Open Data Integration

Renée J. Miller
Northeastern University
miller@northeastern.edu

ABSTRACT

Open data plays a major role in supporting both governmental and organizational transparency. Many organizations are adopting *Open Data Principles* promising to make their open data complete, primary, and timely. These properties make this data tremendously valuable to data scientists. However, scientists generally do not have *a priori* knowledge about what data is available (its schema or content). Nevertheless, they want to be able to use open data and integrate it with other public or private data they are studying. Traditionally, data integration is done using a framework called *query discovery* where the main task is to discover a query (or transformation) that translates data from one form into another. The goal is to find the right operators to join, nest, group, link, and twist data into a desired form. We introduce a new paradigm for thinking about integration where the focus is on *data discovery*, but highly efficient internet-scale discovery that is driven by data analysis needs. We describe a research agenda and recent progress in developing scalable data-analysis or query-aware data discovery algorithms that provide high recall and accuracy over massive data repositories.

PVLDB Reference Format:
Renée J. Miller. Open Data Integration. *PVLDB*, 11 (12):2130-2139, 2018.
DOI: https://doi.org/10.14778/3229863.3240491

1. INTRODUCTION

In data science, it is increasingly the case that the main challenge is not in integrating known data, rather it is in finding the right data to solve a given data science problem. This is in stark contrast to the data integration challenges faced in the 1990's. At that time, one of the biggest complaints of academic database researchers was that it was hard to get data on which to test integration solutions. Real, complex, relational data was precious, expensive, and guarded. Researchers needed lawyers, time and money to get access to such data and could not share it even for research purposes. It was in this environment that many of the early influential data integration results were developed. The impact on research was clear. If data is so precious, then it is of paramount importance that the integration be correct. And once data was integrated, research provided robust efficient DBMS for querying integrated data efficiently.

Proceedings of the VLDB Endowment, Vol. 11, No. 12

DOI: https://doi.org/10.14778/3229863.3240491

The landscape has obviously changed. *Data* used to be the countable plural of *datum*. Today, data is a mass (uncountable) noun like dust, and data surrounds us like dust, even lovely structured data. Data is so cheap and easy to obtain that it is no longer important to always get the integration right and integrations are not static things. Data integration research has embraced and prospered by using approximation and machine learning. But generally not opaque learning. In data integration, the solutions that have had impact are those that can be explained to a human. Indeed, explaining integration [73] and keeping humans in the loop during integration [48] are two very active and important areas of research.

In this paper, we look past the dust to consider future data integration needs. The focus is on data science and analysis. Data science is often done over massive repositories, sometimes called *data lakes*, with large numbers of different datasets. The datasets may have little or no schema, for example, they may be CSV (comma-separated-value) files [64] or JSON files [17] (or in other common semi-structured formats). The repository itself is often very dynamic. Even private, enterprise repositories, with rich schemas, may be growing too fast for data scientists to keep up. So to do data science over these evolving, growing repositories, a data scientist needs scalable data discovery systems.

1.1 Data Science Examples

To illustrate data discovery and some of the data science requirements for data discovery, consider the following two examples.

Example 1: A data scientist is interested in building a model that predicts the CO2 emission caused by the usage of a certain types of fuel in and around London. She has access to a dataset (Table 1) that contains the green house gas emission indicators of different fuels and sectors of various London boroughs over several years. However, she finds that the attributes (features) she has collected are not sufficient to predict the CO2 emission. However, the data scientist intuits that the properties of locations is a key factor in green house gas emission indicators. Thus, she is interested in finding tables that contain such properties. Note that finding properties for only a few Boroughs does not help her, nor the properties for Boroughs outside of London. Similarly, finding properties that cannot be aligned with and assigned to a Borough does not help her. She has a very specific problem. She needs tables that join with her table on the `Borough` attribute and that contain all (or at the very least most) of the attribute values she has. Ideally, her search would be at interactive speeds even if her query table is not in the data lake and if the search cannot make use of precomputed inclusion or foreign key dependencies. She needs a search engine that will quickly return joinable tables. As an example, such a search engine may return Table 2 that contains new information for each Borough and can be joined with Table 1. After doing the join, the

Table 1: Greenhouse Gas Emission in London.

Borough	Data Year	Fuel	ktCO2	Sector	...
Barnet	2015	Electricity	240.99	Domestic	
Brent	2013	Gas	164.44	Transport	
Camden	2014	Coal	134.90	Transport	
City of London	2015	Railways diesel	10.52	Domestic	

Table 2: London Borough Profiles - Joinable Table with Query in Table 1.

Area_name	Population_Estimate	Average_age	Female_employment_rate	Unemployment_rate	...
City of London	8800	43.2	-	-	
Camden	242500	36.4	66.1	4	
Barnet	389600	37.3	62.9	8.5	
Enfield	333000	36.3	66	3.8	

Table 3: Greenhouse Gas Emission of Washington State - Unionable Table with Query in Table 1.

County	Year	Commodity Type	Total Emissions (MT CO2e)	Source	...
Benton	2015	Gasoline	64413	ConAgra Foods...	
Kittitas	2015	Fuel oil (1, 2...	12838	Central Wash...	
Grays Harbor	2015	Aviation fuels	1170393	Sierra Pacific...	
Skagit	2015	liquefied petroleum	59516	Linde Gas...	

data scientist can do the interesting part of her work, testing if any of the new attributes, alone or in combination, with old attributes can be used to build a better model. □

As a second example, consider the need to find unionable tables.

Example 2: Having succeeded at her first task, the data scientist is emboldened and wants to extend her results beyond London. She is ambitious and wants to see if there is sufficient data available to analyze global trends of CO2 emission, for as many years as she can find. Her dataset currently only contains green house gas emission indicators for areas in a small region. To have a comprehensive analysis, she needs to build a list of various geographical locations and their emission indicators. This involves finding tables (like Table 3) that contain (new) tuples with attributes that can be meaningfully aligned with attributes of Table 1 – unionable tables. Suppose her data lake is contains open government data. A keyword search for "greenhouse gas emission" in open data portals or standard search engines returns an overwhelming number of tables or pages that need to be manually examined to understand if the information can be meaningfully unioned. Even more damaging, the tables returned omit many important tables because they are tagged differently ("John Doe's Study of Greater Washington") or not tagged at all. While organizations may value Open Data Principles that stress the value of metadata, the person in charge of disseminating a dataset may not share these values or have sufficient incentives for ensuring high quality metadata. And the words "greenhouse", "gas", and "emission" do not appear in the tables themselves (attribute names or data). And even when tables have meaningful schemas, schema and entity-based search approaches [15, 49, 75] designed to identify entities in a table (in this case parts of London), and find other tables containing the same type of entities (locations), are likely to find a large amount of information on locations that is unrelated to the data science question at hand.

An automated solution for more effectively finding unionable tables (especially one with high recall even over schema-less data) would make this a much less laborious task for the data scientist. What this data scientist needs is a scalable, high performance search engine for finding unionable tables. Notice that the definition of what is unionable is not immediately obvious. The search engine must be able to recognize other attributes, in other tables that contain `Years` (this does not seem too hard), `Geographic Regions` (not just Boroughs, the engine needs to be able to generalize this concept even if the attribute is not named), and `Fuels` (even if they are described using related, but very different terms not mentioned in any available ontology). □

1.2 Data Integration for Data Science

In this paper, we introduce a new paradigm for thinking about data integration over massive, growing data repositories that focuses on data exploration and discovery. Integration is driven by data analysis needs and hence data discovery is done to facilitate this analysis. Data analysis requires discovery of data that joins, unions, or aggregates with existing data in a precise way – a paradigm we call *query-driven data discovery*. Importantly, the input to the discovery algorithms are relational tables and optionally a query operator. This means that solutions must be able to handle small and large query tables (even tables with millions of tuples) and effectively search over repositories containing tables of vastly different size. And for data science, the goal will be high accuracy in the search, especially high recall in finding results, even when there are few tables that satisfy the query. We will not assume all tables are entity-attribute tables containing an identifiable subject attribute containing entity names and will not use keyword-search to narrow the scope of a search to a managable set before joining or unioning them – techniques that have proven to be very valuable in finding *related tables* in Web or HTML data [45, 63, 75].

Our work has been motivated by a study of open data and the characteristics of open data. We want to understand what data management techniques are needed to make open data accessible to data scientists. We also are motivated by a goal of making data integration more open to large, growing repositories of data so that data scientists can more easily find and use this data.

We begin by (briefly) placing this work into historical perspective (Section 2). The goal is not a comprehensive survey of data

integration research, rather we discuss a (biased) selection of some high impact results and some of the general trends. We then discuss how data repositories are being created and some important characteristics of these lakes (Section 3). Next, we present new results on finding joinable tables (Section 4) and finding unionable tables (Section 5). The paper concludes with some open problems and research challenges.

2. A BRIEF HISTORY

We briefly review some of the major innovations in data integration over that last few decades. An important theme is that the way data integration has been studied and the solutions developed are largely driven by human concerns, that is, the way people produce and consume data.

2.1 1980's Data Federation

In the 1980's and 1990's, a major theme in data integration was data federation or mediation [33]. As the name suggests, data federation refers to the uniting or integration of smaller databases into a single centralized unit (often represented by a global or mediated schema). Because of the centralized control, the federation has design autonomy and can create a global schema that best represents the data from the smaller independent data states. Hence, the scientific tools used to study federation included data design principles where the goal is to find the best design for the global schema [6] and to understand the semantics of data transformations [12, 53] and schemas [56]. Concepts like information capacity [35] were used to understand when a (global) schema represents the same information as another [55].

In keeping with the federation notion, query processing and optimization is coordinated by a centralized unit which understands the capabilities of each independent state or system. Innovations in this area came in the way source capabilities are modeled [46] and in how queries are executed and optimized over heterogeneous DBMS, in systems such as Garlic [26]. This work and these systems reflected the general zeitgeist in which data was precious, people put time into designing and maintaining it, and there is tremendous value in creating clean integrated schemas and systems to efficiently query and use this valuable data.

Open data was already being studied in this period, including biological data (both private and public data), with systems that permitted querying over data in different formats [11].

2.2 2000's Data Exchange

The Web made data sharing easier. It also changed how we thought about data integration. To share data, it is not necessary to have any centralized or federated control. Rather, automous systems or peers can share data. Data integration then becomes the task of fitting data received from a source peer that has been designed independently into the design chosen by a receiving peer, the target [54]. This observation led to the development of *data exchange* [23, 24]. In data exchange, a source schema and target schema are given along with an instance of the source database and a schema mapping representing the relationship between the source and target schemas. The problem is to create a target database that conforms to the target schema and mapping, and that best models the source data. Although easy to state, data exchange has led to a surprisingly rich literature studying how database instances (data exchange solutions) relate to each other and how schema mappings can be composed or inverted. Data exchange remains a vibrant and influential research area [42].

Before data exchange can be done, one must design or discover a schema mapping. The Clio system pioneered the creation and use of schema mappings, declarative representations of the relationship between two schemas, and introduced a paradigm for mapping creation based on query discovery [22, 54]. Initially, this was done using inference over relational constraints. Spicy++ [51] and other systems [52] have generalize schema-mapping and data-exchange systems to a larger class of applications. Mapping discovery is facilitated by the discovery of inclusion dependencies within a database [69] or the use query logs to suggest how tables may be combined [21]. Finding joinable tables is an important first step in query discovery. However, in the context of data exchange it is made easier by having prescribed databases with known schemas. Other approaches learn mappings using data and meta-data (schema) evidence [70] or using probabilistic inference over data and meta-data evidence that may be incomplete or inconsistent [41]. This has been complemented by important work on learning mappings from examples (see ten Cate et al. [68] for a survey of these approaches) and many human-in-the-loop guided mapping refinement approaches.

Notably the initial mapping language for Clio has been greatly enhanced by others including generalization to represent entity resolution or linking and other complex transformations [34]. Similar declarative mapping languages have largely replaced the older (brittle) procedural extract-transform-load scripts for (centralized) data warehouses. Data exchange can be seen as a self-centered way of managing integration. Everyone has full freedom over their data and full responsibility for doing their integration – integrating other people's data into their own organization. A benefit was quick industry adoption (Clio was commercialized by IBM within three years of the first research result), but we are now seeing that data exchange is insufficient for data lakes.

Of course this is not a survey of data integration and there were many other advancements beyond mapping and data exchange during this time – tremendous advancements in entity-resolution and data linking to name just two areas [16]. And the importance of unstructured and semi-structured information was further recognized with the development of dataspaces [25] and with work on inferring structure and semantics from data [28, 40]. Researchers observed that the quality of data can often be improved by aligning it with other, higher quality data sources [10, 13, 31]. And this era saw the creation of important collections of structured web datasets, some public and some private [14, 44]

2.3 2020's Query-Driven Data Discovery

In data federation and data exchange, the data to be integrated is known and a central problem is either to design (or discover) a good global schema or to design (or discover) a good mapping between schemas. Once this is done, there are interesting systems and algorithmic challenges in efficiently answering queries over, or exchanging data between, these structures. In data science, however, it is increasingly the case that the main challenge is not in integrating known data, rather it is in finding the right data to solve a given data science problem.

3. EXAMPLE DATA LAKES

Data science is often done over repositories with massive numbers of different datasets, repositories that we will call data lakes. In this section, we take a look at some of the applications and societal trends that are driving the creation of data lakes and consider the characteristics of some of the lakes being created.

3.1 Open Data

Open data is "structured data that is machine-readable, freely shared, used and built on without restrictions."[1] Perhaps one of the most valuable data lakes being created today is by governments through Open Data Initiatives like the U.S. `data.gov` or Canada's `open.canada.ca`. What makes these initiative interesting from a data science perspective is that many governments (federal, provincial, and municipal) are adopting Open Data Principles [67]. These principles state that open data should be complete, primary (including the original data and metadata on how it was collected), timely, and permanent (with appropriate version control over time). Even when imperfectly achieved, these principles make this data valuable and sometimes irreplaceable for data scientists. And when achieved, this data is data on which we can do exquisite data science by taking into account data collection methodologies and bias. And it is not just governments who are embracing these principles. Public and private organizations alike are using these principles when transparency is in their own business interest.

We have been using and curating open data for over a decade. Our efforts include LinkedCT, a project under which we curated the NIH International Clinical Trial Data and published it as linked open data [5, 29, 30]. And we have also been watching and tracking open government data more broadly. Based on our observations, at least 28 countries around the world are publishing substantial amounts of open data. On `data.gov`, the U.S. federal government's open data portal, the number of indexed data files grew by 400% in the year before March 2017. Reacting to the increasing availability of open data, data scientists have made it an important data source: according to a 2017 Data Scientist Report by CrowdFlower, 41% of data scientists say they are using publicly available datasets [18].

We have collected a repository of open data [58]. Our lake contains data collected from three federal governments and is a crawl of 215,393 CSV tables from U.S., U.K., and Canadian open data portals as of March 2017. This is by no means all government open data, rather it is a collection of easily crawled data (obtained using a commonly supported API). Some basic statistics on this data, which we call *Open Data* in the remainder of this paper, are shown in Table 4. Notice that while the number of tables in our crawl is modest in size, the number of attributes is still large. The average number of attributes per table is 16 with a large variance, some tables have many hundreds of attributes. The average attribute size is 465, but among string attributes (arguably the most used in joins) the average attribute size is 1,540.

Table 4: Characteristics of three data lakes

	#Attrs	MaxSize	AvgSize	#UniqVals
Open Data	3,367,520	22,075,531	465	609,020,645
WebTable	252,766,759	17,033	10	193,071,505
Enterprise	2,032	859,765	4,011	3,902,604

3.2 Mass Collaboration

Mass collaboration is often associated with wiki projects, but of course has been applied to structured data creation in projects like DBPedia, WikiData, and others. Community members contribute data and data is also harvested, using information extracted from community created resources. To this end, WebTables [44] can also be thought of as a data lake created through mass collaboration. WebTables is a collection of millions of structured tables extracted from the largest mass collaboration project, the Web.

While different in nature and content from open government data, mass collaboration repositories also have great value to data scientists. In our studies, we have used a 50M table snapshot of English WebTables as a representative of a mass collaboration data lake [44]. Table 4 contains some basic statistics on WebTables. Notice that while more massive in number than our crawl of open data, individual WebTables are on average smaller (both in number of attributes, with an average of five attributes per table, and in the size of those attributes, with an average of ten values per attribute) than open data.

3.3 The Modern Enterprise

Many modern enterprises have significant investment in massive data warehouses. As one example, the MIT warehouse has 2,400 tables [19]. Statistics on a subset of 167 of these table are given in Table 4. Notice that although this is a small subset it already shows interesting trends. Tables have on average slightly fewer attributes (twelve), but are even larger than open data. And the vocabulary of even this small subset is already very large (though perhaps expectedly not as large and heterogeneous as our open data crawl or WebTables).

While one could argue that a standard warehouse like this is not a data lake, increasingly warehouses are reaching the scale that to a data scientist, even one familiar with portions of the warehouse, they can look like a data lake. And furthermore, warehouses are increasingly being used in concert with internal enterprise data lakes. These lakes include data that has been purchased or harvested opportunistically for one project but is saved strategically for reuse in others. And in many cases, we have surpassed the limits of being able to maintain complete and consistent meta-data or join graphs over these repositories.

Enterprises are recognizing the challenge of finding even local enterprise data. IBM's LabBook provides a collaborative data analysis platform over an enterprise data lake [38]. They automatically collect and organize extensive metadata about data, queries, visualizations, and users. They use this to provide query and data recommendations. Similarly, Goods [27] also organizes enterprise data sets using query logs and metadata, and exposes these through faceted search on the metadata of data sets. Skluma is a system for organizing large amounts of data (organized in different file types) which extracts "deeply embedded" metadata, latent topics, relationships between data, and contextual metadata derived from related files [8]. Skluma has been used to organize a large climate data collection with over a half-million files. Skluma uses statistical learning to allow data to be discovered by its content and by discovered topics, something that is especially important for headerless-files which also appear in open data. Important open issues remain on how to maintain all this metadata as lakes evolve. Our work on query-aware search complements these meta-data focused approaches for organizing data.

4. FINDING JOINABLE TABLES

Given a *query table* $T_q(A_1, A_2, \dots)$ with join attribute A_j, a joinable table is a relation $T(B_1, B_2, \dots)$ such that T has at least one attribute B_i that is equi-joinable with A_j. To be more precise, the values in the two attributes must have significant overlap. As illustrated in Example 1, a search engine for joinable tables is a valuable tool for data scientists.

[1]Canada's open data portal: https://open.canada.ca 07/15/2018

4.1 Past approaches

One approach to data discovery (including joinable table discovery) would be to create a global schema [6]. Federated data integration works well when individual data sources come with high quality schemas, the number of data sources is tractable, and the entire collection of data sources remains reasonably static (though incremental approaches have been studied). The landscape we have described is significantly different. In many instances, such as open data or in some cases data from the Web, tables do not have any schema information or lack meaningful attribute names. At Internet scale, the number of tables can easily reach millions or more. Finally, open data publishing platforms (including the common APIs used to publish open data) allow continuous addition of new data sources over time. For data lakes, deriving and maintaining a global schema is quite impossible.

Other data integration approaches such as Clio [22] and Data Civilizer [19] maintain a set of known join paths which are declared or mined from the database. The join paths require knowledge of the database schema such as foreign key constraints of the underlying tables. Using the join paths, one can find joinable tables. And at the cost of additional pre-computation, one can maintain a join graph with tables that are approximately or mostly joinable. However, managing pre-computed join-paths is also intractable for an open platform at Internet scale. In addition, we believe that it is highly desirable to support *ad hoc* discovery of joinable tables with interactive response times. This allows for joinable table-search even when the query table is not in the repository.

Of course joinable table search can be formulated as a classic *set similarity search* problem which is well studied in information retrieval. In this formulation, attributes are sets and containment (the normalized version of set overlap) is used as the similarity function. While numerous, the solutions for this problem have focused on relatively small sets such as keywords and emails (albeit containing a very large number of sets) [9, 20, 47, 71, 72, 74]. Cannonical datasets used in the evaluations of these approaches (including ENRON email, DBLP, and an AOL dataset) have average set sizes ranging from 3 to a little over 100 and maximum set sizes in the low thousands [50]. In developing LSH Ensemble (which we present next), we experimented with these solutions and found they could not provide interactive search over the data lakes we are considering. This was due both to the much larger set sizes (especially in open data), but also to the very large dictionaries in data lakes (number of unique values) that make inverted indices built over data lakes massive and unwieldy.

The Mannheim Search Join, and earlier approaches for table extension [15, 75] on WebTables, assume tables are entity-attribute tables with a subject attribute containing pseudo keys for entitites [45]. Tables are retrieved using attribute names or using subject values. These approaches do not use containment and generally are also evaluated over small query tables (less than 10,000 values).

4.2 LSH Ensemble

We have studied the joinable table search problem as a problem of *domain search* [77], where we treat each attribute of a table as a domain. Thus, finding joinable tables for a query table T_q and join attribute A_j becomes finding tables with an attribute similar to A_j. We introduced a distributed index structure, *LSH Ensemble*, which efficiently indexes all the domains of the tables in a data lake, and supports fast domain search queries.

Our algorithm has the following characteristics that make it particularly suitable for join-driven data discovery.

- *Scalability*: our system is designed to handle Internet scale data lakes. Using data sketch techniques, we are able to achieve very compact index sizes per table, so we have a small query footprint. Our implementation can handle joinable table search for over 260 million tables with three second query response time on a cluster of five machines (each with 64 CPUs).

- *Open world assumption*: the index structure is highly distributed, and can accommodate efficient incremental re-indexing for newly added data sets. We also do no have any assumption on a fixed vocabulary (or dictionary) for the data values. So, the data lake can grow over time, and our index can scale gracefully.

- *Robust for skewed distribution*: at Internet scale, the power-law emerges over domain (attribute) sizes (see Figure 1 for the cardinality distribution of string attributes in Open Data and Figure 2 for string attributes in WebTables). This means that the cardinality of the data sets will vary by orders of magnitudes. Our index uses a provably optimal binning strategy to handle all data sets, small or large, with equal efficiency.

- *Containment*: contrary to traditional set similarity approaches which use Jaccard similarity (which is commonly used with LSH [2]), our algorithm uses the *containment score* as a measure of relevance. Using the containment score, joining the

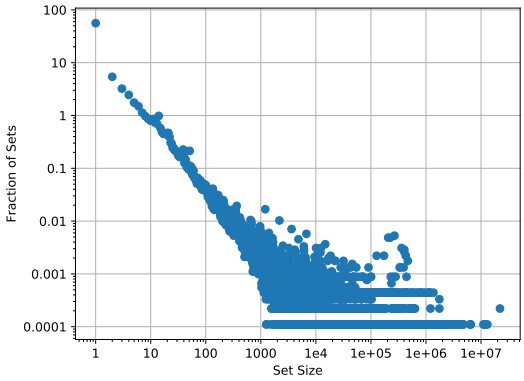

Figure 1: Set size distribution of string attributes from Open Data

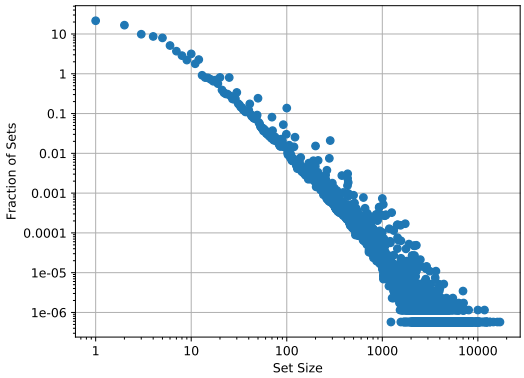

Figure 2: Set size distribution of string attributes from WebTables

query table with a found table will yield the largest number of facts or tuples. This is an important property that is not true of Jaccard similarity.

4.2.1 Containment with MinHash

Consider a query table T_q, a candidate table T, and the equi-join using $T_q \bowtie_{A=B} T$ and A and B being attributes of T_q and T respectively. Our objective is to preserve as many facts in T_q as possible in the join result. This means that we want B to contain as many values from A as possible. Hence, we define the relevance of B to be measured by the containment score. To keep the notation simple, we use A (and B) for both the attributes and for the set of values contained in the attribute.

$$\text{containment}(A, B) = \frac{|A \cap B|}{|A|}$$

A closely related relevance measure is the *Jaccard similarity*.

$$\text{jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard has been a popular choice in measuring the relevance of sets. Unfortunately, it is not appropriate for data discovery on real data lakes because it favors tables with small domains, and unfairly rejects tables with large domains [1]. That said, the Jaccard similarity measure has a highly desirable property that it can be computed very efficiently for very large numbers of domains using locality sensitive hashing (LSH) [2] and MinHash [36].

Consider the following relationship between the containment score and Jaccard similarity.

$$\text{jaccard}(A, B) = \frac{\text{containment}(A, B)}{\frac{|B|}{|A|} + 1 - \text{containment}(A, B)}$$

Using this, we are able to translate a containment threshold based query to a Jaccard threshold based query. This lets us use a MinHash LSH index structure to index the domains of the data lake. Note however, that we do not know *a priori* the size of B (a candidate answer domain). In order to map a containment threshold to a Jaccard threshold, we need a conservative upper bound u on the domain size of B. We showed that the error of this estimate $u - |B|$ creates additional false positives in the LSH index lookup. Thus, the less accurate the estimation of $|B|$ the worse the index performance will be. Coupled with the power law in the domain sizes of data lakes, the naive LSH will be ineffective as an indexing structure [77].

4.2.2 Coping with Skewed Distributions

We proposed to divide the data lake into partitions and distribute each partition across a cluster of machines to overcome the skewed distribution of domain sizes. The distributed collection of LSH indexes that cover a partitioned data set is what we call *LSH Ensemble* [77]. We created a cost model that describes the performance of LSH Ensemble for an arbitrary partitioning. Using the cost model, we formulated the problem of data partitioning for LSH Ensemble as an optimization problem. We showed that for any power law distribution, an optimal partitioning can be obtained. A surprising outcome of our analysis is that a simple geometric partitioning scheme is near-optimal for power-law distributions:

$$u_{i+1} - l_{i+1} \simeq \alpha \cdot (u_i - l_i)$$

where $[l_i, u_i]$ is the range of domain sizes for partition i, and $i = 0, 1, 2, \ldots$

Our experimental evaluation shows that the performance of LSH Ensemble is not very sensitive to the choice of α, and the empirical value of $\alpha = 2$ works well for both the Open Data (Figure 1) and for WebTables (Figure 2) which follow different power law distributions of their domain cardinalities. Since the size of the query affects the transformation from containment to Jaccard similarity and the query size is only determined at run time, we use LSH Forest [7] to dynamically tune the LSH index for different query sizes at run time.

The simplicity of the partitioning scheme is particularly attractive in the context of data discovery. Given that we need to handle large numbers of attributes, we cannot afford to perform overly elaborate data processing. The geometric partitioning only requires a single pass and the partitioning can be updated incrementally when new data sets are added. Our implementation of LSH Ensemble can handle joinable table search for over a couple hundred million tables and we have used it to build a demonstration of open data search that provides an interactive and engaging user experience for data scientists [78].

5. FINDING UNIONABLE TABLES

Given a *query table* $T_q(A_1, A_2, \ldots, A_n)$ with n attributes, a unionable table is a table $T(B_1, B_2, \ldots, B_k)$ such that T has at least one attribute B_i that is unionable with some attribute A_j from the query table. To understand table unionability, we must first understand attribute unionability. Given attribute A_j and its domain (set of values), and attribute B_i and its domain, we want to understand the likelihood that there exists a third domain D from which both A_j and B_i are sampled. Then, we must define a search problem that deals equitably with tables that union over all n attributes with those that union over a subset of attributes.

As illustrated in Example 2, a search engine for unionable tables is a valuable tool for researchers to discover more data to expand their data analysis.

5.1 Past Approaches

An approach to finding unionable tables is through schema matching, where the problem is to match the attributes of two or more tables (or schemas) [32, 57, 60, 65]. Two tables that match on i attributes can presumably be unioned on those attributes. Matching is done largely heuristically using similarity functions over schema (attribute names) and sometimes values (for example, using a set similarity measure) or value distributions [39]. These approaches have generally not been evaluated for accuracy as to how well they find tables whose attribute values are really drawn from the same domain (and indeed most were not explicitly designed for this application). Although scalable schema matching and ontology alignment have been studied extensively, for table union search, matching would need to be studied as a search problem – find the best matches for a given query table.

As an example of matching applied to table union, Das Sarma et al. [63] define two HTML tables as *entity-complements* if they contain information about related sets of entities. They rely on the signals mined from high coverage private ontologies curated from massive quantities of Web data as well as publicly available ontologies (such as YAGO [66]) to determine if tables have subject attributes about related sets of entities. They determine if the non-subject attributes provide similar properties of entities by finding instance-based and schema-based matchings between non-subject attributes. The strength of entity-complement search is tied to the ontology coverage. However, due to the breadth of tables in data lakes, including open data, ontology-based techniques, especially

using open ontologies, are not always reliable and the assumption that a subject attribute can be identified does not always hold.

Lehmberg and Bizer [43] have built upon the work by Ling et al. [49] on stitching tables with similar schemas and created union tables of stitched tables by means of a set of schema-based and instance-based matching techniques. These techniques are designed for WebTables and rely on schema information, which in some data lakes might not be available for many tables, leading to low recall.

To find related tables to keyword queries, Octopus performs Web document-style search on the content and context of WebTables [15]. These tables are then clustered into groups of unionable tables by means of syntactic measures on attribute values. If two tables do not match on keywords, even though unionable, they will not be found, and this can lead to low recall. Pimplikar and Sarawagi present a search engine that finds tables similar to a query table that is represented by a set of keywords each describing a column of the query [59]. This is the closest work to union table search, but uses a keyword search similarity score to find unionable attributes. In contrast, we present an approach below that is not motivated by keyword search but rather by relational table union. The input to table union search is a full relational table and we use this to find other tables in the data lake that can be meaningfully unioned with the query table

5.2 Table Union Search

Before defining table union search, we define precisely what it means for attributes to be unionable.

5.2.1 Attribute Unionability

We introduce the problem of *attribute unionability* [58] as the likelihood that two attributes contain values that are drawn from the same domain. We define three types of domains for attributes and three statistical tests: *set-unionability* for domains containing values (this test can be applied to any attribute), *sem-unionability* for domains containing some values that represent entities belonging to classes in an ontology (e.g., the `Borough` attribute in Table 1), and *NL-unionability* for attributes containing natural-language (e.g., the `CommodityType` attribute in Table 3). Our goal is to have an effective way of evaluating the hypothesis that two attributes come from the same domain. The first two tests use the overlap in values or entities (respectively), but not heuristically using thresholding. Rather, since the size of the set overlap follows a hypergeometric distribution, we are able to compute the likelihood of two attributes being drawn from the same domain (either a set domain or an ontologic domain). This is given by the Cumulative Distribution Function of the hypergeometric distribution of their overlap. Suppose we have a domain D and we want to know how likely it is that A and B are drawn from this domain. Assume $|A \cap B| = t$ is the actual intersection size. The set-unionability $\mathbf{U}_{\text{set}}(A, B)$ is the following sum of probabilities of having an intersection size of s conditioned on the sizes of A, B, and D.

$$\mathbf{U}_{\text{set}}(A, B) = \sum_{0 \le s \le t} p(s \;\Big|\; |A|, |B|, |D|) \qquad (1)$$

Of course, the domain D is hypothetical and we do not know its size. We instead approximate $|D|$ as the size of the disjoint union of A and B. The sem-unionability $\mathbf{U}_{\text{sem}}$ of A and B is defined in a similar way using the size of the overlap of the ontology classes to which the entities (represented by values in an attribute) belong.

Not all values represent entities, but they may still have strong semantic proximity. We model this semantic closeness using NL-unionability. Each value v in an attribute can be represented by its word embedding (a multi-dimensional vector $\vec{v}$) [37]. An attribute can be modeled with a multivariate normal distribution on the word embedding vectors of its values centered around μ_A (the *topic vector* of the attribute) with some covariance matrix Σ_A. We then define two attributes to be NL-unionable if they are likely sampled from the same distribution (which would be a distribution that represents their common domain). This can be computed using the distance between the topic vectors of attributes (that is, the estimated mean of the set of embedding vectors). Following a similar development as for $\mathbf{U}_{\text{set}}$ and $\mathbf{U}_{\text{sem}}$ but using a different test statistic appropriate for these topic vectors, we defined a third notion of unionability, $\mathbf{U}_{\text{nl}}$ [58].

Of course, users will not be able to chose *a priori* which measure to use. Using the maximum of the three scores (or any function over the scores) is not meaningful as the measures are incomparable. Hence, we provide a measure called *ensemble unionability* which automatically selects, based on the statistics in the data lake, the unionability measure that is the best to use with a pair of attributes.

5.2.2 Table Unionability

As motivated above, we need a way of equitably comparing tables that union on different numbers of attributes and determining the best attributes to use for the union. Note that pairs of tables may have some highly unionable attributes and others with much lower values. How do we determine which to use? Obviously, an *a priori* threshold is an unappealing solution. To solve this problem, we make use of *alignments*, meaning one-to-one mappings between subsets of the attributes of two tables. The table unionability of two tables with respect to an alignment is then simply the product of the ensemble attribute unionability of the attributes in the alignment. To compare table unionability over alignments of different sizes, we again use statistics of all unionability scores in the repository, a technique we call *distribution-aware*. Intuitively, the *goodness* of an alignment is the likelihood of the alignment being the most unionable of the alignments of a specific size in a repository.

5.2.3 Table Union Search

The mathematics is all well and good, but for data discovery, we need to make it scale. Our system for table union search uses several system innovations to achieve scalability. First, to compute unionability scores efficiently (at query time), we developed LSH-based indexes that permit us to efficiently retrieve an approximate set of candidate unionable attributes. Different indices are required to approximate different unionability measures. Second, for both ensemble-unionability and table-unionability, we require statistics about pair-wise unionability scores over the entire data lake. Obviously, this is too expensive to compute let alone maintain. Instead, we empirically showed that we can quite accurately estimate these statistics efficiently using the data sketches in our LSH indexes. Our empirical evaluation shows that our union search is much more accurate (in precision and recall) than previous approaches based on keyword search and matching [15, 43] making it better for the data science applications we are targeting. We attribute this to using full relational tables as the query, not just keywords. Moreover, our NL-unionability leverages word embeddings to uncover unionability when attributes have very little value and ontology overlap. The pruning power of NL-unionability makes our approach orders of magnitude faster than our implementation of Octopus [15] (note the original system is not open source).

We have publicly shared an *Open Data Table Union Benchmark* that we created for evaluating accuracy [58] and hope it will be used for further advances in table union search.

6. CONCLUSION AND FUTURE WORK

Our work began with a study of integration over open data. In the process, the techniques we developed were necessarily open to the integration of new data (both new data sets and new unseen values). Our goal is to provide the principles and systems to allow data scientists to effectively find and integrate open data and to make private data repositories more open in that new data can easily be added and integrated. The methodology we are using is based on providing open software or by contributing to well-know software ecosystems.

We are using open data in part so we can share not only our systems, but our empirical data and results. We are also developing shared benchmarks (with gold standard answers) so that others can reproduce our results and compare their systems to ours on the same data. This is something that has been done for evaluating matching systems (for example, with the T2D Gold Standard [61]), but our community needs to do more data preparation, curation, and sharing of data for the purpose of enhancing empirical comparisons of our research proposals [62], In data exchange, the metadata generator iBench [3] has been widely used to create rigorous empirical evaluations of data exchange systems. It is important in data discovery, that we continue as a community to invest in creating and sharing evaluation tools and benchmark datasets [4].

In the short term, there are many avenues for expansion. Extending joinable table search to multiple attributes and to non-equijoins is an interesting direction. To what extent could an entity-resolution strategy be integrated into search to produce an algorithm that measures containment based on resolved values and produces a joined result based on aligning resolved values? Or, could an auto-join approach [76], which learns how to transform values for the join, be incorporated into a joinable table data discovery system? And of course, in the spirit of using the whole table as input, finding joinable tables that have new attributes (which contain new information in comparison to the other attributes in the query table) is an interesting problem. Extending unionable table search to numbers (quantities) is also a very interesting open problem. And schema inference over data lakes remains a important challenge.

The focus of this work has been on data discovery as that is an important and necessary first step in enabling data integration over evolving data lakes. But we should not throw away the past. The lessons learned from data exchange and schema mapping discovery through query-discovery should still inform the future. Ultimately, our goal should be full query-discovery over data lakes.

Acknowledgments

I would like to give a special thank you to Fatemeh Nargesian and Ken Q. Pu for their help and insights on this paper, and to Erkang Zhu for his help and the statistics. Many of these ideas were also influenced by collaborations with: Periklis Andritsos, Bahar Ghadiri Bashardoost, Christina Christodoulakis, and Carolina Simoes Gomes. I would also like to thank Laura Haas, Lucian Popa, and Wang-Chiew Tan for very helpful comments on a draft of this paper. A final thank you to Dong Deng for providing statistics on the MIT warehouse. NSERC funded this work along with the Northeastern College of Computer and Information Science.

7. REFERENCES

[1] P. Agrawal, A. Arasu, and R. Kaushik. On indexing error-tolerant set containment. In *ACM SIGMOD*, pages 927–938, 2010.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[3] P. C. Arocena, B. Glavic, R. Ciucanu, and R. J. Miller. The iBench integration metadata generator. *PVLDB*, 9(3):108–119, 2015.

[4] P. C. Arocena, B. Glavic, G. Mecca, R. J. Miller, P. Papotti, and D. Santoro. Benchmarking data curation systems. *IEEE Data Eng. Bull.*, 39(2):47–62, 2016.

[5] B. G. Bashardoost, C. Christodoulakis, S. H. Yeganeh, R. J. Miller, K. Lyons, and O. Hassanzadeh. VizCurator: A visual tool for curating open data. In *WWW*, pages 195–198, 2015.

[6] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[7] M. Bawa, T. Condie, and P. Ganesan. Lsh forest: Self-tuning indexes for similarity search. In *WWW*, pages 651–660, 2005.

[8] P. Beckman, T. J. Skluzacek, K. Chard, and I. T. Foster. Skluma: A statistical learning pipeline for taming unkempt data repositories. In *Scientific and Statistical Database Management*, pages 41:1–41:4, 2017.

[9] A. Behm, C. Li, and M. J. Carey. Answering approximate string queries on large data sets using external memory. In *IEEE ICDE*, pages 888–899, 2011.

[10] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[11] P. Buneman, S. B. Davidson, K. Hart, G. C. Overton, and L. Wong. A data transformation system for biological data sources. In *VLDB*, pages 158–169, 1995.

[12] P. Buneman, S. B. Davidson, and A. Kosky. Semantics of database transformations. In *Semantics in Databases*, pages 55–91, 1995.

[13] D. Burdick, M. A. Hernández, H. Ho, G. Koutrika, R. Krishnamurthy, L. Popa, I. Stanoi, S. Vaithyanathan, and S. R. Das. Extracting, linking and integrating data from public sources: A financial case study. *IEEE Data Eng. Bull.*, 34(3):60–67, 2011.

[14] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: Exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

[15] M. J. Cafarella, A. Y. Halevy, and N. Khoussainova. Data integration for the relational web. *PVLDB*, 2(1):1090–1101, 2009.

[16] P. Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer, 2012.

[17] D. Crockford. The application/json media type for javascript object notation (JSON). *Request for Comment*, 4627:1–10, 2006.

[18] CrowdFlower. 2017 Data Scientist Report. http://visit.crowdflower.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf, Date accessed: July 15, 2019.

[19] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang. The data civilizer system. In *CIDR*, 2017.

[20] D. Deng, G. Li, J. Feng, and W. Li. Top-k string similarity search with edit-distance constraints. In *IEEE ICDE*, pages 925–936, 2013.

[21] H. Elmeleegy, A. K. Elmagarmid, and J. Lee. Leveraging query logs for schema mapping generation in U-MAP. In *ACM SIGMOD*, pages 121–132, 2011.

[22] R. Fagin, L. M. Haas, M. A. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis. Clio: Schema mapping creation and data exchange. In *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*, pages 198–236, 2009.

[23] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224, 2003.

[24] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[25] M. J. Franklin, A. Y. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Record*, 34(4):27–33, 2005.

[26] L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *VLDB*, pages 276–285, 1997.

[27] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang. Goods: Organizing google's datasets. In *ACM SIGMOD*, pages 795–806, 2016.

[28] O. Hassanzadeh, A. Kementsietsidis, L. Lim, R. J. Miller, and M. Wang. A framework for semantic link discovery over relational data. In *CIKM*, pages 1027–1036, 2009.

[29] O. Hassanzadeh, A. Kementsietsidis, L. Lim, R. J. Miller, and M. Wang. LinkedCT: A linked data space for clinical trials. *CoRR*, abs/0908.0567, 2009.

[30] O. Hassanzadeh and R. J. Miller. Automatic curation of clinical trials data in LinkedCT. In *ISWC*, pages 270–278, 2015.

[31] O. Hassanzadeh, K. Q. Pu, S. H. Yeganeh, R. J. Miller, L. Popa, M. A. Hernández, and H. Ho. Discovering linkage points over web data. *PVLDB*, 6(6):444–456, 2013.

[32] B. He and K. C. Chang. Statistical schema matching across web query interfaces. In *ACM SIGMOD*, pages 217–228, 2003.

[33] D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM Trans. Inf. Syst.*, 3(3):253–278, 1985.

[34] M. A. Hernández, G. Koutrika, R. Krishnamurthy, L. Popa, and R. Wisnesky. HIL: a high-level scripting language for entity integration. In *EDBT*, pages 549–560, 2013.

[35] R. Hull. Relative information capacity of simple relational database schemata. *SIAM J. Comput.*, 15(3):856–886, 1986.

[36] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM STOC*, pages 604–613, 1998.

[37] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *ACL*, 2017.

[38] E. Kandogan, M. Roth, P. M. Schwarz, J. Hui, I. G. Terrizzano, C. Christodoulakis, and R. J. Miller. Labbook: Metadata-driven social collaborative data analysis. In *IEEE Big Data*, pages 431–440, 2015.

[39] J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *ACM SIGMOD*, pages 205–216, 2003.

[40] G. Kasneci, M. Ramanath, F. M. Suchanek, and G. Weikum. The YAGO-NAGA approach to knowledge discovery. *SIGMOD Record*, 37(4):41–47, 2008.

[41] A. Kimmig, A. Memory, R. J. Miller, and L. Getoor. A collective, probabilistic approach to schema mapping. In *IEEE ICDE*, pages 921–932, 2017.

[42] P. G. Kolaitis. Reflections on schema mappings, data exchange, and metadata management. In *ACM PODS*, pages 107–109, 2018.

[43] O. Lehmberg and C. Bizer. Stitching web tables for improving matching quality. *PVLDB*, 10(11):1502–1513, 2017.

[44] O. Lehmberg, D. Ritze, R. Meusel, and C. Bizer. A large public corpus of web tables containing time and context metadata. In *WWW*, pages 75–76, 2016.

[45] O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, and C. Bizer. The mannheim search join engine. *J. of Web Semantics*, 35:159–166, 2015.

[46] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *VLDB*, pages 251–262, 1996.

[47] C. Li, J. Lu, and Y. Lu. Efficient merging and filtering algorithms for approximate string searches. In *IEEE ICDE*, pages 257–266, 2008.

[48] G. Li. Human-in-the-loop data integration. *PVLDB*, 10(12):2006–2017, 2017.

[49] X. Ling, A. Y. Halevy, F. Wu, and C. Yu. Synthesizing union tables from the web. In *IJCAI*, pages 2677–2683, 2013.

[50] W. Mann, N. Augsten, and P. Bouros. An empirical evaluation of set similarity join techniques. *PVLDB*, 9(9):636–647, 2016.

[51] B. Marnette, G. Mecca, P. Papotti, S. Raunich, and D. Santoro. ++Spicy: an OpenSource Tool for Second-Generation Schema Mapping and Data Exchange. *PVLDB*, 4(12):1438–1441, 2011.

[52] G. Mecca and P. Papotti. Schema mapping and data exchange tools: Time for the golden age. *it - Information Technology*, 54(3):105–113, 2012.

[53] R. J. Miller. Using schematically heterogeneous structures. In *ACM SIGMOD*, pages 189–200, 1998.

[54] R. J. Miller, L. M. Haas, and M. A. Hernández. Schema mapping as query discovery. In *VLDB*, pages 77–88, 2000.

[55] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. The use of information capacity in schema integration and translation. In *VLDB*, pages 120–133, 1993.

[56] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. Schema equivalence in heterogeneous systems: bridging theory and practice. *Inf. Syst.*, 19(1):3–31, 1994.

[57] A. Nandi and P. A. Bernstein. HAMSTER: using search clicklogs for schema and taxonomy matching. *PVLDB*, 2(1):181–192, 2009.

[58] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller. Table union search on open data. *PVLDB*, 11(7):813–825, 2018.

[59] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. *PVLDB*, 5(10):908–919, 2012.

[60] E. Rahm. Towards large-scale schema and ontology matching. In *Schema Matching and Mapping*, pages 3–27. 2011.

[61] D. Ritze, O. Lehmberg, and C. Bizer. Matching HTML tables to dbpedia. In *Web Intelligence*, pages 10:1–10:6, 2015.

[62] S. W. Sadiq, T. Dasu, X. L. Dong, J. Freire, I. F. Ilyas, S. Link, R. J. Miller, F. Naumann, X. Zhou, and D. Srivastava. Data quality: The role of empiricism. *SIGMOD Record*, 46(4):35–43, 2017.

[63] A. D. Sarma, L. Fang, N. Gupta, A. Y. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu. Finding related tables. In *ACM SIGMOD*, pages 817–828, 2012.

[64] Y. Shafranovich. Common format and MIME type for comma-separated values (CSV) files. *Request for Comment*, 4180:1–8, 2005.

[65] W. Su, J. Wang, and F. H. Lochovsky. Holistic schema matching for web query interfaces. In *EDBT*, pages 77–94, 2006.

[66] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[67] J. Tauberer. *Open Government Data (The Book)*. https://opengovdata.io/, 2014. Second Edition. Date accessed: July 15, 2018.

[68] B. ten Cate, P. G. Kolaitis, and W. C. Tan. Schema mappings and data examples. In *EDBT*, pages 777–780, 2013.

[69] F. Tschirschnitz, T. Papenbrock, and F. Naumann. Detecting inclusion dependencies on very many tables. *ACM Trans. Database Syst.*, 42(3):18:1–18:29, 2017.

[70] O. Udrea, L. Getoor, and R. J. Miller. Leveraging data and structure in ontology integration. In *ACM SIGMOD*, pages 449–460, 2007.

[71] J. Wang, G. Li, D. Deng, Y. Zhang, and J. Feng. Two birds with one stone: An efficient hierarchical framework for top-k and threshold-based string similarity search. In *ICDE*, pages 519–530, 2015.

[72] J. Wang, G. Li, and J. Feng. Can we beat the prefix filtering?: an adaptive framework for similarity join and search. In *ACM SIGMOD*, pages 85–96, 2012.

[73] X. Wang, L. M. Haas, and A. Meliou. Explaining data integration. *IEEE Data Eng. Bull.*, 41(2):47–58, 2018.

[74] C. Xiao, W. Wang, X. Lin, and H. Shang. Top-k set similarity joins. In *IEEE ICDE*, pages 916–927, 2009.

[75] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: Entity augmentation and attribute discovery by holistic matching with web tables. In *SIGMOD*, pages 97–108, 2012.

[76] E. Zhu, Y. He, and S. Chaudhuri. Auto-join: Joining tables by leveraging transformations. *PVLDB*, 10(10):1034–1045, 2017.

[77] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller. LSH ensemble: Internet-scale domain search. *PVLDB*, 9(12):1185–1196, 2016.

[78] E. Zhu, K. Q. Pu, F. Nargesian, and R. J. Miller. Interactive navigation of open data linkages. *PVLDB*, 10(12):1837–1840, 2017.