Interactive Data Integration and Entity Resolution for Exploratory Visual Data Analytics

Kristi Morton

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2015

Reading Committee:

Magdalena Balazinska, Chair

Daniel Grossman

Hannaneh Hajishirzi

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Interactive Data Integration and Entity Resolution
for Exploratory Visual Data Analytics

Kristi Morton

Chair of the Supervisory Committee:
Associate Professor Magdalena Balazinska
Department of Computer Science and Engineering

Data has become more widely available to the public for consumption, for example, through the Web and the recent "Open Data" movement. An emerging cohort of users, called *Data Enthusiasts*, want to analyze this data, but have limited technical or data science expertise. In response to these trends, online visual analytics systems have emerged as a popular tool for data analysis and sharing. Current visual analytics systems such as Tableau and Many Eyes enable this user cohort to be able to perform sophisticated data analysis visually at interactive speeds and without any programming.

Together, these two systems have been used by tens of thousands of authors to create hundreds of thousands of views, yet we know very little about how these systems are being used. The first challenge we address in this thesis, thus, is: *how are popular visual analytics systems such as Tableau and Many Eyes being used for data analysis?* To the best of our knowledge, this is the first study of its kind, and presents important details about the use of online, visual analytics systems.

Visual analytics systems provide basic support for data integration. A simple approach for interactive data integration in Tableau was implemented in that tool in the context of this dissertation. Visual analytics, systems, however, do not currently assist users with detecting or resolving potential data quality problems including the well-known deduplication problem. Recent approaches for deduplication focus on cleaning entire datasets and com-

monly require hundreds to thousands of user labels. In this thesis, we address the challenge of deduplication in the context of visual data analytics with an approach that produces significantly cleaner views for small labeling budgets than state-of-the-art alternatives. The key idea behind the approach is to consider the impact that individual tuples have on a visualization and to monitor how the view changes during cleaning.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DEDICATION

To my husband and mother for their ongoing love and support and to my late father who inspired me to continue going when the going was tough.

Chapter 1

INTRODUCTION

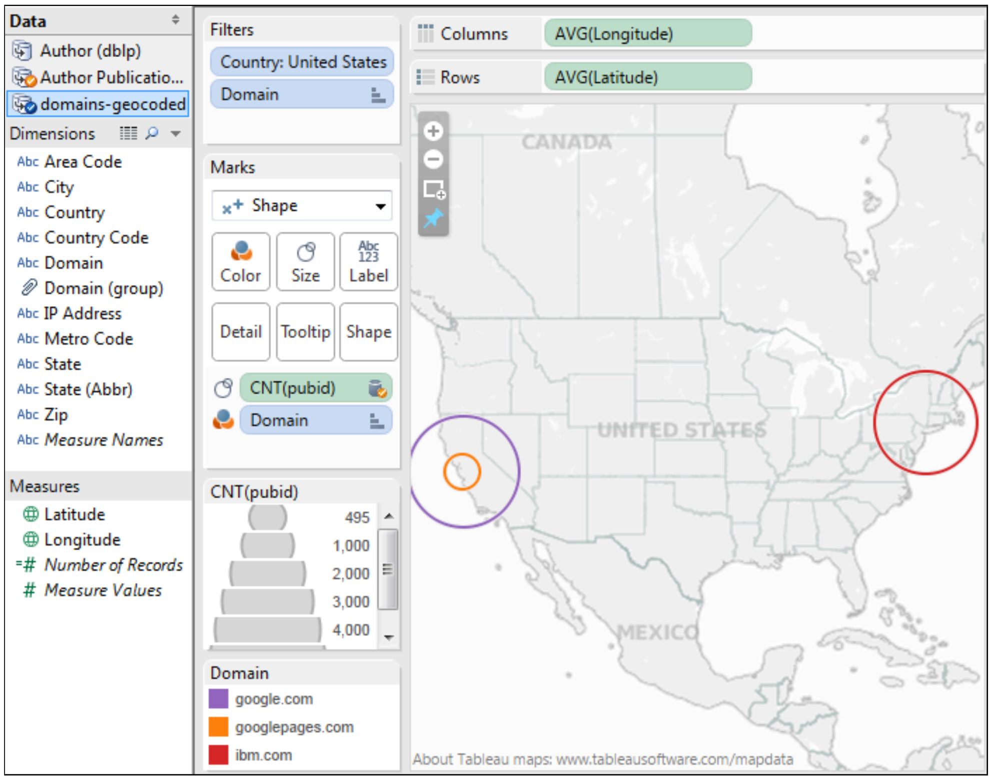
The need for effective analysis of data is widely recognized today, and many tools aim to support professional data scientists from industry and the sciences with this task (*e.g.,* Hadoop [5], Spark [6], and Dato [31]). There is, however, another growing cohort of users who need the ability to analyze data, but have limited technical expertise. These users are without formal training in data science. They are called *Data Enthusiasts* [53, 141]. Such users are increasingly applying data and visualizations to illustrate a story, answer a question, or make a decision. Some common examples include teachers (*e.g.,* to understand student performance across years), real-estate agents (*e.g.,* to anticipate market trends), and Web journalists (*e.g.,* to tell stories based on data). In this thesis, we focus on supporting the data analysis needs of this new class of users.

Data is also increasingly being made available for public consumption on the Web thanks to the recent "Open Data" movement. Since 2009, governments from all over the world (including the US, Canada, and Japan) and governing bodies such as the United Nations and European Union have published their own open data repositories. As of 2015, there are over 426 established local, regional, and national open data catalogues available on the Web through the open source `datacatalogs.org` project. Despite an abundance of public data (estimated to be 2.8 zettabytes in 2012), a "Digital Universe Study" by the International Data Corporation (IDC) [42], found that only a small fraction, 0.5%, is analyzed. The gulf between availability and exploitation presents a significant opportunity for the database community to study the root-causes and bridge the gap by improving the analytical tools for all users, both the data scientists and the data enthusiasts.

1.1 Visual Analytics Systems

Recent *self-service visual analytics systems* strive to support a broad spectrum of users: from beginner data enthusiasts to seasoned business intelligence analysts. Tableau Public [112], Fusion Tables [46], and Many Eyes [125] are among the most popular examples. These visual analytics tools enable the *sensemaking model* [21]: the typical analytical process starts with a question that a data enthusiast seeks to answer. Unless she already has a dataset to explore, the data enthusiast then manually forages for relevant data by, for example, using a search engine such as Google or Bing. Once the appropriate dataset is acquired, the data is explored through a suitable visualization. The user continues to interact with the visualization by, for example, drilling down to the details, rolling up to summarize a dimension, or adding dimensions from other datasets. These existing systems provide several desirable features to support data enthusiasts. They enable users to *visually* explore their data as illustrated in Figure 1.1, which removes the need for learning any programming or query languages. They facilitate the integration and study of *multiple* datasets at the same time. Finally, they support *collaborations* through sharing visualizations and data online for both viewing and editing by others.

Visual interfaces enable data enthusiasts to author sophisticated queries using drag-and-drop actions in a GUI and view the answer(s) through a single visualization (or multiple linked ones). These visualizations are represented internally as database views. Users can create sophisticated views that combine multiple heterogeneous data sets (*e.g.,* Excel spreadsheets, relational databases, data cubes, delimited text files, etc.) along a common dimension or set of dimensions (*i.e.,* shared categorical attributes). This type of visual programming environment supports the sensemaking model well because it provides interactive response times both in authoring the question and in producing visual results; such interactive Q&A is possible in part because the user only has to focus on the semantics of the query and not its syntax. As a concrete example, consider the Tableau visualization in Figure 1.1(top), which compares the academic productivity of the US research branches of IBM and Google. In this visualization, two datasets (DBLP [32] and Freegeoip [40]) are joined on-the-fly using simple drag-and-drop GUI actions. The DBLP dataset lists pub-



```
SELECT [geo].[Latitude] ON ROWS, [geo].[Longitude] ON COLUMNS,
       [geo].[Domain] ON COLOR, COUNT([pub].[pubid]) ON SIZE
FROM [pub]
LEFT JOIN [geo] ON [pub].[Domain] = [geo].[Domain]
WHERE [pub].[Domain] IN {"google.com","googlepages.com","ibm.com"}
      AND [geo].[Country] = "United States"
```

Figure 1.1: The Tableau visual interface and visualization (top) and corresponding VizQL, a formal language describing both the data and visualization (bottom). Two datasets, DBLP and Freegeoip, are combined to show the total publication counts for Google and IBM.

lications by author while the Freegeoip dataset maps author web pages to IP addresses and then geolocations. To create the visualization in Figure 1.1, the user interacts with a relational dataset by dragging a “Dimension” (categorical data such as a domain name) or “Measure” (quantitative data such as a latitude/longitude coordinate) field from the data window in Figure 1.1(left) to the visual canvas (right). In response to the user’s actions, Tableau automatically renders the data points on a map visualization. We explore this example in more detail in Chapter 2.

1.2 Visual Analytics Challenges

While visual analytics systems have rapidly evolved over the past several years, many research challenges remain open.

Challenge 1: Despite their growing popularity, little is known about how these visual analytics systems are being used. Even basic statistics such as the number of users are often not published (*e.g.,* Fusion Tables [46]), let alone any details of user activity. The most prominent system, Many Eyes, started in early 2007, and initial studies [27] indicated a significant uptake, as well as collaboration between users; but there have been no follow-up studies on usage, nor have there been comparable studies of other web-based or web-centric visual data analysis systems. Shortly before Many Eyes, in December 2006, Swivel.com was launched. Swivel was much simpler than Many Eyes, but run as a start-up rather than an experiment. It shut down in summer 2010, casting doubt on whether there was a market for web-based visual data analysis systems. At the same time, there is clearly broad interest in data integration, analysis, and visualization. *The New York Times*, *The Washington Post*, *The Guardian*, and other news media are not only increasingly using visual data analysis as part of news stories, but also experimenting with more sophisticated types of visualizations.

As our society continues to become *data-enabled*, it is important that we continue to improve data management and analysis tools. If we are to build better online data visualization and sharing systems, the first step is to understand how they are being used today. The key contribution addressed in Chapter 4 is to shed light on this exact question: *How are online visual data analysis and sharing systems being used?*

Challenge 2: In recent work [89, 90] (also Chapter 4), we found that today's visual analytics services are attracting hundreds or thousands of new accounts each month, but most users author only one visualization and never return. A recent interview study of Open Government Data consumers [48] corroborates that current visualization tools are underserving their users. One key limitation that make these systems unsuitable for many users is that the tools assume the data is clean and in a well-structured relational format, which is typically not the case. Data enthusiasts have reported that cleaning and transforming their datasets is one of the most time-consuming and tedious steps in their analytical workflows (often comprising 80% of the work [30]). The data is useless until that labor is accomplished up front. Deduplication is one kind of dirty data problem. This problem manifests when there are different representations of the same real world entity or object in the data sources being integrated. For example, the same restaurant may appear under two different phone numbers. The same product may use different abbreviations in its name or may include a different description. Recent approaches for deduplication focus on cleaning entire datasets and require significant human effort. In Chapter 5, we address the problem of deduplication in the context of visual data analytics. The key contribution of this chapter is *View Impact Cleaning*, a new active-learning approach for record deduplication that strives to produce the cleanest visualization/view possible with limited user effort in the form of a limited budget for data labeling.

1.3 Dissertation Contributions

The focus of this dissertation is on improving visual data analytics systems. In the context of supporting data enthusiasts in their exploratory, visual analytic tasks on structured datasets, the contributions of this thesis are the following:

Usage study of two visual data analytics systems (Chapter 4). We take a first step toward understanding how online visual data analysis systems are being used through a longitudinal measurement study of two popular online data visualization and analysis systems: Tableau Public [112] and Many Eyes [125, 83]. Both systems allow users to create visualizations online, and both are free to use. Tableau Public requires the download of a Windows-only client, while Many Eyes is used entirely in the browser. Both systems provide a variety of visualization techniques, which not only generate static images, but which the viewer can interact with in the browser.

We tackle the question of how both of these systems are being used from the perspective of the database community. Through our study, we thus focus on the following core set of questions: (1) How popular are these systems? How many users do they attract and how

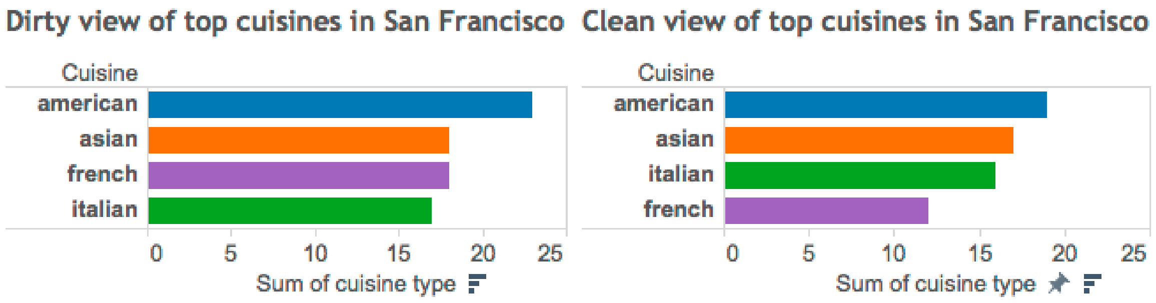


Figure 1.2: Views, dirty (left) and clean (right), over Fodor ∪ Zagat restaurant datasets.

active are these users? (2) How heavily do users leverage the collaborative features of these tools? (3) What do users actually do with the data? How do they analyze it? How much data (in terms of relation cardinality and degree) do users choose to visualize at any given time? And finally (4) Do users integrate multiple data sources in their visualizations? And how do they perform these integrations? To the best of our knowledge, this is the first formal study of these types of systems.

View-driven data cleaning (Chapter 5). Duplicate records may affect a visualization. Figures 1.1 and 1.2 show two examples. In Figure 1.1 there are duplicate entries representing Google publications in the Bay Area: `google.com` (purple circle) and `googlepages.com` (orange circle). These two circles should be instead combined, as they represent publications for the same entity, Google. In Figure 1.2, we see the impact of duplicate entities on the view of the top four cuisine types by quantity of restaurants in San Francisco over a restaurant dataset created from the union of the Fodor and Zagat restaurant ratings datasets in the RIDDLE[1] repository. Duplicate records affect results and should therefore be cleaned. The problem, however, is that data cleaning is a disruptive process. It interrupts the user during her primary data exploration task. Our goal is to clean a user's visualization with minimal interruption.

State-of-the-art techniques for deduplication use active learning [93, 45], where one or

[1]http://www.cs.utexas.edu/users/ml/riddle

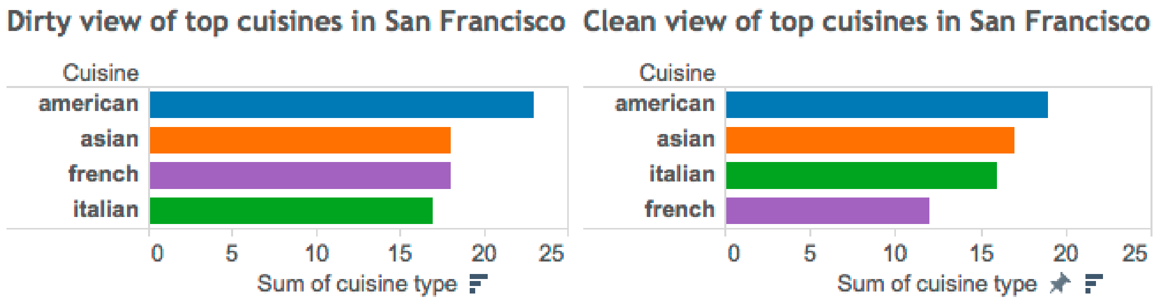


Figure 1.3: Views, dirty (left) and clean (right), over Fodor ∪ Zagat restaurant datasets.

more users label training examples, which enable the system to learn a classifier that categorizes pairs of tuples as either duplicates or not. Active learning iteratively asks users for additional, carefully selected labels and re-trains the classifier until the classifier stops improving. Existing methods produce high-quality classifiers. Because they focus strictly on the data cleaning task, however, existing methods request hundreds to thousands [93, 45] of labels during the data cleaning process.

The time-consuming labeling effort imposed by existing approaches conflicts with the interactive, real-time constraints exploratory visual analytics systems impose to support sensemaking [91]. Worse, current systems do not prioritize labeling pairs that actually impact the view. The user may thus easily find herself labeling data that she is not even analyzing.

In Chapter 5, we develop an approach that addresses the above problem. Our method, called *View Impact Cleaning*, performs deduplication in a manner that focuses on a user's current visualization (or dashboard of visualizations). View Impact Cleaning yields a significantly cleaner view than active learning alone when given a small labeling budget. It only asks the user to label data that is currently being visualized and it automatically stops the cleaning process when it detects that additional labels will not change the visualization further even if they could yield a better overall classifier.

By developing the View Impact Cleaning method, we make the following specific contributions:

1. We define a new notion of *view sensitivity* to duplicate tuples. View sensitivity captures the extent to which a view is affected by duplicate tuples. We also define a new notion of *view impact score* of individual tuples on a visualization. The view impact score measures the extent to which a view will change if a given tuple is found to be a duplicate and is removed.
2. We develop an active-learning-based method that builds an initial classifier and then iteratively improves that classifier. The classifier categorizes pairs of base tuples in the provenance of the user's view as either duplicates or not. The novelty of our approach is in the selection of the training examples. Our approach uses both the view impact scores of individual tuples and the potential of a training example to improve the classifier quality.
3. We develop a new stopping condition for view cleaning that considers not the quality of the classifier but instead considers how the view has been evolving during the cleaning process. An important implication of our approach is that it stops cleaning a view both in the case where a sufficient number of tuples have been removed to clean the view and in the case where a view is not sensitive to duplicate tuples and cleaning has little effect on the view.

We evaluate our approach on nine different views specified on two real-world datasets. We find that, when given a small cleaning budget, our approach yields significantly cleaner views than active learning without consideration for the users's view. It also effectively stops cleaning earlier than active learning alone while delivering cleaner views. Finally, we evaluate and discuss the problem of cleaning a dashboard comprising multiple visualizations.

Data Blending (Chapter 2): In the context of this dissertation, we also implemented a basic approach for interactive data integration in the Tableau system, which we present in Chapter 2, and which is called data blending. The contribution of the data blending feature is in its implementation.

1.4 Dissertation Outline

The dissertation is organized as follows. In Chapter 2, we present an overview of Tableau, a popular visual data analytics service for data enthusiasts, and its current features including data blending, and its limitations. The text for Chapter 2 is published in the Industrial Track of the ACM Special Interest Group on the Management of Data (SIGMOD) in 2012 [92] with the following collaborators from Tableau: Ross Bunker, Jock Mackinlay, Robert Morton, and Chris Stolte. Next, in Chapter 3 we summarize the various challenges that data enthusiasts face in their analytical workflows and what problems related work addresses. The text for Chapter 3 appears in the 2014 International Conference on Very Large Data Bases (VLDB), Vision Track [91]. This publication was the result of a collaboration between the Department of Computer Science & Engineering at the University of Washington (including my advisors, Magdalena Balazinska and Dan Grossman) and Tableau (including my manager, Jock Mackinlay). We then study in Chapter 4 the details of how the Tableau and Many Eyes systems are being used for analytics. The work was originally published in the 2014 SIGMOD Record [90], and we include an extended version of the experimental results. This publication was a collaboration between the University of Washington (Magdalena Balazinska and Dan Grossman) and Tableau (Robert Kosara and Jock Mackinlay). In Chapter 5, we present a new approach to deduplication in the context of an interactive visual analytics system called View Impact Cleaning. This work was done in collaboration with my advisors, Magdalena Balazinska and Dan Grossman, and a machine learning expert from the Department of Electrical Engineering, Hannaneh Hajishirzi. This work is currently in submission. Finally, we discuss future directions and conclude with our overall contributions in Chapter 6.

In summary, in the context of a visual data analytics environment such as Tableau and Many Eyes and motivated by a growing cohort of users with limited technical expertise, this dissertation work addresses the following problems with the goal of improving next generation visual analytics systems: 1) How are two popular systems, Tableau and Many Eyes, being used (Chapter 4) and 2) How can such systems assist users in the complex task of cleaning data that has duplicate entities (Chapter 5).

Chapter 2
BACKGROUND: VISUAL DATA ANALYTICS WITH TABLEAU

In this chapter, we present an overview of Tableau [113], a commercial business intelligence software tool that supports interactive, visual analysis of data. The content of this chapter is an extended version of a SIGMOD Industrial paper published in 2012 [92]. This chapter is organized as follows. We present a high-level discussion of the key features of Tableau in Section 2.1. Next, in Section 2.2, we describe in-depth a data integration feature called *data blending* that supports on-the-fly data integration as part of the data analysis cycle and that we implemented in the context of this dissertation. Finally, in Section 2.3, we present Tableau's data cleaning capabilities.

2.1 Tableau overview

Tableau [113] is a data visualization tool that sits between the end-user and the database and allows the user to create visualizations by dragging and dropping fields from her dataset onto a visual canvas. In response to these actions, Tableau generates formal VizQL (Visual Query Language) [109] statements to build the requested visualization. VizQL is a structured query language with support for rendering graphics and data. As a concrete example, the six lines of VizQL shown in Figure 1.1(bottom) describe the visualization in Figure 1.1(top). The formalism represents the semantics underlying the tool. The user does not author the language directly, but rather her interactions with the data through the GUI automatically result in the generated code. Tableau's high-level architecture is shown in Figure 2.1. The VizQL compiler creates all of the data and visual encodings generated from the user interactions as VizQL statements. Each VizQL statement is compiled into the SQL or Multidimensional Expressions (MDX) queries necessary to generate the data for the visualization. These queries are executed by Tableau's in-memory, column-store data engine [132] if the underlying data can be collocated in the same database; otherwise the queries are federated to the data sources and the result set is pulled into the data engine. As shown in Figure 2.1(top), the types of input data that Tableau supports include any structured relational data, data cubes (see [49] for details), Excel spreadsheets, and flat text files such as comma-separated-values (CSV).

A visualization is a visual representation of a database view computed over a single input dataset or combination of data sets (as in a JOIN operation). For example, during the initial data import process, the user can specify any of the five standard JOIN operations over multiple data sets, *e.g.,* INNER, LEFT OUTER, RIGHT OUTER, FULL OUTER, and CROSS to combine these datasets into a single view that can then further be filtered or aggregated as needed. A visualization is organized by its discrete fields (*e.g.,* categorical attributes) into a series of pages (*e.g.,* a progression of time values) and partitions (*e.g.,* a hierarchy of attribute values), and like a `GROUP BY` clause in SQL, these *grouping fields* comprise the primary key of the visualization. Thus as the user is exploring one dataset through a visualization, other datasets can be *blended* on-the-fly into the visualization, using the primary key of the visualization as the join key. However, this blending operation is restricted to a LEFT OUTER JOIN.

Tableau is equipped with two means of integrating data that have different assumptions of where the data is located. First, in the case where the data sets are collocated (or can be collocated), Tableau formulates a query that joins them (no union) locally, in-memory to produce a visualization. However, in the case where the data sets are not collocated (or cannot be collocated), Tableau federates queries to each data source, and creates a dynamic, *blended* view in-memory that consists of the joined result sets of the queries. Data blending is a complementary technology to the standard collocated approach with the following benefits:

1) Resolves data granularity problems. Frequently a user wants to combine data that may not be at the same granularity. For example, let's say that an employee at company A wants to compare the yearly growth of sales to a competitor company B. The dataset for company B (see Figure 2.2) contains a detailed quarterly growth of sales for B (<u>quarter, year</u> is the primary key), while company A's dataset only includes the yearly sales (<u>year</u> is the primary key). If the employee simply joins these two datasets on yearly earnings, then

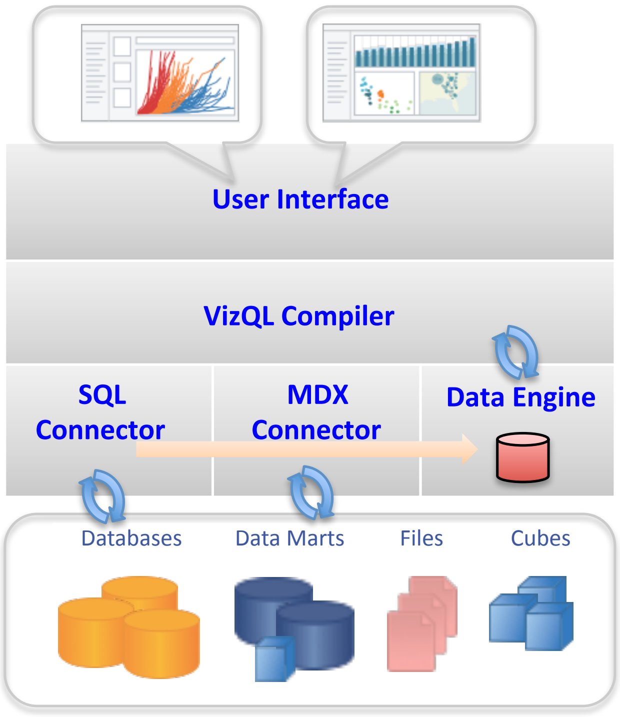
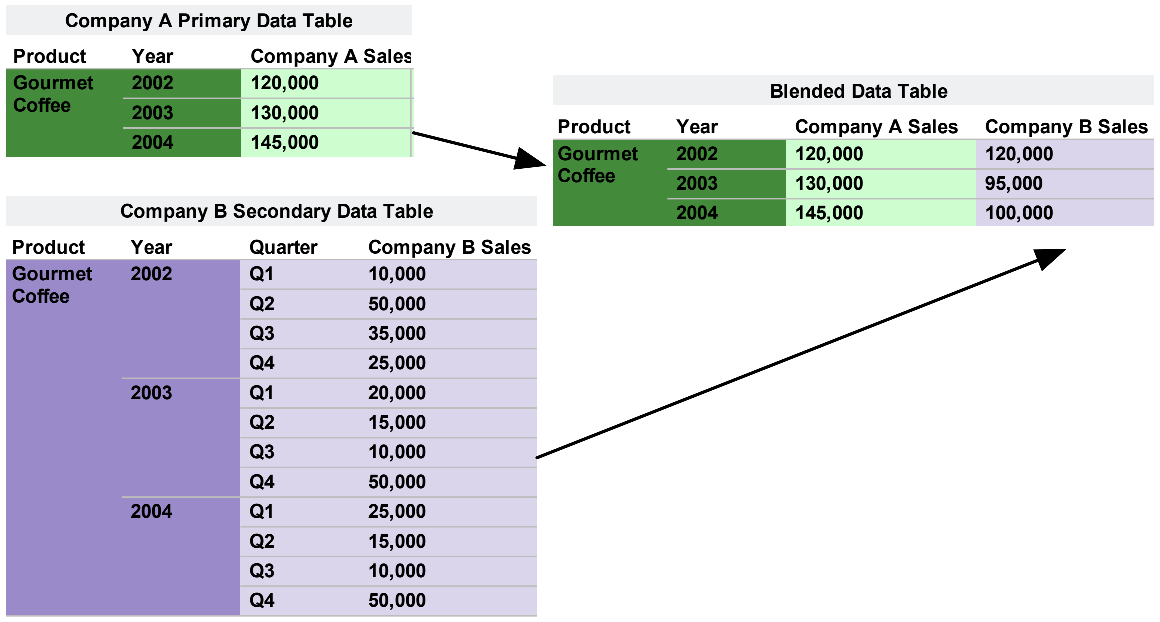


Figure 2.1: Architecture of Tableau

each row from A will be duplicated for each quarter in B for a given year resulting in an inaccurate overestimate of A's yearly earnings. This duplication problem can be avoided if, for example, company B's sales dataset were first aggregated to the level of year, then joined with company A's dataset. In this case, data blending detects that the data sets are at different granularities by examining their primary keys and notes that in order to join them, the common field is year (based on textual similarity of the attribute name). In order to blend the data on year, an aggregation query is issued to company B's dataset, which returns the sales aggregated up to the yearly level as shown in Figure 2.2. This result is blended with company A's dataset to produce the desired visualization of yearly sales for companies A and B. The blending feature does all of this on-the-fly without user-intervention.



Company A Primary Data Table

Product	Year	Company A Sales
Gourmet Coffee	2002	120,000
	2003	130,000
	2004	145,000

Company B Secondary Data Table

Product	Year	Quarter	Company B Sales
Gourmet Coffee	2002	Q1	10,000
		Q2	50,000
		Q3	35,000
		Q4	25,000
	2003	Q1	20,000
		Q2	15,000
		Q3	10,000
		Q4	50,000
	2004	Q1	25,000
		Q2	15,000
		Q3	10,000
		Q4	50,000

Blended Data Table

Product	Year	Company A Sales	Company B Sales
Gourmet Coffee	2002	120,000	120,000
	2003	130,000	95,000
	2004	145,000	100,000

Figure 2.2: Company A and B data tables (left) and blended result (right)

2) Resolves collocation problems. Many of Tableau's BI users are faced with challenges in integrating external data sources into IT-managed data repositories. Some of them simply cannot collocate their datasets for integration either because they lack the appropriate permissions or because moving their large data to their own locally-managed repository is expensive and untenable. In other cases, the data repository may have rigid structure, as with relational data cubes, to ensure performance, support security or protect data quality. Furthermore, it is often unclear if it is worth the effort of integrating an external data set that has uncertain value. The user may not know until she has started exploring the data if it has enough value to justify spending the time to integrate and load it into her repository. Thus, one of the paramount benefits of data blending is that it allows the user to quickly start exploring her data, and during exploration the integration happens automatically as a natural part of the analysis cycle. An interesting final benefit of the blending approach is that it enables users to seamlessly integrate across different types of data (which usually exist in separate repositories) such as relational, cubes, text files, spreadsheets, etc.

3) Adapts to needs of exploratory visual analytics. A key benefit of data blending is its flexibility; it gives the user the freedom to view blended data at different granularities and control how data is integrated on-the-fly. The blended views are dynamically created as the user is visually exploring the datasets. For example, the user can drill-down, roll-up, pivot, or filter any blended view as needed during her exploratory analysis. This feature is useful for data exploration and what-if analysis.

2.2 Data integration in Tableau: data blending

In this section, we discuss in greater detail how data blending works, beginning with a high-level overview in Section 2.2.1, a discussion of the blending architecture in Section 2.2.2, the semantics of blending in Section 2.2.3, how filters work in a blended view in Section 2.2.4, join key inference in Section 2.2.5, and a discussion of the cardinality of blended views in Section 2.2.6. We conclude this section with some illustrative blending examples in Section 2.2.7 taken from Tableau's users.

2.2.1 Data blending overview

The *data blending* feature, released in Tableau 6.0, allows an end-user to dynamically combine and visualize data from multiple heterogeneous sources (*e.g.,* Excel spreadsheets, text files, and relational databases) without any upfront integration effort. This feature automatically creates mediated schemas and wrappers as the user interactively builds a visualization on-the-fly. It also joins in only the necessary information from a data source (e.g., as specified by the user through the GUI) to create the view with minimal data movement, as queries are federated to the data sources in the case where the data cannot be collocated. A key aspect of the Tableau data blending feature is its ability to integrate data without causing any significant disruption to the analysis cycle: Tableau automatically infers how to combine the two datasets. Its inference abilities, however, are limited to joining datasets on the columns that share the same name and aggregating the new dataset if necessary and possible.

A user authors a visualization starting with a single data source (known as the *primary*), which establishes the context for subsequent blending operations in that visualization. Data

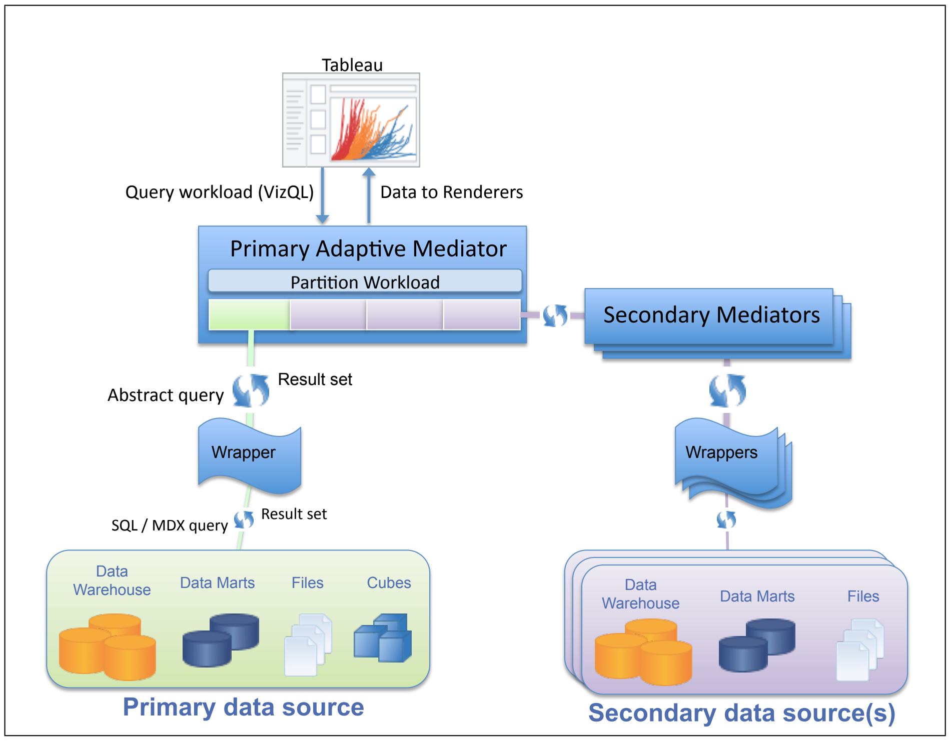


Figure 2.3: Tableau's data blending architecture

blending begins when the user drags in fields from a different data source, known as a secondary data source. Blending happens automatically, and only requires user intervention to resolve conflicts. Thus the user can continue modifying the visualization, including bringing in additional secondary data sources, drilling down to finer-grained details, etc., without disrupting their analytical flow. The novelty of this approach is that the entire architecture supporting the task of integration is created at runtime and adapts to the evolving queries in typical analytical workflows.

2.2.2 Data blending architecture

The data blending system, shown in Figure 2.3 takes as input the VizQL query workload generated by the user's GUI actions and data source schemas, and automatically infers how to query the data sources remotely and combine their results on-the-fly. The system features a two-tier, mediator-based architecture in which the VizQL query workload is analyzed and

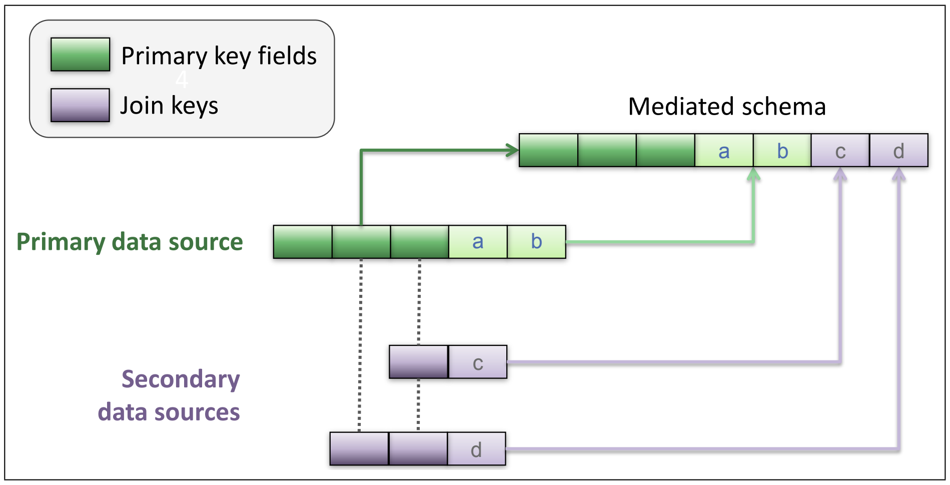
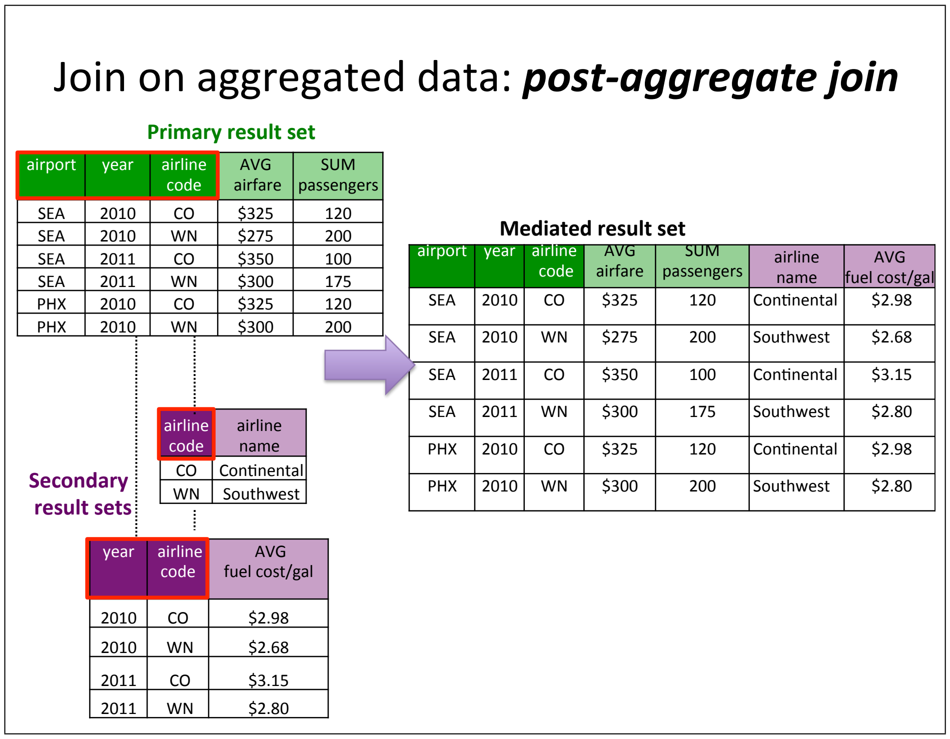


Figure 2.4: Components of the mediated schema

partitioned at runtime based on the corresponding data source fields being used. The primary mediator initiates this process by removing the visual encodings from the VizQL query workload to yield an *abstract query*. The abstract query is partitioned for further processing by the primary mediator and one or more secondary mediators. The primary mediator creates the mediated schema for the given query workload. It then federates the abstract queries to the primary data source as well as the secondary mediators and their respective data sources. The wrappers compile the abstract queries into concrete SQL or MDX queries and instantiate the semantic mappings between the data sources and the mediated schema for each query. The primary mediator joins all the result sets returned from all data sources to produce the mediated result set used by the rendering system.

2.2.3 Blending semantics: post-aggregate left join

In a blended visualization, the grouping fields from the primary data source become the primary key of the mediated schema. In Figure 2.4 these are shown as the dark-green fields in the primary data source, and the light-green fields represent the aggregated data. Each secondary data source must contain at least one field that matches a visualization grouping field in order to blend into the mediated schema. The matching fields in a secondary data



Join on aggregated data: ***post-aggregate join***

Primary result set

airport	year	airline code	AVG airfare	SUM passengers
SEA	2010	CO	$325	120
SEA	2010	WN	$275	200
SEA	2011	CO	$350	100
SEA	2011	WN	$300	175
PHX	2010	CO	$325	120
PHX	2010	WN	$300	200

Secondary result sets

airline code	airline name
CO	Continental
WN	Southwest

year	airline code	AVG fuel cost/gal
2010	CO	$2.98
2010	WN	$2.68
2011	CO	$3.15
2011	WN	$2.80

Mediated result set

airport	year	airline code	AVG airfare	SUM passengers	airline name	AVG fuel cost/gal
SEA	2010	CO	$325	120	Continental	$2.98
SEA	2010	WN	$275	200	Southwest	$2.68
SEA	2011	CO	$350	100	Continental	$3.15
SEA	2011	WN	$300	175	Southwest	$2.80
PHX	2010	CO	$325	120	Continental	$2.98
PHX	2010	WN	$300	200	Southwest	$2.80

Figure 2.5: Concrete example of post-aggregate join involving two secondary data sets and a primary dataset on airline data. If the secondary table at the bottom had fuel costs/gal per quarter, then the data would have to be aggregated into a per-year value.

source comprise its join key, and fields that appear in the `GROUP BY` clause issued by the secondary mediator wrappers. The aggregated data from the secondary data source, shown in light-purple, is then left-joined along its join key into the mediated result set. We refer to this left-join of aggregated result sets as a *post-aggregate join*. A concrete example is shown in Figure 2.5 involving airline data.

2.2.4 Filtering data from a blended view

Tableau provides several options for filtering data. Data may be filtered based on aggregate conditions, such as excluding airlines having a low total count of flights. A user can filter aggregate data from the primary and secondary data sources in this fashion, which results in rows being removed from the mediated result set. In contrast, row-level filters are only allowed for the primary data source. To improve performance of queries sent to the secondary data sources, Tableau will filter the join keys to exclude values which are not present in the domain of the primary data source result set, since these values would be discarded by the left-join.

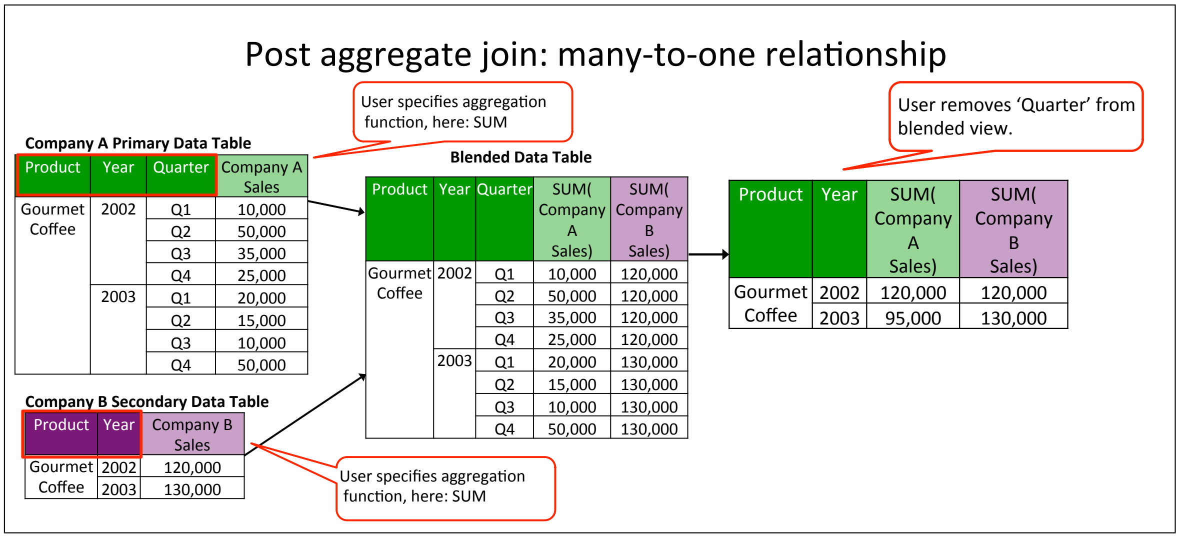
2.2.5 Join key inference

Tableau uses very simple rules for automatically detecting candidate join keys: 1) the secondary data source field name must match a field with the same name in the primary data source, 2) the data types must match 3) if they are date/time fields, they must represent the same granularity date bin in the date/time hierarchy, *e.g.,* both are MONTH. A user can intervene to force a match either by providing field captions to rename fields within the Tableau data model, or by explicitly defining a link between fields using a simple user interface.

2.2.6 Discussion of cardinality of blended view

As a user blends additional data into her visualization/view, she is guaranteed that the blended data will not affect the cardinality of the view. The left-join ensures that no rows from the primary data source result set are lost due to missing data from a secondary data source. Additionally there cannot be a one-to-many mapping between the domain values of the primary key and those of the secondary join key, because the secondary join key is a subset of the primary key and contains only unique values in the aggregated secondary result set. We find that this approach is the most natural for augmenting a visualization with secondary data sources of uncertain value or quality, which is a common scenario for Tableau users.

Data blending supports many-to-one relationships between the domain values of the primary key and each secondary join key. A many-to-one relationship can occur when the secondary data source contains coarser-grained data than the mediated result set. Figure 2.6 shown such an example with a simple coffee dataset. Since the join key in a secondary result set may match a subset of the blended result set primary key, portions of the secondary


Post aggregate join: many-to-one relationship
User specifies aggregation function, here: SUM
User removes 'Quarter' from blended view.
User specifies aggregation function, here: SUM

Company A Primary Data Table

Product	Year	Quarter	Company A Sales
Gourmet Coffee	2002	Q1	10,000
		Q2	50,000
		Q3	35,000
		Q4	25,000
	2003	Q1	20,000
		Q2	15,000
		Q3	10,000
		Q4	50,000

Company B Secondary Data Table

Product	Year	Company B Sales
Gourmet Coffee	2002	120,000
	2003	130,000

Blended Data Table

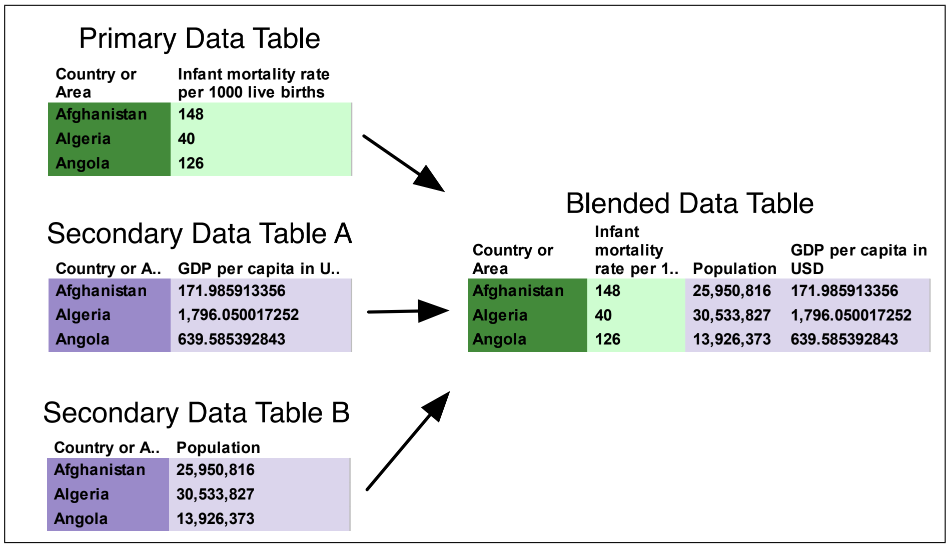
Product	Year	Quarter	SUM(Company A Sales)	SUM(Company B Sales)
Gourmet Coffee	2002	Q1	10,000	120,000
		Q2	50,000	120,000
		Q3	35,000	120,000
		Q4	25,000	120,000
	2003	Q1	20,000	130,000
		Q2	15,000	130,000
		Q3	10,000	130,000
		Q4	50,000	130,000

Product	Year	SUM(Company A Sales)	SUM(Company B Sales)
Gourmet Coffee	2002	120,000	120,000
	2003	95,000	130,000

Figure 2.6: Concrete example of blending a coarser-grained secondary data source with a finer-grained primary dataset on synthetic coffee data.

result set may be duplicated across repeated values in the mediated result set. This does not pose a risk of double-counting measure values, because all aggregation is performed prior to the join. And once an attribute has been aggregated, Tableau does not permit any further aggregations to be performed on top of an already aggregated data value. However, in the example in Figure 2.6, the user can remove the 'Quarter' attribute from the blended table to roll up the view to only show the aggregated `SUM` of values for each year. When a blended visualization uses multiple secondary data sources, each secondary join key may match any subset of the primary key, as shown in the example from Figure 2.5. The primary mediator handles duplicating each secondary result set as needed to join with the mediated result set.

Finally, a secondary dimension which is not part of the join key (and thus not a grouping field in the secondary query) can still be used in the visualization. If it is functionally dependent on the join key, a secondary dimension can be used without affecting the result set cardinality. Tableau references this kind of non-grouping dimension using both `MIN` and `MAX` aggregations in the query issued to the secondary data source, which allows Tableau



Primary Data Table

Country or Area	Infant mortality rate per 1000 live births
Afghanistan	148
Algeria	40
Angola	126

Secondary Data Table A

Country or A..	GDP per capita in U..
Afghanistan	171.985913356
Algeria	1,796.050017252
Angola	639.585392843

Secondary Data Table B

Country or A..	Population
Afghanistan	25,950,816
Algeria	30,533,827
Angola	13,926,373

Blended Data Table

Country or Area	Infant mortality rate per 1..	Population	GDP per capita in USD
Afghanistan	148	25,950,816	171.985913356
Algeria	40	30,533,827	1,796.050017252
Angola	126	13,926,373	639.585392843

Figure 2.7: Small sample tables of infant mortality rates, GDP, and population in 2000

to determine if the dimension is functionally dependent on the join key. For each row in the secondary result set, if the two aggregated values are the same then the value is used as-is, reflecting the functional dependence on the grouping fields. If the aggregated values differ, Tableau represents the value using a special form of `NULL` called *ManyValues*. This is represented in the visualization as a '*', but retains the behavior of `NULL` when used in calculated fields or other computations. The visual feedback allows a user to distinguish this lack of data from the `NULL` values which occur due to missing or mismatched data.

2.2.7 Blending examples from Tableau users

All of the data and visualizations discussed in this section are from real users of Tableau who made them publicly available through Tableau's online version, *Tableau Public.*[1].

Simple blending example: Infant mortality vs. GDP. Three unique data sources (see left half of Figure 2.7 for sample tables) are blended together to create the visualization shown in Figure 2.8. In this example, the Tableau user wants to understand if there is a

[1]https://public.tableau.com

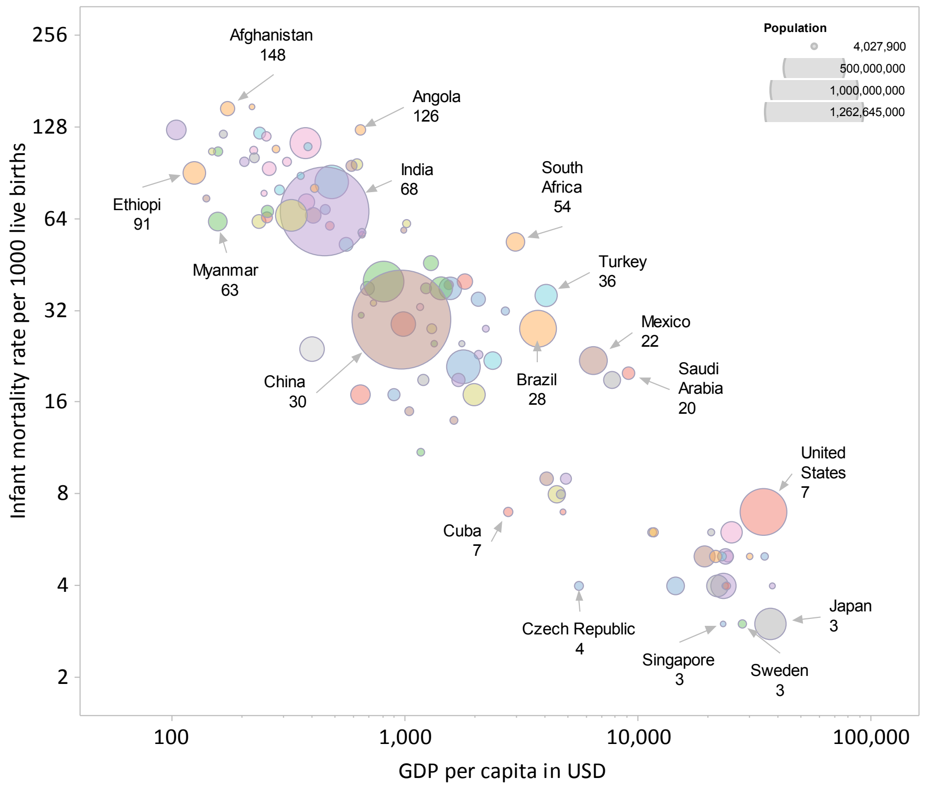


Figure 2.8: Blended view of infant mortality rates, GDP, and population in 2000

connection between infant mortality rates, GDP, and population. She has three distinct spreadsheets with the following characteristics: the first data source contains information about the infant mortality rates per 1000 live births for each country, the second contains information about each country's total population, and the third source contains country-level GDP. For this analysis task, the user drags the fields, `"Country or Area"` and `"Infant mortality rate per 1000 live births"`, from her first data source onto the blank visual canvas. Since these fields were the first ones selected by the user, then the data source associated with these fields becomes the primary data source. This action produces a visualization showing the relative infant mortality rates for each country. But the user wants to understand if there is a correlation between GDP and infant mortality, so she then drags the `"GDP per capita in US dollars"` field onto the current visual canvas from Data Table A. The step to join the GDP measure from this separate data source happens automatically: the

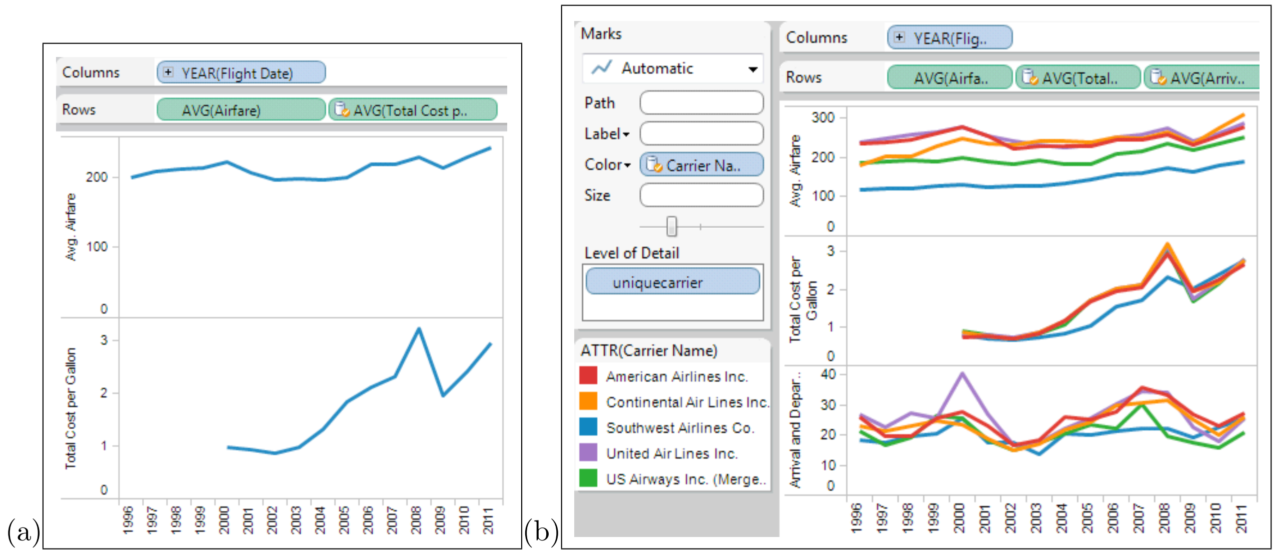


Figure 2.9: Screenshots of blended airline data from the US Bureau of Transportation Statistics:
(a) Two blended datasets: airfare price (324M rows) and fuel costs (8K rows) by year
(b) Four blended datasets: both datasets from (a), on-time performance (140M rows), and carrier names (1.5K rows).

blending system detects the common join key (i.e. `"Country or Area"`) and combines the GDP data with the infant mortality data for each country. Finally, to complete her analysis task, she adds the `"Population"` measure from Data Table B, to the visual canvas, which produces the visualization in Figure 2.8 associated with the blended data table in Figure 2.7.

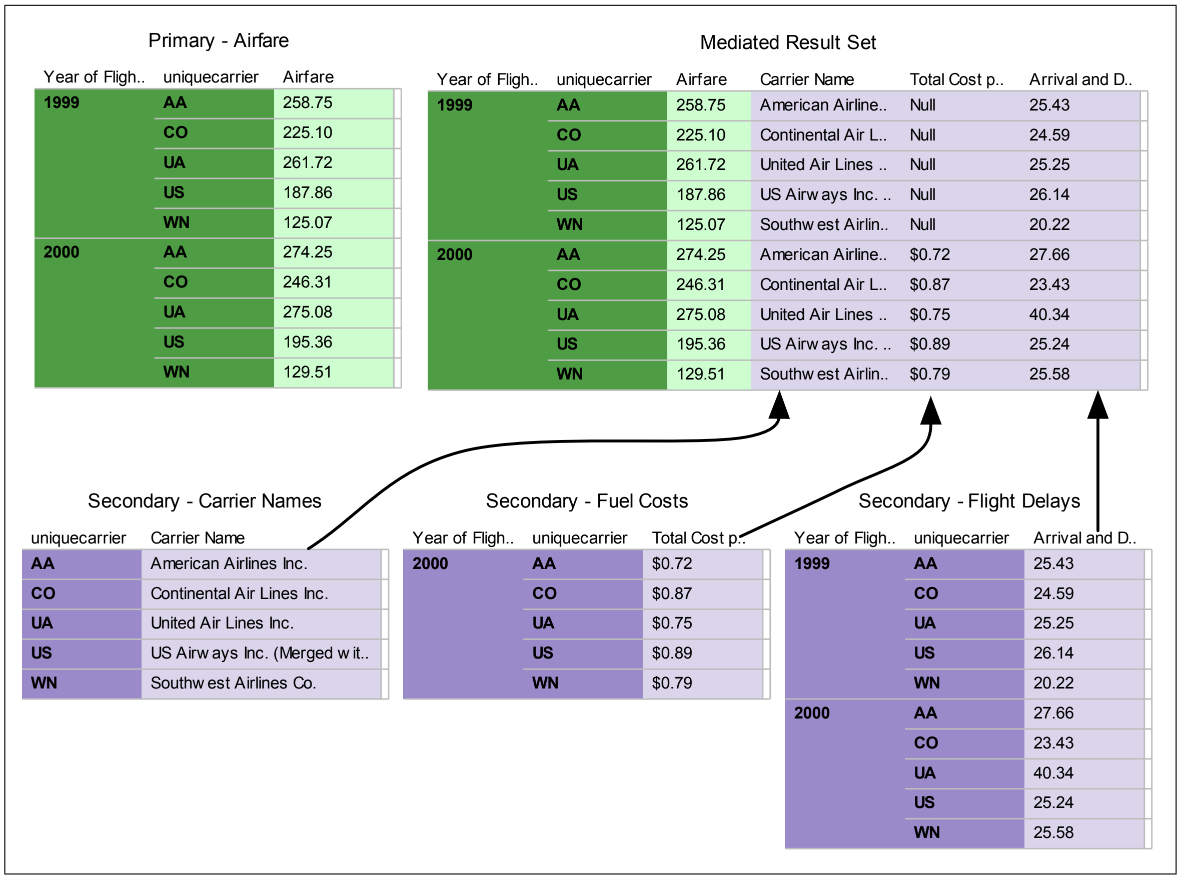
Simple blending example: Airfare price vs. fuel costs. A Tableau data blending scenario is shown in Figure 2.9, which includes multiple views that were composed in minutes by uniquely combining four different airline datasets, the largest of which includes a 324 million row ticket pricing database and a 140 million row on-time performance database. A user starts by dragging fields from any dataset on to a blank visual canvas, iteratively building a VizQL statement which ultimately produces a visualization. In this example, the user first drags the VizQL fields, `YEAR(Flight Date)` and `AVG(Airfare)`, from the pricing dataset onto the visual canvas.

Data blending occurs when the user adds fields from a separate dataset to an existing

VizQL statement in order to augment their analysis. Tableau assigns the existing dataset to the primary mediator and uses secondary mediators to manage each subsequent dataset added to the VizQL. The mediated schema has a primary key composed of the grouping VizQL fields from the primary dataset (*e.g.,* `YEAR(Flight Date)`); the remaining fields in the mediated schema are the aggregated VizQL fields from the primary dataset along with the VizQL fields from each secondary dataset. Continuing our example, the user wishes to drag `AVG(Total Cost per Gallon)` from the fuel cost dataset to the visualization. The schema matching algorithm examines the secondary dataset for one or more fields whose name exactly matches a field in the primary key of the mediated schema. While the proposed matches are often sufficient and acceptable, the user can specify an override. Since the fuel cost dataset has a field named `Date`, the user provides a caption of `Flight Date` to resolve the schema discrepancy.

At this point the mediated schema is created and the VizQL workload is then federated to the wrappers for each dataset. Each wrapper compiles VizQL to SQL or MDX for the given workload, executes the query, and maps the result set into the intermediate form expected by the primary mediator. The mapping is performed dynamically, since both the VizQL and the data model evolve during a user's iterative analytical workflow. Finally, the primary mediator performs a left-join of each secondary result set along the primary key of the mediated schema. In this example, the mediated result set is rendered to produce the visualization shown in Figure 2.9(a).

Evolved Blending Example: Airfaire price vs. fuel costs and delays. Figure 2.9(b) shows further evolution of the analysis of airline datasets, and demonstrates several key points of data blending. First, the user adds a unique ID field named `uniquecarrier` from the primary dataset to the VizQL to visualize results for each airline ID over time. The mediated schema adapts by adding this field to its primary key, and the secondary mediator automatically queries the fuel cost dataset at this finer granularity since it too has a field named `uniquecarrier`. Next, the user decorates the visualization with descriptive airline names for each airline ID by dragging a field named `Carrier Name` from a lookup table. This dataset is at a coarser granularity than the existing mediated schema, since it does not



Primary - Airfare

Year of Fligh..	uniquecarrier	Airfare
1999	AA	258.75
	CO	225.10
	UA	261.72
	US	187.86
	WN	125.07
2000	AA	274.25
	CO	246.31
	UA	275.08
	US	195.36
	WN	129.51

Mediated Result Set

Year of Fligh..	uniquecarrier	Airfare	Carrier Name	Total Cost p..	Arrival and D..
1999	AA	258.75	American Airline..	Null	25.43
	CO	225.10	Continental Air L..	Null	24.59
	UA	261.72	United Air Lines ..	Null	25.25
	US	187.86	US Airw ays Inc. ..	Null	26.14
	WN	125.07	Southw est Airlin..	Null	20.22
2000	AA	274.25	American Airline..	$0.72	27.66
	CO	246.31	Continental Air L..	$0.87	23.43
	UA	275.08	United Air Lines ..	$0.75	40.34
	US	195.36	US Airw ays Inc. ..	$0.89	25.24
	WN	129.51	Southw est Airlin..	$0.79	25.58

Secondary - Carrier Names

uniquecarrier	Carrier Name
AA	American Airlines Inc.
CO	Continental Air Lines Inc.
UA	United Air Lines Inc.
US	US Airw ays Inc. (Merged w it..
WN	Southw est Airlines Co.

Secondary - Fuel Costs

Year of Fligh..	uniquecarrier	Total Cost p..
2000	AA	$0.72
	CO	$0.87
	UA	$0.75
	US	$0.89
	WN	$0.79

Secondary - Flight Delays

Year of Fligh..	uniquecarrier	Arrival and D..
1999	AA	25.43
	CO	24.59
	UA	25.25
	US	26.14
	WN	20.22
2000	AA	27.66
	CO	23.43
	UA	40.34
	US	25.24
	WN	25.58

Figure 2.10: Sample data tables for airline blending example

represent changes to the carrier name over time. The data blending system automatically handles this challenge by allowing the left-join to use a subset of the mediated result set primary key, and replicating the carrier name across the mediated result set. Figure 2.10 demonstrates this effect using a tabular view of a portion of the mediated result set, along with portions of the primary and secondary result sets. The figure also demonstrates how the left-join preserves data for years which have no fuel cost records. Last, the user adds average airline delays from a 140 million row dataset which matches on `Flight Date` and `uniquecarrier`. This is a fast operation, since the wrapper performs mapping operations on the relatively small, aggregated result set produced by the remote database. Note that none of these additional analytical tasks required the user to intervene in data integration

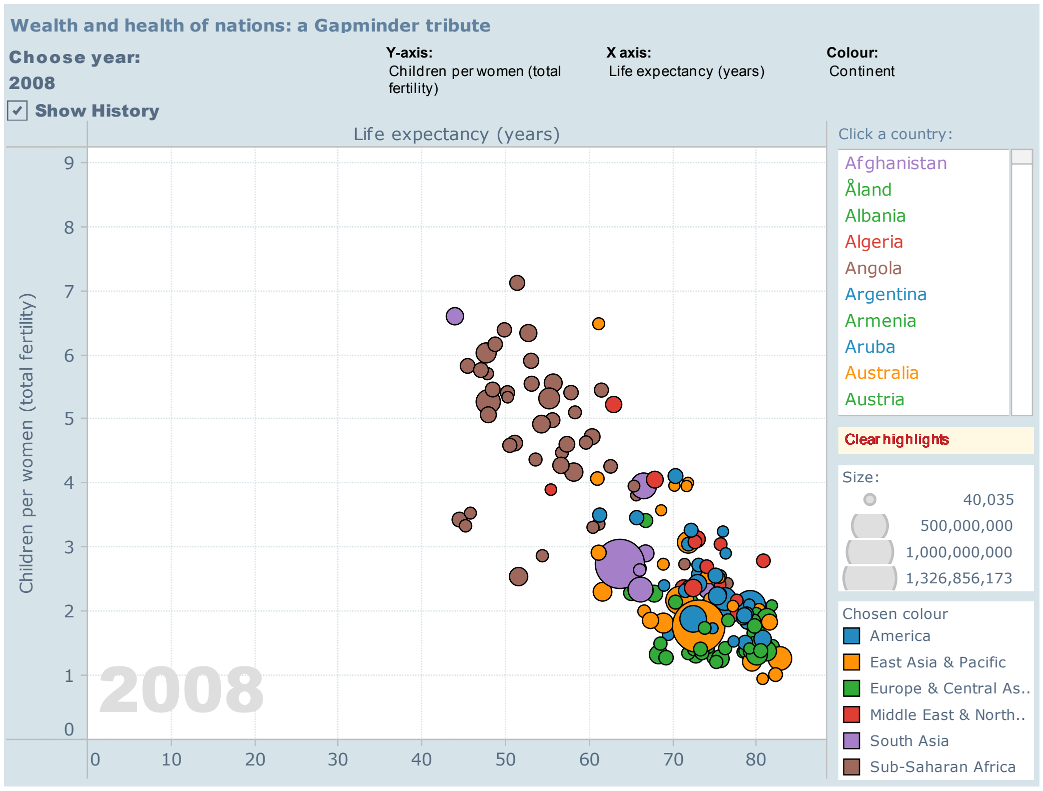


Figure 2.11: Gapminder tribute demonstrating on-the-fly creation of blended views

tasks, allowing their focus to remain on finding insight in the data.

Wealth and health of nations: a Gapminder tribute. A Tableau user created an interactive visualization inspired by Gapminder [43] (see Figure 2.11), which allows users to select and plot various demographic data in a scatterplot, with each data point representing a country. This is a nice example that demonstrates the flexibility of data blending. A user can select from three different data sources to blend on the x-axis (including life expectancy, CO_2 emissions, and population) and three sources for the y-axis (including number of children per woman, energy use, and income per person). This demonstrates the flexibility of the data blending feature, namely that users can dynamically change their blended views by pivoting on different data sources and measures to blend in their visualizations.

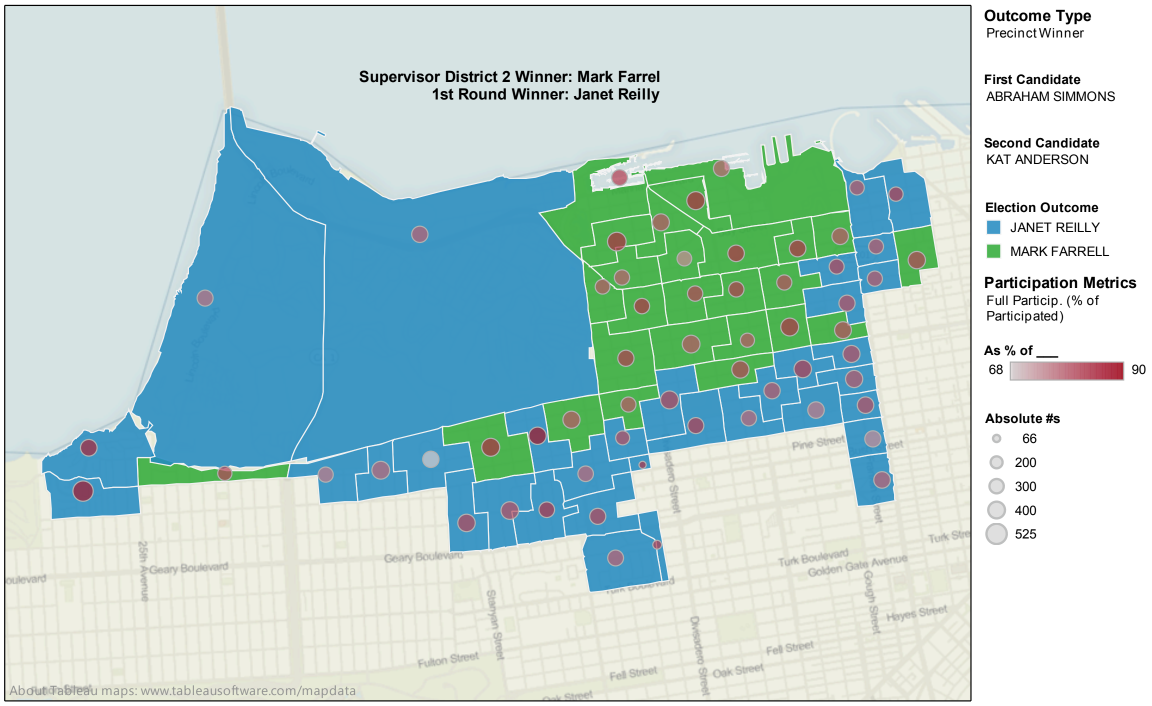


Figure 2.12: San Francisco election outcome modeling

Modeling election outcomes. Figure 2.12 illustrates the possible outcomes of an election for District 2 Supervisor of San Francisco. With this type of visualization, the user can select different election styles and see how their choice affects the outcome of the election. What's interesting from a blending standpoint is that this is an example of a many-to-one relationship between the primary and secondary datasets. This means that the fields being left-joined in by the secondary data sources match multiple rows from the primary dataset and results in these values being duplicated. Thus any subsequent aggregation operations would reflect this duplicate data, resulting in overestimates. The blending feature, however, prevents this scenario from occurring by performing all aggregation prior to duplicating data during the left-join.

2.3 Data cleaning in Tableau

In this section, we discuss the current data cleaning capabilities of Tableau, including support for changing data values with aliases, calculated fields, and ad-hoc groupings. It should be noted that all of these cleaning actions are manually specified by the user while they are exploring their data visually.

Correcting data values with aliases. Tableau supports manual user intervention in resolving field names when schema matching fails (exact string match on the attribute name). And once the schemas match and data is blended, the visualization can help provide feedback regarding the validity of the underlying data values and domains. If there are any data inconsistencies, users can provide aliases for a field's data values which will override the original values in any query results involving that field. The primary mediator performs a left-join using the aliases of the data values, allowing users to blend data despite discrepancies from data entry errors and spelling variations. Tableau provides a simple user interface for editing field aliases.

Transforming data values with calculated fields. Calculated fields are another aspect of Tableau's data model which support data cleaning. Calculated fields support arbitrary transformations of original data values into new data values, such as trimming whitespace from a string or constructing a date from an epoch-based integer timestamp. As with database fields, calculated fields can be used as primary keys or join keys.

Manual entity resolution with ad-hoc groups. Last, Tableau allows users to organize a field's related data values into groups. These *ad-hoc groups* can be used for manual entity resolution, such as binding multiple variations of business names to a canonical form. Ad-hoc groups also allow constructing coarser-grained structures, such as grouping states into regions. Data blending supports joins between two ad-hoc groups, as well as joins between an ad-hoc group and a string field.

Chapter 3
RELATED WORK

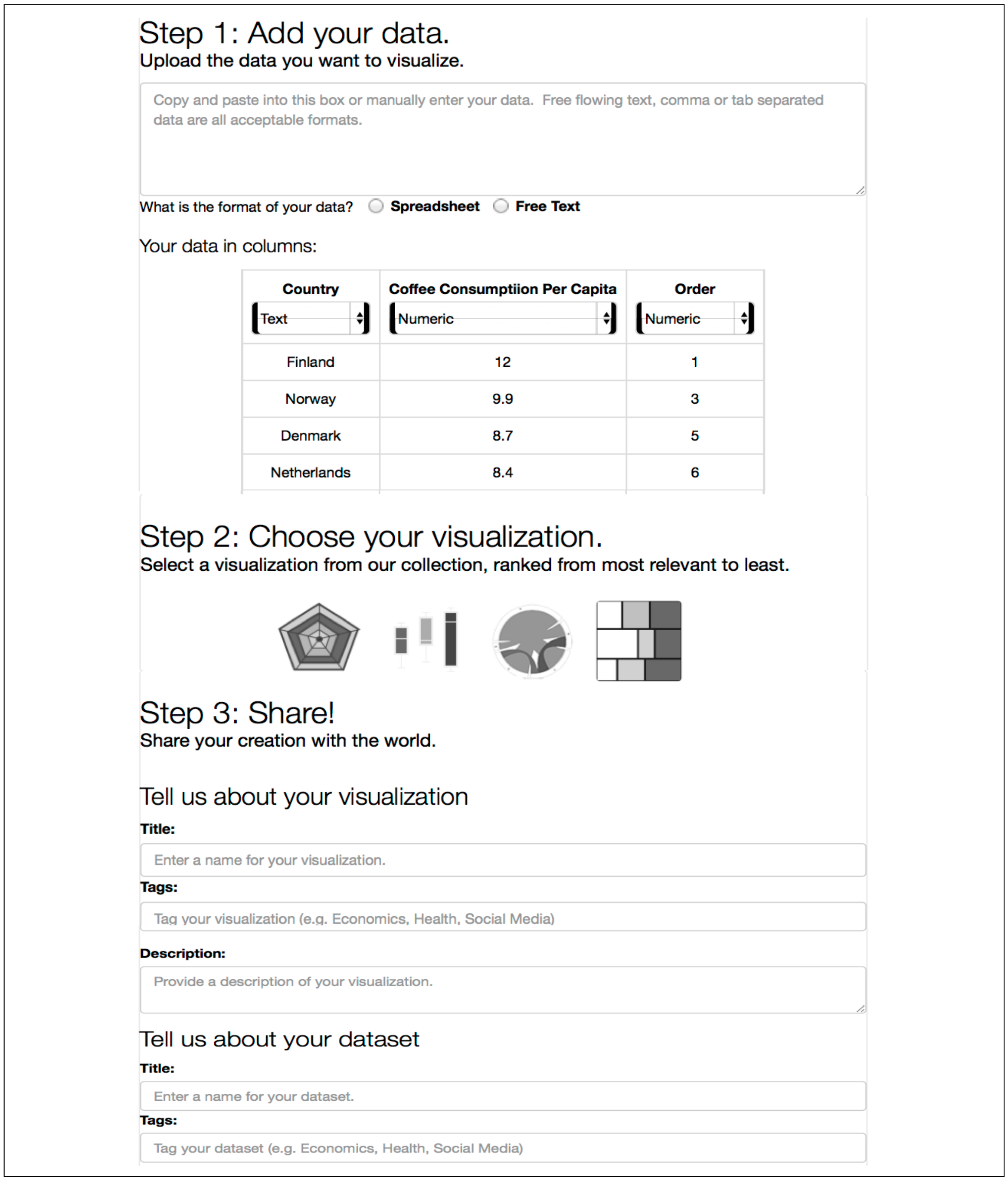
In this chapter, we discuss the existing literature and tools that support data enthusiasts in their analytical workflows. This chapter is organized as follows. In Section 3.1, we first present the capabilities and limitations of the state-of-the-art interactive visual analytics tools. Then in Section 3.2, we discuss a complex analytical workflow step, data integration, for which existing visual analytics tools provide limited support. Integrating heterogeneous datasets is challenging because across datasets, the data values and schemas are not canonicalized or standardized. We discuss the latest techniques that address common data quality problems through interactive cleaning of data in Section 3.3 and resolving duplicate entities, or *deduplication*, in Section 3.4. Note that this thesis does not solve any specific data integration problem other than deduplication. The related work thus focuses primarily on that latter problem.

3.1 Interactive Visual Data Analytics

There has been a wealth of commercial and academic visual analytics systems that support data exploration through a visual interface with *visual queries*. Users author such queries through drag-and-drop interactions with the visualization system's Graphical User Interface (GUI). Some popular visual analytics systems from industry include Tableau [113] and its web-facing Tableau Public [112], Fusion Tables [46, 47], PowerPivot [86], Qlikview [98], and Spotfire [1, 107]. Some notable systems from academia include Many Eyes [125],Visage [102], DEVise [80], DataSplash [142], imMens [79], and VIQING [95]. These tools support what is called the *sensemaking model* [21]: The typical analytical process starts with a question that a data enthusiast seeks to answer. The data enthusiast then forages for relevant data unless she already has a dataset to explore. Once the appropriate dataset is acquired, the data is explored through an appropriate visualization. The user continues to interact with the visualization by, for example, drilling down to the details or pivoting out some dimensions. To support the sensemaking process for data enthusiasts, these analytics systems share the common usability characteristic that no programming or database experience is necessary to query and explore the data; users instead author queries and visualizations through their interactions (*e.g.,* drag-and-drop or pull-down a menu) with the visualization system's GUI. Figure 1.1 shows Tableau's GUI and Figure 3.1 shows Many Eyes's GUI as two illustrative examples.

Tableau, however, has the following distinguishing characteristics from other related visual analytics systems, which motivates our focus on the Tableau system in this dissertation:

1. A formal declarative language that combines query, analysis, and visualization into a single framework (called Visual Query Language, or VizQL [110, 54]). See Figure 1.1 (bottom) for an illustrative example. VizQL is not visible to the user, but rather is automatically generated by the system in response to the user's interactions with the GUI. It is simply an intermediate representation describing the data and visualization.
2. An architecture that compiles the VizQL specification automatically into two parts: the queries (SQL or MDX) and the drawing commands necessary to generate the visualization. This gives Tableau users the unique flexibility of changing the query used to fetch the data and their view of it simultaneously.
3. Incremental view/visualization construction in which the user explores a subset of the full dataset (*e.g.,* applying a row-level filter or projecting a subset of the columns) at-a-time. The systems from related work, in contrast, load the entire data set first from disk to the in-memory database before the user can issue queries. This model either requires that the user already knows what questions to ask (not natural for exploring a database) or incurs the overhead of loading unnecessary data.
4. Query federation, which supports flexible exploration of one or multiple heterogeneous data sources at a time, integrating their result sets on-the-fly. In contrast, the visualizations systems from related work that also integrate data [46, 47], require that the entire data sources be integrated before they can be explored and visualized. Again, this forces the user to know what queries to ask in advance, which is not natural for



Step 1: Add your data.
Upload the data you want to visualize.

Copy and paste into this box or manually enter your data. Free flowing text, comma or tab separated data are all acceptable formats.

What is the format of your data? ○ Spreadsheet ○ Free Text

Your data in columns:

Country	Coffee Consumptiion Per Capita	Order
Text	Numeric	Numeric
Finland	12	1
Norway	9.9	3
Denmark	8.7	5
Netherlands	8.4	6

Step 2: Choose your visualization.
Select a visualization from our collection, ranked from most relevant to least.

Step 3: Share!
Share your creation with the world.

Tell us about your visualization

Title:
Enter a name for your visualization.

Tags:
Tag your visualization (e.g. Economics, Health, Social Media)

Description:
Provide a description of your visualization.

Tell us about your dataset

Title:
Enter a name for your dataset.

Tags:
Tag your dataset (e.g. Economics, Health, Social Media)

Figure 3.1: GUI for uploading data and creating a visualization in the Many Eyes system.

exploration, or to perform potentially unnecessary extra work .

5. Basic manual data cleaning support for: (1) entity resolution, (2) aliases for fixing inconsistent data values, and (3) basic string transformations (*e.g.,* trim whitespace, upper/lower case, etc.). However, these cleaning operations are all manually specified and the system does not provide any assistance on predicting data cleaning issues nor on resolving them.

These features make it possible to incrementally build data views and visualizations, which is key to smoothly supporting interactive data exploration in the sensemaking process. Users author VizQL expressions by dragging and dropping fields from their datasets onto a visual canvas. In this dissertation, we thus take a pay-as-you-go approach to data cleaning by focusing on cleaning the data that actually impacts the view/visualization. We refer the reader to Chapter 2 for more details on Tableau.

In contrast to Tableau, many of the current visual analytics systems lack support for data federation, integration, and cleaning. Separate tools also exist for supporting data federation (*e.g.,* IBM's InfoSphere Federation Server [62] and Cisco's Data Federation tool [25]), data integration (*e.g.,* Microsoft Data Explorer [82]) and data cleaning (*e.g.,* Trifacta [117] and Wrangler [68]). However, in this dissertation, we focus on Tableau as it provides the closest unified model of visualization with initial support for data cleaning and integration.

In Chapter 4, we present a measurement study of Many Eyes and Tableau Public. We study how these systems are being used for visual data analysis and find that they have much room for growth: they attract large numbers of users but most users do not push the limit of what these tools can do.

3.2 Interactive Data Integration

Integrating different data sources is challenging due to the fact that data is heterogeneous: different schemas, deviations of values from a canonical form, different data granularities, different domains, etc [34]. There are two primary activities that require significant human effort upfront: creating the mediated schema and the semantic mappings between the individual data source schemas and the mediated schema [34]. The mediated schema is the schema of the result set or view over the original data sets. A concrete example is shown in Figure 2.5. The semantic mappings are typically a set of wrapper code, which allow information to be retrieved directly from original data sets. Such integration activities require detailed knowledge of the domain as well as an understanding of the queries that need to be supported.

Data integration is an old problem involving matching the schemas between two data sets (called *schema matching*) and issuing queries to the mediated schema such that they can be mapped to the original data sets to retrieve the requested information (called *schema mapping*). Many solutions have previously been proposed for the tasks of schema mapping and matching [65, 82, 41, 97, 99, 59]. Related work on dataspace systems [39, 28] advocates automating these two primary integration activities as much as possible. For example, assisting users with the schema matching by allowing fuzzy matches on identifier names instead of insisting on exact-matches. The resulting integration should give best-effort answers and allow for improving the system in a pay-as-you-go fashion. In other words, the user should get immediate benefit of investing time in using the data management functionality. For example, visualization systems like Tableau and Fusion Tables [46, 47] can help infer some missing schema-level information (*e.g.,* data types of columns) by checking the type of the data values in that column. Such systems also leverage the type information of the columns to suggest relevant visualizations (*e.g.,* if the column contains latitude/longitude coordinates then the system suggests a map view). Other pay-as-you-go data integration only systems like InfoChimps [65], Microsoft Data Explorer (formerly the Montego project [82]), Clio [41, 97], Clip [99] (which builds on Clio), and MDQ [59] generate schema matches and mappings semi-automatically using a combination of inference techniques and human-supplied annotations (*e.g.,* by drawing lines across schema elements or textual ones that carry transformation semantics). In summary, the pay-as-you-go approach to data integration is a natural fit for exploratory visual analysis, allowing the user to pay the cost of integrating her datasets incrementally as she explores them.

Currently there are two primary types of pay-as-you-go integration systems: federation [34] and transformation (*i.e.,* extract-transform-load, or ETL) [34]. The federated approach queries the data at the sources (only moving the data that satisfies the query) and combines the result sets on-the-fly using a distributed query processing engine. In contrast, ETL systems load the data into a local data store in advance of query processing. Recent prior work that combines visual analysis with data integration [46, 47, 98, 106, 107] take an ETL approach. Tableau, in contrast, employs a federated approach that integrates data that has been first aggregated at the sources. This approach is well-suited to interactivity, as it (1) only moves data necessary to answer a question (2) only moves aggregated data, and (3) leverages the power of the fast database systems.

In this dissertation, we address a common data quality problem, deduplication, which occurs in the context of integrating two data sources.

3.3 Interactive Data Cleaning

Data can contain a variety of quality issues such as inconsistent or wrong values, missing values, and duplicate entries. Data cleaning deals with the detection and removal of such errors and inconsistencies from data in order to improve the quality of data. Data quality problems are present in single data sources, such as text files and databases, *e.g.,* due to misspellings during data entry, missing information, or other invalid data. Data quality problems are worsened by data integration because of inconsistencies in data representation between the original data sources and the presence of redundant but not identical information. As a result, data cleaning is often a first step taken by users when they integrate heterogeneous data sources because the sources often contain redundant data in different representations (*e.g.,* deduplication). Different data representations need to be canonicalized and deduplicated to ensure that the user's analytical queries are over accurate and consistent data.

State-of-the-art data cleaning systems such as Wrangler [68], Potter's Wheel [100], Google Refine [61], and Microsoft Data Explorer (formerly Montego [82]) support an ETL-style of first loading all of the data into a local data store and performing the cleaning operations holistically. These systems present the data visually as a spreadsheet and support basic data cleaning transformations such as splitting or merging data columns, table pivot/unpivot, upper/lower case values, etc. This approach to cleaning, however, is expensive and conflicts with the real-time constraints in interactive, exploratory environments.

Moreover, it forces the user to either clean ahead of time, delaying time to first visualization, or to context-switch away from the visualization. Interestingly, only one of the interactive data cleaning systems from the literature takes an incremental approach to data cleaning (DBWipes [144]). In this system, users execute aggregate queries, and the system presents the result visually where the user can interactively detect and clean data value errors in the query results. While the DBWipes system helps users identify the source of data quality problems while exploring the data visually in increments, it requires the user to write aggregate SQL queries and provides no support for complex data cleaning challenges such as deduplication. This system, thus, has limited applicability to data enthusiasts. Moreover, none of these state-of-the art interactive systems support cleaning/detection operations over integrated data sources. An important cleaning operation, deduplication, which we discuss next, is thus unsupported by the current visual analytics systems. Even state-of-the-art industrial data cleaning systems such as Trifacta [117] offer only basic support for deduplication (*i.e.,* exact matches on all attributes).

3.4 Deduplication

In this section, we present the problem of deduplication in detail. We begin with a description of the deduplication problem and present some basic examples in 3.4.1. Then in 3.4.2, we discuss the common deduplication workflow steps taken by state-of-the-art systems from the literature. Deduplication is typically modeled as a binary classification problem, in which a matching function is used to classify pairs of entities as matching or not. However, manually creating an accurate matching function requires technical expertise. Thus, in supporting data enthusiasts, we focus on presenting the literature in which this matching function is a classifier that can be automatically learned from a labeled set of examples in 3.4.3.

3.4.1 Deduplication Problem

Deduplication is a fundamental problem in data integration dealing with multiple different representations of the same entity (due to non-canonicalized data values across the data sources being integrated). One common data quality problem is the existence of duplicate

Table 3.1: Rows r_1 and r_2 are exact duplicates and rows r_1 and r_3 are fuzzy duplicates.

	`First Name`	`Last Name`	`Address`	`Phone`
r_1	Jennifer	Smith	123 Elm St.	(206) 867 5309
r_2	Jennifer	Smith	123 Elm St.	(206) 867 5309
r_3	Jenny	Smith	123 Elm Street	(206) 867 5309

records that refer to the same real-world entity. This problem can occur in any dataset, but it is most common in datasets that are the result of data integration. For example, the same restaurant may appear under two different phone numbers. The same product may use different abbreviations in its name or may include a different description. Therefore, in order to integrate two or more data sources it is necessary to recognize representations that refer to the same real-world entity. In integration scenarios where the duplicate entities are exact replicas, they are trivial to identify and remove. A simple select-distinct query suffices. However, in most cases the duplicate entities are not exact replicas (*e.g.,* the data may contain misspellings, be incomplete, incorrect, or inconsistent). Given the tendency for data to exhibit heterogeneity in values, the task of identifying duplicates is challenging. This setting is the one that we consider in Chapter 5 of this dissertation. Thus, such duplicates are also called *fuzzy duplicates* [24]. We use the term duplicate to refer to fuzzy duplicates. In Table 3.1, records r_1 and r_2 are exact duplicates since they match exactly on the data values for all four attributes. Row r_3 is a fuzzy duplicate with either r_1 or r_2 since two of the attributes (`Last Name` and `Phone`) match perfectly on the data values, but the other two (`First Name` and `Address`) attributes are close enough that they are likely to be representing the same entity: “Jennifer” vs. “Jenny” and “123 Elm St.” vs. “123 Elm Street”.

Deduplication (also known as entity matching, entity resolution, duplicate detection, or record matching) has a long history in the databases literature (see [24, 34, 44] for the basic principles and [37] for a recent survey). In the information retrieval literature, it is known as near-duplicate detection [52], and is applied in the context of detecting duplicate

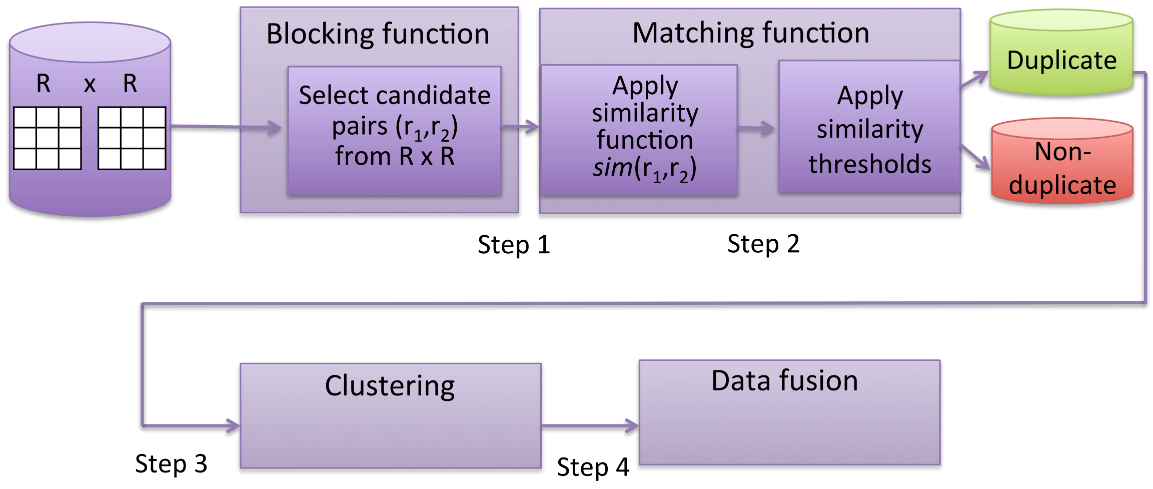


Figure 3.2: Duplicate detection workflow steps for a relation R, which is suspected to contain duplicates.

documents returned by (Web) search queries.

3.4.2 Steps in Deduplication

The high-level approach to deduplicating a set of records, R, takes the following steps. The process requires as input a similarity function that takes a pair of tuples (t_1, t_2) from R and produces a similarity score [33]. This similarity function, together with a similarity threshold, can be applied to all pairs of records in R to determine which ones match [37]. Alternatively, a system may rely solely on users to indicate which tuples match [71, 14]. Multiple tuples can correspond to one entity and such clusters further need to be identified [2, 12, 129, 122, 130]. Once matching tuples are identified, they must be merged [51, 16]. To make the previous steps more compute-efficient by reducing the number of record comparisons, blocking techniques are used [13]. Blocking is an inexpensive heuristic filtering step that either partitions the tuples that get compared or removes pairs with low similarity scores.

Figure 3.2 shows a typical duplicate detection process. A set of records R is suspected to contain duplicates. The key deduplication steps are as follows:

Step 1: Define a similarity function. The promising record pairs are input to a similarity function, which produces a quantitative value $\in$ [0,1] that represents how close the data values are to each other for individual attributes. These quantitative values (*i.e., features*) are stored in a *feature vector* for each pair, as shown for our running example in Table 3.2. Commonly-used similarity measures for string typed attributes, for example, include string equality, edit distance, jaccard, and cosine similarity [37]. For numeric types, some functions include euclidean and manhattan distance.

Step 2: Define a matching function. Next, to compute the overall similarity between two records (called the *matching function*), the building blocks are the similarity measures between individual attributes in these records. The similarity function can be complex, consisting of a weighted set of conjunctive and/or disjunctive terms of similarity functions. The similarity function applies similarity thresholds to individual similarity measures, which are used to decide whether the pair is indeed a duplicate or not [33].Alternatively, a system may rely solely on users to indicate which tuples match [71, 14].

Step 3: Cluster together similar duplicate pairs. Multiple tuples can correspond to one entity and such clusters further need to be identified [2, 12, 129, 122, 130]. One common method is to compute the transitive closure among the duplicate pairs. This dissertation does not apply clustering to pairs, but it could be augmented to do so to help speed up the deduplication effort.

Step 4: Data fusion. Once duplicates have been identified, the next step is to combine or merge them and thus produce a single, possibly more complete representation of that real-world object. During this step, possible data conflicts among the multiple representations must somehow be resolved. This step is called data fusion and is not covered in this dissertation. Instead, we refer the reader to a recent survey on data fusion [16].

Blocking. To make the above steps more compute-efficient by reducing the number of record comparisons, blocking techniques are used [13]. Blocking is an inexpensive heuristic filtering step that either partitions the tuples that get compared or filters/eliminates pairs with low similarity scores. The idea behind blocking by partitioning is to reduce the number of comparison operations by restricting the matching comparisons to occur only between the tuples in their given partition. Tuples are placed in partitions based on having similar data

Table 3.2: Example table of feature vectors computed for R x R. String equality is applied to the single-word attributes, `First Name` and `Last Name`. The jaccard similarity measure is applied to `Address` and `Phone`, which each contain a set of strings.

	StrEq(`First Name`)	StrEq(`Last Name`)	Jaccard(`Address`)	Jaccard(`Phone`)
r_1,r_1	1.0	1.0	1.0	1.0
r_1,r_2	1.0	1.0	1.0	1.0
r_1,r_3	0.0	1.0	0.5	1.0
r_2,r_1	1.0	1.0	1.0	1.0
r_2,r_2	1.0	1.0	1.0	1.0
r_2,r_3	0.0	1.0	0.5	1.0
r_3,r_1	0.0	1.0	0.5	1.0
r_3,r_2	0.0	1.0	0.5	1.0
r_3,r_3	1.0	1.0	1.0	1.0

values on a particular attribute, so only the comparisons are applied to tuples that share a common attribute. One example of this partitioning method is called *locality sensitive hashing* [108]. A recent survey comparing all blocking methods can be found in [108].

Supervised learning. A simple approach to composing a matching function is to compute the similarity between individual attributes for a pair of records and then compute a weighted aggregate similarity between the records [33]. However, determining a good weighting strategy requires an expert to build and tune. Thus, the current state-of-art uses a learning based approach [23, 45, 52, 72, 73, 93, 116] to automatically generate an appropriate matching function. These systems typically learn and return a classifier rather than a function and threshold. To learn a classifier, these methods require a set of training example pairs. A good set of training example pairs is difficult to generate, which is a drawback of this approach.

Unsupervised learning. Supervised learners build a matching function from a labeled training set. The unsupervised learners, in contrast, expect this function (created by a domain expert) as input to identify all the records that correspond to the same entity. This line of work typically focuses on the cases where more than two records can correspond to the same entity and the goal is to identify such clusters of matching records. There are two primary bodies of work for unsupervised classification (in chronological order): probabilistic matching [38, 140] and clustering [37].

The earliest work on deduplication, probabilistic matching, uses a Bayesian approach to classify records [38]. This approach assumes that it is given a feature vector x for each pair and a matching function as input and assigns each pair to a class (match or non match). The matching function is based simply on probabilities for a given pair (α,β): if the probability of the match class, given x, is greater than the probability of the non-match class, then (α,β) is classified as a match. However, in order to build an accurate classifier this approach needs to first compute these probabilities as probability distributions of each of the match classes for the given dataset. To estimate these probability distributions, later work [140] applies the general expectation maximization algorithm over a sample of the data. However, this classification technique was shown to only work well in certain conditions, for example, when the dataset is separable and has a large percentage of matches (*i.e.,* greater than 5%) [139].

Since it is hard to construct accurate matching functions, the most recent work on clustering [2, 12, 14, 50, 71, 122, 129, 130, 131, 134] assumes it is given a matching function as input. The approach then is to use this function to identify all of the clusters of records that correspond to the same entity, with the idea that similar records will be near each other. A standard approach to generating the clusters is to compute the transitive closure on record pairs (taken from the cross-product S) that have a similarity above a pre-specified global threshold, t [129]. The hard part is that it can be computationally intensive to apply the matching function to all the pairs. [136] addresses that problem with the goal to produce cleaner datasets incrementally by first applying the matching function to the pairs of tuples more likely to be a match, *i.e.,* with similarities over a threshold t. However, since all clustering techniques from related work require this matching function as input, it has limited applicability to data enthusiasts and visual analytics systems.

3.4.3 Learning a Classifier for Deduplication

Supervised learning: A supervised learning algorithm is a machine learning task that builds a model from a sample of labeled examples. In the context of deduplication, these examples are pairs that are sampled from the cross-product, $S = R$ x R, of the input relation, R, that we seek to clean. The algorithm produces a classifier, which is then used for mapping new examples to either of the two output classes (match or not). Prior work on supervised techniques has considered a variety of classifiers such as Support Vector Machines (SVMs) [116, 93], decision trees [45], and naive Bayes [38]. The goal of a supervised learning algorithm is to correctly assign the class labels for unseen instances by applying the learned function. There are two main approaches to create such a function: passive and active learning, which is discussed next.

Passive learning. The basic learning algorithm selects a random sample of pairs from S, asks the user to label them as either duplicates or not, and then learns a classifier using that training data. Passive learners for deduplication can be found in the databases literature [23, 73] and in the information retrieval literature [52, 72]. One problem faced by passive learners for deduplication is in obtaining enough or *any* examples of duplicates in the samples due to their relative imbalance to non-duplicates in S. The state-of-the-art techniques [23, 73] rely on blocking to reduce the relative representation of non-duplicates in $|S|$ before sampling. Even after blocking, however, the fraction of duplicate examples in a random sample can remain small, leading to a poor classifier.

Active learning. Active learning improves on the above approach by iteratively training classifiers on increasingly large and carefully selected training examples. As above, the initial step is to learn a classifier on a random sample of training examples. Active learning then selects additional training examples with the purpose of improving the classifier's quality. Several methods exist to select the additional examples that are most *informative.* Current methods to assess the informativeness of training examples measure the disagreement of the component classifiers in its prediction of labels. A component classifier is a classifier that is trained on a random sample (with replacement) of the original training dataset. There will be multiple component classifiers created. The intuition is that the more the component classifiers disagree on the label for an example pair in the unlabeled set, the more likely that the classifier will learn something new from this example and thus (on the re-training step) produce an improved classifier. One method applies the bootstrap [36] to estimate the classifier's uncertainty [93] in its predictions of labels. The uncertainty is defined as the variance of the classifier in its predictions of labels for examples in the testset/unlabeledset. The bootstrap estimates the overall uncertainty of a classifier by considering the overall agreement among the component classifiers on predicting labels for pairs in the unlabeled set. Another method measures the disagreement among an ensemble of decision trees (called a *random forest* [17]) to estimate the classifier's entropy [105] in its label predictions in the unlabeled set [45]. The higher the entropy, the stronger the disagreement, and the more informative the example pair is. Whatever the method, active learning then retrains a new classifier and repeats the process. Learning stops when the classifiers stop improving across iterations.

While existing methods produce high-quality classifiers, one key problem is that it often comes at a cost to the user: requiring hundreds to thousands [45, 93] of labels to clean entire datasets that contain hundreds to thousands of rows. This time-consuming labeling effort imposed by existing approaches conflicts with the interactive, real-time constraints exploratory visual analytics systems impose to support sensemaking. The focus of the deduplication techniques in this thesis, in contrast, is to reduce the user's labeling effort. In Chapter 5, we develop a new method that addresses this problem. It is most similar to the aforementioned state-of-the-art active learning systems with the key differences of 1) only asking the user to label data that is currently being visualized, 2) selecting examples to learn from that actually impact the visualization/view, and 3) stopping the cleaning process when it detects that additional labels will not change the visualization further even if they could yield a better overall classifier. This new method builds on the state-of-the-art active learning algorithms [45, 93]. We describe these algorithms along with our modifications to them in greater detail in Chapter 5.

Chapter 4

MEASUREMENT STUDY OF VISUAL ANALYTICS SYSTEMS

Visual data analytics systems have been available for several years now [113, 63, 46]. Such systems let people upload data, create visualizations, and share them. The most prominent, Many Eyes, started in early 2007, amid considerable attention from both the academic community and the media. There is little information, however, on how and how much these systems are actually used. Early studies of Many Eyes [27] indicated a significant uptake, as well as collaboration between users; but there have been no follow-up studies on usage, nor have there been comparable studies of other web-based or web-centric visualization systems.

To answer these and other questions, we study Many Eyes and Tableau Public [90]. Many Eyes is the oldest system (now as part of IBM Watson Analytics [63]), and is also the only one that has been studied by the visualization community so far. Tableau Public allows users to download and use the Tableau Desktop application for free (with some limitations, see Section 4.1.3). While Tableau provides more powerful features than Many Eyes, it also presents a much more complex user interface and requires more experimentation and learning. Furthermore, the system architecture (see Figure 2.1), integrates a visualization front-end with a database management system (DBMS) back-end [132, 46]. Analytical workload queries are federated across a variety of heterogeneous data sources (*e.g.,* files, cubes, data marts, and databases) to obtain the necessary data to render each visualization. Please see Chapters 2 and 3 for details about Many Eyes and Tableau.

In spite of their growing popularity, little is known about how these systems are being used. In most cases, even basic statistics such as the number of users are not published (*e.g.,* Fusion Tables), let alone any details of user activity. As our society continues to become “data-enabled”, it is important that we continue to improve data management and analysis tools. If we are to build better online, data visualization and sharing systems, the first step

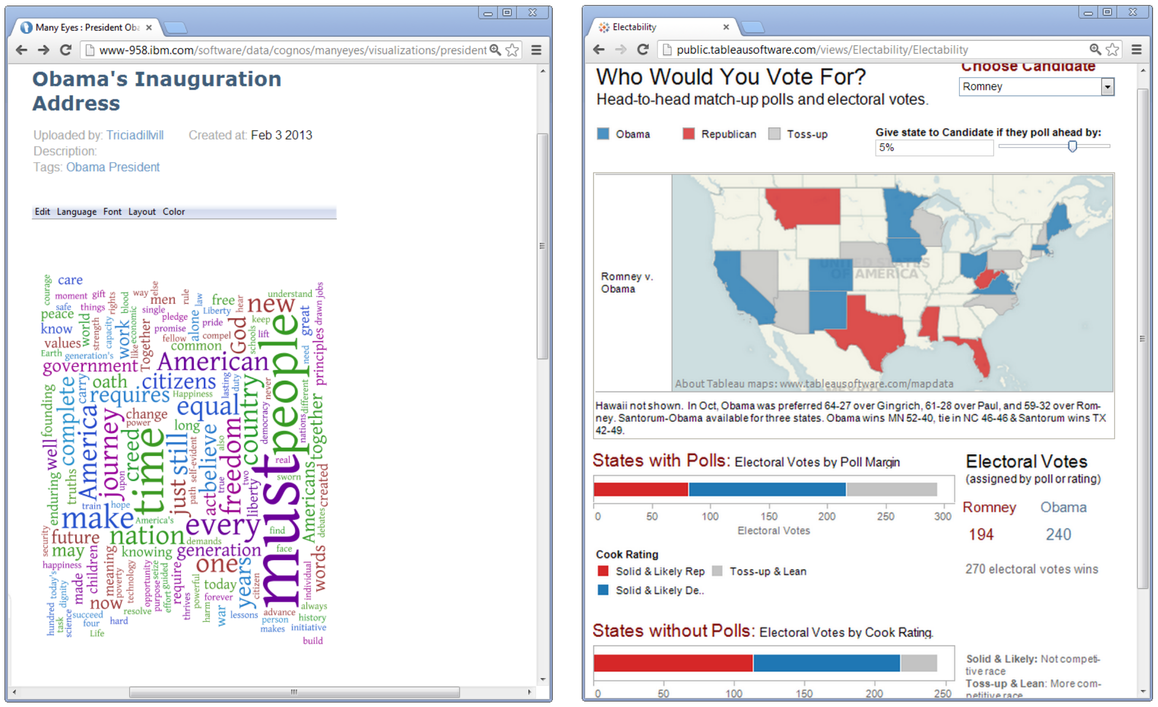

Many Eyes : President Ob
www-958.ibm.com/software/data/cognos/manyeyes/visualizations/president
Obama's Inauguration Address
Uploaded by: Triciadilvill
Created at: Feb 3 2013
Description:
Tags: Obama President
Edit Language Font Layout Color
must people time every American nation make America journey citizens requires equal freedom country
Electability
public.tableausoftware.com/views/Electability/Electability
Who Would You Vote For?
Head-to-head match-up polls and electoral votes.
Choose Candidate
Romney
Obama
Republican
Toss-up
Give state to Candidate if they poll ahead by:
5%
Romney v. Obama
About Tableau maps: www.tableausoftware.com/mapdata
Hawaii not shown. In Oct, Obama was preferred 64-27 over Gingrich, 61-28 over Paul, and 59-32 over Romney. Santorum-Obama available for three states. Obama wins MN 52-40, tie in NC 46-46 & Santorum wins TX 42-49.
States with Polls: Electoral Votes by Poll Margin
Electoral Votes
Electoral Votes
(assigned by poll or rating)
Romney
Obama
194
240
270 electoral votes wins
Cook Rating
Solid & Likely Rep
Toss-up & Lean
Solid & Likely De..
States without Polls: Electoral Votes by Cook Rating
Solid & Likely: Not competitive race
Toss-up & Lean: More competitive race

Figure 4.1: A common visualization type on Many Eyes is the word cloud (left); complex multi-view dashboards are popular on Tableau Public (right).

is to understand how they are being used today. The key contribution of this chapter is to shed light on this exact question: *How are online data visualization and sharing systems being used?*

Research questions. We tackle the question of how both of these systems are being used from the perspective of the database community. Our study is focused on three topics: users, visualizations and the data sets they are based on, and advanced features.

We lack demographic and other information about users, but we can analyze their behavior as far as publishing visualizations and data sets are concerned. In particular, we ask the following questions (Section 4.2.1):

- *How is the number of authors (a user who creates a data visualization and, optionally, shares it with others) growing over time?*

- *How productive are users or authors?*
- *How well do both systems retain authors or users more generally?*
- *How do data sizes and query workloads impact the user experience?*

Next, since these two systems were designed for sharing and collaboration, we assess the degree of interaction with the online published content and how often users collaborate over a common dataset or visualization. In Section 4.2.2 we answer the following questions:

- *How do users interact with published content?*
- *Do authors collaborate over a common dataset or visualization to create new content?*

Turning to the core, visualizations and data sets, we ask a number of questions about the data people use and the visualizations they create (Section 4.2.3):

- *How many rows of data do people work with?*
- *What dimensionality does the data have?*
- *How many of the data dimensions are used in visualizations?*
- *What types of data do people want to visualize?*
- *What visualization techniques do they use?*

Finally, we consider advanced features that include multiple data sets and/or multiple views, either as small multiples or as multi-view dashboards (Sections 4.2.4 and 4.2.5):

- *Do users analyze multiple data sets when they can?*
- *How do they join/blend data sets?*
- *Do users create multiple views when they can?*
- *Can users construct interaction links between multiple views?*

Background and method. The data was collected from each system, Many Eyes and Tableau Public, from their inception up to December 31, 2012 (Table 4.1). For Many Eyes, the data thus spans 24 quarters or six years: Q1/2007 through Q4/2012. For Tableau Public, the collected data spans 12 quarters (three years): Q1/2010 through Q4/2012.

System	Start Date	# Visualizations	# Workbooks	# Datasets	Users
Many Eyes	January 1, 2007	149,395 (3.2/user)	n/a	358,880 (7.8/user)	46,048
Tableau Public	February 10, 2010	269,609 (11/user)	73,404 (3/user)	107,596 (4.4/user)	24,563

Table 4.1: Summary of the collected data from Many Eyes and Tableau Public, from each system's inception until December 31, 2012.

The data contains 46,048 Many Eyes user accounts and 24,563 accounts from Tableau Public (only counting users who have published at least one visualization or data set).

For Tableau Public, each workbook specifies the data sources analyzed (including all of the schema metadata), the types of visualizations produced, and all of the specific VizQL definitions [109] that produce each visualization. For Many Eyes, data was collected on the visualization types used as well as a heuristics-based classification of data into data types.

4.1 Related work

In this section, we present a high-level overview of the two visual analytics systems that we study in this section, Many Eyes and Tableau. For more details on these two systems please see Chapters 2 and 3.

4.1.1 Visualization for the masses

In the wake of highly successful "web 2.0" websites like YouTube, the idea of socially-driven visualization websites [75] spawned a number of experiments, both from large, established enterprises (like IBM's Many Eyes) and small start-ups (like the now-defunct Swivel). Similar to YouTube, the goal was to enable anybody to create and publish visualizations, embed them in blogs, comment, and collaborate by sharing data.

Collaboration was a driving force behind Many Eyes and also similar systems like the short-lived experiment sense.us [58]. Later work extended the idea of commenting to more structured collaboration for sense-making [138].

Crowd-sourcing and citizen science experiments have a longer history in the sciences, such as with the Pathfinder experiment [81]. Both sites studied in this chapter rely on users organically creating content, though this has been called into question recently; a more goal-oriented approach creates more and higher-quality responses [137].

4.1.2 Many Eyes

Many Eyes [83, 125] is a Web-based visualization service that allows users to upload datasets and create visualizations. Unlike Tableau Public, all visualizations are created and published directly through a Web browser. The site was launched in early 2007 by IBM's *Visual Communication Lab* as the first online service that provided ways to not only create static charts, but interactive visualizations that could easily be embedded in blogs and other websites. While both systems share many of the same view types (*i.e.,* bar, line, text, pie, area, scatter, and maps), Many Eyes includes a number of unique techniques that are not available in any other software. In particular, Many Eyes' text views – including word clouds [123], phrase nets [121], and word trees – let users experiment with text data in ways that are still unmatched in most other visualization tools and services.

In addition to the visualization tools themselves, Many Eyes also pioneered the notion of social visualization. The typical Web 2.0 feature of leaving a comment has the added twist that it also contains a live thumbnail reflecting the configuration of the visualization the user was looking at when writing the comment. Users can also browse existing data sets and visualizations and create new ones from what others have uploaded. Many users can thus benefit from the work of somebody scraping or otherwise collecting data.

Other than the original papers by the people behind Many Eyes [27], we are aware of only one other study that looked into visualization activity on the site. *The Guardian* published an informal analysis of Many Eyes in April 2012 [114]. They studied the provenance of the data sources, and reported that the US Census Bureau was one of the most widely used sources. They also presented the most common topic tags for visualizations, most active users, and the number of data sets uploaded per user.

4.1.3 Tableau Public

Tableau Public [112] is a Web-based visualization platform that launched in February of 2010 (Figure 4.1(right)). In contrast to Many Eyes, visualization views are created in (currently Windows-only) Desktop client and then published to the web. Tableau Public is a variation of the commercial Tableau Desktop, with the following restrictions: visualizations are limited to 100,000 rows of data, accounts are limited to 50 MB of storage, and content can only be saved by publishing to the Web-facing Tableau Public servers.

Similar to Many Eyes, all content published on Tableau Public can be downloaded by anybody, including the data and the workbook containing all visualization definitions. Visualizations can also be embedded on other websites or shared through social media or email. Tableau Public, however, was designed to have a low "author to consumer" ratio whereas Many Eyes focused more on collaboration and conversation between author and viewer. As of late February 2013, Tableau Public visualizations have been viewed over 100 million times.

Tableau Public allows for more flexibility in the creation of visualization, though it lacks some of the visualization types that Many Eyes has (in particular ones for text visualization). Interaction is generally richer, with control over mouse-over tooltips, selection, etc. It is also possible to build multiple-view *dashboards* that can have actions between the views to filter or highlight data based on user interaction.

4.1.4 Terminology

Many Eyes and Tableau Public differ in their terminology and the way data and visualizations are organized. Many Eyes treats data sets and visualizations as independent units: users can publish data without creating visualizations, and create visualizations from existing data already on the site (their own or others') without the need to upload first. Tableau Public, on the other hand, packages data and visualization definitions into *workbooks*. Workbooks typically contain multiple *worksheets* that each contain one type of visualization. Worksheets can be combined into multi-view *dashboards* that can also include interaction (highlighting, filtering) between the individual views.

Workbooks can include multiple data sets, and individual visualizations can be created with a single data source or by joining multiple data sources together; the latter is referred to as *blending* [92] and is described in detail in Chapter 2. To review, the data blending feature combines data on the fly from multiple heterogeneous sources without having write a query or specify a data schema with dependencies. A user authors a visualization by selecting the columns from an initial (*primary*) data source which establishes the context for subsequent blending operations in that visualization. Data blending happens when the user drags in fields from a different data source, known as a *secondary* data source. Additionally, the visualization can be further modified by, for example, adding more secondary data sources or drilling down to finer-grained details.

4.2 Analysis results

We present the key results of our analysis, organized around our five core questions about overall workload and author behavior, collaborations, single-dataset analytics, multi-dataset analytics, and multi-view visualizations.

4.2.1 Author behavior

In the following, authors are defined as users who have published at least one data set or visualization. Due to the nature of the data collection, users who never publish anything are not included.

New authors. To better understand the long-term behavior of authors on these systems, in this section we first answer the question, *what is the growth of new authors until the end of 2012?* Since its inception in January 2007, Many Eyes, has grown to over 46,000 authors who have published over 358,000 data sources and more than 149,000 visualizations. For Tableau Public, its user-base includes 24,500 authors who have contributed over 73,000 workbooks, 107,500 datasets, and 269,000 visualizations (Table 4.1).

Figure 4.2 shows how the systems are growing over time in terms of the number of opened accounts. As the figure shows, since its inception in January 2007, Many Eyes, has grown to over 46,000 authors who have published over 358,000 data sources and more than 149,000 visualizations. For Tableau Public, its user-base includes 24,500 authors who have

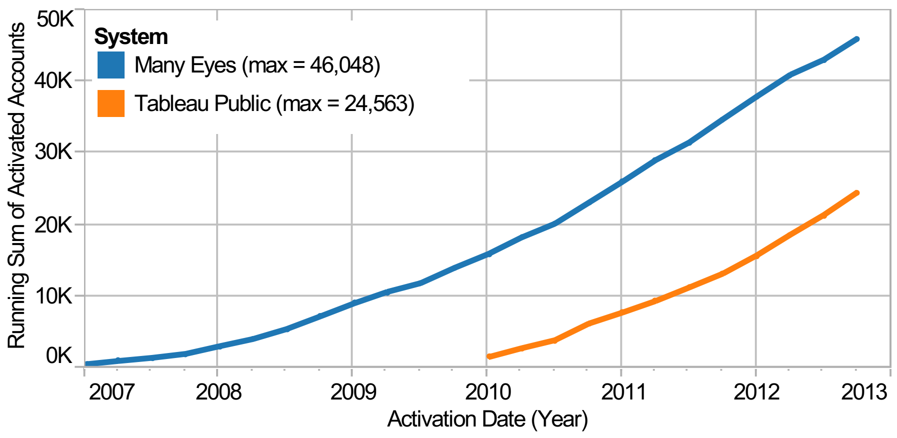


Figure 4.2: Cumulative growth of Many Eyes and Tableau Public activated user accounts. This graph shows the running count of active user accounts since the inception of Many Eyes (Q1, 2007) and Tableau Public (Q1, 2010). By the end of Q4, 2012 (total time span of 6 years for Many Eyes and 3 years for Tableau Public), Many Eyes had 46,048 user accounts and Tableau Public had 24,563.

contributed over 73,000 workbooks, 107,500 datasets, and 269,000 visualizations (Table 4.1). We define authors to be users who have published at least one data set or visualization. **These systems thus have moderate numbers of users today, but their popularity is continuing to grow significantly each year.**

Author productivity. How productive are authors in publishing content? Two types of content can be published to Many Eyes and Tableau Public: data sources and visualizations. Figure 4.3 shows the Probability Distribution Functions (PDFs) of the number of published data sources (top) and visualizations (bottom) per author on Many Eyes and Tableau Public. The publication trends for authors on both systems are quite similar:

Overall these statistics reveal that most authors on Many Eyes and Tableau Public publish only a few data sources and visualizations. As Table 4.2 shows, **half the users (or almost half) are *one-time* users who publish only one dataset or visualization. The remaining users are mostly *light* users who publish two to four visualizations. Only 10% to 17% are *prolific* users who publish five or more data sets**

	Number of Data Sources Published				
	1	≤2	≤3	≤4	≤5
Many Eyes	44%	65%	76%	83%	86%
Tabeau Public	45%	63%	73%	79%	83%

	Number of Visualizations Published				
	1	≤2	≤3	≤4	≤5
Many Eyes	52%	72%	82%	87%	90%
Tabeau Public	53%	71%	80%	85%	88%

Table 4.2: Cumulative fraction of users who publish up to a given number of data sources or visualizations (e.g., 80% of Tableau users publish 3 visualizations or less).

or visualizations. Furthermore, the long tail starting around 14 data sources shows that while these most prolific authors are the minority (with 2–5% representation), their contributions are quite varied (ranging up to 555 data sources for Many Eyes and 715 for Tableau Public). Similarly, for visualizations published, there is a long tail of the remaining top 10% contributing authors. The number of workbooks (visualizations for Many Eyes) published for these authors range from six to 553 for Tableau Public and five to 13,284 for Many Eyes.

The latter number is an outlier caused by the fact that Many Eyes allows users to create visualizations without logging in, so the most prolific user on that system is *Anonymous*. The second-most productive user has only created just over 1,000 visualizations, with the number rapidly decreasing in a typical power-law distribution from there. Among the top 20 users, we find only one member of the Many Eyes team. Tableau Public does not have the notion of anonymous users and thus also does not have a clear outlier like Many Eyes. It does have more active participation from its own employees though, with four of the top 20 contributors being Tableau employees.

Given these results, we categorize Many Eyes and Tableau Public authors into three main groups based on their publication activity: 1) *one-time* users publish one dataset or

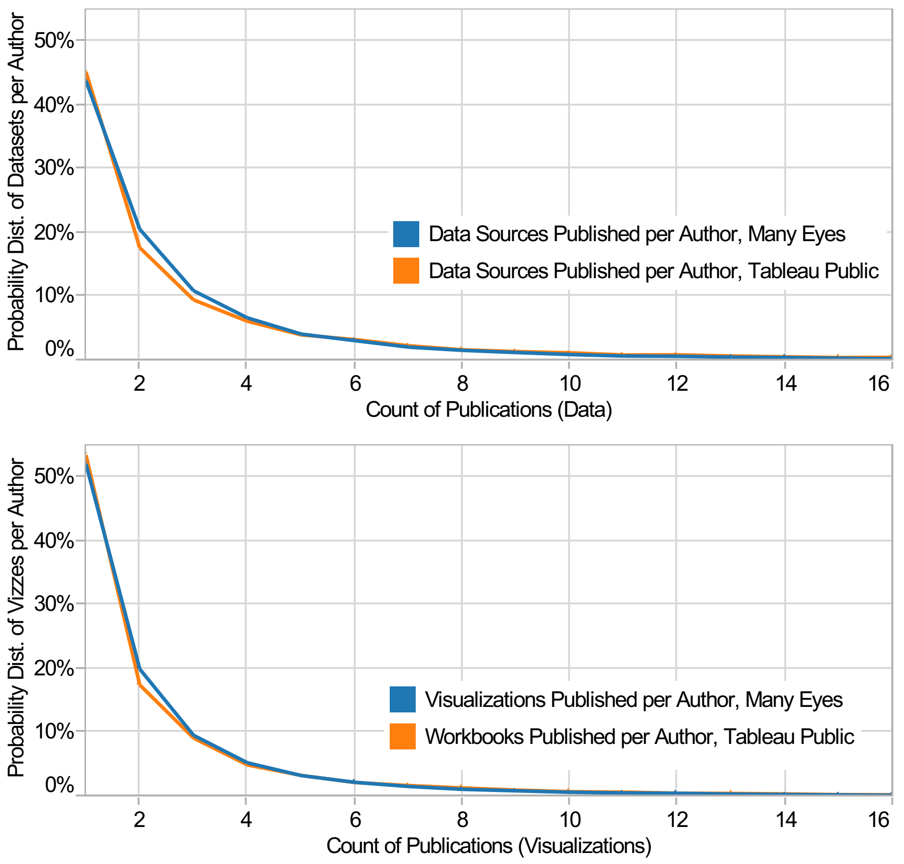


Figure 4.3: PDFs of Many Eyes and Tableau Public per author publications of data (top) and visualizations (bottom). Most authors on both systems have one or a small number of data sources and content, while a small number are prolific. The x-axis was cropped for readability, there are authors with hundreds of visualizations and data sets.

visualization; 2) *light* users publish two to four; 3) *prolific* users publish five or more data sets or visualizations (Figure 4.4). The user-base on Many Eyes and Tableau Public is dominated by one-time and light users.

Author retention and churn. How well do both systems retain authors? Figure 4.5 presents the trends of author retention and churn. We group users into cohorts based on the quarter in which they published their first visualization (workbook on Tableau Public)

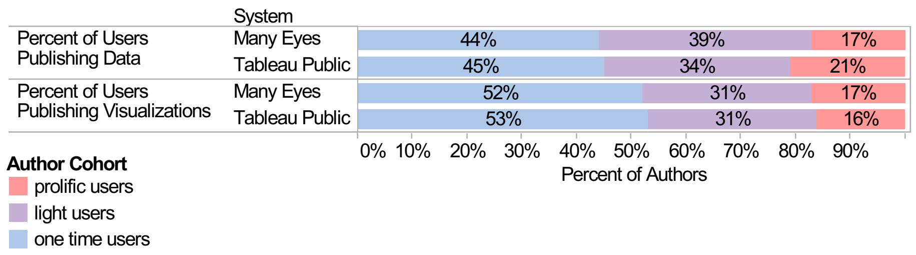


Figure 4.4: Percent of authors who publish data and visualizations on Many Eyes and Tableau Public (grouped by author cohort). Authors are divided into three main cohorts: *one-time* users publish one dataset or visualization *light* users publish two to four; *prolific* users publish five or more data sets or visualizations.

and track their publications. In comparing the activities of authors on these two systems, we trace each cohort of authors for their first 12 quarters (three years).

Figure 4.5(b) shows that by the end of the 12th quarter, Many Eyes has 2,100 actively publishing authors and Tableau Public has over 4,000 active accounts. Additionally, over 6,800 new visualizations are published on Many Eyes in its 12th quarter, (12,000 on Tableau Public), as shown in Figure 4.5(a). Even though the total number of users and visualizations is greater in Tableau Public than Many Eyes, both systems show a strikingly similar pattern in terms of workload distribution between new and returning users in Figure 4.5(c). In the graph all author accounts are new in the first quarter and thus not returning users. This corresponds to the authors who joined Many Eyes and Tableau Public in the first quarter that the systems were deployed on the Web (*i.e.,* the 2007 Q1 cohort on Many Eyes and 2010 Q1 on Tableau Public). Among all of the authors who published in Q2, only 12% of these authors on Many Eyes and 24% on Tableau Public had come from Q1 and had returned to publish. In other words, by Q2 88% of the authors on Many Eyes were new and 76% of the authors on Tableau Public were new. For Many Eyes, we see that by the last (12th) quarter, only 15% are returning authors; for Tableau Public, however, 37% of all authors are returning authors. Furthermore, these returning authors contributed 25% of the published visualizations that quarter on Many Eyes (51% for Tableau Public). Overall, both systems

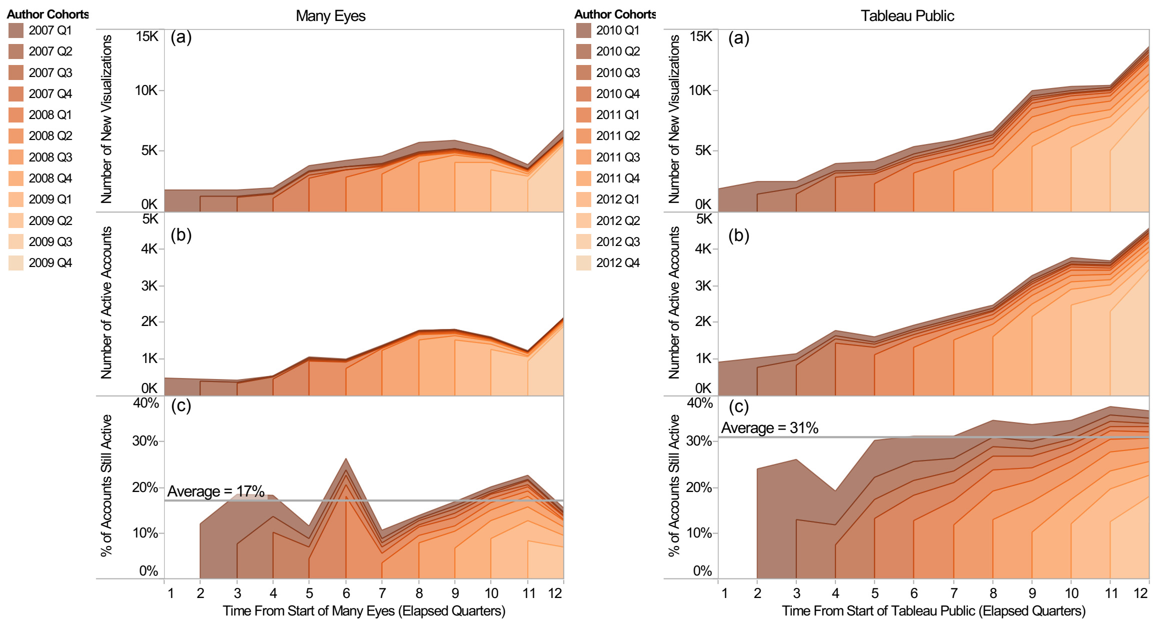


Figure 4.5: Many Eyes (left) and Tableau Public (right) author cohorts for the first 12 quarters (3 years). Authors are grouped into cohorts based on the quarter in which they published their first visualization or workbook. Every cohort contains a set of prolific authors, which suggests that (a) overall usage will grow over time as (b) more people use these systems. Tableau Public also has a high rate of prolific authors and a higher rate of author retention than Many Eyes, as shown in (c), which suggests that users value the richer visualizations.

exhibit the trend of high author turn-over. Looking at the percent of actively publishing accounts by returning authors for each quarter in Figure 4.5(c), Many Eyes averages 17% and Tableau Public averages 31% (not taking the first quarter into account).

To better understand this trend of author retention, Figure 4.6 shows the publication activity of each author cohort group over time (*i.e.,* percentage of workbooks contributed by each cohort as that cohort ages). We compute the per quarter publication activity of each author cohort group as the ratio of the number of workbooks published by a single cohort in that quarter divided by the total number of workbooks published that quarter. After the first quarter of activity, we see a significant drop off in the next quarter; no single cohort contributed more than 27% of the workbooks on Tableau Public. Many Eyes has a similar

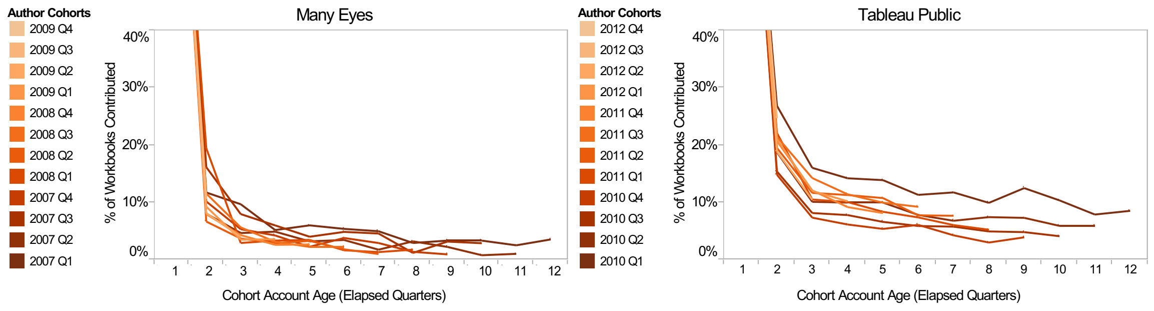


Figure 4.6: Publication activity over time: % of workbooks contributed by author cohorts on Many Eyes (left) and Tableau Public (right). In their second quarter of activity, no cohort contributed more than 19% of the workbooks on Many Eyes. Similarly, on Tableau Public less than 27% of workbooks came from returning authors in their second quarter of activity. Over time, each cohort group publishes fewer workbooks on each system. However, on Tableau Public the cohorts are more productive overall in publishing visualizations.

publication activity trend: in their second quarter of activity, no cohort contributed more than 19% of the workbooks. As each cohort group ages on each system, we see that they publish fewer workbooks overall. However, the author cohorts on Tableau Public exhibit a higher overall publishing activity as the accounts age.

Low retention after initial use is common for free, Web-based services. According to a 2009 Nielsen report [84] only 40% of Twitter users returned to use the site after the first month. However, other websites like MySpace and Facebook achieved retention rates closer to 60%. This result was measured for these other three systems at the same point in their respective user growth curves.

Account size limit. According to Figure 4.7 (top), we see that 90% of user accounts use less than half of their 50MB quotas. Since each account contains datasets and workbooks consisting of a collection of visualizations, we further study the sizes of workbooks published on the site (see bottom of Figure 4.7) to see if the sizes of the visualizations are a limiting factor. The figure shows that 90% of all workbooks are less than 762KB in size – which means that most authors can publish multiple workbooks to their accounts and still be well under the 50MB quota.

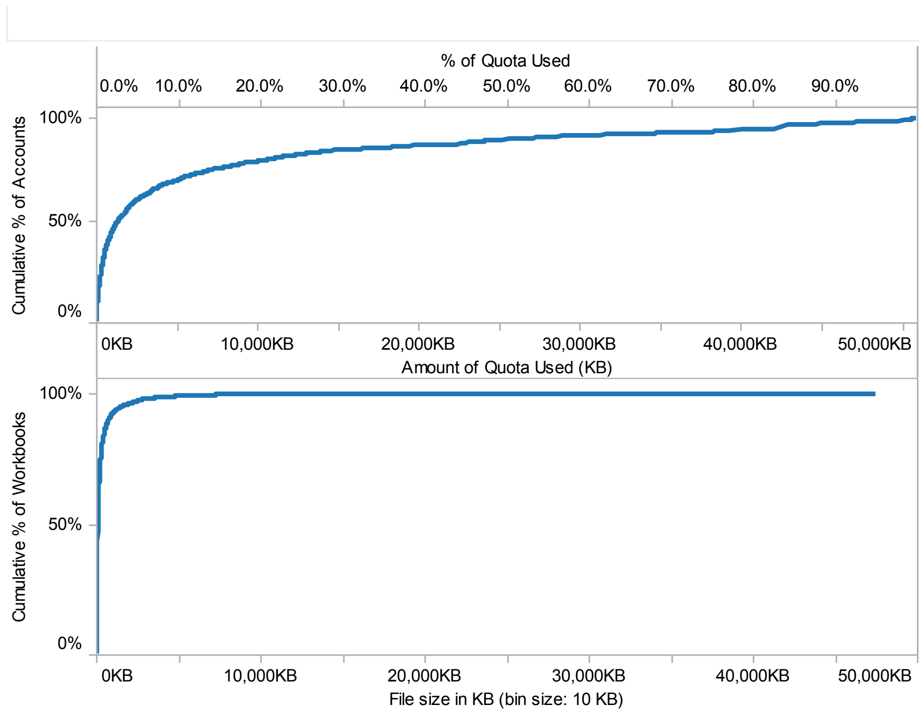


Figure 4.7: Cumulative Distribution Function (CDF) of account quota usage (top) and workbook file size (bottom) on Tableau Public

Query load. According to a study published on Web users' tolerable waiting time [94], 2 seconds is considered an acceptable waiting time for loading Web pages. In Figure 4.8 we see that 84% of all visualizations on Tableau Public take less than 2 seconds to load (includes both query and rendering time) and 98% are under 10 seconds (the accepted limit for keeping a user's attention focused on a given task [88]). Although attitudes and expectations change over time, the basic capability of human attention has not changed over the decades [20, 88]. Thus, our results indicate that the majority of load times should not negatively impact Tableau Public's users.

Discussion. Overall, the results thus show a continued growth in users but a low retention rate of these users. The overwhelming majority of users are either "one-time users" or

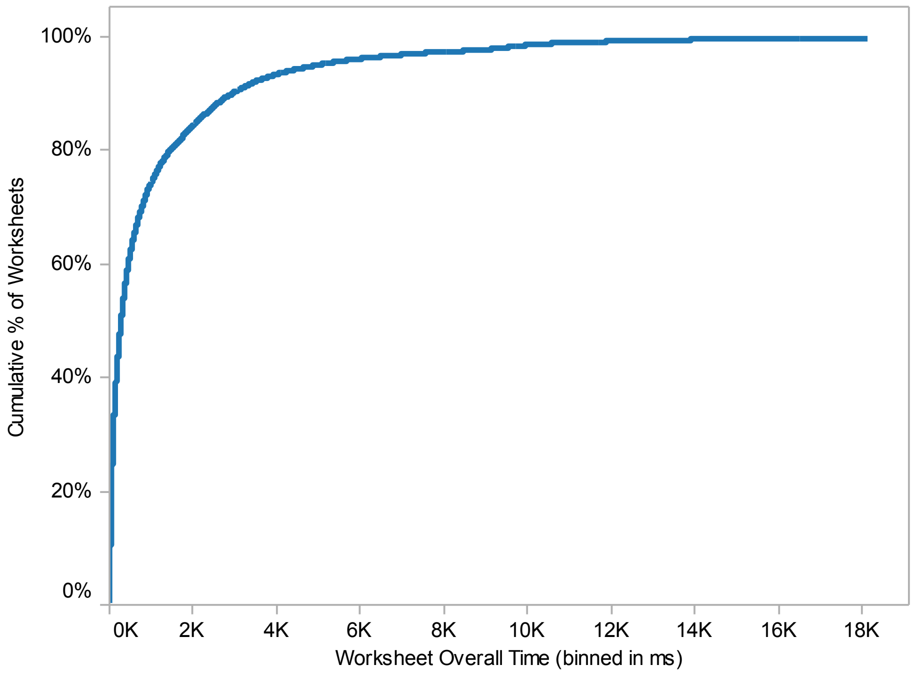


Figure 4.8: CDF of worksheet load times on Tableau Public

"light" users. At the same time, users do not appear to be hindered by constraints on the size of their accounts or the size of their visualizations. Similarly, query performance is below well-known thresholds for user attention. A few direct implications of these results are that (1) online visual analytics systems today have a user-base primarily comprised of users with little to no experience. At the same time, (2) while attracting new users to these systems is not a problem (Figure 4.2), retaining them beyond their first visualization appears to be a critical challenge. While users are not limited by the performance of these systems nor data sizes, there must be other more fundamental causes (perhaps relating to usability or the fact that users tend to not be regular visualization creators) that lead users to abandon the site. Finally, these systems focus strictly on small-data users. It would be interesting to see if the above trends would change if the systems had support for big-data users.

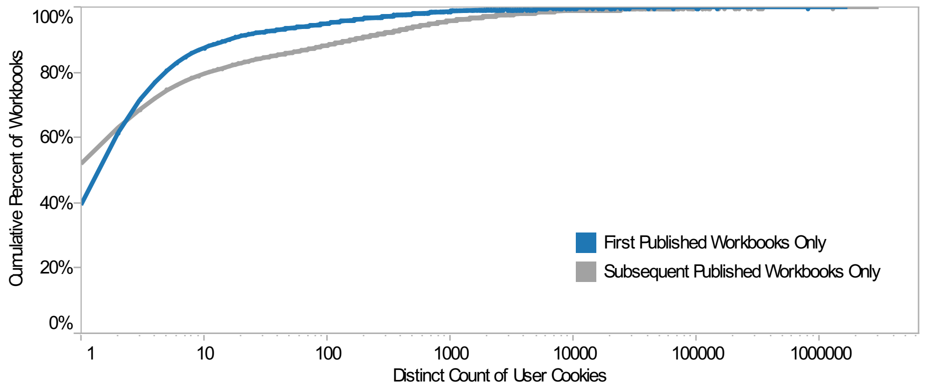


Figure 4.9: CDF of workbook popularity on Tableau Public (max # user cookies = 52.0 Million)

4.2.2 User interaction and collaboration

Since both systems are designed for sharing visualizations and collaboratively analyzing data, we explore the frequency of viewership, collaboration, and sharing in this section.

User interaction with published content. Based on a distinct count of user cookies, we found that there are around 52.0 million unique visitors to Tableau Public. The visitors are thus *several orders of magnitude* more numerous than the authors (only 24,500 authors). Additionally, we found that the top 50% of all Tableau Public traffic is attributed to 244 distinct workbooks (or 0.3% of all workbooks). For the results presented in this subsection, we did not have access to the equivalent traffic and viewership information for Many Eyes.

Figure 4.9 shows the distribution of workbooks by their viewing popularity. In this graph, we split the workbooks into two groups to compare their relative popularity distributions: those that were an author's first publication and those that were a later publication. Since first-time publications make up a sizable fraction of the overall total number of publications (29%), we observe the viewership trends of this group of workbooks in comparison to the trends of subsequent published workbooks. In this figure, we see that 42% of workbooks that were an author's first publication on Tableau Public are only viewed by a single user.

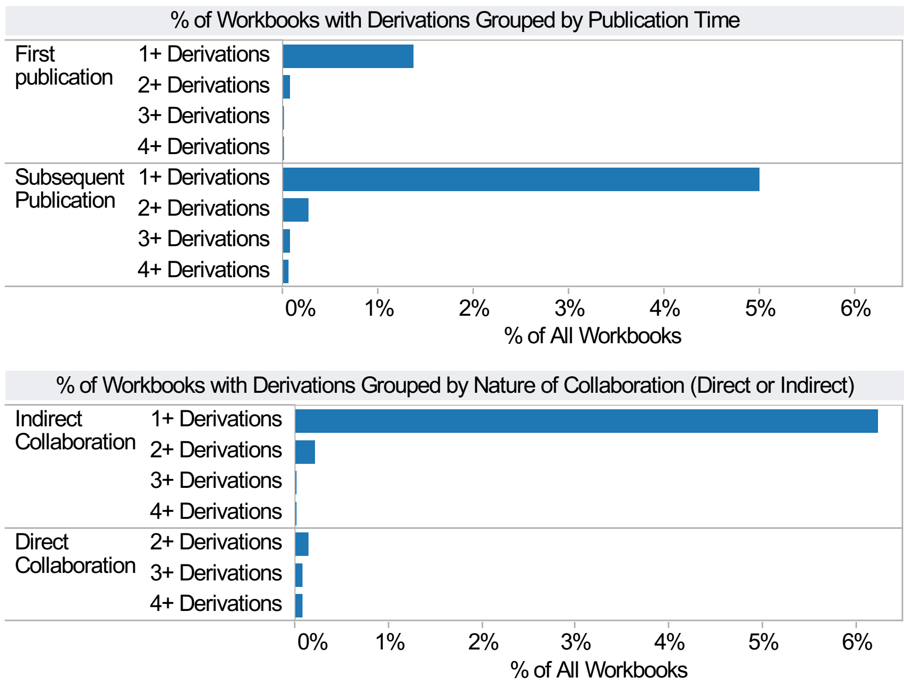


Figure 4.10: Derived Tableau Public workbooks partitioned by publication time (top) and nature of collaboration (bottom)

Interestingly, we see that, likewise, 53% of subsequently published workbooks are viewed by one user. As expected, the curve for the most popular workbooks that were an author's first publication is sharper than workbooks that were not first publications (*i.e.,* popular workbooks tend to not come from first-time authors): At the 90th percentile, we see almost an order of magnitude difference in viewership with only 15 unique users for first-time publications compared to as many as 151 for workbooks that were not the author's first publication. The top 1% of first-time publications received at least 1,500 views, with a maximum viewership of over 1.7 million. In contrast, the top 1% of subsequent publications received at least 10,000 views with a maximum viewership of 3.0 million.

Collaborations and new content published. To get a sense of the degree of collaboration between authors, where multiple authors edit the same visualization, on Tableau

Public, we first explore how often authors take existing content and evolve it for their own analytical needs (*e.g.,* by changing the visualization content to explore some other dimension or measure) and then republish it with their insights. In our approach, we traced the provenance of workbooks that were created by one author and edited and republished by a different author (called a *derivation*). We initially found that a small fraction of workbooks (only 6.4%) contain visualizations that were derived from other workbooks. Since so few workbooks are derived, we tested to see if this was due to the fact that a lot of authors (55%) simply publish a single workbook and never return. Figure 4.10(top) shows the breakdown of workbook derivations grouped by whether or not it was the author's first publication. We see that although a workbook is about five times more likely to be derived if it is not the author's first publication, the probability of derivation remains minuscule.

Finally, Figure 4.10(bottom) shows the extent of the derivations behavior: most derived workbooks are not derived again by other authors, but some workbooks have been extensively evolved by different authors (*e.g.,* 28 workbooks were derived $>$ 4 times). We thus categorize workbooks into two main groups: 1) those that were derived multiple times, but by alternating between the same two authors as in a *Direct Collaboration* and 2) those with one or more derivations, but by a different author each time as in an *Indirect Collaboration.* Interestingly, this figure provides evidence that some authors actually are directly collaborating back and forth with each other and publishing new derived workbooks. This finding on direct collaboration demonstrates that interesting visualizations can spark creative discussions amongst authors. And since there is no feature in Tableau Public that supports such direct collaborations, authors are forced to manually search for visualizations using a search engine and collaborate through some third party website (such as a blog) or through email. We see a need, therefore, for online visual analytics systems to better support collaborative analytics.

Unfortunately, no such equivalent derivation information is available for Many Eyes. However, in order to get a sense of the degree of influence one author's contributions have on other authors, we show in Figure 4.11 how often authors reuse datasets uploaded and shared by others for their visual analysis in Many Eyes. In this figure, we see that only 6% of datasets are used by multiple authors and that 20% of datasets are used in multiple visualizations. Some users are publishing multiple visualizations for a given dataset. We similarly cannot plot Figure 4.11 for Tableau Public because published workbooks make a copy of the data being visualized.

Overall, however, the frequency of reuse of other authors' data on Many Eyes is consistent with the derivation results presented for Tableau Public.

Discussion. The clear conclusion from the above results is that online visual analytics systems are read-heavy today: Orders of magnitude more people are viewers compared to authors. Additionally, as is typically the case for database access patterns, viewership is skewed toward a small fraction of *hot* visualizations. Furthermore, as expected, first-time publications, which account for a large fraction of all publications, are less likely to be shared, derived, or viewed by a large audience than subsequent publications. At the same time, however, some first-time publications can be extremely popular. Also, in general workbooks are not likely to be derived from other workbooks and republished. Hence, true collaboration remains limited between users. Incentivizing and supporting collaborations remain critical challenges for today's online, visual data analytics systems.

Overall, we find that authors tend to bring their own data and do not leverage the contributions of content from other authors. Since one of the main points of these online analysis systems is collaboration, we need to develop tools that can help them find and connect to other good quality data contributed by other users.

4.2.3 Single-dataset analytics

A potential limitation when using online systems are dataset sizes, because both Many Eyes and Tableau Public impose restrictions on the amount of data that can be used. We wanted to find out what sizes of data users work with, and whether they run into the size limits.

Data set size (number of rows). Today's online visual analytics systems are designed for small data. Most of these systems put a bound on the size of datasets that can be processed. On Many Eyes, data sizes are limited to 5MB, while on Tableau Public, each

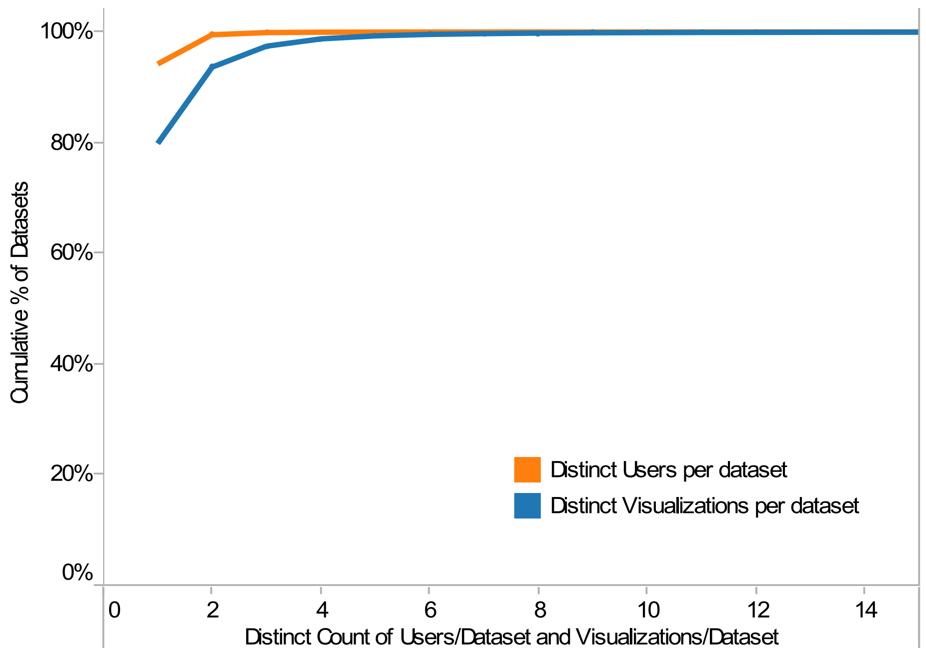


Figure 4.11: Cumulative distribution of Many Eyes users per dataset and Many Eyes visualizations per dataset

	Number of Rows in Visualizations				
System	≤100	≤1K	≤10K	≤50K	≤100K
Many Eyes	63%	90%	98%	99%	100%
Tabeau Public	28%	53%	84%	95%	100%

Table 4.3: Cardinality of visualized relations (*i.e.,* number of rows in visualization/view).

user gets a 50MB account and a visualization/view can have at most 100,000 rows.

Given these restrictions, we see from the Cumulative Distribution Functions (CDFs) in Figure 4.12 that the median number of rows in a visualization is low. We also see that there is a significant shift in the curves for these two systems indicating a greater demand for authors on Tableau Public to create visualizations with larger data.

Tableau Public also offers a paid tier, Tableau Public Premium, which allows a small num-

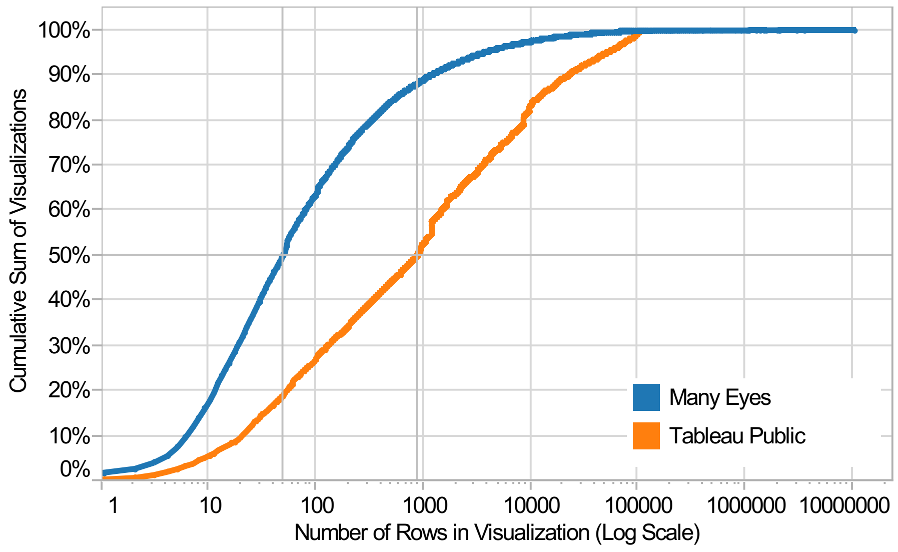


Figure 4.12: CDF of number of rows in visualizations on Many Eyes and Tableau Public. 50% of datasets on Many Eyes are 50 rows or less, while that number is just under 900 for Tableau Public.

	Number of Columns in Data Source				
System	≤2	≤10	≤20	≤100	≤300
Many Eyes	49%	84%	93%	99%	100%
Tabeau Public	2%	28%	52%	90%	99%

Table 4.4: Degree of base relations.

ber of accounts to go beyond the 100,000 rows limit. These accounts (along with some accounts on Many Eyes) visualize more than an order of magnitude more data, which seems to imply the need for the online visualization of bigger data too.

Data set dimensionality. Just like with the number of rows, there is also a large variation in the number of data columns (Figure 4.15 and Table 4.4).

Clearly, data sets are not only larger but also contain many more dimensions on Tableau

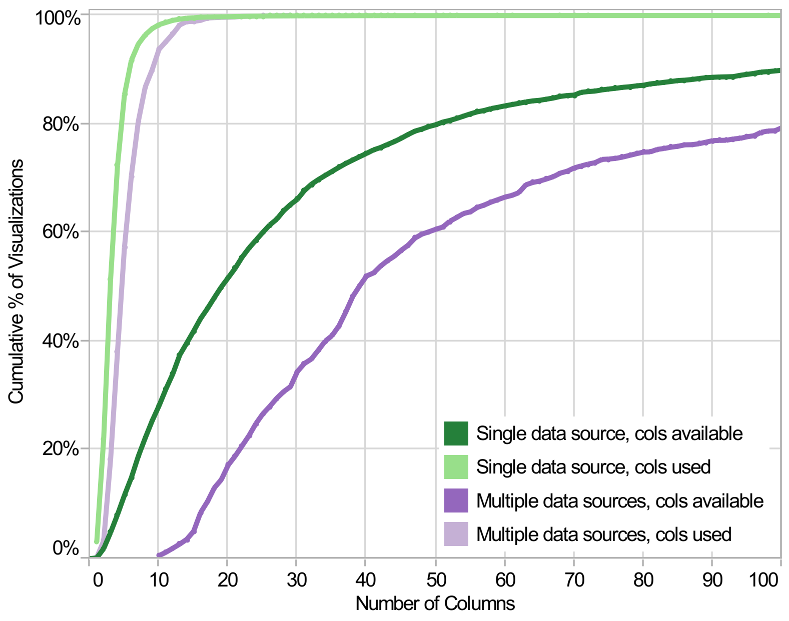


Figure 4.13: CDF of the number of columns in visualizations with one vs. multiple (joined) data sources on Tableau Public. As expected, the distribution of the columns available is much broader, indicating that there are many more columns available that are not being leveraged by the visualization.

Public than Many Eyes. This is presumably due to the fact that individual visualizations on Many Eyes are typically limited to a small number of dimensions that can be shown at the same time, while Tableau Public allows users to build complex multi-view dashboards. Both systems let the user pick the dimensions to be displayed from the available ones for exploration, however.

View dimensionality on Tableau Public. Since the data sets uploaded to Tableau Public tend to be multi-dimensional, is it also the case that the visualizations are multi-dimensional? Figure 4.13 shows the breakdown of data columns used versus available in visualizations with a single data source versus multiple (joined) data sources on Tableau Public (no equivalent information was available for Many Eyes). First, we see that 52% of visualizations with a single data source use at most 3 columns and 90% use at most 6. As

Number of	Number of columns in visualization													
data sets	1	2	3	4	5	6	7	8	9	10	11	12	13	14
One	Text (68%)	Bar (53%)	Bar (47%)	Bar (32%)	Map* (27%) Bar	Map (32%)	Map* (27%) Bar	Map* (26%) Bar	Bar* (25%) Map	Line (32%)	Map* (24%) Bar	Circle (34%)	Bar (30%)	Bar (20%)
Multiple	Text (75%)	Bar (48%)	Bar (50%)	Bar (41%)	Map* (22%) Bar	Map (35%)	Map (40%)	Text (46%)	Map (49%)	Map (56%)	Scatter* (38%) Map	Scatter* (36%) Bar	Circle (64%)	Map (42%)

Table 4.5: Most common visualization types vs. number of columns in the visualization on Tableau Public. For 1–6 columns, the same visualization techniques are used for single and multiple (joined) data sets: text, bar, and map indicating that certain visualization types depend greatly on the dimensionality of the underlying data.

expected, the distribution of the columns available is much broader, indicating that there are many more columns available that are not being leveraged by the visualization. For example, 50% of single data sources contain 25 or more columns.

Table 4.5 shows the breakdown of the most common visualization types used for a given number of columns. The values denoted with a '∗' in Table 4.5 show that a second visualization type was within 5% from the top choice for that given number of columns. For single data sources, we see that the text table is the most common type when there is only one data column present in the visualization. As the number of columns increases, we see a shift in visualization techniques used: bar views become the dominant technique for 2–4 columns and maps are the most popular for 5–8 columns. This behavior is not too surprising since map views have a minimum requirement of two geographic dimensions (*i.e.,* latitude and longitude).

Visualization types. We study the visualization types most commonly used in both systems. Figure 4.14 shows the results for Many Eyes (left) and Tableau Public (right). We first focus on the visualizations that are common between the two systems: bar, line, map, pie, and scatter. Our first observation is that bar, map, pie, and area views have the same relative order in both systems. On Tableau Public, for example, there are over three times as many bars (38%) than maps (10%), three times as many maps than pies (3%), and three

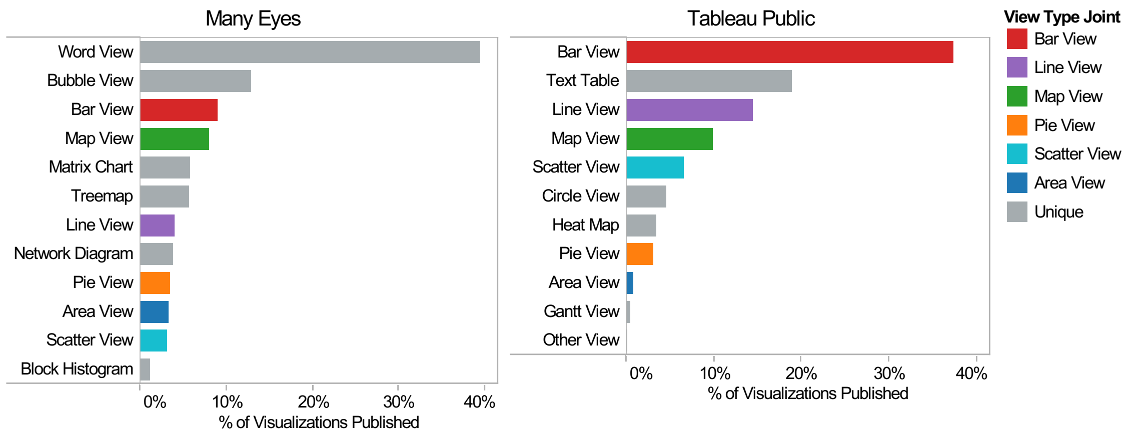


Figure 4.14: Common visualization types for single datasets on Many Eyes (left) and Tableau Public (right). Many Eyes is dominated by text visualizations, while bar charts are the most popular type on Tableau Public.

times as many pies than area views (1%). On Many Eyes, bar views (9%) are almost as frequent as maps (8%) and maps are twice as likely to be found than pie (4%) or area views (3%). Overall, the most common visualization type that exists in both systems is the bar view (38% on Tableau Public and 9% on Many Eyes).

The right half of Figure 4.14 shows that the most common visualization techniques with a single data source on Tableau Public are the bar view (38%), text table (18%), and line view (14%). This result is consistent with Table 4.5, in which the bar and text table dominate for visualizations containing 1 to 4 columns, and Figure 4.15 where 86% of visualizations use 5 or fewer columns. For Many Eyes the most common ones are the word view (40%), bubble view (13%) and bar view (9%).

One of the main differences between these systems worth noting is with regard to text data (Figure 4.14, left). The large number of word views is due to the variety and quality of text visualization views on Many Eyes, most which are not available anywhere else (Tableau Public's text table is just a table, unlike the rich interactive text views on Many Eyes). Similarly, bubble views are attractive but also rather uncommon in visualization and spreadsheet software.

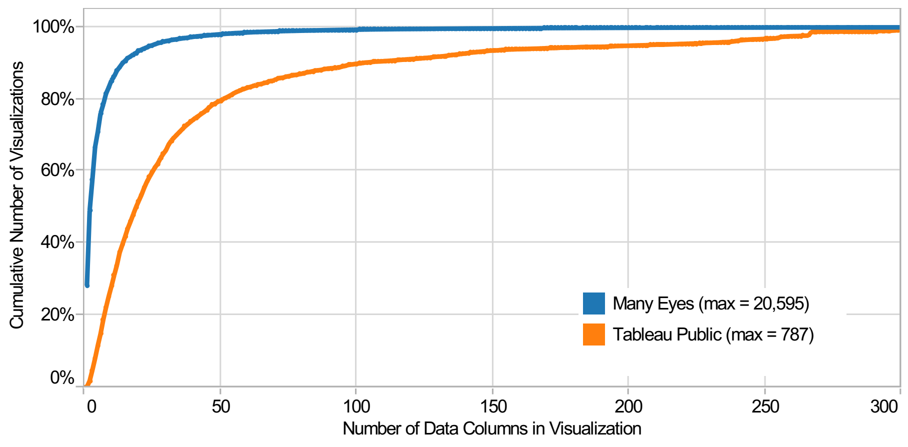


Figure 4.15: CDFs of the number of columns in data sets on Many Eyes and Tableau Public. Almost 50% of data sets on Many Eyes have one or two dimensions, while that number is 25 for Tableau Public.

We thus see consistent results for the two most common visualization types used on Tableau Public; the bar view and text table are the most common. Many Eyes, with its stronger focus on text data, has more popular word views than any other type. Hence some types of visualizations are clearly preferred by users over other types of visualizations but there is room for innovative and specialized visualizations.

Data types. We next consider whether the most appropriate type of visualization depends on the visualized data. For this, we look at the most common visualizations used when viewing attributes with different types. First, in Figure 4.16(a), we see that `Number` (51%) and `String` (44%) are the most common data types in visualizations of a single dataset on Tableau Public. It is interesting that their use is fairly balanced, while intuition would indicate that numbers might be more common due to the quantitative nature of business analytics. The `Number` data type includes both integers and reals. Finally, we see fewer specialized types such as `Datetime` and `Date`, which indicates that visualizations of time-based data are less prevalent.

Many Eyes, however, has a skewed distribution of `String/Categorical` types. In Fig-

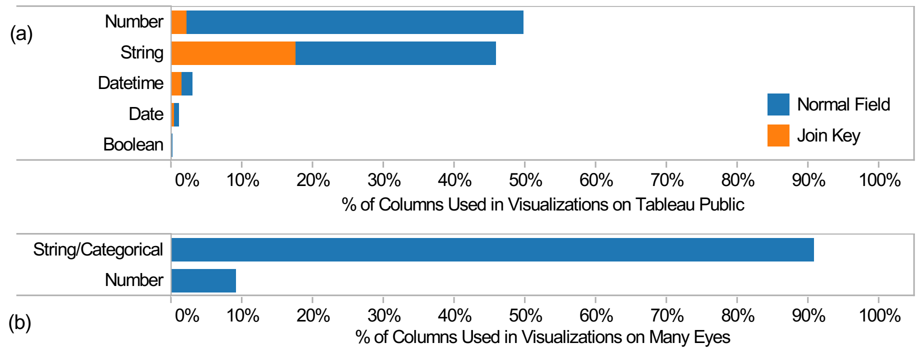


Figure 4.16: Data types of columns in visualizations on Tableau Public (a) and Many Eyes (b). Tableau Public is split between numbers and strings, and word-oriented Many Eyes is heavily skewed toward strings.

ure 4.16(b), we see that 91% of columns on Many Eyes are of this type. This finding is consistent with the previous one regarding the dominance of text-based visualizations on Many Eyes.

Data sets per view on Many Eyes. Many Eyes treats data sets and visualizations as entirely different entities, while Tableau Public packages the data into the workbook. The goal of Many Eyes's approach is to share interesting data sets that many users can build visualizations from.

Interestingly, the number of data sets is much larger than the number of visualizations, and the number of data sets per visualization has increased dramatically over time (Figure 4.17). While there were about 1.2 data sets per visualization in the first quarter of Many Eyes's existence, that number more than doubled to 3.2 during the last quarter of 2012. A sampling of recent data sets shows that many are uploaded multiple times, either because of upload errors or because users are not aware that the data is already in the system. It is unclear, however, why that number has increased over time.

Discussion. In summary, most visualizations have modest data sizes, and seem to not be

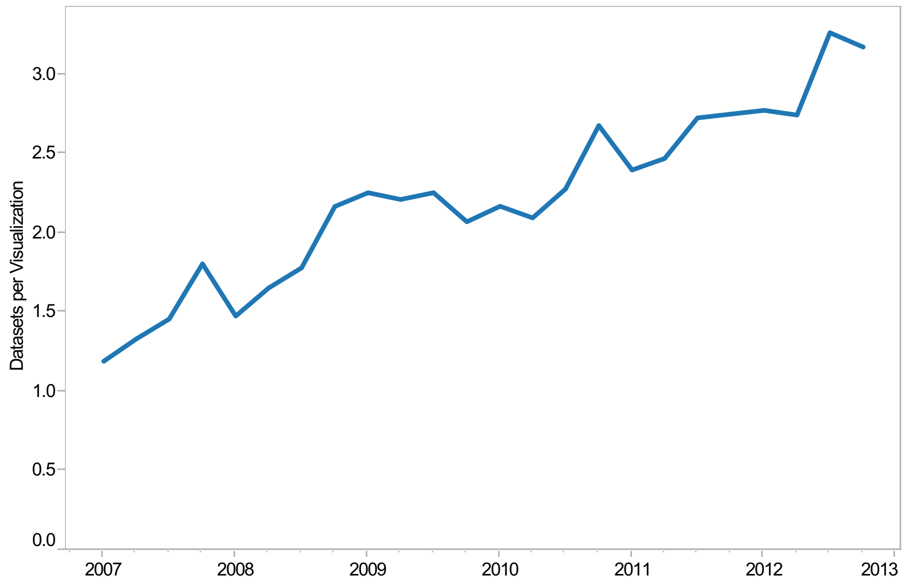


Figure 4.17: Number of datasets per visualization on Many Eyes. This number is steadily rising, which means that users are uploading increasing numbers of datasets compared to the number of visualizations created.

limited by the 100,000 rows restriction, although some users with special privileges visualize datasets with more than one million rows. Additionally, the data sources uploaded to these systems are multi-dimensional. There is thus potential in these systems to support an entirely different class of users with much greater visualization requirements. Furthermore we see that as the number of columns used increases, so does the complexity of the visualization type (*e.g.,* maps require more columns than other types like bar views.) Additionally, visualizations of single datasets tend to use many fewer columns than available. One explanation for this gulf can be drawn from the use of map visualizations in Tableau Public; 62% of such visualizations rely on a Tableau-supplied geocoding database for translating location names into latitude and longitude, since many data sources do not include this necessary context. Finally, visualizations on Tableau Public and Many Eyes contain columns of type `Number` or `String`; the split is very even between these two data types for Tableau Public, and Many Eyes is dominated by `Strings` due to the prevalence of text-based visualizations.

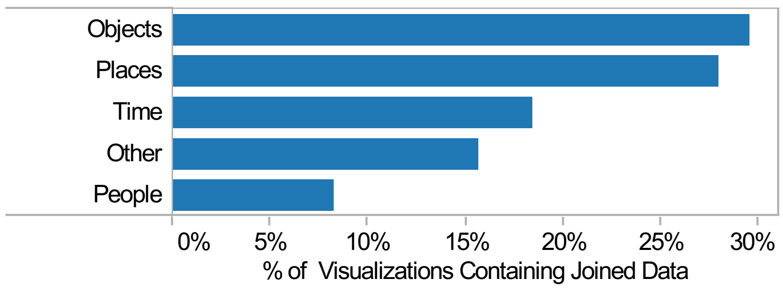


Figure 4.18: Common semantic entities of join keys in visualizations with multiple (joined) data sources. Tableau Public authors tend to combine data sets on the same object entities (as in unique identifiers for product codes) or on the same location (as in zip codes on a map).

4.2.4 Integrating multiple data sets

In this section we study the trends in data and visualization on Tableau Public in the context of blending data from multiple data sources. We omit Many Eyes from this section because the platform currently does not support blending data.

Semantic entities for data blending. On Tableau Public, there are 5,532 visualizations that were created by joining multiple data sets. Of these visualizations, we ask *how do authors combine data sets for their analysis?* To answer this question we manually categorized all of the join keys for the 5,532 visualizations (2%) that have blended data to get a sense of the most popular semantic entities. This process entailed inspecting the column name, data type, and data values of each join key. In the case where the column name was in a foreign language, we used Google Translate on the name and (in some cases) values of that column. If we were still unsure, we opened the workbook to inspect the visualization that was associated with that join key. Figure 4.18 summarizes the semantic entities of the join keys in five different categories: people, places, time, objects, and other. The people category contains any information pertaining to people, including names and demographics. The places category is restricted to geolocations and other identifying characteristics such as zip codes, regions, states, countries, continents, etc. As expected, the time category refers to dates and date times and objects refer to any physical entity that is not a person, place, or time. Objects consist mainly of opaque identifiers like alphanumeric product codes as well as more well-known, descriptive entities such as "university", "department", or "team". Finally, Figure 4.18 shows that visualizations of multiple data sources tend to join on objects (30%), places (28%), and time (18%).

Data columns per visualization. Figure 4.13 shows the breakdown of data columns used versus available in visualizations with a single data source versus multiple (joined) data sources on Tableau Public. From this CDF, we see that visualizations with columns from multiple (joined) datasets tend to be more complex than those containing columns from a single dataset. For example, 43% of the blended views contain 5 or more columns, while only 15% of views with columns from a single dataset contain 5 or more attributes. Furthermore, we see a familiar trend as with single data sources: there is a sizable gulf between the number of columns used and the number of columns available in the blended data sources. Additionally, Table 4.5 shows that, like for single data sources, that visualizations containing a single column tend to be text tables (75%). We also see that the bar view dominates for visualizations containing 2–4 columns and map views for 5–7 columns. This finding is consistent with the distribution of visualization types for single data sources. This behavior is not too surprising since map views have a minimum requirement of two geographic dimensions (*i.e.,* latitude and longitude).

Finally, for visualizations of multiple datasets, the maximum available columns is 793 and the maximum used columns is 29. Similarly for visualizations of a single dataset, the maximum available columns is 792 and the maximum used columns is 133 (this workbook has a poorly designed dataset that contains one column for each day for 4.5 months).

Data types in blended data. Recall from the previous section that Figure 4.16(a) shows that `Number` (51%) and `String` (44%) are the most common data types for visualizations with a single data source on Tableau Public. Additionally, the stacked orange bars represent the data types of the join keys, and `String` (18%) and `Number` (3%) types are the most

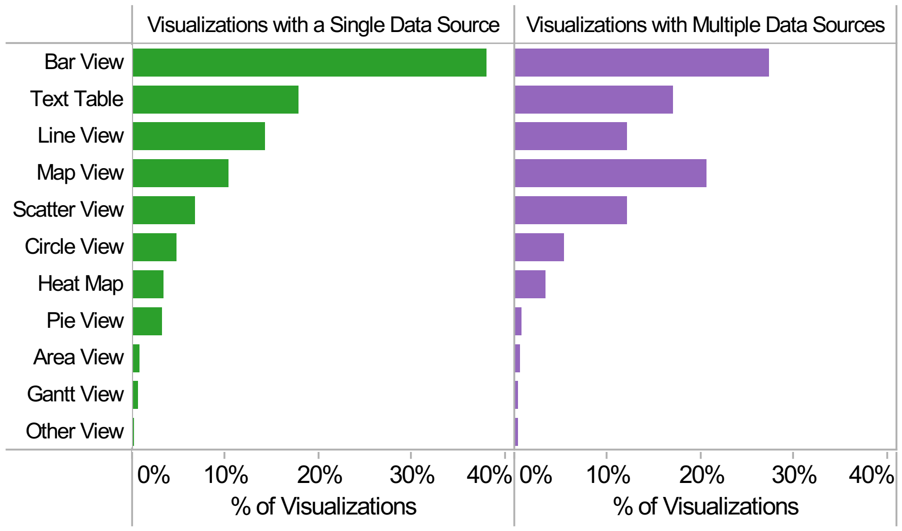


Figure 4.19: Common visualization types on Tableau Public for single and multiple (joined) data sources. Single dataset visualizations are more prevalent as bar, text table, and line views. Blended views are found more often as map views and scatter views (which require higher dimensional data).

common overall.

View types for blended data. Figure 4.19 shows that the most common ways to visualize blended data are with a bar view (27%), map view (21%), or text table (17%). Compared to the distribution for single data sets (recall bar views made up 38%, 18% for text tables, and 14% for line views), we see fewer bar views and more maps. This result is consistent with Table 4.5, in which the text table and map view dominate for visualizations containing 5–10 columns, and Figure 4.13 where 42% of blended visualizations use 5 or more columns.

Figure 4.19 also shows the visualization types with higher percentages of blended views: (in order) map views, scatter views, and text tables. Map views are a special case in Tableau Public, because prior to Tableau version 7 (*i.e.,* before January of 2012), filled maps required tricks involving polygon shapes that were placed using blending. This inflates the number of blended views using maps somewhat, though there are also many other use cases where maps can be used as part of blended views. For example, a common blending pattern for maps is to join on a secondary datasource containing detailed latitude/longitude values. Scatter views are generally used for visualizing correlations, and for authors on Tableau Public, this visualization type is useful for showing correlations between measures from two different data sources. Finally, text tables are often used as a trial/debugging tool for checking out the resulting values from the join operation (*e.g.,* how many `Null` values appear?).

Discussion. In summary, data blending occurs primarily by combining multiple attributes about the same uniquely identified entities from different data sources. This type of blending is more common than simply placing multiple entities at the same location or at the same point in time, although the latter two dominate when considered together. This finding is especially interesting for data integration tools. For example, a recent tool provides recommendations of potentially useful data to integrate with a given database [29]. This tool does not consider joining on place or time. It only considers extending semantic entities with additional attributes. With our study, it becomes clear that such a tool would ignore more than half of all blending scenarios. Additionally, blended visualizations tend to be more complex (*i.e.,* use more columns and have more columns available) than unblended ones. However, the distribution of the most common visualization types for a given number of columns is similar for blended visualizations and those using only single datasets. We also see different trends in visualization techniques for those containing blended data versus single data. Blended views tend to be more prevalent in map views and scatter views; these visualization types tend to be more complex (*i.e.,* use more columns).

4.2.5 Multi-view visualizations

In contrast to Many Eyes, Tableau Public supports visualizations with multiple views, including small multiples as well as multi-view dashboards. These are typically used for creating more complex displays of multi-dimensional data than would be possible with a single view. These are clearly advanced features, so we were wondering how often they are actually used by authors.

Small multiples. With small multiple views, the user can compare a quantitative measure across the members of a (categorical) dimension. Figures 4.5 and 4.19 are examples of small multiple views. To create a small multiples view, the user has to move a dimension field onto a shelf that already contains a dimension or measure. There is no built-in mechanism to suggest how to do this, and Tableau's *Show Me* feature also does not contain small multiples as an option.

We found that 39% of the visualizations published on Tableau Public are small multiple views.

Multi-view dashboards. Dashboards are a complementary technique to small multiple views, providing multiple coordinated displays of data. In contrast to small multiples, where the individual views are identical, dashboards can contain any combination of different visualizations. While adding fields to a single data view tends to make the view more complex and harder to work with, multiple coordinated displays can help split such views into separate displays, which makes it easier to follow.

Each display is explicitly linked to the other views and this allows users to simultaneously explore multiple dimensions of a data source. For example, a link can be defined between views to filter or highlight the members of a dimension that are common to the views. This technique can also be used to explore data from multiple, heterogeneous data sources.

Overall, we found that a majority (74%) of Tableau Public visualizations are featured on a dashboard. Of these dashboard views, we found that 62% are actually true multi-view displays (the rest were using the dashboard for special formatting features of a single visualization), and 42% of those contain actions between views (highlighting, filtering, etc.).

Discussion. The number of both small multiple views and dashboards on Tableau Public was surprising to us. Both require a fairly sophisticated user, or at least considerable experimentation for users without training (which is the vast majority of Tableau Public users). But motivated users with burning questions will not only learn to use tools to be able to answer them, but also explore and make use of advanced features.

This result clearly demonstrates the need for, and usefulness of, advanced features like multi-view visualizations in online visualization tools, which are currently missing from the majority of them.

4.3 Conclusions

In this chapter, we studied five primary dimensions of two popular online visual analytics systems: (1) what types of users are leveraging these systems and what are their workloads, (2) how are users collaborating and interacting with the published content, (3) what are the trends for doing visual analysis over a single-dataset, (4) how do users analyze data joined from multiple sources, and (5) how often are advanced visualization techniques such as multi-view displays utilized.

First, we found that current systems need to effectively support novice users with small datasets. These findings also point to the lack of online, visual analytics tools that would better support users with larger datasets and more sustained data analysis, visualization, and sharing needs.

Second, we showed that in today's collaborative analytics systems, authors tend to bring their own data and do not leverage the contributions of content from other authors. We also discovered that most visualizations of single-dataset (and multi-datasets) tend to use far fewer columns than available. Since both systems have a large repository of potentially useful data sets, we need tools that can help connect users to other good quality data to aid them in their analysis (and in the case where additional data context is necessary to take advantage of a column that would otherwise go unused).

Finally, the use of advanced analytics features like data blending, multiple views, and actions should be encouraging to the data and visualization communities; when turned into usable tools, these features get picked up by users even when they are provided with little guidance, but are self-motivated to answer questions about data.

Chapter 5
DEDUPLICATION FOR VISUAL ANALYTICS SYSTEMS

Visual analytics systems such as Tableau are increasingly popular for interactive data exploration. These tools, however, do not currently assist users with detecting or resolving potential data quality problems including the well-known deduplication problem. Recent approaches for deduplication focus on cleaning entire datasets and commonly require hundreds to thousands of user labels. In this chapter, we address the problem of deduplication in the context of visual data analytics. We present a new approach for record deduplication that strives to produce the cleanest view possible with a limited budget for data labeling. The key idea behind our approach is to consider the impact that individual tuples have on a visualization and to monitor how the view changes during cleaning. With experiments on nine different visualizations for two real-world datasets, we show that our approach produces significantly cleaner views for small labeling budgets than state-of-the-art alternatives and that it also stops the cleaning process after requesting fewer labels.

5.1 Requirements, challenges, and contributions

Visual analytic systems such as Tableau [109] are becoming increasingly popular for data exploration and analysis. These tools enable users to interactively query data through a drag-and-drop interface, and the results are rendered on-the-fly as visualizations. These visualizations are represented internally as database views. Users can create sophisticated views that combine multiple heterogeneous data sets (*e.g.,* Excel spreadsheets, relational databases, data cubes, delimited text files, etc.) along a common dimension or set of dimensions.

Today's visual analytics systems assume that the data sets being consumed are clean and consistent with respect to each other (*e.g.,* all entities in canonical form). However, data (especially on the Web) is often subject to data quality problems. Deduplication is one

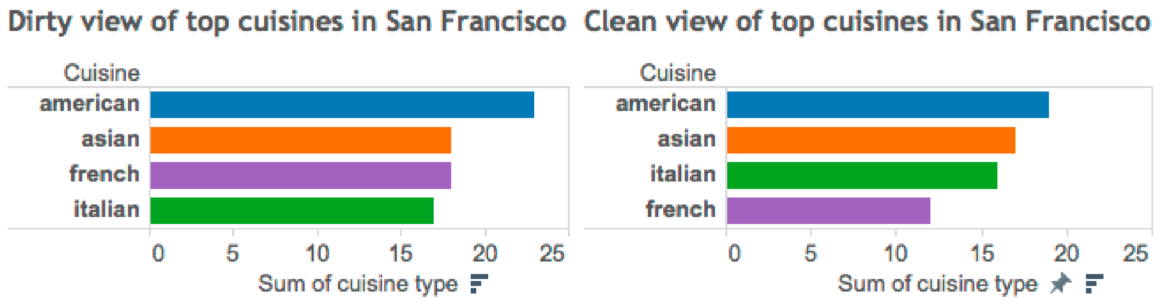


Figure 5.1: Views, dirty (left) and cleaned after 25 labels (right), over Fodor ∪ Zagat restaurant datasets.

kind of dirty data problem. This problem manifests when there are different representations of the same real world entity or object in the data sources being integrated. For example, the same restaurant may appear under two different phone numbers. The same product may use different abbreviations in its name or may include a different description.

Duplicate records may affect a visualization. Figure 5.1 shows an example. The figure shows the top three types of cuisines by quantity of restaurants in San Francisco over a restaurant dataset created from the union of the Fodor and Zagat restaurant ratings datasets in the RIDDLE repository [101]. The figure shows the visualization using either the dirty or clean data. Duplicate records affect results and should therefore be cleaned. The problem, however, is that data cleaning is a disruptive process. It interrupts the user during his primary data exploration task. Our goal is to clean a user's visualization with minimal interruption.

The high-level approach to deduplicating a set of records, R, takes the following steps. The process requires as input a similarity function that takes a pair of tuples (t_1, t_2) from R and produces a similarity score [33]. This similarity function, together with a similarity threshold, can be applied to all pairs of records in R to determine which ones match [37]. Alternatively, a system may rely solely on users to indicate which tuples match [71, 14]. Multiple tuples can correspond to one entity and such clusters further need to be identified [2, 12, 129, 122, 130]. Once matching tuples are identified, they must be merged [51, 16]. To make the previous steps more compute-efficient by reducing the number of record comparisons, blocking techniques are used [13]. Blocking is an inexpensive heuristic filtering step that either partitions the tuples that get compared or removes pairs with low similarity scores.

Since manually devising an accurate similarity function requires an expert, state-of-the-art techniques for deduplication use active learning instead [93, 45], where one or more users label training examples (*i.e.,* pairs of tuples) as either duplicates or not, which enables the system to learn a classifier that categorizes the remaining pairs of tuples. Active learning iteratively asks users for additional, carefully selected labels and re-trains the classifier until the classifier stops improving.

Active-learning-based deduplication is a promising approach for cleaning data visualizations. As a simple example, consider a typical data enthusiast, a food journalist, who wants to publish some visualizations that tell a story about restaurants in San Francisco by the end of the workday. After downloading a US restaurant ratings dataset from the Web that has duplicate entities and before visually exploring it using Tableau (or some other system), the journalist may choose to clean the data. The active learning method [9, 7, 11] would select pairs of records and would ask the user to label them as either duplicates or not. It would then use the labels to build a classifier. Active learning repeats the process until the classifier stops improving. The classifier then serves to label all remaining pairs. After the classification completes, matching records can be merged to yield the clean dataset.

Existing active learning methods produce high-quality classifiers, but at great cost to the user – requiring hundreds to thousands [93, 45] of labels during the data cleaning process, which is a lot for a data enthusiast who most likely just wants to create one or a few visualizations [90]. Several systems use the crowd to perform the cleaning [45], but that approach takes days to complete, which is also inconsistent with our data enthusiast scenario.

In this chapter, we develop an approach that addresses the above problem. Our approach develops a new active-learning-based method to classify tuples as either duplicates or not. In contrast to prior work described above, our approach (1) focuses on producing the cleanest visualization (2) with a small budget of labels from the user (no crowd). We do not address the problem of how best to merge duplicate tuples [16] (we simply drop one of the tuples from each duplicate pair), nor how to handle labeling errors [93] (we assume correct labels). These additional techniques are complementary to the approach developed in this chapter. Additionally, our approach relies on the data enthusiast to provide labels directly. We do not use the crowd.

Our method, called *View Impact Cleaning*, performs deduplication in a manner that focuses on a user's current visualization (or dashboard of visualizations). View Impact Cleaning yields a significantly cleaner view than active learning alone when given a small labeling budget. It only asks the user to label data that is currently being visualized and it automatically stops the cleaning process when it believes that additional labels will not change the visualization further even if they could yield a better overall classifier.

By developing the View Impact Cleaning method, we make the following specific contributions:

1. We define a new notion of *view sensitivity* to duplicate tuples. View sensitivity captures the extent to which a view is affected by duplicate tuples. We also define a new notion of *view impact score* of individual tuples on a visualization. The view impact score measures the extent to which a view will change if a given tuple is found to be a duplicate and is removed (Section 5.3.1).
2. We develop an active-learning-based method that builds an initial classifier and then iteratively improves that classifier. The classifier categorizes pairs of base tuples as either duplicates or not. The novelty of our approach is in the selection of the training examples. Our approach uses both the view impact scores of individual tuples and the potential of a training example to improve the classifier quality (Section 5.3.2).
3. We develop a new stopping condition for view cleaning that considers not the quality of the classifier but instead considers how the view has been evolving during the cleaning process. An important implication of our approach is that it stops cleaning a view both in the case where a sufficient number of tuples have been removed and in the case where a view is not sensitive to duplicate tuples and cleaning has little effect on the view (Section 5.3.3).

We evaluate our approach on nine different views specified on two real-world datasets.

We find that, when given a small cleaning budget, our approach yields significantly cleaner views than active learning without consideration for the users's view. It also effectively stops cleaning earlier than active learning alone while delivering cleaner views. Finally, we evaluate and discuss the problem of cleaning a dashboard comprising multiple visualizations. Our results show that cleaning one view with our approach effectively helps to clean other views even though cleaning is view-driven. As such, our approach helps to make data cleaning a pay-as-you-go task.

5.2 Background

Consider a relation R with attributes $(a_1, \ldots, a_n)$. Two tuples t_1 and t_2 in R are duplicates if they refer to the same real-world entity such as the same restaurant, product, or citation. These tuples, however, are not necessarily identical. For example, the same restaurant may appear twice but with different phone numbers. To clean the data, one needs to identify duplicate tuples and reconcile them. In this paper, we focus on the problem of identifying duplicate tuples and assume any of the existing techniques [44] to reconcile the duplicates once they have been identified.

Learning a Classifier: One can apply machine learning techniques to build a classifier to identify duplicate records [7, 9, 45, 73, 93, 122, 135] in a relation. The approach works as follows. Consider the cartesian product $S = R \times R$. For each tuple in S, compute a feature vector that captures distance information between the individual attributes of the two R tuples that form the S tuple. One can use zero, one, or more distance functions for each attribute. Our implementation uses one function per attribute. The feature vector thus takes the form: $\forall i \in [1, n]$ $dist_{i1}(a_i, a_i), \ldots, dist_{im}(a_i, a_i)$. Commonly used distance functions include edit distance, Jaccard, Jaccard Containment, and Cosine distance (see [26, 37] for detailed descriptions) for string attributes and Euclidean distance for numerical attributes. Other functions are possible [26].

The basic learning algorithm selects a random sample of pairs from S, asks the user to label them as either duplicates or not, and then learns a classifier using that training data.

Because $|S|$ can be large and because the number of positive examples is typically small compared with the number of negative examples, a blocking function serves to reduce $|S|$

before the selection of the training examples. The blocking function takes the form of a selection predicate on S, where the predicate retains only pairs with distance between specific attributes below a pre-defined threshold. For example, a blocking function can retain only pairs of restaurants whose names have an edit distance below some threshold.

Active Learning: Active learning improves on the above approach by iteratively training classifiers on increasingly large and carefully selected training examples. As above, the initial step is to learn a classifier on a random sample of training examples. Active learning then selects additional training examples with the purpose of improving the classifier. Several methods exist to select the additional examples that are most *informative*. Common methods to measure the informativeness of training examples try to measure the disagreement of the component classifiers using uncertainty [93] or entropy [45]. Whatever the method, active learning then retrains a new classifier and repeats the process. Learning stops when the classifiers stop improving across iterations.

5.3 Approach

The goal of our approach is to clean a user's current visualization (or dashboard of visualizations) to the point where duplicate tuples no longer affect it, while making the user do the least amount of work. We employ an active-learning-based approach with the same fundamental setting as presented in Section 5.2: We consider a relation R that contains duplicate tuples. The relation may be the result of the integration of two or more datasets or may contain duplicate tuples for other reasons. We assume R to be given and we do not require knowledge of where individual tuples in R come from. The user builds a view, $V(R)$, and a visualization that displays it. We do not consider the details of the visualization itself. Instead, we focus on the relation $V(R)$ and consider that any change to $V(R)$ affects the visualization. Our approach currently supports views that correspond to select-project queries with optionally aggregation, grouping, sorting, and top-K restrictions.

To clean $V(R)$, our approach is to build a classifier that takes as input all pairs of tuples (t_1, t_2) with $t_1 \neq t_2 \wedge t_1 \in R \wedge t_2 \in R$ and classifies each one as either a duplicate pair or a non-duplicate pair. To clean the view, one tuple from each duplicate pair is removed, but any tuple-merging algorithm can be applied with our approach. Our goal is to produce the cleanest possible view for a given label budget l. Additionally, we require that l be small in the order of tens or low hundreds of labels rather than the thousands of labels commonly used in prior work [45, 93].

In this section, we first describe our model to reason about the quality of a view and the impact of individual tuples on that quality (Section 5.3.1). We then present our active-learning approach to view cleaning, which is based on this model (Sections 5.3.2 and 5.3.3).

5.3.1 Modeling view quality

Consider a relation R and a view, $V(R)$, defined on that relation. Our goal is to clean $V(R)$. We define R_{clean} to be the relation R with all duplicate tuples removed. For convenience, we refer to the original view, $V(R)$, as V_{dirty}, to $V(R_{clean})$ as V_{clean}, and to the same view V computed on a partially cleaned relation as V_{curr}. We define the quality of a view as follows:

Definition 5.3.1 *`Quality(V)` of any view V is `1 - Distance(V,` V_{clean}`)`, for some distance function, `Distance`* $\in [0, 1]$.

The quality of a view thus depends on the distance to the view computed on clean data. In our implementation, we use the well-known Earth Mover's Distance [103] to compute distances between views as we describe in Section 5.4.

We observe that the quality of a view can also capture its *sensitivity to duplicate tuples*. Some views can be resilient to duplicate tuples. For example, a view that displays median values is not easily affected by duplicate results. These views change little as a result of data cleaning. Other views, in contrast, such as those displaying top-k results for example, can change significantly when the data is dirty. We thus use the value of `Distance(V,`V_{clean}`)` as proxy for the sensitivity of a view to duplicate tuples.

During the view cleaning process, the quality of a view can never be measured directly because the system does not have access to R_{clean} and thus V_{clean}. Instead, our approach operates on distances between either consecutive views obtained during the iterative view cleaning process, `Distance(`V_{curr}`,`V_{curr+1}`)` or the distance from the original, dirty view to the current view: `Distance(`V_{dirty}`,`V_{curr}`)`.

An important component of the cleaning process is the identification of pairs of tuples to show to the user for labeling. To select these tuple pairs, our approach takes into consideration how much each tuple affects the view $V(R)$. We call this the *view impact score* of a tuple:

Definition 5.3.2 *The `view impact score` of a tuple $t \in R$ on a view $V(R)$ denoted as `Impact(V,t)` is `Distance(V(R),V(R-t))`*

The impact of a tuple is important during cleaning because it captures the potential improvement in quality if the tuple is determined to be a duplicate and is removed. For example, consider the top-k view of cuisine types by quantity of restaurants in Figure 5.1: a tuple whose `cuisine` attribute is 'American', 'Asian', 'French', or 'Italian' would have a higher impact score than one with a rare type such as 'Indonesian'.

5.3.2 View cleaning

In this section, we present our active-learning-based algorithm for cleaning views by taking into account changes in view quality and view impact scores of individual tuples.

Initial classifier. The first step in the active learning process is to select a set, L_0, of training examples, ask the user to label them, and train an initial classifier using those labels. A training example is a pair of tuples, (t_1, t_2) with $t_1 \in R \wedge t_2 \in R \wedge t_1 \neq t_2$. When a pair has duplicate tuples, it is a positive example. Otherwise, it is a negative example.

Recent prior work [93] randomly selects a 3% sample of such pairs to train the initial classifier. A known challenge with record deduplication, however, is that the number of positive examples is extremely small even when a dataset contains many duplicate tuples. For example, if each duplicate tuple in a relation of size $|R|$ participates in one positive example it also participates in $|R| - 2$ negative examples. As a result, a small random sample of training examples can easily fail to include any positive examples, leading to a poor initial classifier, especially when $|R|$ is large. A common approach to alleviate this problem is to use blocking, where all tuple-pairs with low similarity scores for one or more features are discarded before the data cleaning process even begins. For example, pairs of restaurants with names that are not at all similar should be discarded. We further use a second blocking method: We focus only on tuples that participate in the view. Instead of cleaning $|R|$, we clean only those tuples in $|R|$ that pass the selection condition in the query. We denote these tuples with $\texttt{Provenance}(V(R))$ since they correspond to the why-provenance [18] of $V(R)$ if we ignore any top-k clauses in the query. Table 5.2 shows the fraction of duplicates for two datasets that we describe further in Section 5.5. The table shows the result for both the dataset as a whole and for the subset of the data in the views that we use in the evaluation. Even after applying both types of blocking (blocking on the view and the features), the fraction of positive examples is only 2.3% and 9.4% for the two views (we describe the exact blocking function in Section 5.5).

The second challenge with learning a classifier for record deduplication is that the features themselves used to train the classifier may be poor. In our application domain, in particular, the user's goal is to create and analyze a given set of visualizations. The user is not seeking to clean the data. As a result, the system cannot rely on the user to determine a good set of features. Instead, the feature selection process must be automated, which complicates the identification of a good set of features as we describe further in Section 5.4. Poor features make it difficult to learn a good classifier.

The above two challenges make it difficult to build high quality classifiers as we show in the evaluation, and lead us to develop a different strategy for training an initial classifier. Our key idea is to get the user to label tuple-pairs where at least one tuple has a high view impact score. The intuition is that these pairs will not necessarily be worse training examples than random pairs. At the same time, correct labels for these pairs have the highest potential to improve the quality of the view. For example, in Figure 5.1, tuples that correspond to American, French, Asian, and Italian restaurants will have higher view impact scores than others and pairs containing such tuples should be weighted more heavily when selecting examples to label.

The approach to learn the initial classifier has three main steps: view impact score computation (Algorithm 1), training-example selection, and training of the initial classifier (Algorithm 2 lines 1 through 14). The view impact computation proceeds as follows:

Algorithm 1 View Impact Scores ($V(R)$)

```
1: Input: Base relation $R$ and view $V(R)$
2: Output: PairScores, map of pairs to view impact scores
3: Let TupleScores = PairScores = $\emptyset$
4: for each tuple t $\in$ Provenance(V(R)) do
5:     score = Distance($V(R)$, $V(R - t)$)
6:     TupleScores = TupleScores $\cup$ (t, score)
7: end for
8: for each pair (t, score) $\in$ TupleScores do
9:     for each u $\in$ Provenance($V(R)$) - {t} do
10:        PairScores = PairScores $\cup$ ((t,u), score)
11:    end for
12: end for
13: Return PairScores
```

1. For each tuple $t \in$ `Provenance`$(V(R))$, we compute its view impact score, `score`, as per Definition 5.3.2. For example, in the view in Figure 5.1, we only compute the view impact for restaurants in San Francisco. Other tuples necessarily have a view impact score of zero. We store the results in a relation called `TupleScores`.
2. For each tuple $t \in$ `TupleScores`, we generate $|$`Provenance`$(V(R))| - 1$ potential training examples of the form $((t, u), \text{score})$, where $u \in$ `Provenance`$(V(R)) - \{t\}$ and score is the view impact score for t. We store the results in a relation called `PairScores`.

Selecting the initial training examples proceeds as follows:

1. First, we apply a blocking function that removes obvious non-matches from the previously computed `PairScores`. The blocking function drops all pairs with at least one attribute that has a similarity below a pre-defined threshold (or distance above a pre-defined threshold). Section 5.5 describes the blocking function that we use in the experiments. This step corresponds to function `block` on line 6 of Algorithm 2.
2. Second, we select $|L_0|$ examples from the filtered `PairScores` by using weighted random sampling with weights equal to the view impact scores. To train a new classifier,

we need a set of examples that are not only informative but also diverse [45], and hence as prior work [93, 45], we use weighted random sampling rather than selecting the top-k pairs with highest view impact. This approach heavily weighs the pairs with high scores while randomly breaking ties between pairs that have the same score, such as pairs generated from the same initial tuple. This step corresponds to function $\texttt{Select}_{bias}$ on line 7 of Algorithm 2.

Finally, the user labels the selected pairs and these labels serve as training examples for the initial classifier (lines 8 through 10 in Algorithm 2). The duplicate tuples identified explicitly by the user or implicitly by the classifier are then removed from the input data and the view is recomputed as shown in lines 11 and 12 in Algorithm 2.

Subsequent training examples. To improve the initial classifier, the active learning method selects additional training examples for the user to label. As described in Section 5.2, active learning strives to select examples that are most *informative* and thus have the highest potential to help improve the classifier. It then learns a new classifier on the expanded training data.

As in the case of the initial classifier, we propose to take a different approach and leverage view impact scores when selecting additional training examples. We propose the following two approaches:

1) View Impact Method (ViewImpact): As in the case of learning the initial classifier, this approach favors training examples where at least one tuple has a high view impact score. Instead of breaking ties randomly, however, in the case of these subsequent iterations, we select those samples that can help improve the current classifier the most. The tie breaker is to select the samples with small *margin distance* i.e., the samples with the minimum absolute confidence score from the classifier. If the margin distance is small, the classifier is less confident. Therefore, the sample is better chosen for labeling as it should help fine-tune the classifier. This tie-breaker is similar to the uncertainty method, in which the examples that are closest to the decision boundary are selected. In our case, however, uncertainty is secondary to view impact. This selection algorithm corresponds to method

Algorithm 2 View Impact Cleaning(l,b,b_{L0},$V(R)$)

```
1: Input: l is the total labeling budget,
2: Input: b is the batch size, $b_{L0}$ is $L_0$ size
3: $V_{dirty} = V(R)$ is the view to clean
4: Output: $V_{curr}$ the current cleaned view
5: PS = ViewImpactScores($V_{dirty}$) // PS is a map of pairs to their view impact scores
6: PS = block(PS) // filter candidate pairs with blocking function
7: $L_0 = Select_{bias}(b_{L0}$,PS) // select pairs for user to label
8: $L_0$ = Label($L_0$) // label the $L_0$ pairs
9: PS = PS - $L_0$ // remove labeled pairs from the score map
10: VL = $\theta_{L_0}$(PS) //train $\theta$ on $L_0$ & label remaining pairs
11: dups = matches($L_0 \cup$ VL) //get dups from $L_0$ & VL
12: $V_{curr} = V(R - \{\text{dups}\})$
13: view_change = Distance($V_{curr}, V_{dirty}$)
14: $l = l$ - | $L_0$ | // remove user labeled pairs from budget
15: while $l > 0$ & NOT Converged(view_change) do
16:     $T = Select_{top}$(b,PS) //top scoring pairs, applies tie breaker
17:     $T$ = Label($T$) // label the $T$ pairs
18:     $T_{acc} = T_{acc} \cup T$ // accumulate the $T$ pairs
19:     PS = PS - T // remove user-labeled pairs from PS
20:     VL = $\theta_{L_0 \cup T_{acc}}$(PS) // train $\theta$ on $L_0 \cup T_{acc}$ & use it to label PS
21:     dups = matches($L_0 \cup$ VL) // get dups from $L_0$ & VL
22:     $V_{prev} = V_{curr}$
23:     $V_{curr} = V(R - \{\text{dups}\})$
24:     view_change = Distance($V_{curr}$,$V_{prev}$)
25:     $l = l$ - | T | // remove user labeled pairs from budget
26: end while
27: Return $\mathbf{V_{curr}}$
```

$Select_{top}$(b,PS) in Algorithm 2 and Algorithm 3 and it works as follows:

- The function takes as input the `PairScores` data structure. For each subset of pairs $\{((t, u), s)\} \in$ `PairScores` associated with the same original tuple t, the approach retains only the entry with the lowest margin distance, which it appends to a new

Algorithm 3 $Select_{top}$(b,PS)

```
1: Input: b is the batch size,
2: PS is a map of pairs to their view impact score
3: Output: TopPairScores is a map of pairs to their view impact score
4: TopPairScores = $\emptyset$
5: BestPairs = $\emptyset$
6: PM = getAbsMarginDistance(PS) //map each pair in PS to its absolute margin distance
7: UPM = getCandidateUids(PS) //group pairs by the candidate duplicate tuple UID
8: for uid $\in$ UPM do
9:     pairs = UPM[uid]
10:    bestPair = $\emptyset$
11:    bestMargDist = 0
12:    //for each UID, find the pair with the min margin distance
13:    for pair $\in$ pairs do
14:        if bestPair == $\emptyset$ or PM[pair]<bestMargDist then
15:            bestPair = pair
16:            bestMaginDist = PM[pair]
17:        end if
18:    end for
19:    BestPairs = BestPairs $\cup$ bestPair
20: end for
21: for (pair, impactScore) $\in$ PS do
22:    //keep only pairs with the smallest margin distances
23:    if pair $\in$ BestPairs then
24:        TopPairScores = TopPairScores $\cup$ (pair, impactScore)
25:    end if
26: end for
27: TopPairScores = weightedSampling(b,TopPairScores)
28: Return TopPairScores
```

data structure, `TopPairScores`.

- The algorithm then selects b pairs from `TopPairScores` using weighted sampling where the view impact score s serves as the weight.

Hybrid Method (Hybrid): We also explore a second approach that is a hybrid between the traditional method of selecting the most "informative" training examples and the ViewImpact method. The hybrid approach computes the same `TopPairScores` structure as the ViewImpact method above. However, it then assigns the following hybrid weight to each pair of tuples in that set before selecting the next batch of b examples using weighted sampling. `ClassifierUncertainty` measures the classifier uncertainty as in state-of-the-art active learning methods for record deduplication [93] (See Section 5.2).

$$\text{HybridScore } = \alpha \text{ViewImpactScore} + (1-\alpha)\text{ClassifierUncertainty} \tag{5.1}$$

Hence, in contrast to ViewImpact, which weighs samples based on their view impact score alone, Hybrid can weigh both the ClassifierUncertainty and ViewImpactScore with an adjustable relative weight, α.

Hence, in contrast to ViewImpact, which weighs samples based on their view impact score alone, Hybrid can weigh both the ClassifierUncertainty and ViewImpactScore with an adjustable relative weight, α.

Algorithm 2, lines 12 and following capture how the above methods fit within the overall active learning part of the cleaning process.

5.3.3 Stopping condition

The state-of-the-art stopping condition for deduplication [45] is based on when the classifier stops improving in accuracy. The idea is to check the *confidence*, or agreement among classifiers on a set of example pairs, over a fixed window of time. Before active learning, a small (3%) random sample of pairs from the underlying data is set aside as a holdoutset for evaluating classifier quality. As the classifier learns from more informative examples, the confidence values will increase. However, when there are few informative examples left to learn from the confidences level off. Once the confidence values have stabilized within +/- $epsilon = 0.01$, over a window size of $n_{converge} = 20$ iterations, then the training stops.

Our approach, in contrast, is to check the convergence of the view quality. Intuitively, the view cleaning process should stop when the view stops improving or when the user has exhausted her labeling budget. A view stops changing as a result of cleaning either because the data has been cleaned or because the view is no longer sensitive to the remaining duplicate tuples. We say that a view has *converged*:

Definition 5.3.3 *A view has converged if Distance(V_{curr}, V_{prev}) $\leq \epsilon$ for $n_{converged}$ iterations.*

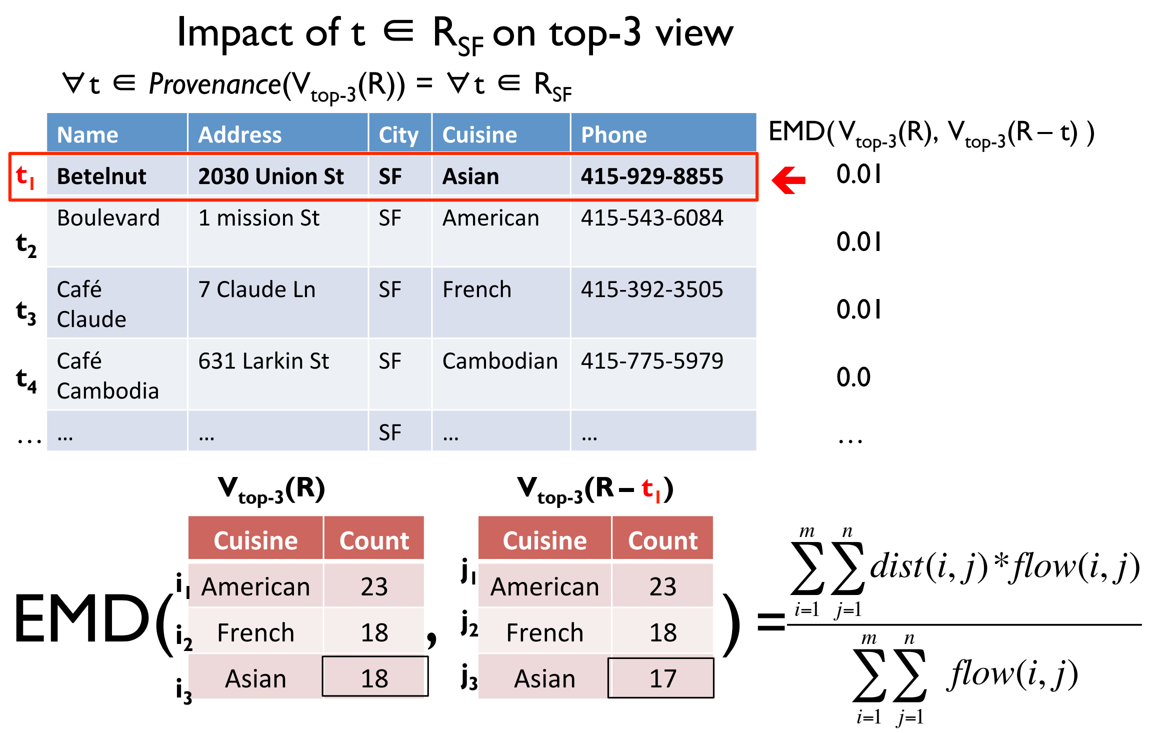
The condition for stopping the process of cleaning a view is thus based on the convergence – within some ϵ– of the `Distance` function computed between consecutive views during cleaning.

5.4 Implementation

Different implementations of the View Impact Cleaning approach are possible. In this section, we discuss our choices for the distance function for views, the classifier, and the features.

Distance function for views We use the Earth Mover's Distance (EMD) [103] to compute distances between views. Other distance measures could be used as well. EMD is a method to evaluate dissimilarity between two multi-dimensional distributions. Intuitively, given two distributions, one can be seen as a mass of earth spread in space, the other as a collection of holes in that same space. Then, the EMD measures the least amount of work needed to fill the holes with earth. Here, a unit of work corresponds to transporting a unit of earth by a unit of ground distance.

To compute the EMD between two views V_1 and V_2, we need a distance function for individual tuples in these views and a weight for each tuple. For the weight, we assign each tuple in a view V the same weight equal to $\frac{1}{|V|}$. Following prior work [77], we consider each tuple t with n attributes as an n-dimensional vector and use Euclidean distance to compute the distance between two tuples $i \in V_1$ and $j \in V_2$. For individual attributes in a tuple, we use Euclidean distance to compute the distance between numeric attributes (normalized



Impact of $t \in R_{SF}$ on top-3 view

$\forall t \in Provenance(V_{top\text{-}3}(R)) = \forall t \in R_{SF}$

	Name	Address	City	Cuisine	Phone	$EMD(V_{top\text{-}3}(R), V_{top\text{-}3}(R-t))$
t_1	**Betelnut**	**2030 Union St**	**SF**	**Asian**	**415-929-8855**	0.01
t_2	Boulevard	1 mission St	SF	American	415-543-6084	0.01
t_3	Café Claude	7 Claude Ln	SF	French	415-392-3505	0.01
t_4	Café Cambodia	631 Larkin St	SF	Cambodian	415-775-5979	0.0
…	…	…	SF	…	…	…

$V_{top\text{-}3}(R)$

	Cuisine	Count
i_1	American	23
i_2	French	18
i_3	Asian	18

$V_{top\text{-}3}(R-t_1)$

	Cuisine	Count
j_1	American	23
j_2	French	18
j_3	Asian	17

$$EMD(V_{top\text{-}3}(R), V_{top\text{-}3}(R-t_1)) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} dist(i,j) * flow(i,j)}{\sum_{i=1}^{m}\sum_{j=1}^{n} flow(i,j)}$$

Figure 5.2: (top) The EMD is computed for each tuple t in $\texttt{Provenance}(V_{top-3}(R))$ from Figure 5.1 denoted R_{SF} to assess the tuple's view impact. (bottom) The EMD between $V_{top-3}(R)$ and $V_{top-3}(R-t_1)$ is 0.01.

to the $[0, 1]$ range) and 1 - string equality as the distance between categorical attributes. To illustrate the distance computation consider the following two views (in Figure 5.2): (1) $V_{top-3}(R)$, a top-3 view of cuisines in San Francisco over a dirty restaurants dataset, R, and (2) the same top-3 view but over a cleaner relation R' where one duplicate tuple t has been removed.

The SQL statement for this view appears in Table 5.3. The distances are then calculated between all combinations of tuples (i, j) where $i \in V_{top-3}(R)$ and $j \in V_{top-3}(R-t)$, as shown in Table 5.1. Each tuple is given the same weight of $1/|V_{top-3}|$ and we call the library from [35] to solve the linear program that computes the minimum flow to move the earth between the views using the pre-computed distances. The solution to the linear programming problem is shown bolded in Table 5.1. The EMD returned is $\frac{0*1/3+0*1/3+0.043*1/3}{1/3+1/3+1/3}$,

$i \in V(R_{SF})$ $[i_{Cuisine}, i_{Count}]$	$j \in V(R_{SF} - t_1)$ $[j_{Cuisine}, j_{Count}]$	Attribute dist(i,j) = [1-StrEq($i_{Cuisine}$,$j_{Cuisine}$),normEuclid(i_{Count},j_{Count})]	Tuple dist(i,j) = Euclid(Attribute dist(i,j))	flow(i,j)
[“American”, 23]	**[“American”, 23]**	**[0.0, 0.0]**	**0.0**	**1/3**
[“American”, 23]	[“French”, 18]	[1.0, 0.217]	1.023	0.0
[“American”, 23]	[“Asian”, 17]	[1.0, 0.261]	1.033	0.0
[“French”, 18]	[“American”, 23]	[1.0, 0.217]	1.023	0.0
[“French”, 18]	**[“French”, 18]**	**[0.0, 0.0]**	**0.0**	**1/3**
[“French”, 18]	[“Asian”, 17]	[1.0, 0.043]	1.001	0.0
[“Asian”, 18]	[“American”, 23]	[1.0, 0.217]	1.023	0.0
[“Asian”, 18]	[“French”, 18]	[1.0, 0.0]	1.0	0.0
[“Asian”, 18]	**[“Asian”, 17]**	**[0.0, 0.043]**	**0.043**	**1/3**

Table 5.1: EMD computation details for tuple, t_1, and the views, $V(R_{SF})$ and $V(R_{SF} - t_1)$, from Figure 5.2. $\forall$ pairs $(i,j) \in V(R_{SF}) \times V(R_{SF} - t_1)$ dist(i,j) is computed by applying the Euclidean distance to the set of attribute distances, Attribute dist(i,j). For example, the tuple distance for the third row in this table is computed as: $\sqrt{(1.0)^2 + (0.261)^2} \approx 1.033$. Each attribute has *one* type-based distance function applied to it, *e.g.,* since *Count* is a `num` type, the Euclidean distance is used (and normalized by the max value in the table so that the result is in the [0,1] range). For example, the `normEuclid`$(23, 17) = \frac{\sqrt{(23-17)^2}}{23} \approx 0.261$. With the tuple dist($i$,$j$) and per-view tuple weights $= 1/|V| = 1/3$ as input, we call the EMD library in [35] to solve for the flow(i,j)that minimizes movement of earth between the two views, or dist(i,j)*flow(i,j).

which is 0.01.

Classifier. We use a common choice for classification, support vector machines, to build the classifier. Our implementation uses libsvm [22]. We use either linear kernels or Gaussian kernels by tuning over the data. Because the number of positive and negative examples is imbalanced, we set the weights for the positive and negative label classes to the reciprocals of their respective cardinalities.

Features: Selecting the right features to give to a machine learning algorithm (or *feature engineering*) is a well-known, challenging problem. Recent prior work leaves the feature selection decision to an expert user [4], or in the case of Corleone [45], the system randomly selects a subset of attributes as features. For the applications that we target, we cannot require that users define features. In our implementation, we use generic, type-based features. We find, however, that such generic features do not always work well, complicating

Dataset with blocking method	Rows	Pairs	Cols	Dup Pairs	Dups (%)
Restaurants:(Fodor ∪ Zagat)	864	7.4x10^5	5	224	0.03
block on view:SFrestaurants	148	2.1x10^4	5	36	0.17
block on view & features:SFrestaurants	148	384	5	36	9.4
Products:(Amazon ∪ Google)	4,589	2.1x10^7	4	1,300	0.006
block on view:MfrProducts	291	8.3x10^4	4	162	0.19
block on view: & features:MfrProducts	291	6.9x10^3	4	162	2.3

Table 5.2: Datasets used in evaluation. Table shows the cardinality, number of pairs, degree, number of duplicate (matching) pairs, and fraction of pairs that are duplicates for the base data and after view and feature blocking.

the problem of building a good classifier and emphasizing the benefit of View Impact Cleaning. In the experiments, we manually select one type-based distance function to serve as feature for each pair of attributes. We leave the full automation of the feature selection step as future work.

5.5 Experiments

We evaluate the effectiveness of our View Impact Cleaning approach by measuring the view quality that it achieves (we measure and show $\texttt{Distance}(V_{curr}, V_{clean})$ where $\texttt{Distance} = \texttt{EMD}$) as compared with view-agnostic active-learning. We focus on the results that each approach achieves for small numbers of labels since our main goal is to limit the amount of work that the user needs to perform.

Data sets. We use two datasets (Table 5.2) that differ in their degrees of difficulty to build a good classifier for duplicate tuples. The restaurants dataset was collected from the culinary rating sites, Fodor and Zagat. Both tables have the following five attributes: name, address, city, cuisine, and phone. Figure 5.2 shows example records. The restaurant dataset retains this schema and comprises the union of the Fodor (533 tuples) and Zagat

(331 tuples) tables.

Products, a more challenging dataset to deduplicate, combines electronics products from Amazon (1,363 rows) and Google (3,226 rows) [3] with schema (name, description, manufacturer, price). An example record is: [*'learning quickbooks 2007','learning quickbooks 2007','intuit',38.99*]. There are more than 21 million tuple pairs in the union of these tables, among which only 1,300 pairs refer to the same entity (0.006% matches).

Features. We compute the features only for tuples in the provenance of the view. For restaurants we compute the following four features: `jaccard`(name), `jaccard`(address), `jaccardContainment`(name), and `jaccardContainment`(address). For products we compute the following five features: `cosine`(description), `normEuclidean`(price), `jaccard`(name), `jaccard`(name_numeric), and `cosine`(name_alpha), where name_numeric contains only the numeric values extracted from the product name and name_alpha retains only the strings. We found this separation to be important because product names in this dataset contained both model numbers and English-language descriptions. We do not use the manufacturer attribute because our views select only tuples that belong to one of three manufacturers.

Views. We study `SELECT`, `PROJECT`, and `AGGREGATE` (*e.g.,* `GROUP/ORDER BY LIMIT` based views. Queries containing joins have not been evaluated, but there is no theoretical limitation to applying the View Impact Cleaning method to such views. Recent prior work on deduplicating views [2] only applies to simple, non-aggregate `SELECT/PROJECT` views, while others such as SampleClean [128, 76] are designed only for aggregate queries without ordering nor top-k clauses. We evaluate our approaches over 9 views (Table 5.3) that we choose for the following reasons: (1) variety of impact that individual duplicates have on the view and (2) variety of the overall view's sensitivity to duplicates.

Blocking methods. When considering the cross-product of tuples in each dataset, the number of positive examples (pairs that match) grows linearly with the size of the data while the number of negative examples grows as the square of the relation size. We use two types of blocking strategies to reduce this class imbalance: view-based and feature-based blocking.

Table 5.2 shows how each type of blocking increases the fraction of positive examples by an order of magnitude. For the restaurants dataset, our views select restaurants in San Francisco (shown as SFrestaurants). For the products dataset, the views include products sold by Microsoft, Apple, and, Adobe (shown as MfrProducts). For feature-based blocking, for restaurants, we drop pairs whose `jaccard` or `jaccardContainment` match scores on the *name* and *address* attributes are less than 0.2. For products, we block on *price* and *name* when the normalized euclidean distance on price is greater than 0.54 and `jaccard` scores are less than 0.17 or `jaccardContainment` scores are less than 0.27.

For all experiments in this section, *all approaches* (including view-agnostic active learning) select pairs from the two views that include *both* blocking strategies, or SFrestaurants and MfrProducts.

Machine learning settings. As indicated in Section 5.4, we use the libsvm [22] libraries for the implementation of the learner. All restaurants experiments use a linear kernel and a cost factor of 10. For the products dataset, all experiments use a Gaussian kernel with a cost factor of 1000 and the default gamma. For both datasets, the weights for the positive and negative label classes were set to the reciprocals of their respective cardinalities.

Experimental setup. We run the restaurants experiments 20 times and the products experiments 100 times. For each experiment, we create a randomly-selected holdoutset, which is not used for training. It serves to evaluate the quality of the classifiers. The size of the holdoutset is approximately equal to half of the size of the initial unlabeled set.

Methods compared. We apply two state-of-the-art active learning methods, which we refer to as Uncertainty and Entropy, as a baseline. For each of these methods, we use an uncertainty or entropy measure to select the subsequent batches of examples for active learning. Our implementation is based on the description in [93]: uncertainty [93, 105] and entropy [45] scores are computed over 10 bootstraps that are sampled with replacement from the trainingset. The examples are ranked by either their uncertainty or entropy scores and selected by applying biased weighted sampling. Both Uncertainty and Entropy measure

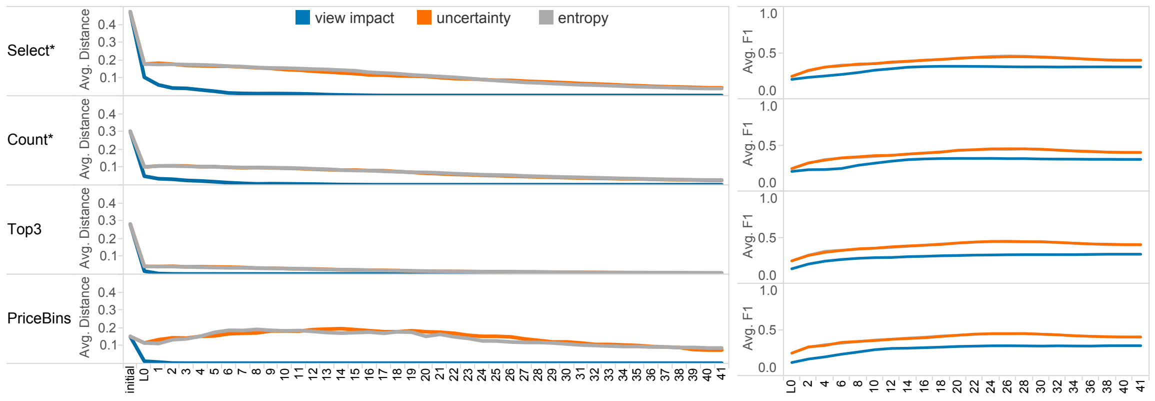


Figure 5.3: All product views are cleaned by View Impact (monotonically decreasing) in less than 18 batches with an L_0 size of 100 pairs and subsequent batch size of 20. View cleanliness (left) is shown with AVG $Distance(View_{curr}, View_{clean})$ and (right) classifier accuracy with avg. F1. Entropy and Uncertainty have similar results for both cleaning ability and classifier accuracy: both require more than 42 batches to completely clean any of the product views and exhibit non-monotonic behavior for the `PriceBins` view.

the disagreement of the classifiers over the holdoutset example labels. Thus, the higher the uncertainty, the stronger the disagreement, and the more *informative* the example is to the learner.

5.5.1 End-to-End Results

We first compare the overall ability of our approach, View Impact Cleaning, and the two state-of-the-art active learning algorithms, Entropy and Uncertainty, to clean views with a small number of user labels. Figures 5.3, 5.4, and 5.5 show `Distance`(V_{cur}, V_{clean}) before cleaning (value shown under "Initial distance"), after cleaning with the initial classifier (labeled L_0), and after the maximum budget of user labels. Each point represents the average of either 20 runs (Restaurant dataset) or 100 runs (Products dataset) and the standard deviation, σ. Since Figure 5.3 shows that Uncertainty and Entropy achieve similar cleaning results and classifier accuracies, we present only Uncertainty in future graphs.

Main result for products. As shown in Figures 5.3 and 5.4(left), all product views are

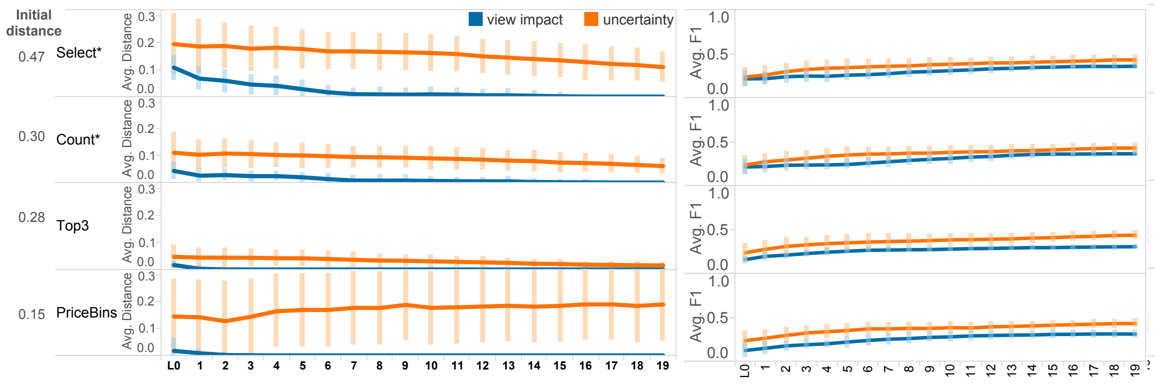


Figure 5.4: All product views are cleaned by View Impact (monotonically decreasing) in less than 18 batches with an L_0 size of 100 pairs and subsequent batch size of 20. View cleanliness (left) is shown with AVG $Distance(View_{curr}, View_{clean})$ +/- σ and (right) classifier accuracy with avg. F1 +/- σ. Uncertainty, however, requires more than 20 batches to completely clean any of the product views and exhibits non-monotonic behavior for the `PriceBins` view.

completely cleaned by View Impact in fewer than 18 batches, or 440 labels (including the initial L_0 batch of size 100 and subsequent batches of size 20). `Top3` and `PriceBins` are cleaned in only three and four batches, respectively (140 to 160 labels). These views are cleaned faster with View Impact than the `Select*` and `Count*` views because only a small number of tuples impacts these view. These are the only tuples with non-zero view impact scores and View Impact biases the selection of tuple-pairs for the user to label toward these tuples. In contrast, all tuples impact `Select*` and `Count*` views and do so equally, leading to a longer cleaning process. Most importantly, for all views, the View Impact Cleaning approach yields rapid improvements in view quality early on in the cleaning process. For the `Select*` view, our approach cuts the distance to the clean view by 4X after the initial classifier (from 0.47 to 0.11). The first subsequent batch cuts the distance by another 50%. In contrast, the Classifier Uncertainty and Entropy methods are unable to completely clean any views in 40 batches.

Figures 5.3 and 5.4(right) show the classifier accuracy (F1) scores achieved by all methods. It is difficult to build a quality classifier for the products dataset as evidenced by the

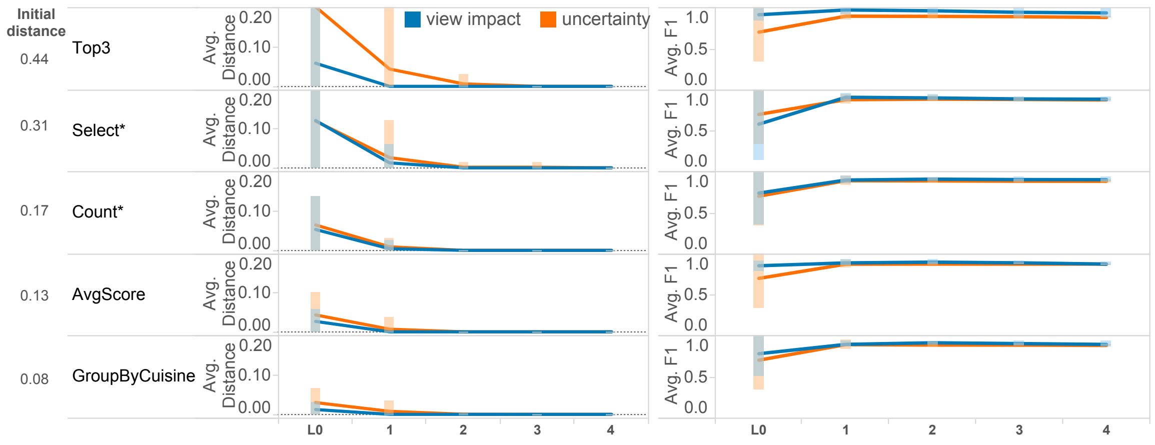


Figure 5.5: All restaurant views are completely cleaned after three batches with view impact and four batches with related work. View cleanliness (left) is shown with avg. $Distance(View_{curr}, View_{clean})$ +/- σ and (right) classifier accuracy with avg. F1. Related work is Uncertainty. The initial classifier is trained on 13 pairs and subsequent batches are of size 20 pairs.

low average F1 scores for all methods. The products dataset contains many data quality problems including missing and wrong values, which complicates feature selection. For example, *name* values were inconsistent even for matching pairs. We thus used this attribute for blocking but not for learning. We observe, however, that as expected the View Impact Cleaning method yields, on average, a classifier with a lower F1 score than the Uncertainty or Entropy methods. This method focuses on the quality of the view rather than the quality of the classifier itself.

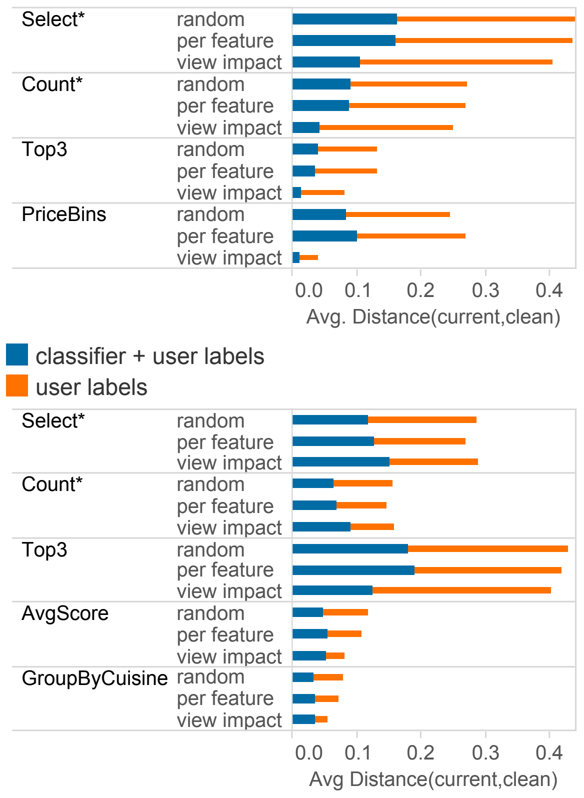
Interestingly, all views are cleaned monotonically with the View Impact Cleaning approach, while some aggregate views such as `PriceBins` exhibit non-monotonic behavior for the other methods. We see this undesirable behavior with Uncertainty and Entropy because they focus on selecting examples that improve the classifier's quality and not the view. Since the classifier's accuracy is low, it is unable to correctly label the pairs that impact the view. The View Impact Cleaning approach, in contrast, favors as training examples those pairs that have a high impact on the view. Since these labels are not the most informative, the classifier it learns is not as good as Uncertainty, but these labels are useful for cleaning the view.

Main result for restaurants. Figure 5.5(right) shows that all approaches exhibit higher average F1 scores on the holdoutset for restaurants than products, which suggests that duplicates in this dataset are much easier to classify. We thus expect that the results for cleaning with all approaches should be similar. We observe, in Figure 5.5(left), that the View Impact Cleaning method cleans all restaurant views in three batches, while Uncertainty needs four batches. Assuming that the initial classifier is learned over a batch of 13 pairs and subsequent batches contain 20 pairs, view impact can clean all views one batch faster than Uncertainty. Furthermore, for the `Top3` view, view impact only requested two batches (33 labels), while Uncertainty required two additional batches of 20. These results indicate that even when a good classifier can be learned with a small number of labeled examples, our technique does not hurt the quality of the view compared with Uncertainty.

5.5.2 Learning an initial classifier

We now study the individual components of the View Impact Cleaning approach. The first component of the approach is the selection of the initial training examples (see Section 5.3.2 for details). The selection occurs after both view-based and feature-based blocking.

We measure the quality of the view obtained after cleaning using the initial classifier learned with View Impact Cleaning. We compare the results to cleaning when using a classifier learned on a strictly random sample of the data taken also after both view-based and feature-based blocking. As discussed in Section 5.3.2 and as shown in Table 5.2, because the number of positive examples is extremely small compared with the number of negative examples, an initial classifier learned on a random data sample may have no positive examples to learn from. We thus also compare with a third approach that biases the selection of the training examples to select a larger fraction of positive examples. We call this last method *per-feature round-robin.* This approach sorts the tuple pairs by decreasing value of each of their features. It creates as many sorted lists as there are features and each pair



		Average F1		
Dataset	View	Random	Per feature	View impact
Products	Select*	29%	32%	27%
Products	Count*	29%	32%	28%
Products	Top3	29%	32%	28%
Products	PriceBins	29%	32%	17%
Restaurants	Select*	84%	84%	71%
Restaurants	Count*	84%	84%	65%
Restaurants	Top3	84%	84%	94%
Restaurants	AvgScore	84%	84%	73%
Restaurants	GroupByCuisine	84%	84%	74%

Figure 5.6: Impact of biasing the initial classifier on avg. Distance to the true clean view for products (top) and restaurants (bottom) and avg. F1 (table below). Initial L_0 batch size is 100 for products and 13 for restaurants.

appears once in each list. It then uses weighted sampling to select the pairs using the rank in the lists as weight.

Products. Figure 5.6 (top) shows the results for the four views over the Products dataset. As the figure shows, the View Impact Cleaning method yields the cleanest views after this initial cleaning step. Because View Impact Cleaning focuses on labeling and cleaning pairs with tuples that have high-impact on the view, an important question that arises is whether a classifier is at all useful or whether all the benefits come from the user labels. The figure also shows the quality of the view if we clean it using only the user labels. As the figure shows, with all three methods, building and using a classifier is critical to cleaning the view. Interestingly, the classifiers help to clean the view even though their average F1 accuracies are low for all sampling approaches (see Figure 5.6's table). This result implies that, for the purpose of quickly cleaning a view, it is not necessary to learn a high-quality classifier; rather it is more important to have the user resolve the most impactful tuples first, and train a classifier using these biased labels.

Restaurants. We see in Figure 5.6(bottom) that all sampling strategies have similar behaviors when cleaning the views for a small L_0 batch of 13 example pairs, which corresponds approximately to 3% of the data (a commonly used initial training set size [45]). Since the restaurants dataset is easier to classify, View Impact Cleaning does not have as much of an advantage as before. However, for three out of five views, view impact still produces a cleaner view than the other approaches.

5.5.3 Tuning the parameter settings

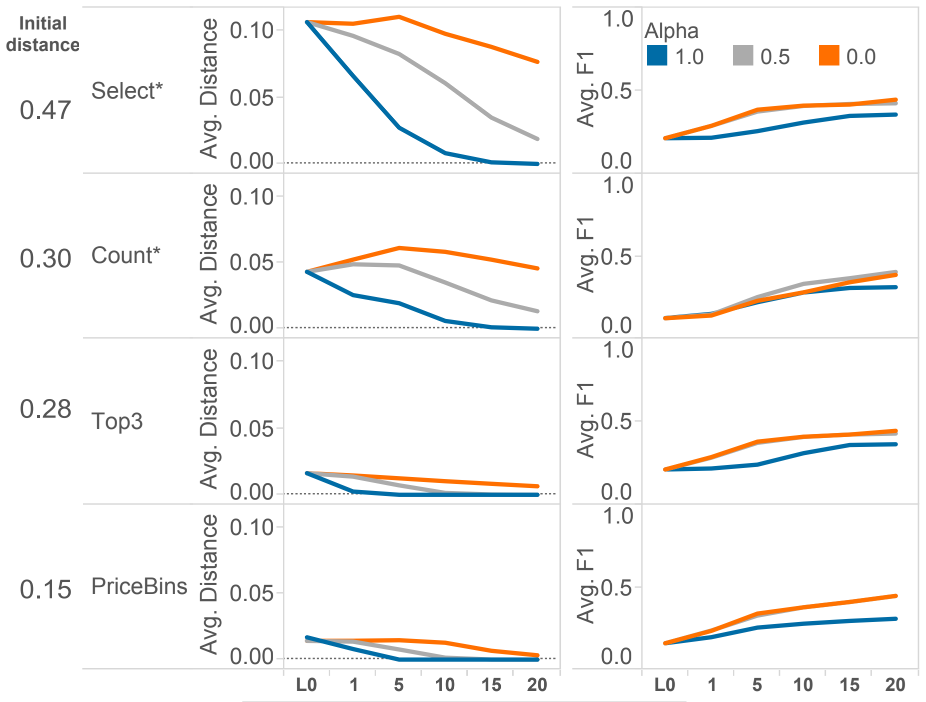
In this section, we study the effect of tuning various settings for the View Impact Cleaning and Uncertainty approaches. We first present the impact of weighting the two cleaning approaches using the α parameter in Equation 5.1 for the Hybrid method that combines View Impact Cleaning with ClassifierUncertainty. We also study the impact of varying the batch sizes.

α **values.** Since a good quality classifier can help save the user in cleaning effort, we study the effect of the weighting factor α in the Hybrid method described in Section 5.3.2. Recall that this method selects pairs for labeling by assigning them the following weight: $\alpha * ViewImpactScore + (1-\alpha) * ClassifierUncertainty$. We consider the two extremes: prioritizing pairs that will improve the classifier ($\alpha = 0$) and prioritizing pairs that impact the view ($\alpha = 1$). We also consider the hybrid method with $\alpha = 0.5$. Our analysis focuses on the products dataset, since the quality of its classifiers was much lower than those for restaurants. Furthermore, we zoom in to the details for two views, which exhibit very different sensitivities to duplicates. Other views showed similar trends.

View with higher relative sensitivity to dups. Figure 5.7 shows the result for the `Select*` view, which has the highest initial sensitivity to duplicates, 0.47. All initial classifiers are trained on 100 example pairs with View Impact Cleaning. The choice of α affects only subsequent batches. As the figure shows, the View Impact Cleaning approach (*i.e.,* $\alpha = 1.0$) is still able to make more progress cleaning than both the hybrid ($\alpha = 0.5$) and ClassifierUncertainty ($\alpha = 0.0$), despite having consistently lower overall classifier accuracy. In fact, View Impact Cleaning is the only technique that completely cleans this view within the budget of 500 labels (20 batches). This result suggests that heavily biasing the selection strategy toward the most impactful pairs is better for cleaning views that are highly sensitive to duplicates and defined over a dataset for which it is difficult to build a high quality classifier.

View with lower relative sensitivity to dups. Figure 5.7 shows the results for the `PriceBins` view, which is close to half as sensitive to duplicates as the `Select*` view. Once again, View Impact Cleaning outperforms the other approaches. It is able to completely deduplicate the `PriceBins` view by batch 5, neither of the other two approaches could to do so by the end of the 500 label budget (20 batches).

Tie breaking strategies. As described in Section 5.3.2, when selecting the next batch of example pairs for active learning, View Impact Cleaning uses the view impact score as sampling weights and the minimum margin distance to the classifier [118] as tie-breaker.



Stdev of Distance						
	$\alpha = 0.0$		$\alpha = 0.5$		$\alpha = 1.0$	
View	σ_{L_0}	σ_{20}	σ_{L_0}	σ_{20}	σ_{L_0}	σ_{20}
Select*	0.05	0.04	0.02	0.02	0.05	0
Count*	0.03	0.03	0.03	0.01	0.03	0
Top3	0.01	0.01	0.01	0	0.01	0
PriceBins	0.02	0.007	0.02	0.01	0.02	0

Figure 5.7: Impact of α on AVG $Distance(V_{curr}, V_{clean})$ and AVG F1 for product views with different sensitivities. Select* (top) has a high initial sensitivity to duplicates (0.47) and PriceBins view has a lower sensitivity (0.15). Initial classifier training set is 100 pairs. Subsequent batches contain 20 pairs each with a total budget of 500.

In this section, we compare the benefits of using margin distance as opposed to randomly breaking ties. Since the results for the views on restaurants exhibited the same behavior as for the products, we present the findings on products for this experiment.

Products. Figure 5.8 shows the results. Recall from Figure 5.4(right) that the quality of the classifiers learned for all the views on products dataset is low. Since the margin

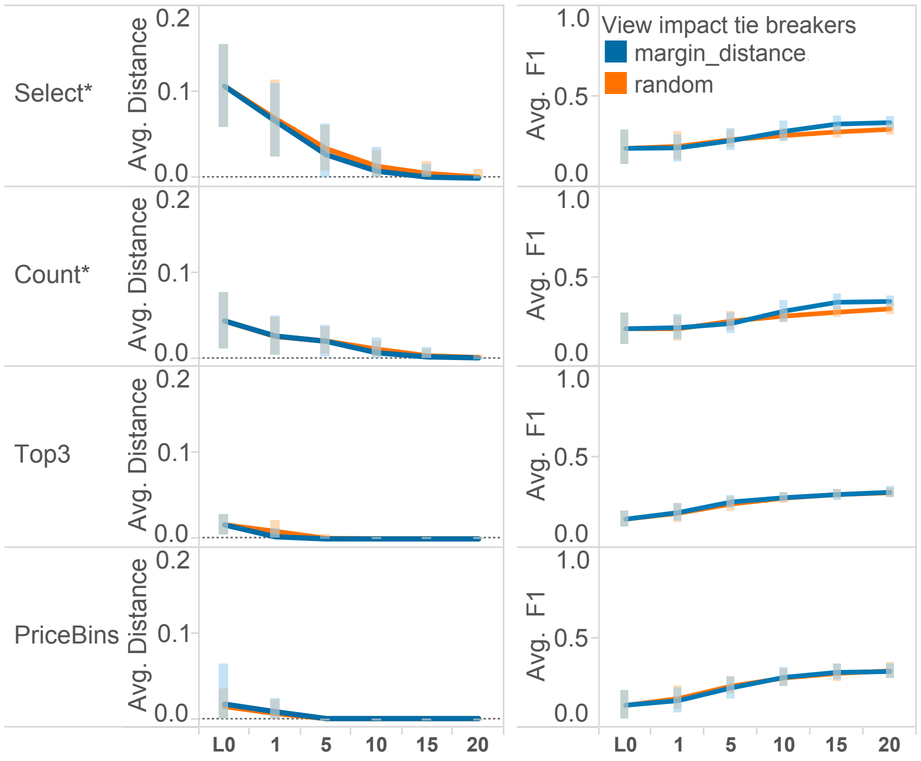


Figure 5.8: Impact of tie-breaking schemes on product views: AVG $Distance(V_{curr}, V_{clean})$ (left) +/- σ and AVG F1 (right)+/- σ. Initial classifier training set is 100 pairs. Subsequent batches contain 20 pairs each with a total budget of 500.

distance approach relies on selecting points that are closest to the classifier, it is sensitive to the classifier's accuracy. However, for all views, margin distance was either the fastest approach to cleaning or just as good as random. For the most sensitive view to duplicates, `Select*`, View Impact Cleaning with margin distance was able to completely clean the view with one batch less than View Impact Cleaning with random. For all other views with lower sensitivities such as `PriceBins`, the tie breaking strategies are able to clean the views within the same number of batches.

Batch size. We study the effect of cleaning views with different batch sizes (10, 20, 50, and 100 example pairs) and budgets (400 and 200) with View Impact Cleaning and Uncertainty.

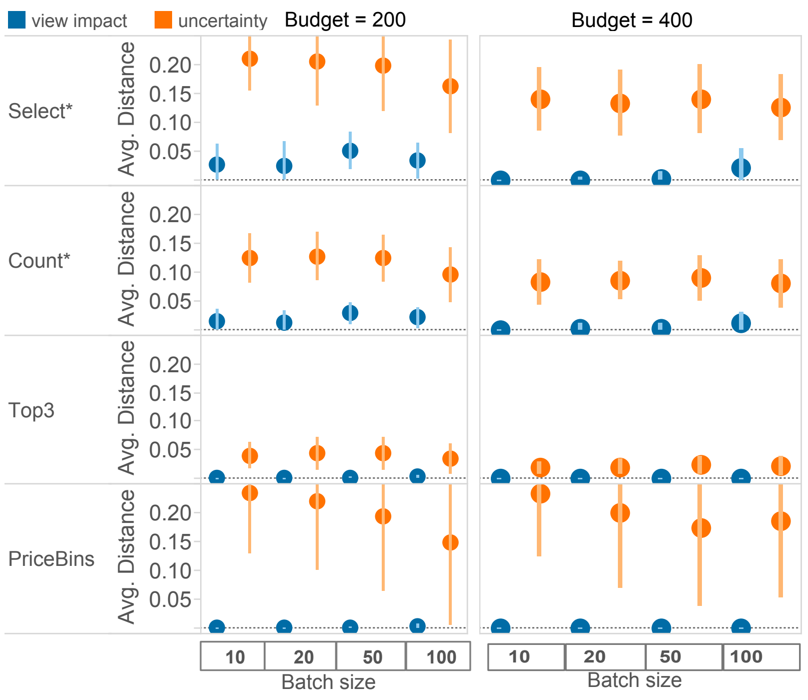


Figure 5.9: For all product views, we see the impact of batch size on average $Distance(V_{curr}, V_{clean})$ +/- σ with a budget of 200 (left) and budget of 400 (right).

We show the result for products in Figure 5.9. We observed similar results for restaurants. Overall, the batch size does not significantly influence the results. For all configurations, View Impact Cleaning is able to clean more than Uncertainty on average. Additionally, the variance for Uncertainty is much higher than for View Impact Cleaning. This result suggests that View Impact Cleaning is a more stable approach to deduplication and that it is not sensitive to the batch size.

5.5.4 Runtime and scalability

View Impact Cleaning complexity. There are two primary sources of computational complexity for the View Impact Cleaning algorithm. First, computing the feature vectors for all pairs is $O(n^2)$, where n is the input dataset. Second, computing the Distance as EMD in View Impact Scores (from Algorithm 1) takes worst-case $O(n \times m^2)$ because the EMD has $O(m^2)$ complexity where m is the size of the view [78] and can be called (worst-case) n times if $|Provenance(V(R))| = |R| = n$. Since computing the EMD grows quadratically with the view size, this approach works best with small views. Interestingly, a recent study of visualizations/views created on Tableau Public and Many Eyes [90] showed that 53% of views have fewer than 1,000 rows. We discuss the empirical findings next.

Empirical runtimes. We run View Impact Cleaning on a desktop machine with dual 2.4 GHz quad-core Intel Xeon processors and 11GB of memory. We use SQLite as our backend database to compute the feature vector table. We present the detailed measurement of runtimes for our approach on the view, `Select*` from products, as this view is the largest of all from Table 5.3 (with 291 rows) and takes the most time. We assume the same experiment settings as prior experiments on this view, where the initial L_0 batch has 100 pairs and subsequent batches have each 20 pairs. We time each of the key steps as follows: (1) Compute view impact scores for all tuples: three minutes, (2) Compute feature vector with four features with view blocking and feature blocking: three minutes (without feature blocking the time is 53 minutes) (3) Pick examples to label per batch: under one second, (4) Learn a new classifier per batch: under one sec, (5) Labels all pairs as either duplicates or not per batch: under three seconds.

As expected, steps (1) and (2) are the only steps that take a significant amount of time. To help with overall interactivity, these steps can be done as a background process while the user first explores the data. Interestingly, these two steps only need to be performed once before the cleaning process begins. Over the course of cleaning a view, the tuple view impact scores tend to not change.

5.5.5 *Stopping condition*

Recall that in practice V_{clean} is not known. We thus do not know exactly when to stop cleaning. The heuristic used is to stop after little to no progress has been made for some interval of time. All we can do is show empirically that this heuristic is effective. The Entropy method does this based on the stability of the confidence values of the classifier.

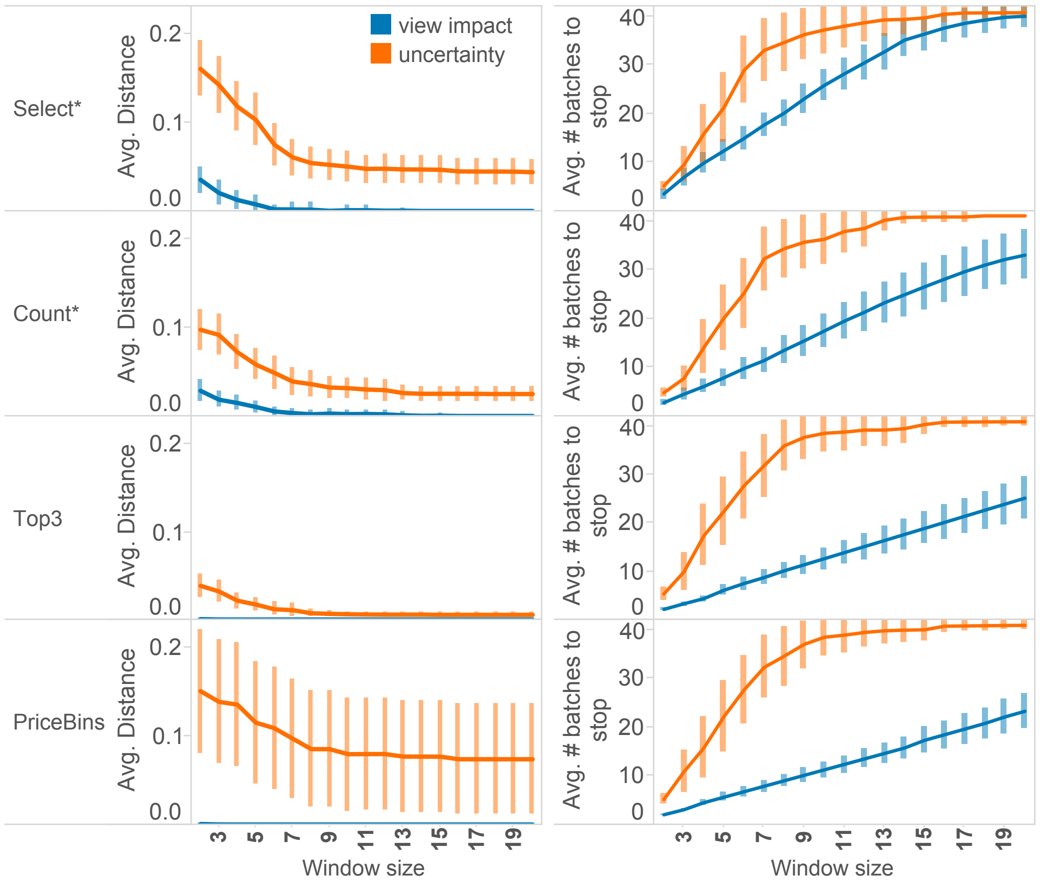


Figure 5.10: For product views, larger window sizes produce cleaner results (left is avg. Distance(V_{dirty},V_{clean})+/- σ). Smaller window sizes save the user in labeling effort (shown right), but result in a less than perfect clean view. View Impact Cleaning converges to a perfect clean view when the window size is 16 (18 - 37 batches of user labels). A window size of 7 is a good compromise. For all views, Uncertainty does not converge for any window size, as its classifier is too unstable. Initial L_0 is 100 pairs, subsequent batches contain 20 pairs. Total budget is 900 pairs.

Since the Uncertainty approach does not specify when learning can stop, we apply the same approach as used in the Entropy work to monitor the stability of the uncertainty values of the classifier. The View Impact Cleaning approach has a more natural and direct way to measure "little change" based on the $Distance(V_{curr},V_{prev})$. The idea is to stop cleaning once we observe that the distances computed between the current view, V_{curr}, and the view cleaned from the previous iteration, V_{prev}, have plateaued (within +/- $epsilon = 0.01$) over a window of size $n_{converge}$ batches. For the product views, shown in Figure 5.10, we evaluate the impact of the window size on the convergence to the true clean view. The figure shows the distance values when cleaning stops (left) and the corresponding labeling effort (right).

Given a much larger budget this time (900 labels, 41 batches total) and using the same $n_{converge}$ = window size = 20 batches and ϵ as reported in the Entropy evaluation section, Uncertainty still fails to converge to the true clean view for all product views. The result on the right indicates that the Uncertainty approach is unstable for a long time: all window sizes require many more batches to stop for Uncertainty than View Impact. This result suggests that the Uncertainty classifier learned is not stable enough to stop given even a large labeling budget of 900. For View Impact Cleaning, we see that a window size of 16 achieves the objective of converging to the true clean view (*i.e.,* Distance = 0) for all views. However, each view requires between 18 to 37 batches of labels given this window size. While this result is consistent with the time period in which the `Select*` and `Count*` views actually converge to the true clean view (see Figure 5.4), the `Top3` and `PriceBins` views require significantly less cleaning effort (only three to four batches). If the user is willing to trade off cleaning quality for effort, a window size of 7 would be an appropriate compromise, as half of the views are completely cleaned and the other half have a small average Distance, 0.002, from the true clean view (99% clean).

5.5.6 Multi-view deduplication

A dashboard is a collection of related visualizations or views typically over a common dataset. In this section, we study the performance of three techniques for data cleaning in the context of such dashboards.

We study three different orderings strategies of the set of four views on the products dataset. We focus on products rather than restaurants because it is a more challenging dataset to classify duplicates. We answer the question: is it faster to deduplicate each view at-a-time in isolation or collectively across all views?

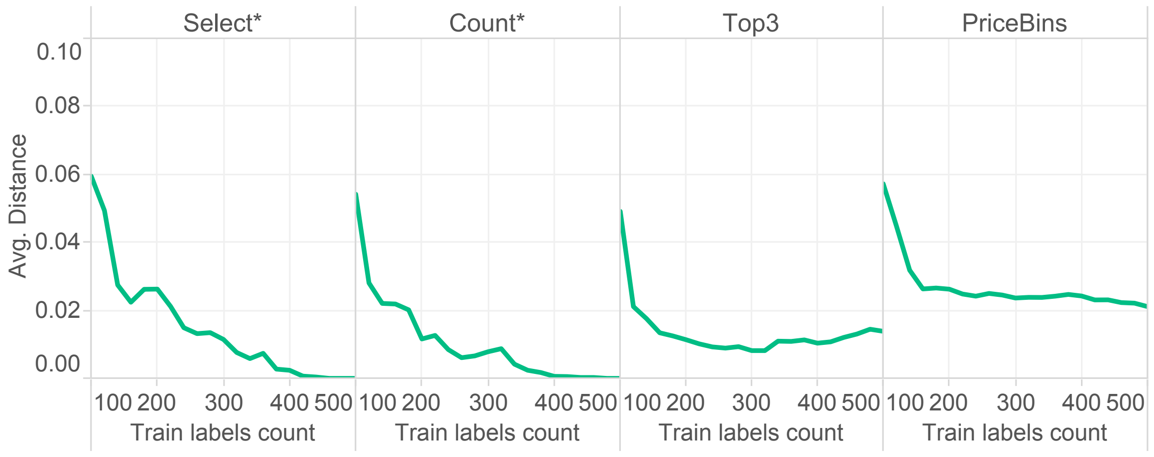


Figure 5.11: Result of fully cleaning each view in products on cleaning the other views given a budget of 500 labels: resolving the duplicates in the `Select*` view (far left) helps clean all the other views the fastest in 460 labels. However, the views are not cleaned monotonically. If we instead clean `Top3` or `PriceBins` first, we see smoother curves but the views are not cleaned as quickly as before. $Distance = Distance(V_{curr}, V_{clean})$ and all batches contain 20 pairs.

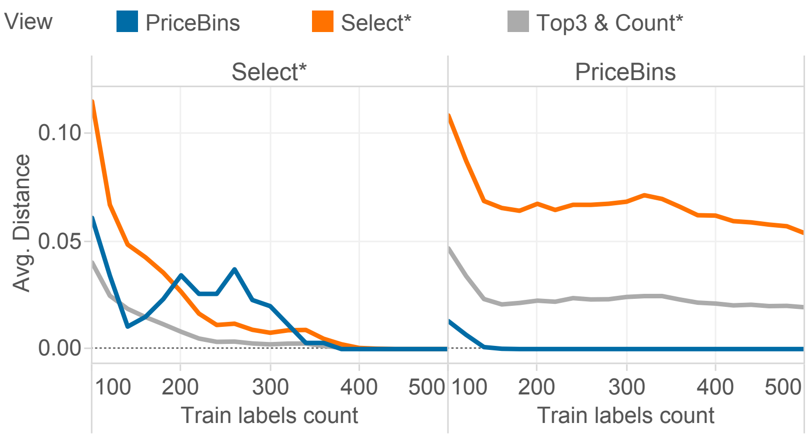


Figure 5.12: The most sensitive view first approach (left) monotonically cleans the other product views except for `PriceBins` with a budget of 500 user labels. Cleaning `Select*` cleans all other views the quickest (460 labels). If we instead clean `PriceBins` first, we see smoother curves but the views are not cleaned as quickly as before. $Distance = Distance(V_{curr}, V_{clean})$ and all batches contain 20 pairs.

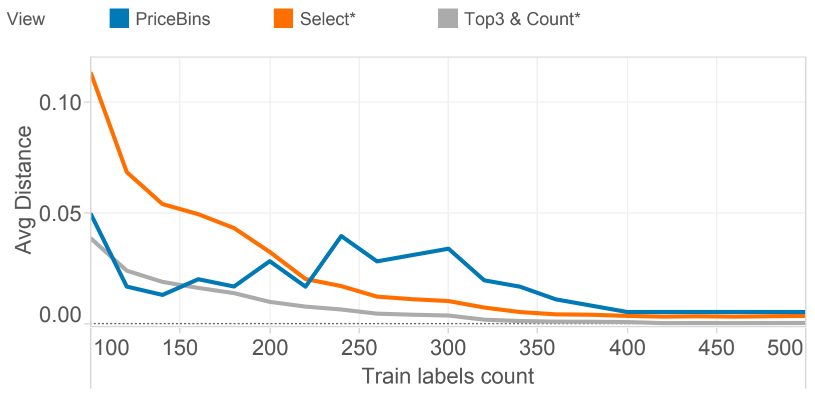


Figure 5.13: View Impact cleans 100 tuples from each view in a round-robin fashion: select the most impactful 100 tuples from each view based on sensitivity to duplicates. This approach cleans all of the other product views monotonically except for `PriceBins`, but requires a bigger budget (580 labels) than the previous approach to clean based on the most sensitive view first. $Distance = Distance(V_{curr}, V_{clean})$ and all batches contain 20 pairs.

(1) Fully clean one view at-a-time. We first study how much cleaning one view in a dashboard can help to clean the other views. Figure 5.11 shows the average distance, $Distance(V_{curr}, V_{clean})$, across all four views for products as we clean one of the four views only. As the figure shows, when cleaning one of the two views with the greatest sensitivity to duplicates, `Select*` and `Count*`, the most progress can be made on simultaneously cleaning the other views: `Select*` cleans all other views in 460 labels and `Count*` cleans them in 480 labels. In these views, all tuples have the same view impact scores and the cleaning process treats them all in the same way helping to clean all views the fastest. Interestingly, as shown in Figure 5.12, deduplicating the `Select*` view using View Impact Cleaning causes temporary, non-monotonic behavior in one view, `PriceBins`, which is possible given that the quality of the classifiers is low and subsequently learned classifiers may change how they classify the most impactful tuples for the `PriceBins` view.

Thus, cleaning one view helps to make progress on other views. However, in the context of cleaning an entire dashboard, the at-a-time method must be done with careful attention to the order in which the views are cleaned.

2) Partially clean one view at-a-time: round-robin view sensitivity ordering.

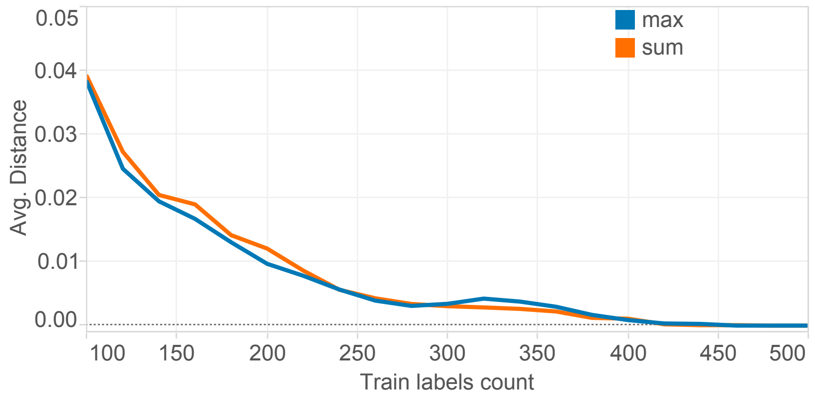


Figure 5.14: Clean all views simultaneously using MAX and SUM across all product views: MAX fully cleans all views together in 460 labels and SUM cleans all views in 480 labels (one batch later). SUM and MAX produce smoother curves for cleaning all views than the comparable first approach studied with `Select*` being cleaned. $Distance = Distance(V_{curr}, V_{clean})$ and all batches contain 20 pairs.

We select batches of the 100 most impactful pairs from each of the views in a round-robin fashion. The batches selected from each view are ordered by the initial overall view sensitivities to duplicates (*i.e.,* Distance(V_{dirty},V_{clean}) with the most sensitive one first: `Select*`, `Count*`, `PriceBins`, and `Top3`. We found that this approach required 100 more label requests than the previous approach (*i.e.,* just cleaning `Select*` fully) to clean all views and exhibited temporary, non-monotonic behavior cleaning the `PriceBins` view.

3) Clean across all views simultaneously using an aggregate measure of sensitivity across all views, MAX and SUM We also evaluate the performance of cleaning a dashboard of visualization as a single view. In this approach, the View Impact score for each tuple in the base relation is either the max or the sum of its impact across all the views. Figure 5.14 shows the results. The results are similar when using either MAX or SUM and the total number of labels required to clean all views is the same as cleaning just `Select*` or `count`. However, the curves for both MAX and SUM are smoother than when cleaning only `Select*` (Figure 5.11). This approach has the double benefit of yielding more stable results across batches and avoiding the problem of selecting which view to clean first.

Discussion. View-ordering is a challenging problem because different orderings have different effects on other views. For example, the strategy that cleaned just the `Select*` view produced non-monotonic behavior in cleaning `PriceBins`, a top-k view. Such non-monotonic behavior can happen in cleaning views that, for example, require an ordering of a small subset of k rows (*e.g.,* any top-k view queries) since the cost of a mistake for a single row is a much higher proportion of the Distance score. For such views, some duplicate tuples may not impact the view at all, while others have a much higher impact. However, for views like `Select*` and `Count*`, each tuple has a similar impact on the view and thus has a small proportion of the Distance score. One challenge we address is to make the View Impact Cleaning approach more robust to of any inter-view ordering constraints that can slow down the progress of cleaning across the views. One idea presented to address this challenge is to order the batches of pairs by their aggregate overall impact (MAX or SUM) across each view being cleaned. This approach achieves the fastest cleaning result of all, only requiring 460 labels to clean the four views collectively.

5.6 Related work

Deduplication has a long history in the literature (see [34, 44]). The state-of-the-art deduplication approaches that are closely related to this work fall into the following categories:
Active learning. Active learning systems for record deduplication [7, 9, 45, 93] focus on cleaning an entire dataset at-a-time. Our work also uses active learning but our focus is doing a minimum amount of work (in the form of user labels) to clean one or more specific views over the dirty data rather than focusing on producing the cleanest base data. All of these systems (except Corleone [45]) require a developer/expert to manage the common learning tasks such as writing the blocking rules. Corelone pushes this expert work to the crowd. In our work, we do not focus on automating the generation of the blocking rules and our approach could be extended with such techniques. Active learning methods carefully select additional training examples at each iteration that are most informative. Common methods to measure the informativeness of training examples try to measure the disagreement of the component classifiers using uncertainty [93] or entropy [45]. In contrast, we use our new notion of View Impact for sampling the initial set and selecting additional

training examples. As a result, while related systems often require the user to provide thousands of labels to clean entire datasets, we show that View Impact Cleaning can yield clean views with only tens to hundreds of labels.

Using crowd workers. Several entity resolution systems rely on feedback from a set of crowd workers (who may provide incorrect or conflicting labels). These systems strive to limit the number of unnecessary label requests to the crowd of resolving duplicates [45, 93, 122, 127]. In contrast, we focus on a single data enthusiast performing the cleaning in the context of his or her data exploration task.

Passive learning. Passive learners for deduplication can be found in the databases literature [23, 73] and in the information retrieval literature [52, 72]. However, we showed in Section 5.5.2 that passive learners are insufficient in completely cleaning any of the views in one shot.

Incremental deduplication. Some techniques focus on the problem of deduplicating newly inserted records once an original dataset had been deduplicated [50, 131, 134]. In contrast, we focus only on the initial data deduplication problem.

Clustering. Several deduplication approaches consider the setting where each tuple can match multiple other tuples [2, 12, 129, 122, 130]. They either leverage the transitive property of the match relation [2, 129, 122] or correlation clustering [12, 130] to infer matching and non-matching pairs based on previously labeled pairs and reduce the labeling effort by users. These approaches are complementary to ours and could be added to our method to further speed-up view cleaning.

5.7 Conclusions

We proposed an active learning algorithm for deduplicating records in an exploratory visual analytic systems, which strives to produce the cleanest view possible within a limited budget. Our key idea is to consider the impact that individual tuples have on a visualization and to monitor how the view changes during cleaning. We demonstrated over a set of nine views that our approach produces significantly cleaner views for small labeling budgets than state-of-the-art alternatives and that it also stops the cleaning process after requesting fewer labels.

View	View SQL	View Description	Rows	% Rows affected by dups	Initial Distance (V_{dirty}, V_{clean})
SFrestaurants: Top3	SELECT cuisine, COUNT(*) FROM SFrestaurantsSelect* GROUP BY cuisine ORDER BY COUNT(*) DESC LIMIT 3	Top 3 restaurants in San Francisco by type of cuisine	3	33	0.44
SFrestaurants: Select*	SELECT * FROM restaurants WHERE city = 'SF'	All restaurants in San Francisco	148	12	0.31
SFrestaurants: Count*	SELECT COUNT(*) FROM SFrestaurantsSelect*	Count of restaurants in San Francisco	1	100	0.17
SFrestaurants: JoinAvgScore	SELECT cuisine, AVG(score) FROM SFrestaurantsSelect-scores GROUP BY cuisine	Restaurants by cuisine & AVG inspection score from the San Francisco Health Department's restaurant inspection scores DB [104]	29	31	0.13
SFrestaurants: GroupByCuisine	SELECT cuisine, COUNT(*) FROM SFrestaurantsSelect* GROUP BY cuisine	A histogram-like view of restaurants in San Francisco grouped by cuisine	29	31	0.08
MfrProducts: Select*	SELECT * FROM products WHERE name LIKE '%Apple%' OR name LIKE '%Microsoft%' OR name LIKE '%Adobe%'	Products manufactured by Apple, Microsoft, or Adobe	291	27	0.47
MfrProducts: Count*	SELECT COUNT(*) FROM MfrProductsSelect*	Count of products manufactured by Apple, Microsoft, and Adobe	1	100	0.30
MfrProducts: PriceBins	SELECT mfr, CASE WHEN price < 10 then 'Bin 1: [0,10)' WHEN price <100 then 'Bin 2: [10,100)' WHEN price < 1000 then 'Bin 3: [100,1000)' ELSE 'Bin 4: 1000+' END AS priceRange, COUNT(*) FROM MfrProductsSelect* GROUP BY mfr, priceRange ORDER BY mfr ASC, priceRange ASC LIMIT 5	For each manufacturer, tally the products in various price ranges limited to the first 5 groupings	5	20	0.28
MfrProducts: Top3	SELECT mfr, COUNT(*) as cnt FROM MfrProductsSelect* GROUP BY mfr ORDER BY cnt DESC LIMIT 3	Top 3 manufacturers sorted on total count in descending order	3	33	0.15

Table 5.3: Views studied on restaurants and products: view sizes, fraction of rows impacted by duplicate entities, and initial view sensitivities to duplicates using $Distance(View_{dirty}, View_{clean})$ where $View_{clean}$ is the true clean view.

Chapter 6

CONCLUSIONS AND FUTURE WORK

Whether in industry or science, everyone today has a big data problem. While many tools are being developed to support expert data scientists with the management and analysis of this data, another class of users, data enthusiasts, are currently under-served. Data enthusiasts are ordinary people who need to manage and analyze data but do not have the skills of professional data scientists.

In this dissertation, we presented the following three contributions to the state-of-the-art in supporting data enthusiasts with their data analysis tasks:

1) Usage study of two visual data analytics systems. In Chapter 4 we provided the first study of how two popular visual analytics systems, Tableau and Many Eyes, were being used. Our study focused on answering the following core set of questions: (1) How popular are these systems? How many users do they attract and how active are these users? (2) How heavily do users leverage the collaborative features of these tools? (3) What do users actually do with the data? How do they analyze it? How much data (in terms of relation cardinality and degree) do users choose to visualize at any given time? And finally (4) Do users integrate multiple data sources in their visualizations? And how do they perform these integrations? The results from this study informed our next contribution, cleaning data in the context of a visual analytics environment.

2) View-driven data cleaning. In the context of visual analytics systems, we developed a new approach to deduplication that builds on the active learning literature. We call our approach View Impact Cleaning (in Chapter 5). Our method cleans only the data that impacts the view (or set of views) and only if the view is sensitive to duplicates. We developed a new measure of view sensitivity to duplicates and consider the impact of each tuple on the view (*i.e.,* the *view impact* of each tuple) during the cleaning process. Our active learning method biases the selection of training examples based on view impact. We also developed a new stopping condition for the algorithm: stop cleaning when the view (or set of views) is no longer sensitive to duplicates. We showed in our evaluation over nine views and two real-world data sets that this method cleans views faster than the state-of-the-art work.

3) Data blending. Finally, in Chapter 2 we presented a simple approach for interactive data integration in Tableau.

Future work: classifying duplicates. There are three primary pain-points in applying an active-learning based approach to identify duplicate entities in a database. Active learning builds a classifier that takes pairs of records as input and labels each pair as either a duplicate or not. First, in order to build a good quality classifier, the learner needs to have a good set of features that can help separate the two classes (duplicates and non-duplicates). This is a well-known problem (called *feature engineering*) and requires some technical expertise, which we cannot assume for our cohort of users, data enthusiasts. Moreover an additional pain-point is in specifying the appropriate initial parameter settings (*e.g.,* which kernel to use), which are specific to the dataset being classified and hard to generalize to other data sets. Finally, the classifier quality is subject to the class imbalance problem and the state-of-the-art approach is to apply blocking rules to boost the fraction of the under-represented class (*i.e.,* duplicates). There are currently no tools that can help automate any of the aforementioned pain-points.

Future work: data enrichment for visual analytics. While many data sources are available on the Web or (more conveniently) shared by other users of the visual analytics service, identifying interesting data to enrich a visualization is challenging. Different datasets have different schemas, different levels of granularity (*e.g.,* we may have state-level unemployment data but zip code-level income data), or different levels of cleanliness. They may also contain different subsets of relevant data. Next-generation visual analytics services should help users *identify* datasets that they can potentially leverage for their current data analysis task. The recommendation needs to take into account the visualizations that the user is creating and could create and not just the underlying data.

Future work: a common formalism. *Most importantly, the following capabilities should be seamlessly combined into a unified framework to support data enthusiasts and their workflows: data cleaning, data recommendation, data integration, and visual exploration.* To the user, it should be a visualization system that enables jumping among the tasks of exploring data, finding new data, integrating data, and cleaning data in a consistent, integrated fashion. Current systems require a mental context switch every time a user needs to integrate another data source by forcing the user to deal with the details of cleaning and transforming it. Presently, no single analytics system can accomplish this task: users must find and learn to use separate secondary tools and then manually reload the cleaned results into the primary analytics tool. We want to avoid these expensive context- and tool-switches.

BIBLIOGRAPHY

[1] Christopher Ahlberg. Spotfire: An Information Exploration Environment. *Sigmod Record*, 25:25–29, 1996.

[2] Hotham Altwaijry, Dmitri V Kalashnikov, and Sharad Mehrotra. Query-driven approach to entity resolution. *VLDB*, 2013.

[3] Amazon-Google Dataset. `http://dbs.uni-leipzig.de/file/Amazon-GoogleProducts.zip`, 2015.

[4] Michael Anderson, Dolan Antenucci, Victor Bittorf, Matthew Burgess, Michael J Cafarella, Arun Kumar, Feng Niu, Yongjoo Park, Christopher Ré, and Ce Zhang. Brainwash: A data system for feature engineering. In *CIDR*, 2013.

[5] Apache Hadoop. `https://hadoop.apache.org/`.

[6] Apache Spark. `http://spark.apache.org/`.

[7] Arvind Arasu, Michaela Götz, and Raghav Kaushik. On active learning of record matching packages. In *SIGMOD*, 2010.

[8] Arvind Arasu, Chris Re, and Dan Suciu. Large-scale deduplication with constraints using dedupalog. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 952–963. IEEE, 2009.

[9] Kedar Bellare, Suresh Iyengar, Aditya G. Parameswaran, and Vibhor Rastogi. Active sampling for entity matching. In *SIGKDD*, 2012.

[10] T. Berners-Lee. The year open data went worldwide. The Huffington Post, `http://www.huffingtonpost.com/tedtalks/tim-berners-lee-the-year_b_490726.html`, 2010.

[11] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th Annual International Conf. on Machine Learning.* ACM, 2009.

[12] Indrajit Bhattacharya and Lise Getoor. Query-time entity resolution. *Journal of Artificial Intelligence Research*, pages 621–657, 2007.

[13] M. Bilenko, B. Kamath, and R.J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *Data Mining, 2006. ICDM '06. Sixth International Conference on*, pages 87–96, Dec 2006.

[14] Mustafa Bilgic, Louis Licamele, Lise Getoor, and Ben Shneiderman. D-dupe: An interactive tool for entity resolution in social networks. In *IEEE Symposium On Visual Analytics Science And Technology*. IEEE, 2006.

[15] A. Black. Open data movement: Where it's been and where it appears to be going. `http://icma.org/pm/`, 2012.

[16] Jens Bleiholder and Felix Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1:1–1:41, January 2009.

[17] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[18] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. Why and where: A characterization of data provenance. In *ICDT*. Springer, 2001.

[19] Michael J Cafarella, Alon Halevy, Zhe Daisy Wang, Eugene Wu, and Yang Zhang. Webtables: Exploring the power of tables on the web. In *Proceedings VLDB*, pages 538–549, 2008.

[20] S. Card, G. Robertson, and J. Mackinlay. The information visualizer, an information workspace. In *CHI*, 1991.

[21] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. Using Vision to Think. In *Readings in Information Visualization*. Morgan Kaufmann, 1999.

[22] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[23] Surajit Chaudhuri, Bee-Chung Chen, Venkatesh Ganti, and Raghav Kaushik. Example-driven design of efficient record matching queries. In *Proceedings of the 33rd international conference on Very large data bases*, pages 327–338. VLDB Endowment, 2007.

[24] Peter Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection.* Springer Science & Business Media, 2012.

[25] Cisco Data Federation. `http://www.compositesw.com/data-virtualization/data-federation/`.

[26] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *SIGKDD*, volume 3, 2003.

[27] C. Danis., F. Viegas, and M. Wattenberg andJ. Kriss. Your place or mine?: visualization as a community component. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 275–284, 2008.

[28] Anish Das Sarma, Xin Dong, and Alon Halevy. Bootstrapping Pay-As-You-Go Data Integration Systems. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of Data*, SIGMOD '08, pages 861–874, 2008.

[29] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. Finding Related Tables. In *SIGMOD*, 2012.

[30] Tamraparni Dasu and Theodore Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley & Sons, New York, NY, 2003.

[31] Dato: Fast, Scalable Machine Learning Platform. `https://dato.com/`.

[32] DBLP. `http://dblp.uni-trier.de`.

[33] Debabrata Dey, Sumit Sarkar, and Pradipta De. Entity matching in heterogeneous databases: a distance-based decision model. In *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*, volume 7, pages 305–313. IEEE, 1998.

[34] AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[35] Earth Mover's Distance Library. `http://robotics.stanford.edu/~rubner/emd/default.htm`, 2015.

[36] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*, volume 57. CRC press, 1994.

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

[38] Ivan P Fellegi and Alan B Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[39] Michael Franklin, Alon Halevy, and David Maier. From Databases to Dataspaces: A New Abstraction for Information Management. *SIGMOD Rec.*, 34(4):27–33, December 2005.

[40] Freegeoip. `http://freegeoip.net`.

[41] Ariel Fuxman, Mauricio A. Hernandez, Howard Ho, Renee J. Miller, Paolo Papotti, and Lucian Popa. Nested mappings: schema mapping reloaded. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 67–78, 2006.

[42] John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the Future*, 2007:1–16, 2012.

[43] Gapminder. `http://www.gapminder.org/`.

[44] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: theory, practice & open challenges. *PVLDB*, 5(12), 2012.

[45] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojin Zhu. Corleone: Hands-off crowdsourcing for entity matching. In *SIGMOD*, 2014.

[46] Hector Gonzalez, Alon Halevy, Christian S Jensen, Anno Langen, Jayant Madhavan, Rebecca Shapley, and Warren Shen. Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud. In *SOCC*, 2010.

[47] Hector Gonzalez, Alon Y Halevy, Christian S Jensen, Anno Langen, Jayant Madhavan, Rebecca Shapley, Warren Shen, and Jonathan Goldberg-Kidon. Google fusion tables: Web-centered data management and collaboration. In *SIGMOD*, 2010.

[48] Alvaro Graves and James Hendler. Visualization tools for open government data. In *Proc. of the 14th International Conf. on Digital Government Research*, 2013.

[49] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.

[50] Anja Gruenheid, Xin Luna Dong, and Divesh Srivastava. Incremental record linkage. *PVLDB*, 7(9), 2014.

[51] Sudipto Guha, Nick Koudas, Amit Marathe, and Divesh Srivastava. Merging the results of approximate match operations. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 636–647. VLDB Endowment, 2004.

[52] Hannaneh Hajishirzi, Wen-tau Yih, and Aleksander Kolcz. Adaptive near-duplicate detection via similarity learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 419–426. ACM, 2010.

[53] P. Hanrahan. Analytic database technologies for a new kind of user: the data enthusiast. In *SIGMOD*, 2012.

[54] Pat Hanrahan. VizQL: A Language for Query, Analysis and Visualization. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 721–721, 2006.

[55] William R. Harris and Sumit Gulwani. Spreadsheet Table Transformations From Examples. In *PLDI 2011*, 2011.

[56] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *TKDE*, 21(9):1263–1284, 2009.

[57] Jeffrey Heer and Maneesh Agrawala. Design considerations for collaborative visual analytics. In *Proc. IEEE VAST*, 2007.

[58] Jeffrey Heer, Fernanda B Viégas, and Martin Wattenberg. Voyagers and Voyeurs: Supporting Asynchronous Collaborative Information Visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1029–1038. ACM Press, 2007.

[59] Martin Hentschel, Laura Haas, and Renée J. Miller. Just-In-Time Data Integration in Action. *Proc. VLDB Endow.*, 3(1-2):1621–1624, September 2010.

[60] David Huynh, Stefano Mazzocchi, and David Karger. Piggy bank: Experience the semantic web inside your web browser. In *Proc. of ISWC*, 2005.

[61] D. Huynhand and S. Mazzocchi. Google Refine. `http://code.google.com/p/google-refine/`.

[62] IBM InfoSphere Federation Server. `http://www-03.ibm.com/software/products/en/ibminfofedeserv`.

[63] IBM Watson Analytics. `http://www-969.ibm.com/software/analytics/manyeyes/`.

[64] iCharts. `http://www.icharts.net/`.

[65] Infochimps: Smart Data for Apps and Analytics. `http://www.infochimps.com/`.

[66] Zachary G Ives, Todd J Green, Grigoris Karvounarakis, Nicholas E Taylor, Val Tannen, Partha Pratim Talukdar, Marie Jacob, and Fernando Pereira. The orchestra collaborative data sharing system. *ACM SIGMOD Record*, 37(3):26–32, 2008.

[67] Zachary G. Ives, Craig A. Knoblock, Steven Minton, Marie Jacob, Partha Pratim Talukdar, Rattapoom Tuchinda, José Luis Ambite, Maria Muslea, and Cenk Gazen. Interactive data integration through smart copy & paste. In *CIDR*, 2009.

[68] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive Visual Specification of Data Transformation Scripts. In *CHI*, 2011.

[69] Sean Kandel, Ravi Parikh, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. Profiler: Integrated Statistical Analysis and Visualization for Data Quality Assessment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 547–554, 2012.

[70] Eser Kandogan. Just-in-time annotation of clusters, outliers, and trends in point-based data visualizations. In *VAST*, 2012.

[71] Hyunmo Kang, Vivek Sehgal, and Lise Getoor. Geoddupe: a novel interface for interactive entity resolution in geospatial data. In *11th International Conference on Information Visualization*. IEEE, 2007.

[72] Alistair Kennedy and Stan Szpakowicz. A supervised method of feature weighting for measuring semantic relatedness. In *Advances in Artificial Intelligence*, pages 222–233. Springer, 2011.

[73] Hanna Köpcke and Erhard Rahm. Training selection for tuning entity matching. In *QDB/MUD*, pages 3–12, 2008.

[74] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, 3(1-2), 2010.

[75] Robert Kosara, Brent Fitzgerald, Hans Rosling, Warren Sack, and Fernanda B Viégas. Panel: The Impact of Social Data Visualization. In *IEEE Visualization Conference Compendium*, pages 128–130. IEEE CS Press, 2007.

[76] Sanjay Krishnan, Jiannan Wang, Michael J Franklin, Ken Goldberg, and Tim Kraska. Stale View Cleaning: Getting Fresh Answers from Stale Materialized Views. In *PVLDB*, 2015.

[77] YongChul Kwon, Guiping Xu, and Magdalena Balazinska. Identifying similar past events in a continuous monitoring system. *VLDB*, 2007.

[78] Haibin Ling and Kazunori Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):840–853, 2007.

[79] Zhicheng Liu, Biye Jiang, and Jeffrey Heer. immens: Real-time visual querying of big data. In *Computer Graphics Forum*, volume 32, pages 421–430. Wiley Online Library, 2013.

[80] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and K. Wenger. DEVise: Integrated Querying and Visual Exploration of Large Datasets. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, SIGMOD '97, pages 301–312, New York, NY, USA, 1997. ACM.

[81] Kurt Luther, Scott Counts, Kristin B. Stecher, Aaron Hoff, and Paul Johns. Pathfinder: an online collaboration environment for citizen scientists. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 239–248, New York, NY, USA, 2009. ACM.

[82] S. Maine, L. Prem, C. Szyperski, J. Terwilliger, and the Microsoft "Montego" Team. Microsoft Codename "Montego" – Data Import, Transformation, and Publication for Information Workers. In *Proceedings of the VLDB Endowment*, volume 4, pages 1454–1457, 2011.

[83] Many Eyes. `http://many-eyes.com/`.

[84] D. Martin. Twitter Quitters Post Roadblock to Long-Term Growth. `http://blog.nielsen.com/nielsenwire/online_mobile/twitter-quitters-post-roadblock-to-long-term-growth/`, 2009.

[85] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[86] Microsoft PowerPivot. `http://www.powerpivot.com`.

[87] Microsoft Windows Azure Marketplace. `https://datamarket.azure.com/`.

[88] R. Miller. Response Time in Man-Computer Conversational Transactions. In *AFIPS Fall Joint Computer Conf.*, 1968.

[89] Kristi Morton, Magdalena Balazinska, Dan Grossman, Robert Kosara, and Jock Mackinlay. A Measurement Study of Two Web-based Collaborative Visual Analytics Systems. Technical Report UW-CSE-12-08-01, U. of Washington, Aug 2012.

[90] Kristi Morton, Magdalena Balazinska, Dan Grossman, Robert Kosara, and Jock Mackinlay. Public data and visualizations: How are many eyes and tableau public used for collaborative analytics? *ACM SIGMOD Record*, 43(2), 2014.

[91] Kristi Morton, Magdalena Balazinska, Dan Grossman, and Jock Mackinlay. Support the data enthusiast: Challenges for next-generation data-analysis systems. *VLDB*, 7(6), 2014.

[92] Kristi Morton, Ross Bunker, Jock Mackinlay, Robert Morton, and Chris Stolte. Dynamic Workload Driven Data Integration in Tableau. In *SIGMOD*, 2012.

[93] Barzan Mozafari, Purna Sarkar, Michael Franklin, Michael Jordan, and Samuel Madden. Scaling up crowd-sourcing to very large datasets: A case for active learning. *VLDB*, 8(2), 2015.

[94] F. Nah. A Study on Tolerable Waiting Time: How Long Are Web Users Willing to Wait? In *Behaviour and Information Technology*, volume 23, 2004.

[95] C. Olston, M. Stonebraker, A. Aiken, and J. M. Hellerstein. Viqing: Visual interactive querying. In *Proceedings of the IEEE Symposium on Visual Languages*, VL '98, pages 162–, Washington, DC, USA, 1998. IEEE Computer Society.

[96] Open Data Protocol. `http://www.odata.org/`.

[97] Lucian Popa, Yannis Velegrakis, Mauricio A. Hernández, Renée J. Miller, and Ronald Fagin. Translating web data. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 598–609, 2002.

[98] Qlikview. `http://www.qlikview.com`.

[99] Alessandro Raffio, Daniele Braga, Stefano Ceri, Paolo Papotti, Mauricio Hernández, et al. Clip: a Visual Language for Explicit Schema Mappings. In *ICDE*, 2008.

[100] Vijayshankar Raman and Joseph M. Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. In *Very Large Data Bases*, pages 381–390, 2001.

[101] RIDDLE Dataset. `http://www.cs.utexas.edu/users/ml/riddle`, 2015.

[102] S. F. Roth, P. Lucas, J. A. Senn, C. C. Gomberg, M. B. Burks, P. J. Stroffolino, A. J. Kolojechick, and C. Dunmire. Visage: A User Interface Environment for Exploring Information. In *Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*, INFOVIS '96, pages 3–, Washington, DC, USA, 1996. IEEE Computer Society.

[103] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[104] San Francisco Dept. of Public Health Food Safety Program. `http://www.sfdph.org/dph/EH/Food/Inspections.asp`, 2015.

[105] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[106] Socrata. http://www.socrata.com/.

[107] Spotfire. http://spotfire.tibco.com/.

[108] Rebecca C Steorts, Samuel L Ventura, Mauricio Sadinle, and Stephen E Fienberg. A comparison of blocking methods for record linkage. In *Privacy in Statistical Databases*, pages 253–268. Springer, 2014.

[109] Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional databases. *IEEE TVCG*, 8(1), 2002.

[110] Christopher Richard Stolte. *Query, Analysis, and Visualization of Multidimensional Databases.* PhD thesis, Stanford University, Stanford, CA, USA, 2003.

[111] 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.

[112] Tableau Public. `http://www.tableaupublic.com/`.

[113] Tableau Software. `http://www.tableau.com/`.

[114] The Guardian Datablog. `http://www.guardian.co.uk/news/datablog/2011/mar/17/visualise-data-trends#data`, 2012.

[115] The New York Times. `http://www.nytimes.com/2012/01/17/science/open-science-challenges-journal-tradition-with-web-collaboration.html?_r=3`, 2012.

[116] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

[117] Trifacta. `http://www.trifacta.com`.

[118] Ming-Hen Tsai, C-H Ho, and Chih-Jen Lin. Active learning strategies using svms. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2010.

[119] Rattapoom Tuchinda, Pedro Szekely, and Craig A. Knoblock. Building mashups by example. In *IUI*, 2008.

[120] Frank van Ham and Bernice Rogowitz. Perceptual organization in user-generated graph layouts. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1333–1339, 2008.

[121] Frank van Ham, Martin Wattenberg, and Fernanda B. Viegas. Mapping text with phrase nets. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1169–1176, 2009.

[122] Norases Vesdapunt, Kedar Bellare, and Nilesh Dalvi. Crowdsourcing algorithms for entity resolution. *VLDB*, 7(12), 2014.

[123] Fernanda B. Viegas, Martin Wattenberg, and Jonathan Feinberg. Participatory visualization with wordle. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1137–1144, 2009.

[124] Fernanda B Viégas, Martin Wattenberg, Matt Mckeon, Frank Van Ham, and Jesse Kriss. Harry Potter and the Meat-Filled Freezer: A Case Study of Spontaneous Usage of Visualization Tools. In *Hawaii International Conference on System Sciences*, pages 159–168, 2008.

[125] Fernanda B Viegas, Martin Wattenberg, Frank Van Ham, Jesse Kriss, and Matt McKeon. Many eyes: A site for visualization at internet scale. *IEEE TVCG*, 13(6), 2007.

[126] ViewShare: Interfaces to our Heritage. `http://viewshare.org/`.

[127] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 2012.

[128] 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 *SIGMOD*, pages 469–480. ACM, 2014.

[129] Jiannan Wang, Guoliang Li, Tim Kraska, Michael J Franklin, and Jianhua Feng. Leveraging transitive relations for crowdsourced joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 229–240. ACM, 2013.

[130] Sibo Wang, Xiaokui Xiao, and Chun-Hee Lee. Crowd-based deduplication: An adaptive approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1263–1277. ACM, 2015.

[131] Michael J Welch, Aamod Sane, and Chris Drome. Fast and accurate incremental entity resolution relative to an entity knowledge base. In *CIKM*, pages 2667–2670. ACM, 2012.

[132] Richard Wesley, Matthew Eldridge, and Pawel T Terlecki. An Analytic Data Engine for Visualization in Tableau. In *SIGMOD*, 2011.

[133] Steven Euijong Whang and Hector Garcia-Molina. Entity resolution with evolving rules. *PVLDB*, 3(1-2), 2010.

[134] Steven Euijong Whang and Hector Garcia-Molina. Incremental entity resolution on rules and data. *PVLDB*, 23(1):77–102, 2014.

[135] Steven Euijong Whang, Peter Lofgren, and Hector Garcia-Molina. Question selection for crowd entity resolution. *VLDB*, 6(6), 2013.

[136] Steven Euijong Whang, David Marmaros, and Hector Garcia-Molina. Pay-as-you-go entity resolution. *TKDE*, 25(5):1111–1124, 2013.

[137] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. Strategies for crowdsourcing social data analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 227–236, New York, NY, USA, 2012. ACM.

[138] Wesley Willett, Jeffrey Heer, Joseph Hellerstein, and Maneesh Agrawala. Commentspace: Structured support for collaborative visual analytics. In *CHI*, 2011.

[139] William E Winkler. Methods for record linkage and bayesian networks. Technical report, Technical report, Statistical Research Division, US Census Bureau, Washington, DC, 2002.

[140] William E Winkler, United States. Bureau of the Census, et al. Improved decision rules in the fellegi-sunter model of record linkage. 1993.

[141] Gary Wolf, A Carmichael, and K Kelly. The Quantified Self. *TED*, 2010.

[142] A. Woodruff, C. Olston, A. Aiken, M. Chu, V. Ercegovac, M. Lin, M. Spalding, and M. Stonebraker. Datasplash: A direct manipulation environment for programming semantic zoom visualizations of tabular data. *Journal of Visual Languages And Computing*, 12(5):551–571, 2001.

[143] Eugene Wu and Samuel Madden. Scorpion: Explaining Away Outliers in Aggregate Queries. In *VLDB*, 2013.

[144] Eugene Wu, Samuel Madden, and Michael Stonebraker. A Demonstration of DB-Wipes: Clean as You Query. *Proc. VLDB Endow.*, 5(12):1894–1897, August 2012.