay tolerant networks. *IEEE Transactions on Mobile Computing*, 14(4):755–769, 2015.

[28] R. E. Roth. An Empirically-Derived Taxonomy of Interaction Primitives for Interactive Cartography and Geovisualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2356–2365, 2013.

[29] S. Sarawagi. Explaining Differences in Multidimensional Aggregates. In *Proceedings of the International Conference on Very Large Data Bases*, pages 42–53, 1999.

[30] S. Sarawagi, R. Agrawal, and N. Megiddo. *International Conference on Extending Database Technology*, chapter Discovery-Driven Exploration of OLAP Data Cubes, pages 168–182. Springer Berlin Heidelberg, 1998.

[31] T. H. Savage and H. T. Vo. Yellow cabs as red corpuscles. In *Proceedings of Workshop on Big Data and Smarter Cities*, 2012.

[32] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1560–1567, 2007.

[33] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva. Querying and re-using workflows with vistrails. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 1251–1254, 2008.

[34] N. Shadbolt, K. O'Hara, T. Berners-Lee, N. Gibbins, H. Glaser, H. Wendy, and M. Schraefel. Linked Open Government Data: Lessons from Data.gov.uk. *IEEE Intelligent Systems*, 27(3):16–24, 2012.

[35] G.-D. Sun, Y.-C. Wu, R.-H. Liang, and S.-X. Liu. A Survey of Visual Analytics Techniques and Applications: State-of-the-Art Research and Future Challenges. *Journal of Computer Science and Technology*, 28(5):852–867, 2013.

[36] TaxiVis. `https://github.com/ViDA-NYU/TaxiVis`.

[37] TLC Trip Record Data. `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`, 2015.

[38] J. Wakefield and A. Kim. A Bayesian Model for Cluster Detection. *Biostatistics*, 14(4):752–765, 2013.

[39] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. v. d. Wetering. Visual Traffic Jam Analysis Based on Trajectory Data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2159–2168, 2013.

[40] H. Wickham. Tidy Data. *The Journal of Statistical Software*, 59, 2014.

[41] K. Zhao, M. P. Chinnasamy, and S. Tarkoma. Automatic City Region Analysis for Urban Routing. In *IEEE International Conference on Data Mining Workshop*, pages 1136–1142, 2015.

[42] K. Zhao, M. Musolesi, P. Hui, W. Rao, and S. Tarkoma. Explaining the power-law distribution of human mobility through transportation modality decomposition. *Nature Scientific Reports*, 5(9136), March 2015.

[43] Y. Zheng, F. Liu, and H. Hsieh. U-Air: When Urban Air Quality Inference Meets Big Data. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1436–1444, 2013.

Data Quality for Temporal Streams

Tamraparni Dasu, Rong Duan, Divesh Srivastava
AT&T Labs - Research, Bedminster, NJ 07921, USA
{tamr, rongduan, divesh}@research.att.com

Abstract

Temporal data pose unique data quality challenges due to the presence of autocorrelations, trends, seasonality, and gaps in the data. Data streams are a special case of temporal data where velocity, volume and variety present additional layers of complexity in measuring the veracity of the data.

In this paper, we discuss a general, widely applicable framework for data quality measurement of streams in a dynamic environment that takes into account the evolving nature of streams. We classify data quality anomalies using four types of constraints, identify violations that could be potential data glitches, and use statistical distortion as a metric for measuring data quality in a near real-time fashion. We illustrate our framework using commercially available streams of NYSE stock prices consisting of aggregates of prices and trading volumes collected every minute over a one year period from November 2011 to November 2012.

1 Introduction

In today's data driven world, decisions are based on analysis applications that process continuous streams of data. Typically, the data are summarized or characterized by a *statistical signal* that represents the data within some bounds of uncertainty. The statistical signal could be as simple as a mean and standard deviation, or as complex and comprehensive as a joint probability distribution. Analysis applications rely on this signal to make predictions, forecasts and to create reports, dashboards and visualizations. Data quality issues interfere with this signal and distort the data, leading to misleading analysis outcomes and potentially bad decisions.

When there are data quality issues in temporal data, such as gaps (missing data), or dips (incomplete data) and spikes (duplicate data), they impact the underlying, constantly evolving density distributions. Some bins in the histogram will have less mass (missing/incomplete) than normal while other bins will have additional mass (duplicates) as compared to the norm resulting in distorted densities. Abnormally high values (inconsistent data) will show up as outliers in the tails of the distribution. Peculiar to streaming data, late arrivals (untimely or out-of-order data) will manifest as holes in the density at their expected arrival time, and spikes in the density when they actually arrive. Figure 1 illustrates some of these concepts using data streams that measure volumes of data transmitted by two correlated data sources depicted in red and blue. We will discuss it in detail in Section 2.3.

Since data are constantly changing over time (for example, level shift in Figure 1), the nature of data quality issues as well as the extent to which they are present changes too. Therefore the notion of what is acceptable in terms of data quality changes over time as well. In addition, it is important to take seasonality, periodicity

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

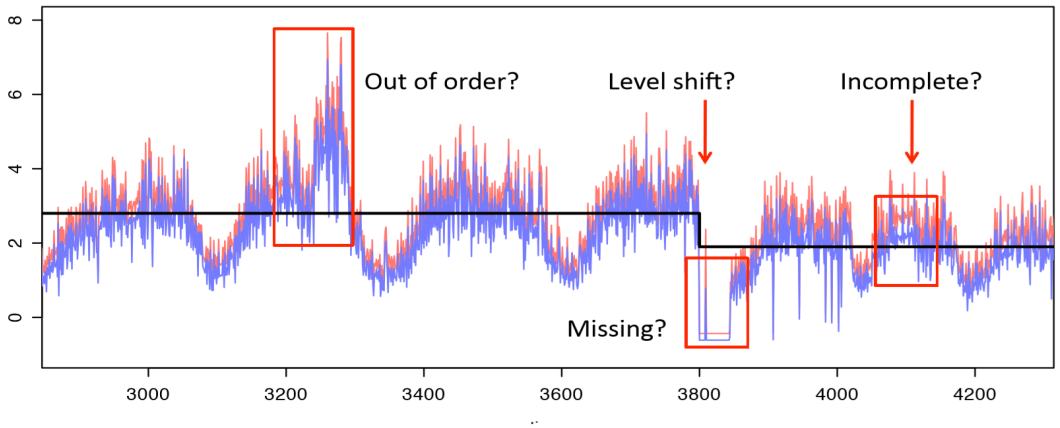


Figure 1: *Two correlated data streams with data quality glitches. Each color represents a different stream. Notice that glitches can be correlated and have complex dependence patterns such as: (a) data that arrives late (timeliness) as indicated by lower volumes followed by higher volumes (out of order), (b) missing values followed by a level shift when a generative process changes, e.g., by dropping a portion of the stream, and (c) incomplete data for the blue stream when the two streams diverge.*

and autocorrelations into account while determining whether a data point is a potential data quality concern. For example, it is certainly the case that trading volumes of stocks are higher at the beginning and end of the day, and very low on the eve of national holidays. It is also critical to consider dependence structure in glitches to prevent overstating of glitches. For example, out-of-order data manifest as missing followed by duplicates. Similarly, in correlated streams, if both streams have the same unusual behavior it might be a genuine feature of the generative process, while divergence between otherwise correlated streams could indicate incompleteness of one of the streams.

In addition to the statistical signal, domain-specific logical constraints need to be embodied as a set of data quality rules to complement the statistical signal. Toward this end, it is important to consult domain experts in formulating these rules as well as in refining them. Otherwise, violations of such constraints may not be data quality issues but more a reflection of incomplete or improperly specified constraints. For example, in the world of telecommunications, call volumes are exponentially higher on certain holidays like Mother's Day. Therefore, any constraint that states that call volumes need to be within a certain range $[Low, High]$ will need to reflect the exceptions.

A variety of logical constraints proposed by the data quality research community can be used to capture domain knowledge specifically in temporal streams, including order dependencies [5], sequential dependencies [4], speed constraints [6] and temporal rules [7]. These can be naturally used within the framework for data quality, discussed next.

2 A Framework for Data Quality in Temporal Streams

We discuss a general, widely applicable framework that addresses data quality concerns unique to temporal data. The methodology applies to numerical data as well as categorical data treated as group-by variables, counts or proportions. The approach is empirically driven and can be applied even in situations where there is no prior

Type	Single Column	Multi-column
Single row	Type 1: Applies to a single attribute and single entity	Type 2: Depends on multiple attributes but only one entity
Multi-row	Type 3: Applies to a single attribute but depends on multiple entities	Type 4: Depends on multiple attributes and multiple entities

Table 1: Types of constraints

domain knowledge. Even if the error rates are high, as long as they are not uniformly spread across the data, the method can isolate concentration of errors in specific subsets of the data.

The framework consists of the following primary stages, stated only briefly here. A detailed discussion follows.

- Formulate or learn rules or constraints that embody the characteristics that data need to meet (within some margin of error or uncertainty) in order to be deemed of a good quality. We call this the *ideal*, D_t^I, at time t.
- Benchmark the observed data D_t against the ideal D_t^I for *unexplained* violations of constraints and flag them as possible *glitches*. Note that violations are not necessarily data quality issues. As demonstrated in [9], many violations have "explanations" that are not captured by extant rules or constraints. These explanations can be used to refine the data quality rule set to prevent them from being flagged in the future.
- The presence of glitches causes the data to deviate from expected or ideal behavior. Capture the extent of deviation using the notion of *statistical distortion* $SD(t)$ introduced in [8]. A high distortion could mean that data D_t might not be suitable for use by downstream applications.
- Using the statistical distortion $SD(t)$ as a *data quality metric*, continuously monitor D_t to identify unusual distortion that falls outside acceptable ranges and alert responsible parties.

We explain in detail in the following sections.

2.1 The Ideal, D_t^I

Data streams are consumed by numerous downstream analysis applications that generate reports, alerts and other information used for decision making. These applications expect the data to meet certain criteria embodied in assumptions like: "Data must contain no missing attributes" (completeness), "a file of size X Megabytes arrives every minute" (timeliness, completeness), "the file size should not exceed Y Megabytes" (presence of duplicates?), "the files should not contain outliers in excess of some threshold, say 1%", or "if the zip code is Z, then the city has to be C". Such constraints describe an ideal D^I of what the data is expected to satisfy.

In the context of databases, multiple data sets could satisfy data quality constraints and match the notion of what is acceptable. And associated with each of these data sets is a data distribution $\mathcal{F}_i$ that captures the statistical properties of that particular data set. Therefore, an ideal D^I could be thought of as a mixture of these distributions $\{\mathcal{F}_i\}$. In the case of temporal data, the ideal itself could be changing with time to accommodate changes in the underlying generative process ("the system will switch from few large files to many smaller files to transmit the same amount of data") or changing needs of applications, and is denoted as D_t^I, where t is time. In the absence of pre-specified formulations of the ideal, it is possible to *learn* D_t^I from historical data.

In this paper we will use the words "ideal" and "expected" interchangeably to describe D_t^I.

2.2 DQ Constraints

As evidenced by the preceding discussion, constraints play an important role in specifying the ideal D_t^I. We classify these data quality constraints by their dependence on *attributes* or features (typically columns in a database table) and *entities* or instances (rows). An entity or row can represent something as simple as an individual identified by a "Customer_ID" and described using attributes like "Age", "Name", and "Phone Number" stored in columns. An entity could also be more complex and hierarchical like a corporation, with branches and sub-branches. Such a classification based only on row-column dependence is conceptually clean and closely relates to the cost of repairs. Logical, representational and statistical constraints all have a place in it and the framework is not dependent on any particular constraint language.

2.2.1 Constraints-Type 1

Type 1 constraints depend on a single row and column, i.e., a single individual value. For example, the constraint "should be an integer" can be validated by examining a single value with no additional reference to other rows or columns. Examples of violations include formatting problems that make the data hard to parse and access. Such representational constraints can be *learned* from existing data using data profiling and data browsing techniques (e.g., see [3] and [1]). Type 1 constraints are inexpensive to validate since they require access to a single cell in the database.

2.2.2 Constraints-Type 2

Type 2 constraints involve other attributes from the same row. For example, in the state of New Jersey sales tax cannot exceed 7% of the price of an item; this can be formulated as a Type 2 constraint, "if State equals NJ, then Sales Tax $<=$ 0.07*Price", which can be naturally expressed as a denial constraint. Note that we need two other columns (State and Price) to validate the value in a single row of the attribute "Sales Tax".

2.2.3 Constraints-Type 3

Next, consider constraints that involve values in a column across multiple rows. These are often based on comparisons between two or more rows (which is the case for many logical constraints) or distributions/aggregates (which is the case for many statistical constraints).

The notion of single-attribute *uniqueness* or *keyness* is a Type 3 constraint. The constraint "Column A is a key" can be specified by simply comparing two rows at a time, and checking that the Column A values are different. Similarly, detection of statistically anomalous values within a single attribute involves validating a Type 3 constraint, e.g., "Data should not lie outside three standard deviations from the mean," where the mean and standard deviation are statistical summaries computed from multiple rows of that column.

2.2.4 Constraints-Type 4

Finally, we define constraints that involve multiple rows as well as multiple columns to be of Type 4. For example, "Attributes A and B should generally trend in the same direction as measured by a correlation of greater than 0.8". Functional dependencies of the form "if two rows have the same SocialSecurityNumber, they must have the same LastName" are also Type 4 constraints. Consistency with an independent, external data source is a Type 4 constraint as well. Identifying multi-attribute *duplicates* has a similar flavor.

Note that the above classification roughly reflects the amount of data that needs to be accessed for validation and changed for repair, capturing the cost as well as the amount of data that could be impacted by potential repairs.

2.3 Identifying Data Glitches

Data glitches are often discovered through data exploration and visualization. Figure 1 shows examples of data glitches in the arrival of two correlated data streams, red and blue. In general, the streams move together. However, between t=3200 and t=3400, there is a drop in the amount of data that arrives, followed by an increase, indicating that data have arrived late causing them to be out of order. At t=3800, there is a short period of missing data followed by consistently lower volumes indicating that the process may have been redesigned to drop some component from transmission. Finally, note the brief period of divergence in the amount of data delivered for the two streams, indicating that the blue stream might be incomplete.

Given how increasingly critical data are to applications and businesses, glitch discovery has become an integral part of the early stages of data management. Data glitches are identified by validating a given instantiation of data for constraint violations. As noted earlier, not all violations of constraints constitute glitches. Some are "suspicious" at first glance but can be explained by experts as legitimate data. For example, in telecommunications, certain holidays have a big impact on network traffic and abnormally high or low volumes are acceptable, and could be explained, e.g., "We expect significantly higher call volumes on Mother's Day."

Domain knowledge in the form of explanations of anomalies can then be used to refine the constraints to reflect the conditions under which the constraint could legitimately be violated, e.g., Mother's Day. Since not all constraint violations are data glitches, it is important to seek glitch explanations to avoid over-treating the data, and embody the explanations as additional constraints to better reflect the ideal, D_t^I. See [9] for details about generating empirical glitch explanations and refining domain knowledge.

2.4 DQ Metric: Statistical Distortion

Analysis applications typically rely on statistical distributions because they drive the signal in the data. Data glitches interfere with this signal, as do indiscriminate attempts to clean the data. In [8], this notion of distortion to the signal in the data caused by cleaning was generalized as *statistical distortion*. Intuitively, statistical distortion is a data quality metric that measures the discrepancy between the ideal that we expect to receive at time t, D_t^I, and the data D_t we actually observe at t, within some acceptable statistical variation. For example, if we expect to receive 10 files every hour, and if we receive 7 files in the current hour, is that too few, or within expected statistical variation?

Formally, given the ideal D_t^I and current data D_t, *statistical distortion* measures the statistical distance $\mathcal{D}$ between them:

$$SD(D_t) = \mathcal{D}(D_t^I, D_t).$$

When the current data D_t is significantly different from the ideal D_t^I, we expect the statistical distortion to be high and vice versa. As a corollary, any good cleaning technique should reduce the statistical distortion and bring the cleaned data within an acceptable distance of the ideal distribution.

Statistical distortion can be captured using many metrics, from simple to complex: it could be a basic difference in proportion of constraint violations between the ideal D_t^I and the given data set D_t, or more data-specific distances like the Mahalanobis Distance (MD), Kullback Leiber Divergence (KLD) or the Earth Mover Distance (EMD). Glitch based distances like the difference in proportions are easy to compute and appropriate for the streaming setting. The difference in actual statistical distributions of the D_t^I and D_t captured by MD, KLD and EMD are computationally more expensive but capture the amount of disparity. These distances are useful for drilling down to the specific attributes or entities that are causing the disparity, and help in identifying the source of the glitches. Application needs and resource constraints determine the choice of distance used.

2.5 Making the Data Usable

Once the data glitches are identified and validated, data are made usable by making changes or repairs to the data to align it with expectation D_t^I, within acceptable limits of statistical distortion. Note that the repairs are often specific to applications. Marketing analysts focus on typical values that represent a group and hence will not tolerate outliers, while network engineers are interested in the rare catastrophic outlier and will not tolerate any missing values. In general, human experts may need to be involved to identify the appropriate repair strategy. In this paper, we will not focus on general repair strategies, but will discuss specifics in the context of our case study below.

3 A Case Study: NYSE Data

We illustrate the concepts discussed in the preceding sections using NYSE stock data, a classic example of temporal streams. We were able to access the raw minute-by-minute aggregates of stock prices and trading volumes. In order to address concerns specific to financial streams, we focus on quality concepts outlined by Thomson Reuters (TR) in their guidelines for cleaning financial data streams [11].

3.1 Description

The case study data consists of "minute bars" from over 2000 stocks traded on the New York Stock Exchange (NYSE) spanning the period from November 2011 to November 2012. The attributes, recorded within every minute interval [DT_t, DT_t + 1 minute) where DT_t is the date-time at time t described below, are:
trading time (YYYY-MM-DD hh:mm) DT_t,
day of the month,
opening price O_t,
highest price observed during the trading minute H_t,
lowest price L_t,
closing price C_t,
and the trading volume V_t.
Not every stock trades every minute. When there is no trade, no record is generated for that stock resulting in an irregular temporal stream. A sample of the data:

```
2011-11-01 09:38,1,21.75,21.78,21.75,21.76,1200
2011-11-01 09:39,1,21.74,21.75,21.73,21.74,1481
2011-11-01 09:40,1,21.75,21.77,21.74,21.76,6675
2011-11-01 09:41,1,21.77,21.81,21.76,21.79,1713
2011-11-01 09:42,1,21.8,21.82,21.78,21.78,2655
```

The Stock ID is embedded in the filename. We consider "StockID + Timestamp" to be a unique key. A separate table provides the mapping from Stock ID to StockTickerName, e.g., StockID=968 and StockTickerName="HAL" for Halliburton.

In addition, we compute derived variables such as the price spread,

$$SP_t = H_t - L_{t'},$$

consecutive changes, delta, in a given attribute, say H_t, the intra-minute high,

$$DL_t = H_t - H_{t'},$$

and the lag, LG_t, between successive trading minutes

$$LG_t = t - t',$$

where t' is the last observed actively traded minute.

We use the NYSE data stream to do two types of data quality checks. The first set of checks monitors the process of *data gathering,* i.e., is the data arriving in an expected fashion? The second set of checks monitors the *data content* for the quality of the contents of data received.

3.2 Glitch Detection

We illustrate using two approaches to glitch detection. Thomson Reuters (TR) proposes a deterministic percent change over previous value i.e they consider D_{t-1}, the previous value, to be the ideal, D_t^I, for the purpose of measuring data quality. In addition to the TR approach, we use the Feed Inspection Tool (FIT) for monitoring streaming data, discussed in detail in [10].

Briefly, FIT computes statistical summaries (mean and higher order moments, quantiles, counts) based on historical data in a sliding window to empirically estimate the ideal D_t^I. The current chunk of observed data D_t is validated against D_t^I. The comparison can be done at various levels of temporal granularity (5 minutes, 15 minutes, every hour) and the summaries can be made fine-grained by partitioning the data into groups such as day-of-week (DoW) and time-of-day (ToD). Note that running a simple outlier detection within each trading day is not very effective. It is well known that trading behavior is dependent on the time of the day. There is heightened activity in the morning when the market catches up with after-hour activity and again at the end of the day when traders try to "flatten their books" by hedging or unwinding risky positions. In addition, important economic numbers are often announced mid-morning causing markets to react. An outlier detection mechanism that does not take the time-of-day effect into account will generate spurious outliers. By limiting the history used in computing expected behavior via an appropriately chosen sliding window, or by deprecating the contribution of data over time using exponential decay models, FIT ensures that the empirically computed ideal evolves with the data and captures changing distributions accordingly. The choice of group by variables ToD and DoW as well as the size n of the sliding window affect the detection of anomalous values by FIT, while the choice of percentage threshold will affect TR's anomalies. We address this aspect in Section 3.5.

3.3 DQ: Data Gathering

We first check for completeness and timeliness by monitoring the number of stocks that report each minute. The reports, known as "minute bars", are delivered as individual files for each stock. The check for file counts by both TR as well as FIT are Type 4 constraints since they use multiple entities (stocks), at multiple times (previous value for TR, and for FIT, values in n rows where n is window size), and multiple attributes for FIT (file counts, and derived attributes ToD and DoW extracted from time stamps).

The left panel of Figure 2 depicts the results of running FIT on the average rate of file arrivals per minute within a given hour, compared to the expected average file rate per minute for that hour based on the sliding window FIT model. We ran FIT with several window sizes and thresholds and did not find any significant change in the results. In the plot, big pink dots represent the morning hours starting at 9:30 AM, gradually changing into smaller blue dots as the day goes by, with the smallest bluest dot for the hour 15:00. We plot only three hours–first, middle and last–to avoid clutter in the plot. The red dots are anomalous values. The size of the red dots allows us to determine which hour of the day they correspond to. According to the plot, there were anomalously low rates of file arrivals per minute during the last hour of July 3, and abnormally high values on Sep 12, to mention just a couple.

The data gathering checks do not require us to open the files, or parse them to examine the content; we just count the number of files, and if needed, their size. This is a fast DQ check and can prevent resources from

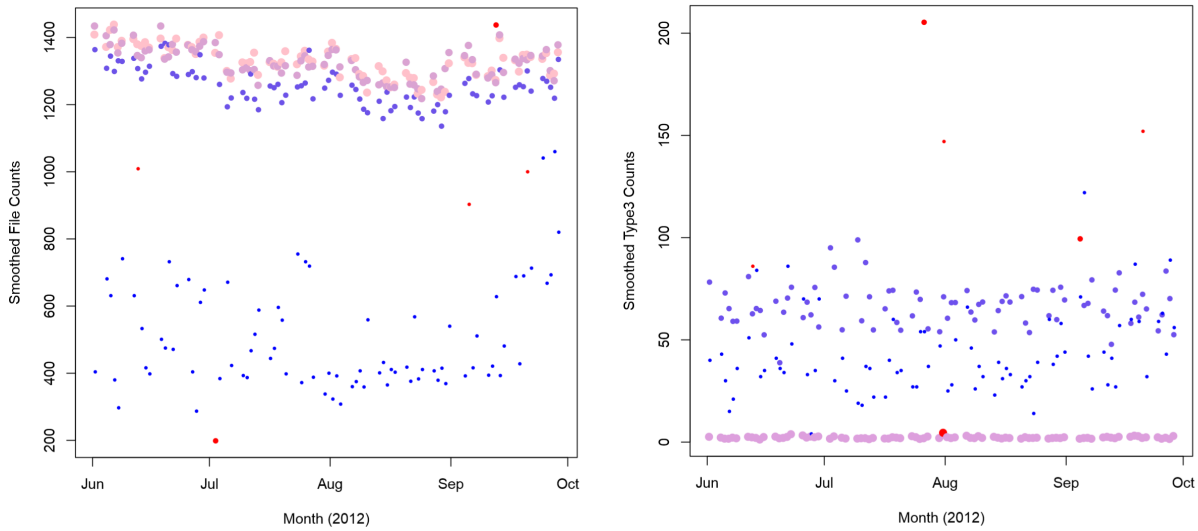


Figure 2: *The panel on left depicts the results of running FIT on the average rate of file arrivals per minute within a given hour. Big pink dots represent the beginning of the trading day at the morning hours gradually changing into smaller blue dots as the day goes by. The red dots are anomalous values. The size of the red dots allows us to determine which hour of the day they correspond to. There were anomalously low rates of file arrivals per minute during the last hour of July 3, and abnormally high values on September 12, to mention a couple. The panel on the right shows anomalous values of statistical distortion as measured by the total of Type 3 glitch counts,* $G^3(t)$.

being wasted on ingesting incomplete files. Figure 3 shows details of two sample anomalous days July 3 and September 12. The box plots in each figure represent the two underlying baseline models to which the days July 3, 2012 and Sep 12, 2012 are compared respectively. The red dots represent the number of stocks that trade each minute within a given hour, for the given day.

On July 3, the trading dropped off at 1 PM, due to an early close of the stock market on the eve of the July 4 holiday. This is an explainable anomaly. On September 12, 2012, there was an unexplained upward trend. The shift here was subtle, hence it took a statistical model like FIT to detect the change. FIT, because it takes into account the day-of-week effect, had already accounted for spikes on Wednesdays caused by Fed announcements. The higher values observed on September 12, 2012 were *outliers* and *inconsistent* with historical data even after taking that fact into account, and could not be explained using a correlated stream of times of Fed announcements.

We decided to investigate further and consult a domain expert. We found that the Fed, for the first time in 2012, had given a definitive extension until 2015 for interest rates, causing stocks to trade more actively as observed by a general shift of red dots toward higher values after 2 PM. (There was an upward trend earlier that day as word leaked out and the markets anticipated the Fed's decision). By seeking additional domain knowledge that is not available freely or in an automated fashion, we were able find an explanation. This is an example where a data anomaly would have been deemed a glitch because the explanation for it is not easily accessed in an automated fashion. Perhaps with sufficient effort, a majority of anomalies might be explained isolating a set of "true" glitches. However, given resource constraints, the process of generating explanations falls short of this clean separation of glitches and explainable anomalies.

Note that glitches captured by monitoring file counts are correlated glitches–movements that affect a large number of stocks.

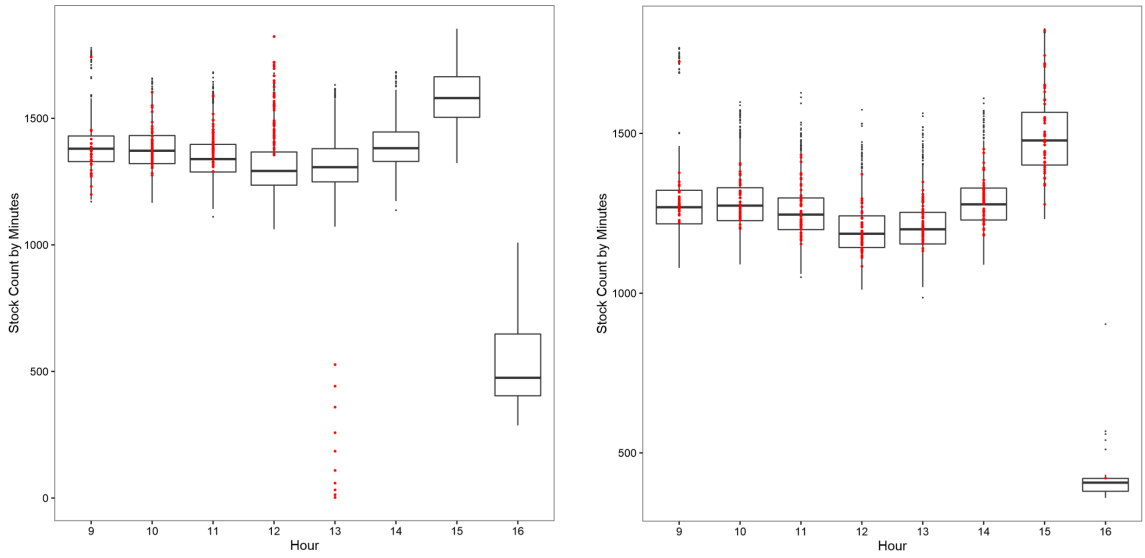


Figure 3: *Two sample anomalous days July 3 and September 12: number of stocks that trade each minute, within each hour. The box plots in each figure represent the two underlying baseline FIT models to which the days July 3, 2012 and September 12, 2012 are compared respectively. The red dots represent the observed values for the given day. While the July 3 dip was obvious, the September 2012 highs were more subtle. Notice the skew of red points toward higher values, particularly pronounced in hours 11,13, 14 and 15. It required an expert Fed watcher to provide an explanation.*

3.4 DQ: Data Content

A vast majority of DQ checks relate to content of the actual data. We consider stocks individually while detecting glitches, but it might be necessary to consider market behavior (correlations among stocks) for explaining or validating glitches, e.g., did all the stocks experience a drop or spike in volumes?

3.4.1 Type 1 Constraints

For each of the NYSE stocks and for each of the trading minutes, we follow the Thomson Reuters guidelines to formulate the following constraints:
(1) no attribute should be missing;
(2) all prices and volumes should be non-negative.

Let violations of these constraints at time t be denoted by $g_1^1(t), g_2^1(t)$, where the superscript denotes glitches of Type 1, and the subscript indexes the glitches as 1 and 2 respectively.

We also aggregate these over all NYSE stocks to compute the total glitches of these types at time t as follows:

$$G_1^1(t) = \sum_{NYSE} g_1^1(t)$$

and

$$G_2^1(t) = \sum_{NYSE} g_2^1(t)$$

respectively, where the summation is over all stocks and t is indexed over the trading minutes. We will use these sums later to compute statistical distortion.

Since the data was obtained from an authoritative source, we could not find any obvious glitches of Type 1. This is not surprising since Type 1 and Type 2 glitches are the easiest to detect (do not require accessing other rows/entities). They are often the ones that are cheaply remedied and hence taken care of by data providers.

3.4.2 Type 2 Constraints

Domain experts often stipulate conditions on inter-relationships between attributes. For example, for a given stock S at time t:
(1) the closing price C_t should be contained in the interval $[L_t, H_t]$ where L_t is the lowest price observed during that minute t, and H_t is the highest.
(2) If the closing price C_t is present, then volume V_t should be present. While this might seem redundant to the first Type 1 constraint mentioned above, it addresses a specific aspect of the process: volume is determined only after the close of the trading minute.
These are examples of Type 2 constraints. Using notation introduced above, let the violations of these two constraints be denoted by $g_1^2(t)$ and $g_2^2(t)$, and the aggregates across all stocks be denoted by

$$G_1^2(t) = \sum_{NYSE} g_1^2(t)$$

and

$$G_2^2(t) = \sum_{NYSE} g_2^2(t)$$

respectively. We could not find Type 2 glitches in this particular data stream that violated the constraints specified above, because, as mentioned earlier, they are cheaply detected and remedied. Since we could not find a Type 1 or Type 2 constraint violation using the TR guidelines, in order to demonstrate a Type 1 constraint, we applied a rule of thumb that states that blue chip stocks need to trade every minute. Since this type of glitch falls out naturally as a Type 3 glitch by learning the number of trades for each stock, we merely demonstrate it here to exemplify a Type 1 constraint violation. There are 390 minutes in a normal trading day, starting at 9:30 AM and ending at 15:59 PM. Sometimes there is an additional 391^{st} minute, a spillover record at the end of the day stamped 16:00 to report the overflow from the previous minute. So we could stipulate a simple Type 1 constraint on the daily totals: "every day should have at least 390 minute files". Figure 4 depicts daily total of trading minutes (corresponds to the number of files received) of the stock HAL. It usually fluctuates between 390 and 391.

There are noticeable violations on the days before Thanksgiving and July 4 respectively. This is expected since the stock exchange closes early on these days. There were less noticeable gaps on May 23, May 25 and June 6 when there were 3, 2 and 20 missing minutes. All these were detected as outlying "lags" by FIT, as a part of Type 3 glitch detection where the model expected the lag between successive trades to be one minute. For a heavily traded stock like HAL, gaps in trading are very unusual. The missing trading minutes on May 23, May 25 and June 6 are deemed data glitches (missing/incomplete data) since we could not find any explanation for them using readily available domain knowledge such as holiday trading patterns.

However, the most interesting missing data detected by FIT, but not by mere counting of total minutes, was on August 13. The total number of trading minutes were 390, a very common and acceptable value, hence not a violation of the Type 1 constraint stipulated above. However, FIT flagged a missing minute. Closer examination showed that the reason FIT generated a lag alarm was because of the missing minute at 13:54 causing an unusual lag of more than a minute in data reporting.

```
2012-08-13 13:52,13,34.76,34.81,34.76,34.81,16704
2012-08-13 13:53,13,34.8,34.8,34.78,34.79,5913
2012-08-13 13:55,13,34.79,34.82,34.79,34.8,14644
```

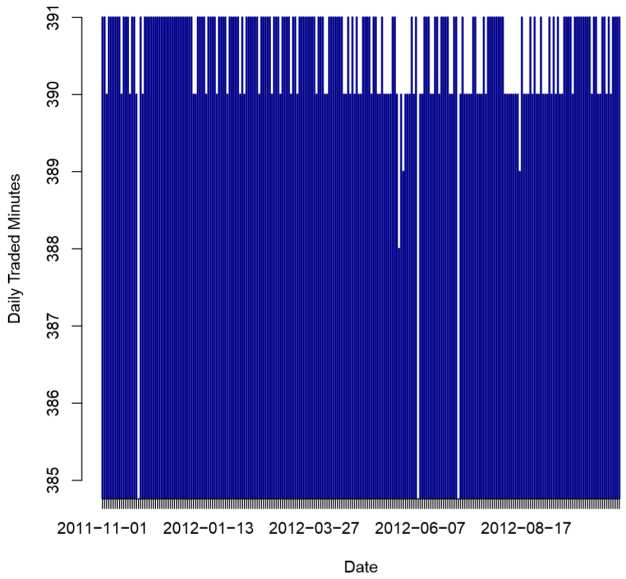


Figure 4: *The total daily trading minutes (corresponds to the number of files received) of the StockID 968 (HAL). It usually fluctuates between 390 and 391. There are noticeable gaps on the days before Thanksgiving and July 4 respectively. This is expected since the stock exchange closes early on these days. There were less noticeable gaps on May 23, May 25 and June 6 when there were 3, 2 and 20 missing minutes. All these were detected as outlying "lags" by FIT (Feed Inspection Tool), as a part of Type 3 glitch detection where the model expected the lag between successive trades to be one minute.*

```
2012-08-13 13:56,13,34.8,34.8,34.78,34.79,9154
```

The total counts remained at 390 due to the presence of the spillover record at 16:00.

```
2012-08-13 15:59,13,35.01,35.04,35,35.03,191139
2012-08-13 16:00,13,35.03,35.03,35.03,35.03,100
```

Thus, FIT provides a mechanism for detecting potentially missing records at the minute level, that could be overlooked by monitoring aggregates. This is an example where a simple Type 1 constraint would have failed, while a Type 3 constraint that takes into account the history of inter-trading lags catches the missing record.

3.4.3 Type 3 Constraints

Next, we validate Type 3 constraints. Because Type 3 and Type 4 constraints capture more interactions and interrelationships in the data, they tend to reveal complex glitches that might not be obvious. We expect the values of the high, low, open and close prices, and volume, to be within certain statistical ranges for each of the stocks. Similarly, derived variables such as the lag between successive trades LG_t, the price spread SP_t, and consecutive changes, e.g., for the high H_t, $DL_t = H_t - H_{t'}$ should all fall within observed statistical ranges.

We compute the expected behavior in two ways. The first, as expressed by [11], as a fixed percentage of the previous value. And second, using the statistical models employed by FIT. These are Type 3 constraints because they use multiple entities but within a single attribute. We articulated eight such constraints, one each for the five variables H_t, L_t, O_t, C_t, V_t and one for each of the derived variables SP_t, DL_t, LG_t. Note that in principle we could have derived variables for each of the original five variables, but for purposes of illustration we focus on

the price spread SP_t, the trading lag LG_t, and the change (delta) DL_t in successive values of the intra-minute high, H_t.

Therefore, there could be eight possible glitches of Type 3,

$$g_i^3(t), i = 1, \ldots, 8,$$

and the aggregates across all stocks and all types are given by:

$$G^3(t) = \sum_i G_i^3(t) = \sum_i \sum_{NYSE} g_i^3(t), i = 1, \ldots, 8.$$

The total of Type 3 glitches, $G^3(t)$, (since Type1 and Type 2 errors are all zero) is a simple measure of *statistical distortion*, the disparity between what we expect and what we actually observe. That is,

$$SD(t) = G^3(t).$$

The panel on the right in Figure 2 shows the results of running FIT on the total of Type 3 glitch counts, $G^3(t)$. Note that the inherent noise in the streams ensures that at any given time there will be a certain number of glitches. We check if the observed number of glitches are *above or below* the statistical variation. It is important to identify too few glitches too because that might indicate a drastic change, e.g., a system has malfunctioned and is sending bogus records, e.g., "price flats" where there is no variation at all. Here, the FIT model takes into account the hour of day. Big pink dots represent the morning hours, gradually changing into smaller blue dots as the day goes by. The red dots are anomalous values. There were several days that featured a high number of glitches. These potentially correspond to global events that affected a large number of stocks, like the Fed decision on September 12.

3.4.4 Type 4 Constraints

Finally, we cross reference with an independent data source to validate our data. We used the daily stock data available on Yahoo to validate the intra-minute high, H_t, on stocks. We formulate the following Type 4 constraint: "The highest price from the daily Yahoo file should be the same as the maximum of all the intra-minute highs in the NYSE minute bar file". The bar chart on the left panel of Figure 5 shows the distribution of the number of stocks that violated that constraint on a given number of days. The tallest blue bar shows that there were around 1100 stocks that matched the constraint almost every day during the observation period of over 250 days. The red bars on the right show stocks that violated that constraint on most days. The panel on the right shows an example stock of each of these types and plots the daily high against the maximum intra-minute high from the minute files. The blue stock is perfectly linear as expected, while the red stock's values are spread out including a very suspicious "price flat" indicative of a spurious interpolation of missing values.

Given the price flats in daily file, we suspect the NYSE data has better quality than the independent source.

3.5 Glitch Prone Stocks

Next, we defined the statistical distortion corresponding to a given stock S to be the proportion of glitches of all types in the entire stream corresponding to that stock. That is,

$$SD(S) = \frac{\sum_i G^i(S)}{N}, i = 1, \ldots, 4$$

where N=the total number of possible glitches. Note the implicit summation over time t. In the simplest case, if there are K data elements (for example, 390 traded minutes on each of 280 days described by 7 attributes would yield 764,400 data elements), and if each of these could have a maximum of C glitches associated with it, then

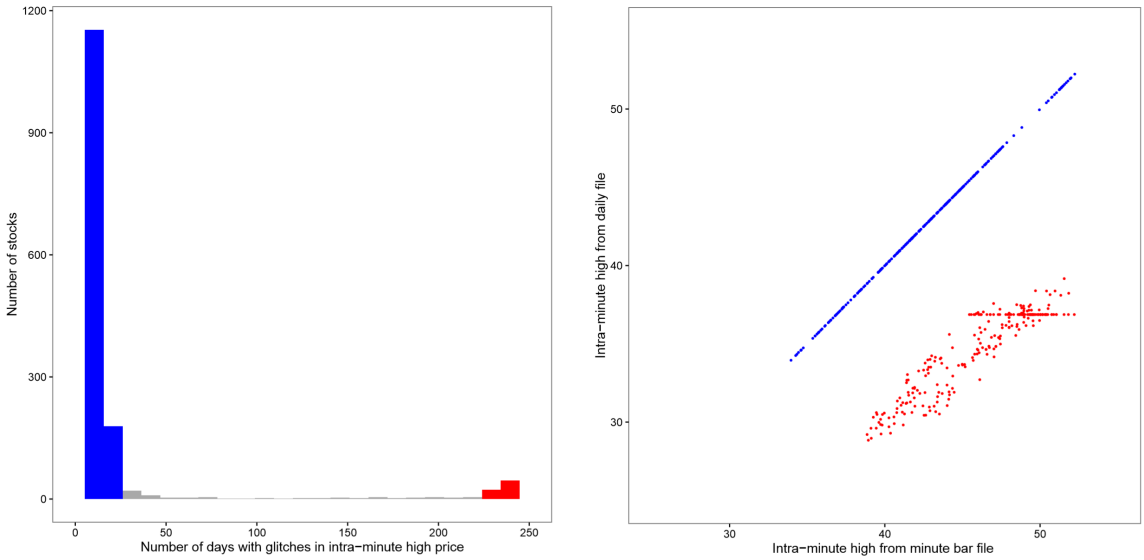


Figure 5: *We validated our data using an independent, external data source and formulating the following Type 4 constraint: "The highest price from the daily Yahoo file should be the same as the maximum of all the intra-minute highs in the NYSE minute bar file". The bar chart in the left panel shows the distribution of the number of stocks that violated a Type 4 constraint on a given number of days on the X-axis. The tallest blue bar shows that there were around 1100 stocks that matched the constraint almost every day during the observation period of over 250 days. The red bars on the right show stocks that violated that constraint on most days. The panel on the right shows an example stock of each of these types and plots the daily high against the maximum intra-minute high from the minute files. The blue stock is perfectly linear as expected, while the red stock's values are spread out including a very suspicious "price flat" indicative of a spurious interpolation of missing values in the external source.*

$N = K * C$. For example, if $C = 2$ (e.g., where the possible glitches are: missing OR duplicate OR outlier OR (duplicate AND outlier)), then $N = 1,528,800$ for the stock S.

We sorted the individual stocks by the statistical distortion during the period of observation. Figure 6, left panel, shows a plot of the glitch proportion (percentage) for individual stocks. Note the elbow in the curve at 3.5%. Based on this empirical observation, it is reasonable to consider the quality of data for stocks with statistical distortion greater than 3.5% to be suspect. The worst stock, with a statistical distortion of almost 7%, is represented by a purple dot at the top, its StockID given by 1881. The "cleanest" stock is shown by a gold dot at the bottom of the curve with a glitch proportion of less than 2%, with a StockID of 968, which we know from our previous studies to be Halliburton. The distributions of glitches in these two stocks are shown in Figure 6, right panel. The box plot for the purple stock 1881 is wider, indicating more glitched data, as well as more glitches in each minute indicating that there were more corrupt attributes per minute as well. Stock 968 on the other hand, when glitchy, had in general fewer glitches per minute, typically just 1.

In Figure 7, we show a typical day for each of these two stocks, for intra-minute high price, H_t. The cyan dots represent Thomson Reuters anomalies, while red dots represent FIT anomalies. For TR, we selected a percentage that yielded a number of anomalies comparable to FIT, since the actual percentage threshold changes from stock to stock. The black solid line represents the ideal D_t^I as computed by the streaming FIT model. The blue line represents observed values. In general, the TR method creates more alarms since it is based only on the previous value and a fixed percentage threshold. The FIT model used here is based on a window of 25 values, by hour, hence more robust but also more sensitive since it is customized to each hour.

Note that 1881 is a penny stock and hence any change is bound to be large on a percentage basis as evidenced

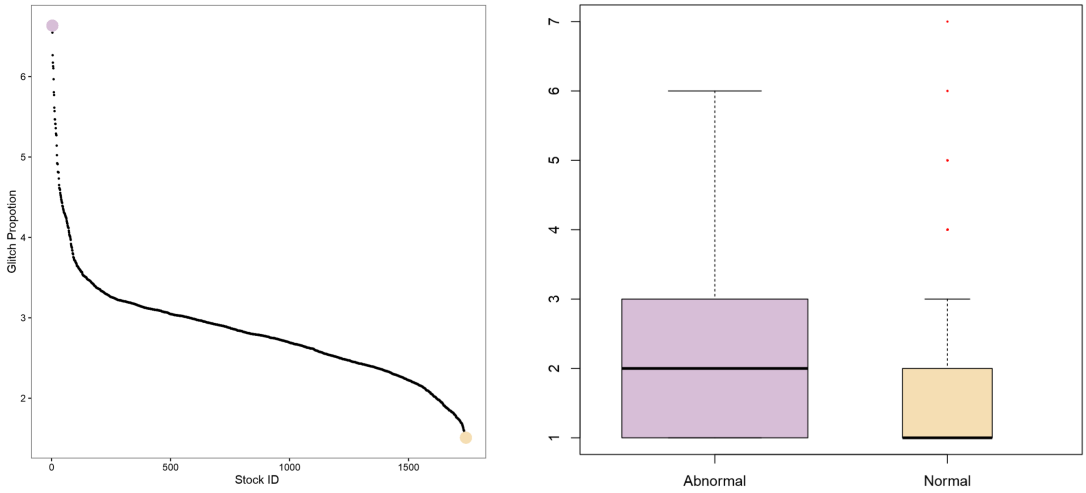


Figure 6: *Individual stocks sorted by the amount of statistical distortion as measured by proportion of glitches during the period of observation. Note the elbow in the curve at 3.5%: stocks with glitch proportion greater than 3.5% are suspect. The worst stock, with a statistical distortion of almost 7%, is represented by a purple dot at the top, Stock ID 1881. The "cleanest" stock is shown by a gold dot at the bottom of the curve with a glitch proportion of less than 2%, StockID 968. The distributions of glitches in these two stocks are shown in the right panel. The box plot for the purple stock 1881 is wider, indicating more glitched data, as well as more glitches in each minute indicating that there were more corrupt attributes per minute as well. Stock 968 on the other hand, when glitchy, had in general fewer glitches per minute, typically just 1.*

by TR's cyan dots. However, FIT alerts less often since it is based on statistical variation and picks up only significant changes. Stock 968 exhibits great volatility in the beginning of the plot. FIT considers this normal for the stock and does not alert while the TR method generates numerous alerts based on percentage change. Toward the middle, there is much less volatility and FIT identifies anomalies in an appropriate fashion even though the variability is smaller in absolute terms as compared to the beginning.

4 Conclusion

In this paper, we presented a framework for monitoring data quality in dynamic temporal streams. We employ four types of constraints in increasing order of data dependence, to define data quality problems or glitches. We use FIT, a statistical stream monitoring tool, to detect the glitches in near real time, capturing instances where current data D_t varies statistically from the ideal D_t^I computed in a data-driven fashion by FIT using historical data. We use *statistical distortion* to measure the disparity between the ideal and observed by tracking the proportion of glitches across stocks and within stocks. Our case study of financial data streams of 2000 NYSE stocks during the period November 2011 to November 2012 revealed interesting data quality phenomena some of which could be explained by our domain expert.

5 Acknowledgments

We would like to thank Kumar Doraiswami for donating a year's worth of NYSE minute bar data, and for being our domain expert, and Simon Urbanek for permitting us to annotate and use the underlying plot in Figure 1.

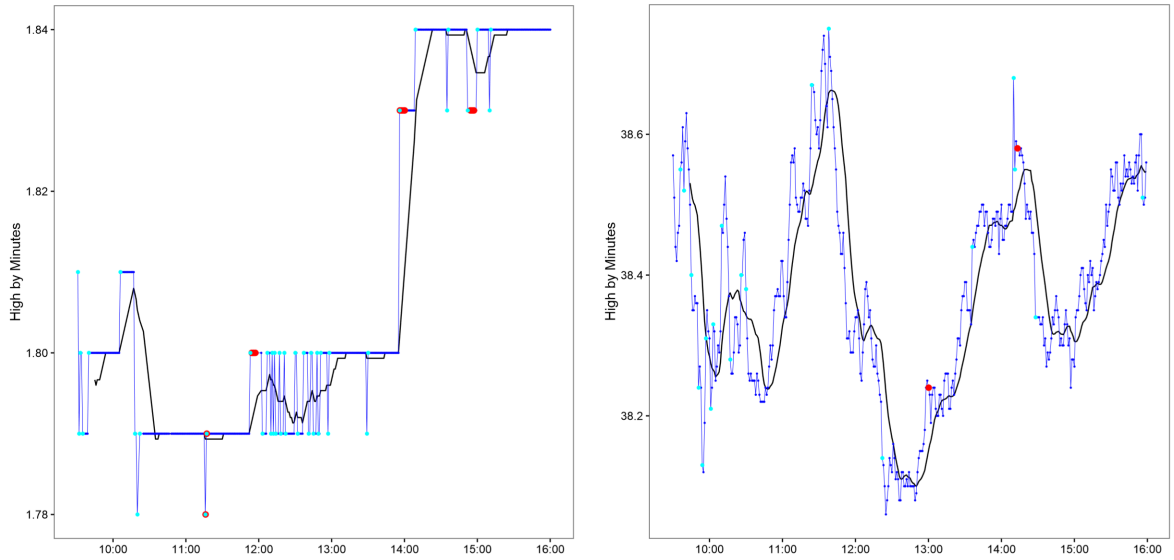


Figure 7: *A typical day for each of the stocks 1881 and 968, for the attribute H_t, the intra-minute high. The cyan dots represent Thomson Reuters anomalies, while red dots represent FIT anomalies. For TR, we selected a percentage that yielded a number of anomalies comparable to FIT, since the actual percentage threshold changes from stock to stock. The black solid line represents the ideal D_t^I as computed by the streaming FIT model. The blue line represents observed values. In general, the TR method generates more alarms since it is based only on the previous value and a fixed percentage threshold. The FIT model used here is based on a window of 25 days, hence more robust but also more sensitive since it is customized to each hour.*

References

[1] Z. Abedjan, L. Golab and F. Naumann, *Profiling relational data: a survey*, VLDB, 2015.

[2] L. Berti-Equille, T. Dasu and D. Srivastava, *Discovery of Complex Glitch Patterns: A Novel Approach to Quantitative Data Cleaning*, ICDE 2011.

[3] T. Johnson and T. Dasu, *Bellman: A Data Quality Browser*, 2001.

[4] L. Golab, H. J. Karloff, F. Korn, A. Saha and D. Srivastava, *Sequential Dependencies*, VLDB, 2009.

[5] S. Ginsburg and R. Hull, *Order Dependency in the Relational Model*, Theor. Comput. Sci., 26, 1983.

[6] S. Song, A. Zhang, J. Wang and P. S. Yu, *SCREEN: Stream Data Cleaning under Speed Constraints*, Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 827–841, 2015.

[7] Z. Abedjan, C. G. Akcora, M. Ouzzani, P. Papotti and M. Stonebraker, *Temporal Rules Discovery for Web Data Cleaning*, PVLDB, 9, 4, 336–347, 2015.

[8] T. Dasu and J. M. Loh, *Statistical Distortion: Consequences of Data Cleaning*, PVLDB, 2012.

[9] T. Dasu, J. M. Loh and D. Srivastava, *Empirical Glitch Explanations*, KDD, 2014.

[10] T. Dasu, V. Shkapenyuk, D. Srivastava, and D. F. Swayne, *FIT: Feed Inspection Tool for Data Streams*, PVLDB, 2015.

[11] Thomson Reuters, *Thomson Reuters Datastream: Data Quality Assurance*, 2010.

Quality-Aware Entity-Level Semantic Representations for Short Texts

Wen Hua[#], Kai Zheng[#†], Xiaofang Zhou[#†]
[#]School of ITEE, The University of Queensland, Brisbane, Australia
[†]School of Computer Science and Technology, Soochow University, Suzhou, China
w.hua@uq.edu.au, {kevinz, zxf}@itee.uq.edu.au

Abstract

Recent prevalence of Web search engines, microblogging services as well as instant messaging tools give rise to a large amount of short texts including queries, tweets and instant messages. A better understanding of the semantics embedded in short texts is indispensable for various Web applications. We adopt the entity-level semantic representation which interpretes a short text as a sequence of mention-enity pairs. A typical strategy consists of two steps: entity extraction to locate entity mentions, and entity linking to identify their corresponding entities. However, it is never a trivial task to achieve high quality (i.e., complete and accurate) interpretations for short texts. First, short texts are noisy, containing massive abbreviations, nicknames and misspellings. As a result, traditional entity extraction methods cannot detect every potential entity mentions. Second, entities are ambiguous, calling for entity linking methods to determine the most appropriate entity within certain context. However, short texts are length-limited, making it infeasible to disambiguate entities based on context similarity or topical coherence in a single short text. Furthermore, the platforms where short texts are generated are usually personalized. Therefore, it is necessary to consider user interest and its dynamics overtime when linking entities in short texts. In this paper, we summarize our work on quality-aware semantic representations for short texts. We construct a comprehensive dictionary and extend traditional dictionary-based entity extraction method to improve recall of entity extraction. Meanwhile, we combine three novel features, namely content feature, social feature and temporal feature, to guarantee precision of entity linking. Empirical results on real-life datasets verify the effectiveness of our proposals.

1 Introduction

Recent decades have witnessed the flourishing of Web search engines, microblogging services, as well as instant messaging tools. This results in an increasing amount of short texts, i.e., length-limited poorly-structured natural language texts. Short texts embed invaluable knowledge. For example, companies can estimate public support for their products by analyzing query logs; governments can discover potential threat by monitoring tweet streams. In order to harvest knowledge from short texts, we need to go beyond raw texts and discover semantics. In this paper, we adopt entity-level semantic representation, namely recognizing entities

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

from short texts. More formally, given a short text s, we need to obtain a sequence of mention-entity pairs $\{\langle m_1, e_1\rangle, \langle m_2, e_2\rangle, ..., \langle m_l, e_l\rangle\}$ where m_i is an entity mention (i.e., a noun phrase) detected from s and e_i refers to a real-world entity recorded in large-scale machine-understandable knowledgebases. We use Wikipedia[1] as an example knowledgebase in the rest of the paper. Fig. 1 depicts an example of entity-level semantic representation wherein three mention-entity pairs are identified from the given tweet, namely $\{\langle m_1$ ="allen iverson", e_1 =*Allen Iverson (basketball)*$\rangle$, $\langle m_2$ ="michael jordan", e_2 =*Michael Jordan (basketball)*$\rangle$, $\langle m_3$ ="shaquille o neal", e_3 =*Shaquille O'Neal (basketball)*$\rangle\}$.

Figure 1: An example of entity-level semantic representation.

A typical strategy for obtaining entity-level semantic representation consists of two steps: entity extraction which locates entity mentions $\{m_1, m_2, ..., m_l\}$ in a given short text s, and entity linking which identifies the entity e_i that each mention m_i refers to. The quality of semantic representation determines the quality of knowledge discovered from short texts, which in turn affects user experience of the aforementioned Web applications. We consider two types of quality criteria in this work: *completeness* and *accuracy*. In other words, we aim to extract every possible entity mentions from a short text, and meanwhile find the most appropriate (i.e., semantically coherent with the context) entities for these mentions. However, the noisy, contextualized and personalized textual sources introduce some unique challenges for obtaining high quality entity-level semantic representations for short texts. In the following, we demonstrate quality issues in short texts with several examples, and discuss the limitations of existing methods to address these problems.

Data quality problem 1: noisy text. There have been extensive efforts on entity extraction which can be classified into two categories, namely linguistic-based and dictionary-based. Linguistic-based approaches incorporate linguistic features, such as capitalization, digitalization, punctuation, part-of-speech tags and so forth, into a machine learning model (e.g., Support Vector Machine [1, 2], Maximum Entropy Model [3, 4, 5], Hidden Markov Model [6] and Conditional Random Field [7]) to detect entity mentions. However, short texts are informal and do not always observe linguistic rules, which makes traditional linguistic features (e.g., capitalization) inapplicable to entity extraction from short texts. Dictionary-based approaches [10, 11] are becoming increasingly popular nowadays due to their simplicity and real-time nature. They extract entity mentions in a streaming manner by checking for existence or frequency of a noun phrase in a predefined dictionary of entity mentions. In particular, the widely-used Longest Cover method searches for longest noun phrases contained in the dictionary while scanning a text. Note that most dictionary-based approaches implicitly require noun phrases to exactly match at least one element in the dictionary. Whereas, short texts are noisy and full of abbreviations, nicknames, and misspellings. For example, "new york city" is usually abbreviated to "nyc" and known as "big apple". Hence, we need to add some flexibility to existing dictionary-based entity extraction methods to guarantee the completeness of semantic representations for short texts.

[1]Wikipedia data is publicly available at `https://dumps.wikimedia.org/enwiki/`

Data quality problem 2: entity ambiguity. A knowledgebase can be regarded as a huge collection of mentions and entities as well as mappings between them. Hence, we can obtain the entity that a mention refers to directly from such mapping information. However, there exists one-to-many mappings between mentions and entities. In other words, a specific entity mention can correspond to multiple real-world entities. For example. "jordan" could be a mention of *Jordan (country)*, *Air Jordan*, *Michael Jordan (basketball)*, as well as *Michael Jordan (machine learning expert)*. An accurate semantic representation requires to find the most appropriate entity for each mention detected in a short text. To this end, it is indispensable for entity linking methods to resolve entity ambiguity. Existing approaches to entity linking [12, 13, 14, 15, 16, 17, 18] are mostly content-based and targeted on documents. They utilize a combination of three features, namely entity popularity, context similarity and topical coherence, to estimate the weights of candidate entities. In particular, the *entity popularity* feature assumes that users' attention is usually focused on a small subset of entities, and hence the historical frequency information can be regarded as a hint for entity linking. Take the entity mention "jordan" as an example. It is more possible that users are talking about entity *Air Jordan* rather than *Jordan (country)* considering its relatively larger popularity. The *context similarity* feature measures mention-entity correspondence by calculating similarity between texts around entity mentions and documents describing entities in a knowledgebase. The *topical coherence* feature assumes that entities mentioned in a single document should be topically coherent, and handles mentions collectively by considering semantic relatedness between corresponding entities. Despite the satisfying performance these features have achieved in documents, the accuracy of entity linking decreases dramatically in short texts due to their length limitation [19]. Short texts cannot provide sufficient information to calculate context similarity accurately, nor can they provide enough mentions to derive a joint and interdependent entity assignment based on cross-entity relationships. Another factor that could affect the accuracy of entity linking in short texts is the personalized nature of Web applications where short texts are generated. Take microblogging services as an example. Users interested in basketball are more likely to talk about *Michael Jordan (basketball)*, while users interested in computer science tend to mention *Michael Jordan (machine learning expert)* in her postings. Consequently, it is necessary to take user interest into consideration when linking entities in such personalized short text dataset. [20, 21, 22] believe that users' interest is scattered in the messages they broadcast, and adopt a content-based method to discover user interest. They construct a graph model between candidate entities detected from the whole set of short texts published by a single user and conduct entity linking collectively. However, the topics of users' postings vary significantly, making it inaccurate to infer user interest from such a diverse stream of short texts. Furthermore, a large amount of users are information seekers with limited broadcasting history [23], which also increases the difficulty of learning their interest. Finally, users' interest can be influenced by recent events and change over time. For example, *Michael Jordan (basketball)* is more likely to be mentioned during NBA seasons while *Michael Jordan (machine learning expert)* is probably a better candidate entity when ICML (International Conference on Machine Learning) is being held. Therefore, the dynamics of user interest should also be considered, in order to guarantee the accuracy of semantic representations for personalized short text sources.

In this paper, we summarize our work on entity-level semantic representations for short texts, with a focus on how we resolve the aforementioned data quality problems and thus guarantee the quality (i.e., completeness and accuracy) of semantic representations. Fig. 2 illustrates an overview of our framework. We adopt the two-step strategy for obtaining semantic representations. Specifically, we construct a large-scale dictionary from existing knowledgebases and extend traditional dictionary-based methods to allow for approximate entity extraction from noisy short texts (Sec. 2); we combine three novel features, namely content feature, social feature and temporal feature, which are calculated based on some pre-collected statistical information to resolve entity ambiguity in short texts (Sec. 3). We empirically evaluate our framework on real-life datasets (i.e., queries and tweets) and present some of the experimental results in Sec. 4.

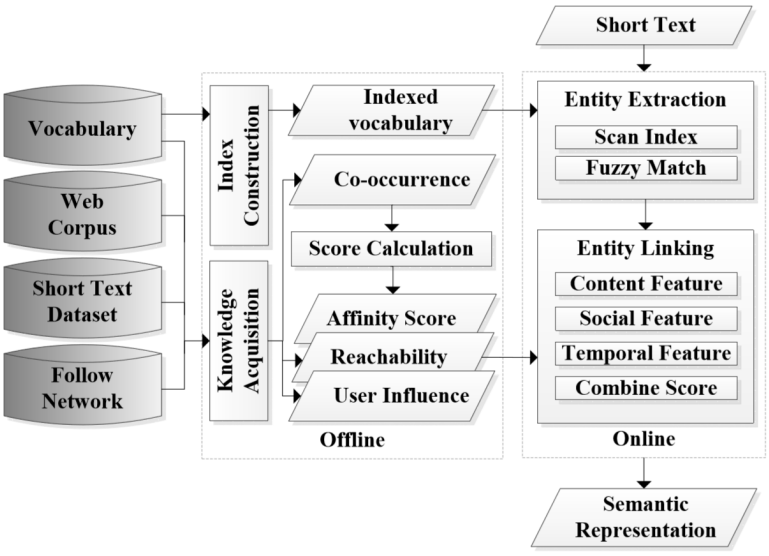


Figure 2: Framework overview.

2 Entity Extraction

Entity extraction is the first step for obtaining entity-level semantic representations. It detects entity mentions from texts written in a natural language. We adopt the widely-used dictionary-based approach to extract entity mentions, considering its simplicity and real-time nature. Since short texts are full of abbreviations, nicknames and misspellings, we need to handle this noise specifically, in order to achieve a complete set of entity mentions from a short text.

2.1 Handling abbreviations and nicknames

A large-scale dictionary (also called vocabulary, lexicon, or gazetteer) is a prerequisite for dictionary-based entity extraction, which is usually constructed from existing knowledgebases. Wikipedia is organized as a collection of Web pages including entity pages, disambiguation pages and redirect pages. Each entity page corresponds to a specific entity, and contains a detailed description about that entity. We can obtain the set of entities from Wikipedia' entity pages. Disambiguation page of a mention consists of a list of hyperlinks to entities it can refer to, from which we can extract the one-to-many mappings between mentions and entities. For example, given disambiguation page of "harry potter", we correspond mention "harry potter" to entities *Harry Potter (book)*, *Harry Potter (film)*, *Harry Potter (character)* and *Harry Potter (journalist)*. Redirect page is a virtual page which jumps to a specific entity page. Therefore, the URIs of redirect page and corresponding entity page can be used to extract abbreviations and nicknames of entities. For instance, the redirect page of "nyc" links to the entity page of "New York City", and hence "nyc" should be an abbreviation of entity *New York City*. Another information that can help reducing noise in short texts is the hyperlink structure between Wikipedia' entity pages. From the anchor texts of hyperlinks (e.g., "big apple") and the entity pages they point to (e.g., *New York City*), we can also obtain the abbreviations and nicknames of entities. In this way, we construct a dictionary which contains a huge collection of entity mentions along with their abbreviations and nicknames[2].

[2]The dictionary of abbreviations and nicknames is publicly available at `http://probase.msra.cn/dataset.aspx`

2.2 Handling misspellings

Approximate entity extraction is necessary to cope with misspellings in short texts. It locates substrings in a text that are similar to some dictionary entries. To quantify the similarity between two strings, many similarity functions have been proposed including token-based similarity functions (e.g., jaccard coefficient) and character-based similarity functions (e.g., edit distance). We choose edit distance as our similarity function since it is more suitable for handling misspellings.

We adopt and extend the trie-based method [30] for approximate entity extraction. That is, given an edit distance threshold τ, we divide each entity mention into $\tau + 1$ segments evenly. The pigeonhole principle guarantees that if a substring is similar to an entity mention with respect to τ, it must contain at least one segment of that mention. We build a segment-based inverted index on the entire dictionary, where the entries are segments and each segment is associated with an inverted list of entity mentions containing the segment. Given a short text, we adopt the search-extension algorithm proposed in [30] to find all possible mentions. In other words, we first enumerate every substring of a short text and check whether it matches a segment using the trie structure. In this way, we obtain a set of segments contained in the short text. Then for each segment and the corresponding substring, we extend the substring to a longer substring similar to a mention in the inverted list. The most notable limitation of the existing trie-based framework is that it utilizes one specific edit distance threshold τ. However, our dictionary contains a large amount of abbreviations as well as multi-word entity mentions which require different edit distance thresholds. For example, in order to recognize misspelled multi-word mentions, we sometimes need a large edit distance threshold of at least 2. But when we apply the same edit distance threshold to abbreviations, it will lead to mistakes (e.g., "nyc" and "ntu" will be regarded as similar). To this end, we extend the trie-based framework to allow for various edit distance thresholds at the same time. The problem is how to determine the value of τ for different entity mentions. It can be expected that τ depends on the length of mentions. In other words, the longer a mention is, the more possible it will be misspelled and the more mistakes there will be. Therefore, we collect a large-scale short text dataset from search engines and microblogging sites, and invite annotators to label misspelled mentions along with their edit distances. We observe a near step-like distribution between edit distance and mention length, which is then used as our guideline for determining edit distance threshold for different entity mentions.

3 Entity Linking

Entity linking resolves entity ambiguity, i.e., the one-to-many mappings between mentions and entities. In other words, it estimates the weights of candidate entities and finds the best entity for a given mention within certain context. As discussed in Sec. 1, traditional content-based entity linking approaches cannot be directly applied to short texts due to length limitation and personalized nature. We introduce three novel features to guarantee the accuracy of entities recognized from short texts. Formally, given the set of candidate entities $E_m = \{e_1, e_2, ..., e_n\}$ for a mention m published by user u, the weight of each entity $S(e)$ is a combination of content feature, social feature and temporal feature.

$$S(e) = \alpha \cdot S_{content}(e) + \beta \cdot S_{social}(u, e) + \gamma \cdot S_{temporal}(e). \quad (1)$$

In Eq. 1, α, β and γ ($\alpha + \beta + \gamma = 1$) are coefficients that represent relative contributions of content feature, social feature and temporal feature to the overall weighing function respectively, which can be manually defined or automatically learned using machine learning algorithms. We describe these three features in detail in the following sections.

3.1 Content feature

Short texts do not have sufficient content to calculate context similarity between mentions and candidate entities accurately. Meanwhile, the number of mentions that can be extracted from a short text are usually limited, making the topical coherence feature between entities inapplicable in short texts. As an alternative, we dig into semantic relatedness between any types of terms (e.g. verbs and adjectives, in addition to entities) to assist entity linking in short texts. Consider the tweet "wanna watch harry potter tonight" as an example. Only one mention "harry potter" can be detected, and hence we cannot apply topical coherence to determine the best entity for "harry potter" in this tweet. However, given the knowledge that the verb "watch" is much more semantically related to *Harry Potter (film)* than *Harry Potter (book)*, *Harry Potter (character)* and *Harry Potter (journalist)*, we can successfully identify *Harry Potter (film)* as the best entity for "harry potter" in this tweet according to such relatedness information.

The key technique here is to ensure the accuracy of relatedness calculation between terms. In this work, we consider relatedness in terms of both similarity and co-occurreence. That is, two terms are related if they are semantically similar or they frequently co-occur within certain context. Therefore, we propose an *affinity socre* $S_{affinity}(x, y)$ to denote semantic relatedness between two terms x and y, which is defined as the maximum value of similarity score and co-occurreence score.

$$\begin{aligned} S_{affinity}(x, y) &= \max(S_{sim}(x, y), S_{co}(x, y)) \\ &= \max(cosine(\vec{c}_x, \vec{c}_y), cosine(\vec{c}_{co(x)}, \vec{c}_y)). \end{aligned} \tag{2}$$

$S_{sim}(x, y)$ in Eq. 2 denotes semantic similarity between terms x and y, which is calculated by the cosine similarity between their category distributions $\vec{c}_x$ and $\vec{c}_y$, namely $S_{sim}(x, y) = cosine(\vec{c}_x, \vec{c}_y)$. Each entity is classified into several categories in Wikipedia. For example, entities *Michael Jordan (basketball)* and *Shaquille O'Neal (basketball)* are semantically similar, since they share a large amount of categories such as "NBA all-stars", "basketball players", "business people" and so on.

$S_{co}(x, y)$ in Eq. 2 represents co-occurreence score between terms x and y. Some existing knowledgebases, such as WordNet, have already incorporated information about co-occurrence or relatedness between terms. However, we observe that terms of different types co-occur with different context. For instance, the verb "watch" co-occurs with entity *Harry Potter (movie)*, while the entity *Watch* co-occurs with entity *Omega SA*. Therefore, a more accurate co-occurrence network should be constructed between terms with specific types. We observe that

- The more frequently two terms co-occur and the closer they locate in a certain sentence, the larger their semantic relatedness should be;
- Common terms (e.g., "item" and "object") are meaningless in modeling semantic relatedness, and thus should be penalized.

Based on these observations, we automatically analyze a large-scale Web corpus (e.g., Wikipedia entity pages or general Web pages) and compute co-occurrence strength based on such factors as frequency, distance, and tf-idf measure. In this way, we obtain a co-occurrence network between verbs, adjectives and entities[3]. During the construction of co-occurrence network, some co-occurring information might be missing due to limited coverage of the Web corpus. As demonstrated in Fig. 3, we cannot find a sentence in the Web corpus that contain both "watch" and "harry potter", and hence the co-occurring information between the verb "watch" and the entity *Harry Potter (film)* is missing. Such a phenomenon will affect the accuracy of semantic relatedness. Therefore, we transform the original entity-level co-occurrence network into a category-level co-occurrence network by mapping entities to their categories. The nodes in the category-level co-occurrence network are

[3]The co-occurrence network is publicly available at `http://probase.msra.cn/dataset.aspx`

verbs, adjectives and categories, and the edge weights are aggregated from the original network. Given the knowledge that "watch" co-occurs with category "film", we can indirectly recover the co-occurring relationship between "watch" and entity *Harry Potter (film)*. Let $\vec{c}_{co(x)}$ and $\vec{c}_y$ denote the set of categories x co-occurs with in the category-level co-occurrence network and the set of categories y belongs to respectively. We observe that the larger the overlapping between these two sets, the stronger the relatedness between terms x and y, namely $S_{co}(x, y) = cosine(\vec{c}_{co(x)}, \vec{c}_y)$.

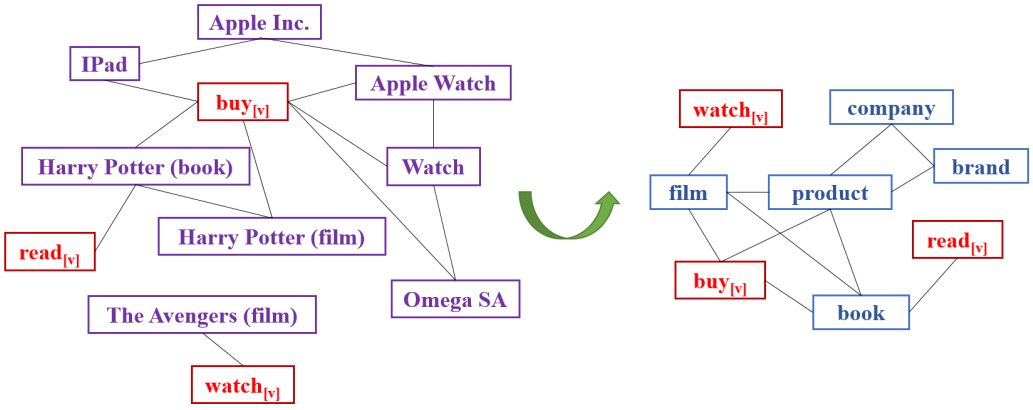


Figure 3: Examples of entity-level and category-level co-occurrence networks.

Based on Eq. 2, we can obtain semantical support, namely affinity score, of any contextual term recognized in short text s for candidate entity e. We choose the largest one as the content feature for entity linking.

$$S_{content}(e) = \max_{x \in s} S_{affinity}(x, e). \tag{3}$$

3.2 Social feature

In personalized web applications where short texts are generated, it is indispensable to consider user interest when conducting entity linking. As discussed in Sec. 1, traditional user interest modeling approaches based on historical broadcastings are inaccurate, due to the diverse range of topics embedded in messages and the existence of information seekers who tweet rarely. In this work, we resort to social interactions between users to indirectly infer user interest. We consider the "following" relationship in microblogging sites as an example, but our model can be easily extended to other platforms.

Microblogging users follow others to subscribe to tweets they are interested in. This means user u's interest in entity e can be reflected by her interest in following the set of users broadcasting about e. We define such a set of users as a *community* U_e which can be obtained by pre-processing a corpus of historical tweets using current state-of-the-art entity linking method [22]. We adopt reachability checking to estimate a user's interest in following another user, as formulated in Eq. 4. There are two issues we need to handle carefully to guarantee the accuracy of user interest estimation.

$$\begin{aligned} S_{social}(u, e) = S_{interest}(u, e) = S_{interest}(u, U_e) &= \frac{\sum_{v \in U_e} Reach(u, v)}{|U_e|} \\ &\approx S_{interest}(u, U_e^*) = \frac{\sum_{v \in U_e^*} Reach(u, v)}{|U_e^*|}. \end{aligned} \tag{4}$$

First, the *small-world* phenomenon in microblogging services [31] indicates that only reachable does not necessarily mean interested. Consequently, reachability which checks connectedness between two users should

be weighted, in order to achieve a more meaningful measurement of user interest. We consider both the distance and the strength of connection to weigh reachability between users. More formally,

$$Reach(u, v) = \frac{1}{d_{uv}} \cdot \frac{|F_{uv}|}{|F_u|} \tag{5}$$

In Eq. 5, d_{uv} is the shortest path distance from u to v. F_u denotes the collection of u's followees, and F_{uv} represents u's followees participating in at least one shortest path from u to v. Therefore, $\frac{|F_{uv}|}{|F_u|}$ actually reflects the strength of connection between u and v.

Second, different people have different influences in a community, and a user's interest in influential people contributes more to her interest in the community. To improve the accuracy of user interest estimation, we propose to detect a collection of most influential users for each community (denoted as U_e^*) and aggregate weighted reachability only with those influential users, as depicted in Eq. 4. Intuitively, a user is influential in a community associated with an entity e if:

- She is enthusiastic in broadcasting about entity e;
- She is discriminative among candidate entities E_m. This means an influential user should have a specific and continuous interest in broadcasting about entity e. For example, NBA's official account in Twitter (i.e., @NBAOfficial) hardly broadcasts about entities *Jordan (country)*, *Air Jordan* or *Michael Jordan (machine learning expert)*, making u's subscription to @NBAOfficial an important hint of her interest in basketball. Therefore, @NBAOfficial can be regarded as a discriminative and influential user in the community associate with entity *Michael Jordan (basketball)*.

Based on these heuristics, we propose a tfidf-based approach and an entropy-based approach to calculate user u' influence in the community associate with entity e. We consider the proportion of tweets published by user u about entity e to formulate the first heuristic in both approaches. As for the second heuristic, the tfidf-based approach measures the percentage of candidate entities u has mentioned in her tweets using the idf model, whereas the entropy-based approach examines the shape of probability distribution of u's historical tweets on candidate entities using the entropy model. In practice, it is common that an influential user in a community (say @NBAOfficial) occasionally tweets about candidate entities of other communities (say *Air Jordan*). Such an incident posting should not cause huge impact on her influence in the original community. In this sense, the entropy-based approach is superior to the tfidf-based approach in modeling user influence.

3.3 Temporal feature

As discussed in Sec. 1, users' interest is dynamic and can be influenced by recent events. Therefore, we need to capture the variation of user interest to further improve the precision of entity linking. Generally speaking, users are interested in entities involved in recent events or those attracting much public attention recently. We propose a temporal feature called *entity recency* to model an entity's recent popularity. Entity recency can be identified when a burst of tweets about that entity occurs during recent time period. In this work, we adopt a simple but effective approach to measure entity recency - sliding window. Formally, given a time window τ, we define entity recency using Eq. 6 where D_e^τ denotes the set of recently-published tweets about entity e.

$$S_{temporal}(e) = Recency(e) = \begin{cases} \frac{|D_e^\tau|}{\sum_{e_i \in E_m} |D_{e_i}^\tau|} & |D_e^\tau| \geq \theta_1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Besides a burst of tweets about entity e, recency can also be indirectly signified by that of related entities. For example, the recency of *Chicago Bulls* and *NBA* enhances that of *Michael Jordan (basketball)*. Similarly, increasing amount of tweets about *ICML* implies more attention on machine learning experts like Michael Jordan *(machine learning expert)*. Therefore, we propose a *recency propagation model* to incorporate mutual reinforcement of recency between related entities.

- Since entity recency is used as a feature for entity linking, it should not be propagated between candidate entities of the same entity mention, such as *Jordan (country)*, *Air Jordan*, *Michael Jordan (basketball)* and *Michael Jordan (machine learning expert)*
- If two entities are more topically related with each other, recency should be propagated between them in a larger extent;
- Only highly-related entities can reinforce each other's recency. This avoids extensive recency diffusion to slightly-related entities.

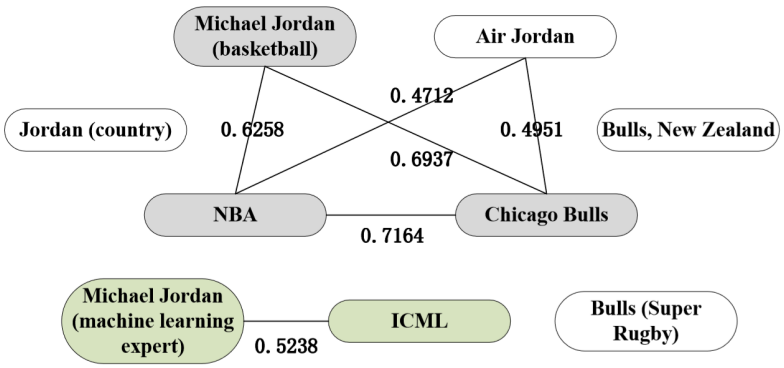


Figure 4: An example of recency propagation model.

Fig. 4 illustrates an example of the recency propagation model, which is formalized as an undirected graph on knowledgebase entities. Based on the above heuristics, edges are added only between highly-related entities corresponding to different entity mentions, and edge weights are defined based on semantic relatedness. We can use the affinity score described in Sec. 3.1 or the well-known Wikipedia Link-based Measure (WLM) [13] to model semantic relatedness between entities. Given the recency propagation model, we adopt a PageRank-like algorithm to combine recency gathered from underlying tweets and that reinforced by related entities.

4 Empirical Evaluation

We conducted extensive experiments on real-life datasets to evaluate the performance of our proposals. All the algorithms were implemented in C#, and all the experiments were conducted on a server with 2.90GHz Intel Xeon E5-2690 CPU and 192GB memory.

4.1 Benchmark

We briefly describe the dictionary and test datasets used in this work. In fact, our framework is generalized and can be applied to other dictionaries and short text platforms with slight extensions.

Dictionary. We downloaded the July 2014 version of English Wikipedia to build our dictionary for entity extraction. The Wikipedia dump contains 19.2 million entity pages, 6.3 million redirect pages, 0.2 million disambiguation pages, as well as 380 million hyperlinks between entity pages. Using the strategy described in Sec. 2.1 we obtained a huge dictionary of 29.3 million mentions including abbreviations and nicknames, and 19.2 million entities.

Test datasets. We constructed the test datasets by randomly sampling queries and tweets from a Web search engine (i.e., Bing) and a microblogging site (i.e., Twitter). We removed queries and tweets which contain entity mentions that cannot be recognized from our dictionary due to insufficient coverage. Altogether we obtained

1478 queries and 649 tweets. We also preprocessed the tweet dataset to remove some tweet-specific features such as @username, hashtags, urls, etc. We invited colleagues to annotate the test datasets, and the final labels were based on majority vote.

4.2 Effectiveness of proposals

Our empirical evaluation was divided into two parts according to the data quality problems discussed in Sec. 1. First, we evaluated whether our approach for entity extraction can effectively resolve textual noise, i.e., abbreviations, nicknames, and misspellings, in short texts. Second, we evaluated the performance of the proposed features, i.e., content feature, social feature, and temporal feature, compared with other features adopted in existing entity linking methods.

Effectiveness of entity extraction. Entity extraction locates entity mentions in a natural language text. In order to cope with abbreviations and nicknames in short texts, we construct a huge dictionary which incorporates not only entity mentions but also their abbreviations and nicknames using the strategy described in Sec. 2.1. We denote this dictionary as dic^*, and compare it with a preliminary dictionary dic containing only entity names. We use exact matching method to find entity mentions, and report the performance in terms of precision, recall and f1-measure in Table 1. Precision is the fraction of detected mentions that are labeled as correct, while recall is the fraction of correct mentions that are detected from the test dataset. More formally, precision $p = \frac{|M_{algo} \cap M_{label}|}{|M_{algo}|}$ and recall $r = \frac{|M_{algo} \cap M_{label}|}{|M_{label}|}$ where M_{algo} and M_{label} represent the set of entity mentions detected from the test dataset and those labeled by annotators respectively. F1-measure $f_1 = 2 \cdot \frac{p \cdot r}{p+r}$. From Table 1 we can see that exact matching based on dic^* can extract more entity mentions than dic, since dic^* enables matching method to recognize abbreviations and nicknames. And the increase of recall in the tweet dataset is slightly larger than in the query dataset, due to more frequent usage of abbreviations and nicknames in tweets. However, dic^* might cause more extraction errors sometimes. For example, given the information that "it" is an abbreviation of entity *Information Technology* in dic^*, exact matching will mistakenly recognize "it" in tweet "@payalpatel95 you're welcome! I never did it lol" as an entity mention. Such a phenomenon is especially common in the tweet dataset, since tweets are usually sentence-like while queries are keyword-like. This explains the much lower precision achieved in the tweet dataset than in the query dataset.

Table 1: Different dictionaries for entity extraction.

		precision	recall	f1-measure
query	dic	0.993	0.826	0.902
	dic^*	0.988	0.847	0.912
tweet	dic	0.707	0.682	0.694
	dic^*	0.627	0.718	0.669

We also apply approximate entity extraction to handle misspellings in short texts. We adopt and extend the trie-based method [30] to allow for approximate entity extraction with varying edit distance thresholds. We compare the performance of our approach (i.e., Trie with Varying edit distance, $TrieV$) with the trie-based method (i.e., $Trie$) and exact matching method (i.e., $Exact$), in terms of precision, recall, and f1-measure. From Table 2 we can see that approximate entity extraction can obtain more entity mentions from short texts than exact matching, at the cost of introducing slightly more extraction errors. By allowing for various edit distance thresholds depending on text length, $TrieV$ improves the precision of $Trie$ by reducing extraction errors caused by short entity mentions, abbreviations, etc. Overall, $TrieV$ achieves the highest f1-measure in both datasets. Note that the increase of recall in the tweet dataset is also larger than that in the query dataset, due to more misspellings in tweets.

Effectiveness of entity linking. We evaluate the performance of entity linking methods in terms of precision, and we only examine whether the correctly detected mentions are correctly linked. Formally, we calculate

Table 2: Different matching strategies for entity extraction.

		precision	recall	f1-measure
query	$Exact$	0.988	0.847	0.912
	$Trie$	0.943	0.922	0.932
	$TrieV$	0.984	0.918	0.950
tweet	$Exact$	0.627	0.718	0.669
	$Trie$	0.579	0.847	0.688
	$TrieV$	0.618	0.833	0.710

precision as $p = \frac{|M^*_{algo}|}{|M_{algo} \cap M_{label}|}$ where M^*_{algo} denotes the set of correctly linked mentions detected from the test dataset. Table 3 depicts the precision of entity linking based on five different combinations of features:

- [19]: consider topical coherence between entities.
- [22]: combine topical coherence between entities and user interest estimated from historical messages;
- $content$: consider semantic relatedness between any types of terms (e.g., verbs and adjectives, in addition to entities) based on co-occurrence relationship;
- $content + social$: combine semantic relatedness between any types of terms and user interest estimated through social interactions;
- $content + social + temporal$: combine semantic relatedness between any types of terms, user interest estimated through social interactions, as well as the dynamics of user interest overtime modeled as entity recency.

In Table 3, we only present the entity linking precision achieved by [19] and $content$ features for the query dataset, since we could not obtain author or timestamp information associated with each query when constructing this dataset which, however, is necessary to compute the social and temporal features. We obtain several observations from Table 3. First, the entity linking precision is consistently higher in the query dataset than in the tweet dataset. This is mainly because a large proportion of entities mentioned in tweets are celebrities and locations which are more ambiguous and harder to disambiguate. Second, $content$ performs better than [19] by considering topical coherence between any types of terms rather than only that between entities. Such a precision improvement is much more significant in the query dataset than in the tweet dataset, also due to the prevalence of celebrities and locations mentioned in tweets which cannot be disambiguated based on verbs or adjectives. Third, the precision of entity linking can be further increased by combining intra-tweet topical coherence with user interest information. Specifically, [22] discovers user interest from historical tweets and achieves larger precision than [19]. However, the improvement is limited due to the existence of information seekers in Twitter who cannot provide sufficient tweeting historical for interest estimation. Our proposed social feature, on the contrary, infers user interest based on social interactions, and hence increases precision to a larger extent. Fourth, the change of user interest overtime is also a crucial factor that should be considered when conducting entity linking in dynamic platforms such as microblogging sites. By combining all the three proposed features, namely content feature, social feature, and temporal feature, our framework achieves the best performance.

Table 3: Different features for entity linking.

	[19]	[22]	$content$	$content + social$	$content + social + temporal$
query	0.7104	-	0.8901	-	-
tweet	0.6667	0.6860	0.6777	0.7273	0.7315

5 Conclusion

Entity extraction and entity linking are basic steps for obtaining entity-level semantic representations for short texts. High quality, i.e., complete and accurate, semantic representations bring tremendous benefits to many Web applications. However, the noisy, personalized and dynamic nature of short text sources imposes unique challenges on both entity extraction and entity linking. In this paper, we summarize our work on semantic representations for short texts with a focus on the quality issues. Specifically, we construct a huge dictionary to incorporate abbreviations and nicknames, and extend the segment-based indexing structure on the dictionary to enable approximate entity extraction and thus reduce the impact of misspellings in short texts. Considering the prevalence of entity ambiguity in short texts and the limitations of traditional content-based entity linking approaches, we propose to combine three novel features, namely content feature (i.e., semantic relatedness between terms), social feature (i.e., user interest by social interactions), and temporal feature (i.e., entity recency which models the dynamics of user interest), to improve the accuracy of entity linking. Details on these features can be found in [24] and [25]. We report in this paper some of our empirical results on real-life datasets to examine the effectiveness of our framework in terms of precision and recall. The experimental results demonstrate significantly better performance of our proposals, compared with current state-of-the-art methods.

6 Acknowledgement

This work was partially supported by the ARC project under Grant No. DP140103171 and the NSFC project in Soochow under Grant No. 61472263.

References

[1] K. Takeuchi and N. Collier. Use of support vector machines in extended named entity recognition. In *CONLL*, pages 1–7, 2002.

[2] H. Isozaki and H. Kazawa. Efficient support vector classifiers for named entity recognition. In *COLING*, pages 1–7, 2002.

[3] H. L. Chieu and H. T. Ng. Named entity recognition: A maximum entropy approach using global information. In *COLING*, pages 1–7, 2002.

[4] O. Bender, F. J. Och, and H. Ney. Maximum entropy models for named entity recognition. In *CONLL*, pages 148–151, 2003.

[5] J. R. Curran and S. Clark. Language independent ner using a maximum entropy tagger. In *CONLL*, pages 164–167, 2003.

[6] G. Zhou and J. Su. Named entity recognition using an hmm-based chunk tagger. In *ACL*, pages 473–480, 2002.

[7] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *CONLL*, pages 188–191, 2003.

[8] X. Liu, S. Zhang, F. Wei, and M. Zhou. Recognizing named entities in tweets. In *HLT*, pages 359–367, 2011.

[9] X. Liu, M. Zhou, F. Wei, Z. Fu, and X. Zhou. Joint inference of named entity recognition and normalization for tweets. In *ACL*, pages 526–535, 2012.

[10] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B. Lee. Twiner: Named entity recognition in targeted twitter stream. In *SIGIR*, pages 721–730, 2012.

[11] D. M. de Oliveira, A. H. F. Laender, A. Veloso, and A. S. da Silva. Fs-ner: A lightweight filter-stream approach to named entity recognition on twitter data. In *WWW*, pages 597–604, 2013.

[12] R. Mihalcea and A. Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *CIKM*, pages 233–242, 2007.

[13] D. Milne and I. H. Witten. Learning to link with wikipedia. In *CIKM*, pages 509–518, 2008.

[14] X. Han and J. Zhao. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *CIKM*, pages 215–224, 2009.

[15] X. Han and J. Zhao. Structural semantic relatedness: A knowledge-based method to named entity disambiguation. In *ACL*, pages 50–59, 2010.

[16] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of wikipedia entities in web text. In *KDD*, pages 457–466, 2009.

[17] X. Han, L. Sun, and J. Zhao. Collective entity linking in web text: A graph-based method. In *SIGIR*, pages 765–774, 2011.

[18] W. Shen, J. Wang, P. Luo, and M. Wang. Linden: Linking named entities with knowledge base via semantic knowledge. In *WWW*, pages 449–458, 2012.

[19] P. Ferragina and U. Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*, pages 1625–1628, 2010.

[20] A. Davis, A. Veloso, A. S. da Silva, W. Meira Jr., and A. H. F. Laender. Named entity disambiguation in streaming data. In *ACL*, pages 815–824, 2012.

[21] X. Liu, Y. Li, H. Wu, M. Zhou, F. Wei, and Y. Lu. Entity linking for tweets. In *ACL*, pages 1304–1311, 2013.

[22] W. Shen, J. Wang, P. Luo, and M. Wang. Linking named entities in tweets with knowledge base via user interest modeling. In *KDD*, pages 68–76, 2013.

[23] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: Understanding microblogging usage and communities. In *WebKDD/SNA-KDD*, pages 56–65, 2007.

[24] W. Hua, Z. Wang, H. Wang, K. Zheng, and X. Zhou. Short text understanding through lexical-semantic analysis. In *ICDE*, pages 495–506, 2015.

[25] W. Hua, K. Zheng, and X. Zhou. Microblog entity linking with social temporal context. In *SIGMOD*, pages 1761–1775, 2015.

[26] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen. Short text conceptualization using a probabilistic knowledgebase. In *IJCAI*, pages 2330–2336, 2011.

[27] D. Kim, H. Wang, and A. Oh. Context-dependent conceptualization. In *IJCAI*, pages 2654–2661, 2013.

[28] W. Wang, C. Xiao, X. Lin, and C. Zhang. Efficient approximate entity extraction with edit distance constraints. In *SIGMOD*, pages 759–770, 2009.

[29] G. Li, D. Deng, and J. Feng. Faerie: efficient filtering algorithms for approximate dictionary-based entity extraction. In *SIGMOD*, pages 529–540, 2011.

[30] D. Deng, G. Li, and J. Feng. An efficient trie-based method for approximate entity extraction with edit-distance constraints. In *ICDE*, pages 141–152, 2012.

[31] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, pages 591–600, 2010.

[32] I. Witten and D. Milne. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *AAAI*, pages 25–30, 2008.

Knowledge-Based Trust: Estimating the Trustworthiness of Web Sources

Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang
Wilko Horn, Camillo Lugaresi, Shaohua Sun, Wei Zhang
Google Inc.
{lunadong|gabr|kpmurphy|vandang|wilko|camillol|sunsh|weizh}@google.com

Abstract

The quality of web sources has been traditionally evaluated using exogenous *signals such as the hyperlink structure of the graph. We propose a new approach that relies on* endogenous *signals such as the correctness of factual information provided by the source: a source that has few false facts is considered to be trustworthy. The facts are automatically extracted from each source by information extraction methods commonly used to construct knowledge bases. We propose a way to distinguish errors made in the extraction process from factual errors in the web source per se, by using joint inference in a novel multi-layer probabilistic model. We call the trustworthiness score we computed* Knowledge-Based Trust (KBT). *We apply our method to a database of 2.8B facts extracted from the web, and thereby estimate the trustworthiness of 119M webpages. Manual evaluation of a subset of the results confirms the effectiveness of the method.*

1 Introduction

"Learning to trust is one of life's most difficult tasks." – Isaac Watts.

Quality assessment for web sources (specific webpages, such as `wiki.com/page1`, or whole websites, such as `wiki.com`) is of tremendous importance in web search. It has been traditionally evaluated using exogenous signals such as hyperlinks and browsing history. However, such signals mostly capture how popular a web source is. For example, the gossip websites listed in [16] mostly have high PageRank scores [4], but would not generally be considered trustworthy. Conversely, some less popular websites nevertheless have very trustworthy information.

In this paper, we address the fundamental question of estimating how trustworthy a given web source is. Informally, we define the trustworthiness or *accuracy* of a web source as the probability that it provides the correct value for a fact (such as Barack Obama's nationality), assuming that it mentions any value for that fact. (There can be other endogenous signals such as the *completeness* and *freshness* of a web source; they are orthogonal to the accuracy measure and out of the scope of this paper.)

We propose using *Knowledge-Based Trust (KBT)* to estimate source trustworthiness as follows. We extract a plurality of facts from many pages using information extraction techniques. We then jointly estimate the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

correctness of these facts and the accuracy of the sources using inference in a probabilistic model. Inference is an iterative process, since we believe a source is trustworthy if its facts are correct, and we believe the facts are correct if they are extracted from a trustworthy source. We leverage the redundancy of information on the web to break the symmetry. Furthermore, we show how to initialize our estimate of the accuracy of sources based on authoritative information, in order to ensure that this iterative process converges to a good solution.

The fact extraction process we use is based on the *Knowledge Vault* (KV) project [9]. KV uses 16 different information extraction systems to extract (subject, predicate, object) *knowledge triples* from webpages. An example of such a triple is *(Barack Obama, nationality, USA)*. A subject represents a real-world entity, identified by an ID such as *mid*s in *Freebase* [2]; a predicate is pre-defined in *Freebase*, describing a particular attribute of an entity; an object can be an entity, a string, a numerical value, or a date.

The facts extracted by automatic methods such as KV may be wrong. There are two main sources of errors: incorrect facts on a page, and incorrect extractions made by an extraction system. As shown in [10], extraction errors are far more prevalent than source errors. Ignoring this distinction can cause us to incorrectly distrust a web source.

The KBT method introduced in this paper uses a novel multi-layer probabilistic model to distinguish errors made in the extraction process from factual errors in the web source per se. This provides a much more accurate estimate of the source reliability. We propose an efficient, scalable algorithm for performing inference and parameter estimation in the proposed probabilistic model (Section 3). We applied our model to 2.8 billion triples extracted from the web, and were thus able to reliably predict the trustworthiness of 119 million webpages and 5.6 million websites (Section 4).

We note that source trustworthiness provides an additional signal for evaluating the quality of a web source. We discuss new research opportunities for improving it and using it in conjunction with existing signals such as PageRank (Section 4.4). Also, we note that although we present our methods in the context of knowledge extraction, the general approach we propose can be applied to many other tasks that involve data integration and data cleaning. This paper is a summary of the paper [11].

2 Problem Definition and Overview

This section formally defines the notion of *Knowledge-Based Trust* (KBT) and gives an overview of our approach.

Input: We are given a set of web sources $\mathcal{W}$ and a set of extractors $\mathcal{E}$. An extractor is a method for extracting `(subject, predicate, object)` triples from a webpage. For example, one extractor may look for the *pattern "*$A*, the president of* $B*, ..."*, from which it can extract the triple *(*$A*, nationality,* $B*)*. Certainly, this is not always correct (*e.g.*, if A is the president of a company, not a country). In addition, an extractor reconciles the string representations of entities into entity identifiers such as Freebase mids, and sometimes this fails too. The different extractors can apply different extraction techniques on different types of data (*e.g.*, Web texts, DOM trees, and Webtables), and each of them may use a large number of different patterns; details of the extractors we used in KV can be found in [10]. In the rest of the paper, we represent such triples as (data item, value) pairs, where the data item is in the form of `(subject, predicate)`, describing a particular aspect of an entity, and the `object` serves as a value for the data item.

We define an observation variable X_{ewdv}. We set $X_{ewdv} = 1$ if extractor e extracted value v for data item d on web source w; if it did not extract such a value, we set $X_{ewdv} = 0$. We use matrix $X = \{X_{ewdv}\}$ to denote all the data. We can represent X as a (sparse) "data cube", as shown in Figure 1, where each cell gives the values extracted by an extractor from a web source on a particular data item. Table 1 shows an example of a single horizontal "slice" of this cube for the case where the data item is d^* *= (Barack Obama, nationality).* We discuss this example in more detail next.

Table 1: Obama's nationality extracted by 5 extractors from 8 webpages. Column 2 (Value) shows the nationality truly provided by each source; Columns 3-7 show the nationality extracted by each extractor. Wrong extractions are shown in italics.

	Value	E_1	E_2	E_3	E_4	E_5
W_1	USA	USA	USA	USA	USA	*Kenya*
W_2	USA	USA	USA	USA	*N.Amer.*	
W_3	USA	USA		USA	*N. Amer.*	
W_4	USA	USA		USA	*Kenya*	
W_5	Kenya	Kenya	Kenya	Kenya	Kenya	Kenya
W_6	Kenya	Kenya		Kenya	*USA*	
W_7	-			*Kenya*		*Kenya*
W_8	-					*Kenya*

Example 1: Suppose we have 8 webpages, $W_1 - W_8$, and suppose we are interested in the data item *(Obama, nationality)*. It is widely believed that Obama has nationality USA, not any other country.

The value stated for this data item by each of the webpages is shown in the left hand column of Table 1. We see that $W_1 - W_4$ provide *USA* as the nationality of Obama, whereas $W_5 - W_6$ provide *Kenya* (a false value)[1]. Pages $W_7 - W_8$ do not provide any information regarding Obama's nationality.

Now suppose we have 5 different extractors of varying reliability. The values they extract for this data item from each of the 8 webpages are shown in the table. Extractor E_1 extracts all the provided triples correctly. Extractor E_2 misses some of the provided triples (false negatives), but all of its extractions are correct. Extractor E_3 extracts all the provided triples, but also wrongly extracts the value *Kenya* from W_7, even though W_7 does not provide this value (a false positive)[2]. Extractor E_4 and E_5 both have poor quality, missing a lot of provided triples and making numerous mistakes.

Knowledge-based trust (KBT): For each web source $w \in \mathcal{W}$, we define its *accuracy*, denoted by A_w, as the probability that a value it provides for a fact is correct (*i.e.*, consistent with the real world). We use $A = \{A_w\}$ for the set of all accuracy parameters. We now formally define the problem of KBT estimation.

Definition 1 (KBT Estimation): The *Knowledge-Based Trust (KBT) estimation task* is to estimate the web source accuracies $A = \{A_w\}$ given the observation matrix $X = \{X_{ewdv}\}$ of extracted triples.

Our solution: To precisely estimate KBT, it is critical to distinguish extraction errors from source errors. Simply assuming all extracted values are actually provided by the source obviously would not work. In our example, we may wrongly infer that W_1 is a bad source because of the extracted *Kenya* value, although this is an extraction error. Instead, we wish to distinguish correctly extracted true triples (*e.g.*, *USA* from $W_1 - W_4$), correctly extracted false triples (*e.g.*, *Kenya* from $W_5 - W_6$), wrongly extracted true triples (*e.g.*, *USA* from W_6), and wrongly extracted false triples (*e.g.*, *Kenya* from $W_1, W_4, W_7 - W_8$).

In this paper, we present a new probabilistic model that can estimate the accuracy of each web source, factoring out the noise introduced by the extractors. We consider two sets of latent variables, one set representing the true value of each data item, and the other representing whether each extraction was correct or not; this allows us to distinguish extraction errors and source data errors. In addition, we define a set of parameters for the accuracy of the web sources (A), and a set of parameters for the quality of the extractors (we formally define them in Section 3.1); this allows us to separate the quality of the sources from that of the extractors. We call the new model the *multi-layer model*, because it contains two layers of latent variables and parameters.

[1]Real example can be found at http://beforeitsnews.com/obama-birthplace-controversy/2013/04/alabama-supreme-court-chief-justice-roy-moore-to-preside-over-obama-eligibility-case-2458624.html.

[2]As an example, KV made such a wrong extraction from webpage http://www.monitor.co.ug/News/National/US+will+respect+winner+of+Kenya+election++Obama+says/-/688334/1685814/-/ksxagx/-/index.html.

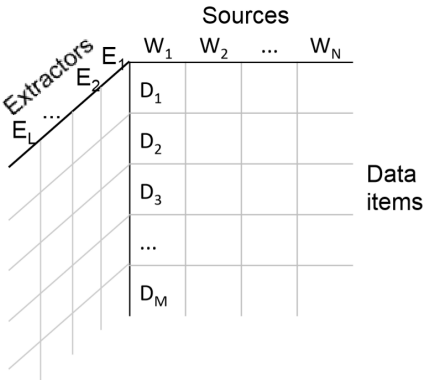


Figure 1: Form of the input data.

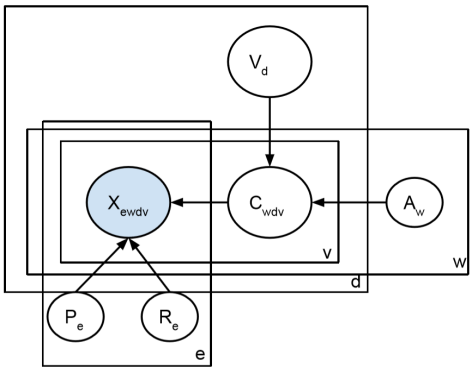


Figure 2: A representation of the multi-layer model using graphical model plate notation.

3 Multi-Layer Model

In this section, we describe in detail how we compute $A = \{A_w\}$ from our observation matrix $X = \{X_{ewdv}\}$ using a multi-layer model.

3.1 The multi-layer model

We assume that each data item can only have a single true value. This assumption holds for functional predicates, such as *date-of-birth*, but is not technically valid for set-valued predicates, such as *child*. Nevertheless, [10] showed empirically that this "single truth" assumption works well in practice even for non-functional predicates, especially when the input data contain a lot of noises; thus, we adopt it in this work for simplicity and we can extend it for multi-valued attributes by applying approaches in [21, 27, 33]. Based on the single-truth assumption, we define a latent variable $V_d \in \text{dom}(d)$ for each data item d to present the true value for d, where $\text{dom}(d)$ is the domain (set of possible values) for data item d.

We introduce the binary latent variables C_{wdv}, which represent whether web source w actually provides triple (d, v) or not. These variables depend on the true values V_d and the accuracy of each of the web sources A_w as follows:

$$p(C_{wdv} = 1 | V_d = v^*, A_w) = \begin{cases} A_w & \text{if } v = v^* \\ \frac{1-A_w}{n} & \text{if } v \neq v^* \end{cases} \tag{7}$$

where v^* is the true value, and n is the number of false values for this domain (*i.e.*, we assume $|\text{dom}(d)| = n+1$). The model says that the probability for w to provide a true value v^* for d is its accuracy, whereas the probability for it to provide one of the n false values is $1 - A_w$ divided by n (as in [7], we assume uniform distribution of the false values).

The likelihood of an observed extraction depends on how likely the extractor extracts a truly provided triple and how likely it extracts an unprovided triple. Following [27, 33], we use a two-parameter noise model for the observed data, as follows:

$$p(X_{ewdv} = 1 | C_{wdv} = c, Q_e, R_e) = \begin{cases} R_e & \text{if } c = 1 \\ Q_e & \text{if } c = 0 \end{cases} \tag{8}$$

Here R_e is the *recall* of the extractor; that is, the probability of extracting a truly provided triple. And Q_e is 1 minus the *specificity*; that is, the probability of extracting an unprovided triple. Parameter Q_e is related to the

Table 2: Extraction correctness and value truthfulness for the data in Table 1. Columns 2-4 show $p(C_{wdv} = 1|X_{wdv})$, and the last row shows $p(V_d|\hat{C}_d)$ (note that this distribution does not sum to 1.0, since not all of the values are shown in the table).

	USA	Kenya	N.Amer.
W_1	1	0	-
W_2	1	-	0
W_3	1	-	0
W_4	1	0	-
W_5	-	1	-
W_6	0	1	-
W_7	-	.07	-
W_8	-	0	-
$p(V_d\|\hat{C}_d)$	.995	.004	0

recall (R_e) and precision (P_e) as follows:

$$Q_e = \frac{\gamma}{1-\gamma} \cdot \frac{1-P_e}{P_e} \cdot R_e \tag{9}$$

where $\gamma = p(C_{wdv} = 1)$ is the prior probability for any $v \in \text{dom}(d)$, as explained in [27].

To complete the specification of the model, we must specify the prior probability of the various model parameters:

$$\theta_1 = \{A_w\}_{w=1}^{W}, \theta_2 = (\{P_e\}_{e=1}^{E}, \{R_e\}_{e=1}^{E}), \theta = (\theta_1, \theta_2) \tag{10}$$

For simplicity, we use uniform priors on the parameters. By default, we set $A_w = 0.8$, $R_e = 0.8$, and $Q_e = 0.2$. In Section 4, we discuss an alternative way to estimate the initial value of A_w, based on the fraction of correct triples that have been extracted from this source, using an external estimate of correctness (based on *Freebase* [2]).

Let $V = \{V_d\}$, $C = \{C_{wdv}\}$, and $Z = (V, C)$ be all the latent variables. Our model defines the following joint distribution:

$$p(X, Z, \theta) = p(\theta)p(V)p(C|V, \theta_1)p(X|C, \theta_2) \tag{11}$$

We can represent the conditional independence assumptions we are making using a graphical model, as shown in Figure 2. The shaded node is an observed variable, representing the data; the unshaded nodes are hidden variables or parameters. The arrows indicate the dependence between the variables and parameters. The boxes are known as "plates" and represent repetition of the enclosed variables; for example, the box of e repeats for every extractor $e \in \mathcal{E}$.

As an example, Table 2 shows the probabilities computed for the latent variables V and C. The multi-layer model is able to decide that *USA* is likely to be true and is likely to be provided by $W_1 - W_4$, contributing positively to their trustworthiness. On the other hand, *Kenya* is likely to be false and is likely to be provided by $W_5 - W_6$, contributing negatively to their trustworthiness. We describe next how we may compute these probabilities.

3.2 Inference

Recall that estimating KBT essentially requires us to compute the posterior over the parameters of interest, $p(A|X)$. Doing this exactly is computationally intractable, because of the presence of the latent variables Z. One approach is to use a Monte Carlo approximation, such as Gibbs sampling, as in [32]. However, this can be

Algorithm 1 MULTILAYER(X, t_{max})

Input: X: all extracted data;
Output: Estimates of Z and θ.

```
Initialize θ to default values;
for t ∈ [1, t_max] do
    Estimate C according to Eq.(8);
    Estimate V according to Eq.(7);
    Estimate θ_1 by Eq.(15);
    Estimate θ_2 by Eqs.(12-13);
    if Z, θ converge then
        break;
    end if
end for
return Z, θ;
```

slow and is hard to implement in a Map-Reduce framework, which is required for the scale of data we use in this paper.

A faster alternative is to use EM, which will return a point estimate of all the parameters, $\hat{\theta} = \text{argmax} p(\theta|X)$. Since we are using a uniform prior, this is equivalent to the maximum likelihood estimate $\hat{\theta} = \text{argmax} p(X|\theta)$. From this, we can derive $\hat{A}$.

As pointed out in [26], an exact EM algorithm has a quadratic complexity even for a single-layer model, so is unaffordable for data of web scale. Instead, we use an iterative "EM like" estimation procedure, where we initialize the parameters as described previously, and then alternate between estimating Z and then estimating θ, until we converge.

We next give an overview of this EM-like algorithm. Algorithm 1 gives a summary of the pseudo code; the details can be found in [11].

In our case, Z consists of two "layers" of variables. We update them sequentially, as follows. First, let $X_{wdv} = \{X_{ewdv}\}$ denote all extractions from web source w about a particular triple $t = (d, v)$. We compute by Bayesian analysis the extraction correctness $p(C_{wdv}|X_{wdv}, \theta_2^t)$, which is our guess about the "true contents" of each web source. This can be done in parallel over d, w, v.

Let $\hat{C}_d = \hat{C}_{wdv}$ denote all the estimated values for d across the different web sources. We then compute by Bayesian analysis $p(V_d|\hat{C}_d, \theta_1^t)$, which is our guess about the "true value" of each data item. This can be done in parallel over d.

Having estimated the latent variables, we then estimate θ^{t+1}. This parameter update also consists of two steps (but can be done in parallel): estimating the source accuracies $\{A_w\}$ and the extractor reliabilities $\{P_e, R_e\}$, as explained next.

3.3 Estimating the quality parameters

For reasons explained in [27], it is much more reliable to estimate P_e and R_e from data, and then compute Q_e using Equation (9), rather than trying to estimate Q_e directly. According to the definition of precision and recall, we can estimate them as follows:

$$\hat{P}_e = \frac{\sum_{wdv:X_{ewdv}=1} p(C_{wdv}=1|X)}{\sum_{wdv:X_{ewdv}=1} 1} \quad (12)$$

$$\hat{R}_e = \frac{\sum_{wdv:X_{ewdv}=1} p(C_{wdv}=1|X)}{\sum_{wdv} p(C_{wdv}=1|X)} \quad (13)$$

Following [7], we estimate the accuracy of a source by computing the average probability of its provided values being true:

$$\hat{A}_w = \frac{\sum_{dv:\hat{C}_{wdv}=1} p(V_d = v|X)}{\sum_{dv:\hat{C}_{wdv}=1} 1} \tag{14}$$

We can take uncertainty of $\hat{C}$ into account as follows:

$$\hat{A}_w = \frac{\sum_{dv:\hat{C}_{wdv}>0} p(C_{wdv} = 1|X)p(V_d = v|X)}{\sum_{dv:\hat{C}_{wdv}>0} p(C_{wdv} = 1|X)} \tag{15}$$

Eq. (15) is the key equation behind Knowledge-based Trust estimation: it estimates the accuracy of a web source as the weighted average of the probability of the facts that it contains (provides), where the weights are the probability that these facts are indeed contained in that source.

4 Experimental Results

This section describes our experimental results on large-scale real-world data. We show that (1) our algorithm can effectively estimate the correctness of extractions and the truthfulness of triples; and (2) KBT provides a valuable additional signal for web source quality.

We implemented the multi-layer model described in Section 3, called MULTILAYER. We initialized the source quality according to a gold standard, as we shall describe shortly. We note that such quality initialization is not required for MULTILAYER, but did significantly improve the predictions (see [11] for detailed comparison).

4.1 Data set

We experimented with knowledge triples collected by Knowledge Vault [9] on 7/24/2014; for simplicity we call this data set *KV*. There are 2.8B triples extracted from 2B+ webpages by 16 extractors, involving 40M extraction patterns (many extractors each learns and applies a large number of extraction patterns). Implementation details for KV are described in [9].

Figure 3 shows the distribution of the number of distinct extracted triples per URL (*i.e.*, webpage) and per extraction pattern. On the one hand, we observe some huge sources and extractors: 26 URLs each contributes over 50K triples (a lot due to extraction mistakes), 15 websites each contributes over 100M triples, and 43 extraction patterns each extracts over 1M triples. On the other hand, we observe long tails: 74% URLs each contributes fewer than 5 triples, and 48% extraction patterns each extracts fewer than 5 triples.

To determine whether these triples are true or not (gold standard labels), we use two methods. The first method is called the *Local-Closed World Assumption (LCWA)* [9, 10, 15] and works as follows. A triple (s, p, o) is considered as `true` if it appears in the Freebase KB. If the triple is missing from the KB but (s, p) appears for any other value o', we assume the KB is locally complete (for (s, p)), and we label the (s, p, o) triple as `false`. We label the rest of the triples (where (s, p) is missing) as `unknown` and remove them from the evaluation set. In this way we can decide truthfulness of 0.74B triples (26% in *KV*), of which 20% are true (in Freebase).

Second, we apply type checking to find incorrect extractions. We found that in the following cases a triple (s, p, o) is often due to an extraction error: 1) $s = o$; 2) the type of s or o is incompatible with what is required by the predicate; or 3) o is outside the expected range (*e.g.*, the weight of an athlete is over 1000 pounds). We discovered 0.56B triples (20% in KV) that violate such rules and consider them both as `false` triples and as extraction mistakes.

Our gold standard include triples from both labeling methods. It contains in total 1.3B triples, among which 11.5% are true.

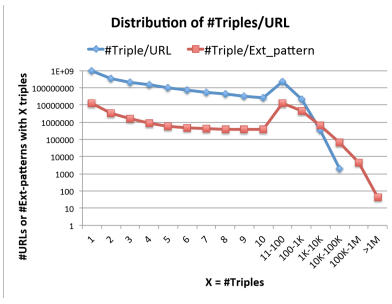


Figure 3: Distribution of #Triples per URL or extraction pattern shows a high variety of the sources and extractors.

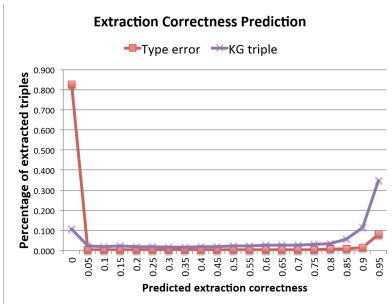


Figure 4: Distribution of predicted extraction correctness shows effectiveness of MULTILAYER.

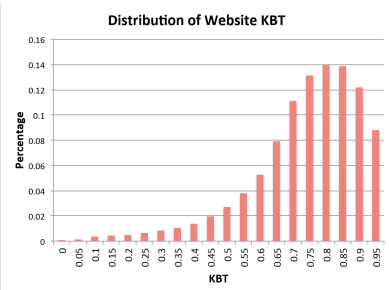


Figure 5: Distribution on KBT for websites with at least 5 extracted triples.

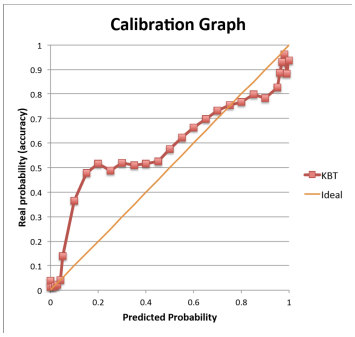


Figure 6: MULTILAYER predicts well-calibrated probabilities.

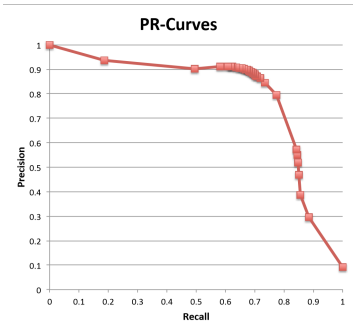


Figure 7: The PR-curve of MULTILAYER results is in good shape.

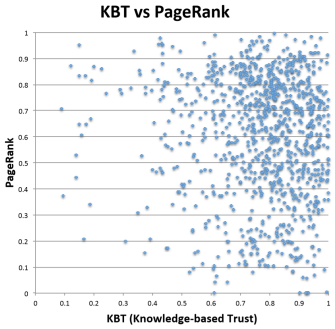


Figure 8: KBT and PageRank are orthogonal signals.

4.2 Correctness of triples and extractions

We divide our triples according to the predicted probabilities into buckets $[0, 0.01), \ldots, [0.04, 0.05), [0.05, 0.1), \ldots, [0.9, 0.95), [0.95, 0.96), \ldots, [0.99, 1), [1, 1]$ (most triples fall in $[0, 0.05)$ and $[0.95, 1]$, so we used a finer granularity there). For each bucket we compute the accuracy of the triples according to the gold standard, which can be considered as the real probability of the triples. Ideally, the predicted probabilities should be the same as the real probabilities. Figure 6 plots the calibration curve, showing that MULTILAYER computes well-calibrated probabilities for the truthfulness of triples. In addition, Figure 7 plots the PR-curve, where the X-axis represents the recall and the Y-axis represents the precision as we order triples according to the predicted probabilities. The PR-curve is also in good shape.

To examine the quality of our prediction on extraction correctness (recall that we lack a full gold standard), we plotted the distribution of the predictions on triples with type errors (ideally we wish to predict a probability of 0 for such extractions) and on correct triples (presumably a lot of them, though not all, would be correctly extracted and we shall predict a high probability for such extractions). Figure 4 shows the results by MULTILAYER. We observe that for the triples with type errors, MULTILAYER predicts a probability below 0.1 for 80% of them and a probability above 0.7 for only 8%; in contrast, for the correct triples in Freebase, MULTILAYER predicts a probability below 0.1 for 26% of them and a probability above 0.7 for 54%, showing effectiveness of our model.

4.3 KBT vs PageRank

We now evaluate how well we estimate the trustworthiness of web sources. Our data set contains 2B+ webpages from 26M websites. Among them, our multi-layer model believes that we have correctly extracted at least 5 triples from about 119M webpages and 5.6M websites. Figure 5 shows the distribution of KBT scores: we observed that the peak is at 0.8 and 52% of the websites have a KBT over 0.8.

Since we do not have ground truth on web-source quality, we compare our method to PageRank. We compute PageRank for all websites on the web, and normalize the scores to $[0, 1]$. Figure 8 plots KBT and PageRank for 2000 randomly selected websites. As expected, the two signals are almost orthogonal. We next investigate the two cases where KBT differs significantly from PageRank.

Low PageRank but high KBT (bottom-right corner): To understand which sources may obtain high KBT, we randomly sampled 100 websites whose KBT is above 0.9. The number of extracted triples from each website varies from hundreds to millions. For each website we considered the top 3 predicates and randomly selected from these predicates 10 triples where the probability of the extraction being correct is above 0.8. We manually evaluated each website according to the following 4 criteria.

- *Triple correctness*: whether at least 9 triples are correct.
- *Extraction correctness*: whether at least 9 triples are correctly extracted (and hence we can evaluate the website according to what it really states).
- *Topic relevance*: we decide the major topics for the website according to the website name and the introduction in the "About us" page; we then decide whether at least 9 triples are relevant to these topics (*e.g.*, if the website is about business directories in South America but the extractions are about cities and countries in SA, we consider them as not topic relevant).
- *Non-trivialness*: we decide whether the sampled triples state non-trivial facts (*e.g.*, if most sampled triples from a Hindi movie website state that the language of the movie is Hindi, we consider it as trivial).

We consider a website as truly trustworthy if it satisfies all of the four criteria. Among the 100 websites, 85 are considered trustworthy; 2 are not topic relevant, 12 do not have enough non-trivial triples, and 2 have more than 1 extraction errors (one website has two issues). However, only 20 out of the 85 trustworthy sites have a PageRank over 0.5. This shows that KBT can identify sources with trustworthy data, even though they are tail sources with low PageRanks.

High PageRank but low KBT (top-left corner): We consider the 15 gossip websites listed in [16]. Among them, 14 have a PageRank among top 15% of the websites, since such websites are often popular. However, for all of them the KBT are in the bottom 50%; in other words, they are considered less trustworthy than half of the websites. Another kind of websites that often get low KBT are forum websites. For instance, we discovered that *answers.yahoo.com* says that *"Catherine Zeta-Jones is from New Zealand"* [3], although she was born in Wales according to *Wikipedia*[4].

4.4 Discussions

Although we have seen that KBT seems to provide a useful signal about trustworthiness, which is orthogonal to more traditional signals such as PageRank, our experiments also show places for further improvement as future work.

1. To avoid evaluating KBT on topic irrelevant triples, we need to identify the main topics of a web source, and filter triples whose entity or predicate is not relevant to these topics.

[3]https://answers.yahoo.com/question/index?qid=20070206090808AAC54nH.
[4]http://en.wikipedia.org/wiki/Catherine_Zeta-Jones.

2. To avoid evaluating KBT on trivial triples, we need to decide whether the information in a triple is trivial. One possibility is to consider a predicate with a very low variety of objects as less informative. Another possibility is to associate triples with an IDF (inverse document frequency), such that low-IDF triples get lower weight in KBT computation.
3. Our extractors (and most state-of-the-art extractors) still have limited extraction capabilities and this limits our ability to estimate KBT for all websites. We wish to increase our KBT coverage by extending our method to handle open-IE style information extraction techniques, which do not conform to a schema [14]. However, although these methods can extract more triples, they may introduce more noise.
4. Some websites scrape data from other websites. Identifying such websites requires techniques such as copy detection. Scaling up copy detection techniques, such as [6, 7], has been attempted in [23], but more work is required before these methods can be applied to analyzing extracted data from billions of web sources.
5. Finally, there have been many other signals such as PageRank, visit history, spaminess for evaluating web-source quality. Combining KBT with those signals would be important future work.

5 Related Work

There has been a lot of work studying how to assess quality of web sources. PageRank [4] and Authority-hub analysis [19] consider signals from link analysis (surveyed in [3]). EigenTrust [18] and TrustMe [28] consider signals from source behavior in a P2P network. Web topology [5], TrustRank [17], and AntiTrust [20] detect web spams. The knowledge-based trust we propose in this paper is different from all of them in that it considers an important *endogenous* signal–the correctness of the factual information provided by a web source.

KBT estimation is closely related to the *knowledge fusion* problem [10], where the goal is to decide the true (but latent) values for each of the data items, given the noisy extraction. It is also relevant to the body of work in *Data fusion* (surveyed in [1, 12, 23]), where the goal is to resolve conflicts from data provided by multiple sources, but assuming perfect knowledge on the values provided by each source (so no extraction error). Most of the recent work in this area considers trustworthiness of sources, measured by link-based measures [24, 25], IR-based measures [29], accuracy-based measures [7, 8, 13, 22, 27, 30], and graphical-model analysis [26, 31, 33, 32]. However, these papers do not model the concept of an extractor, and hence they cannot distinguish an untrustworthy source from a low-quality extractor.

Graphical models have been proposed to solve the data fusion problem [26, 31, 32, 33]. In particular, [26] considers single truth, [32] considers numerical values, [33] allows multiple truths, and [31] considers correlations between the sources. These prior works do not model the concept of an extractor, and hence they cannot capture the fact that sources and extractors introduce qualitatively different kinds of noise. In addition, the data sets used in the experiments of traditional data fusion works are typically 5-6 orders of magnitude smaller in scale than ours, and their inference algorithms are inherently slower than our algorithm.

6 Conclusions

This paper proposes a new metric for evaluating web-source quality–knowledge-based trust. We proposed a sophisticated probabilistic model that jointly estimates the correctness of extractions and source data, and the trustworthiness of sources. In addition, we presented an algorithm that dynamically decides the level of granularity for each source. Experimental results have shown both promise in evaluating web-source quality and improvement over existing techniques for knowledge fusion.

References

[1] J. Bleiholder and F. Naumann. Data fusion. *ACM Computing Surveys*, 41(1):1–41, 2008.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[3] A. Borodin, G. Roberts, J. Rosenthal, and P. Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *TOIT*, 5:231–297, 2005.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[5] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: Web spam detection using the web topology. In *SIGIR*, 2007.

[6] X. L. Dong, L. Berti-Equille, Y. Hu, and D. Srivastava. Global detection of complex copying relationships between sources. *PVLDB*, 2010.

[7] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *PVLDB*, 2(1), 2009.

[8] X. L. Dong, L. Berti-Equille, and D. Srivastava. Truth discovery and copying detection in a dynamic world. *PVLDB*, 2(1), 2009.

[9] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, 2014.

[10] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang. From data fusion to knowledge fusion. *PVLDB*, 2014.

[11] X. L. Dong, E. Gabrilovich, K. Murphy, V. Dang, W. Horn, C. Lugaresi, S. Sun, and W. Zhang. Knowledge-based trust: Estimating the trustworthiness of web sources. *PVLDB*, 2015.

[12] X. L. Dong and F. Naumann. Data fusion–resolving data conflicts for integration. *PVLDB*, 2009.

[13] X. L. Dong, B. Saha, and D. Srivastava. Less is more: Selecting sources wisely for integration. *PVLDB*, 6, 2013.

[14] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open information extraction: the second generation. In *IJCAI*, 2011.

[15] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, pages 413–422, 2013.

[16] Top 15 most popular celebrity gossip websites. http://www.ebizmba.com/articles/gossip-websites, 2014.

[17] Z. Gyngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with TrustRank. In *VLDB*, pages 576–587, 2014.

[18] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *WWW*, 2003.

[19] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *SODA*, 1998.

[20] V. Krishnan and R. Raj. Web spam detection with anti-trust rank. In *AIRWeb*, 2006.

[21] F. Li, X. L. Dong, A. Langen, and Y. Li. Knowledge verification for long tail verticals. Technical report, 2016. `www.comp.nus.edu.sg/~furongli/factCheck_report.pdf`.

[22] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD*, pages 1187–1198, 2014.

[23] X. Li, X. L. Dong, K. B. Lyons, W. Meng, and D. Srivastava. Scaling up copy detection. In *ICDE*, 2015.

[24] J. Pasternack and D. Roth. Knowing what to believe (when you already know something). In *COLING*, pages 877–885, 2010.

[25] J. Pasternack and D. Roth. Making better informed trust decisions with generalized fact-finding. In *IJCAI*, pages 2324–2329, 2011.

[26] J. Pasternack and D. Roth. Latent credibility analysis. In *WWW*, 2013.

[27] R. Pochampally, A. D. Sarma, X. L. Dong, A. Meliou, and D. Srivastava. Fusing data with correlations. In *Sigmod*, 2014.

[28] A. Singh and L. Liu. TrustMe: anonymous management of trust relationshiops in decentralized P2P systems. In *IEEE Intl. Conf. on Peer-to-Peer Computing*, 2003.

[29] M. Wu and A. Marian. Corroborating answers from multiple web sources. In *Proc. of the WebDB Workshop*, 2007.

[30] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. In *Proc. of SIGKDD*, 2007.

[31] X. Yin and W. Tan. Semi-supervised truth discovery. In *WWW*, pages 217–226, 2011.

[32] B. Zhao and J. Han. A probabilistic model for estimating real-valued truth from conflicting sources. In *QDB*, 2012.

[33] B. Zhao, B. I. P. Rubinstein, J. Gemmell, and J. Han. A Bayesian approach to discovering truth from conflicting sources for data integration. *PVLDB*, 5(6):550–561, 2012.

Data Engineering

It's FREE to join!

TCDE

tab.computer.org/tcde/

The Technical Committee on Data Engineering (TCDE) of the IEEE Computer Society is concerned with the role of data in the design, development, management and utilization of information systems.

- Data Management Systems and Modern Hardware/Software Platforms
- Data Models, Data Integration, Semantics and Data Quality
- Spatial, Temporal, Graph, Scientific, Statistical and Multimedia Databases
- Data Mining, Data Warehousing, and OLAP
- Big Data, Streams and Clouds
- Information Management, Distribution, Mobility, and the WWW
- Data Security, Privacy and Trust
- Performance, Experiments, and Analysis of Data Systems

The TCDE sponsors the International Conference on Data Engineering (ICDE). It publishes a quarterly newsletter, the Data Engineering Bulletin. If you are a member of the IEEE Computer Society, you may join the TCDE and receive copies of the Data Engineering Bulletin without cost. There are approximately 1000 members of the TCDE.

Join TCDE via Online or Fax

ONLINE: Follow the instructions on this page:

www.computer.org/portal/web/tandc/joinatc

FAX: Complete your details and fax this form to **+61-7-3365 3248**

Name _____

IEEE Member # _____

Mailing Address _____

_____

Country _____

Email _____

Phone _____

TCDE Mailing List

TCDE will occasionally email announcements, and other opportunities available for members. This mailing list will be used only for this purpose.

Membership Questions?

Xiaofang Zhou
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
zxf@uq.edu.au

TCDE Chair

Kyu-Young Whang
KAIST
371-1 Koo-Sung Dong, Yoo-Sung Ku
Daejeon 305-701, Korea
kywhang@cs.kaist.ac.kr

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398