Making Open Data Transparent:
Data Discovery on Open Data

Renée J. Miller[1], Fatemeh Nargesian[1]
Erkang Zhu[1], Christina Christodoulakis[1], Ken Q. Pu[2], Periklis Andritsos[1]
[1]University of Toronto, [2]UOIT
{miller,ekzhu,fnargesian,christina}@cs.toronto.edu,
ken.pu@uoit.ca, periklis.andritsos@utoronto.ca

Abstract

Open Data plays a major role in open government initiatives. Governments around the world are adopting Open Data Principles *promising to make their Open Data complete, primary, and timely. These properties make this data tremendously valuable. Open Data poses interesting new challenges for data integration research and we take a look at one of those challenges,* data discovery. *How can we find new data sets within this ever expanding sea of Open Data. How do we make this sea transparent?*

1 Introduction

The goals of Open Data are laudable including the promotion of civil discourse, improving public welfare, and ensuring citizens may participate more fully in their governments. Governments in many countries have recognized that the provision of open access to their data is crucial and beneficial to the public good [2, 33]. Therefore, both federal and local governments have been putting policy frameworks in place in order to enable access to their data sources. When open data is used effectively, citizens can explore and analyze public resources, which in turn allows them to question public policy, create new knowledge and services, and discover new (hidden) economic value useful for existing and new business initiatives. Efforts to support true transparency often involve dedicated teams that manually gather data from tens or hundreds of sources and tediously edit and compile it to produce consumable reports on issues like federal spending, the environment, and many others.

Given the value of Open Data, especially Open Government Data, an important question is how well are we doing in providing adequate tools for integrating Open Data? We take a look at the first task in data integration, *data discovery*. We consider data sets of structured data which may be from a relational DBMS, but more often are extracted from spreadsheets or document collections and published in formats such as CSV or JSON. We will use the term data set and table interchangeably for any dataset in which a set of (not necessarily named) attributes can be identified. We consider three important data discovery problems: finding joinable tables (Section 2), finding unionable tables (Section 3), and creating an organization over a massive collection of tables (Section 4). We consider solutions that apply specifically to Open Data (given the characteristics of Open Data) versus those developed for other types of data such as enterprise data or web tables (meaning tables extracted from web pages). We conclude with a discussion of open problems in data discovery.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

2 Finding Joinable Tables

One of the basic ways of finding new tables is to start with an existing table (which we will call a query table) and look for other tables that join with the query table. Before defining this problem precisely and presenting a solution for finding joinable tables in Open Data, we motivate this problem with an example.

Example 2.1: A journalist is working on a piece about the state of industrial research in Canada. Being knowledgable about the topic, he searches using the keyword "NSERC"[1] on *open.canada.ca*, the Canadian Open Data portal, to find data sets related to the disbursement of government funding for scientific research. The keyword search returns tables on individual funding disbursements and the industry partners associated with some of the disbursements.[2] The search results are interesting, and include detail on the funding amount, research topic, proposal keywords, along with other information. As one example, by joining and processing the tables on funding disbursements and industry partners, the journalist creates a table on companies and the total funding received from NSERC, in 2004, as shown in Table 1.

company name	total funding amount	number of fundings	...
Bell Canada,	4,505,988	26	
Kruger Inc.	1,551,303	7	
Pfizer Canada Inc.	464,725	2	
Bombardier Inc.	120,000	1	

Table 1: The query table for finding joinable tables for attribute "company name"

The journalist is not yet satisfied because all the information in the table is related to the funding alone without information on the industry partners other than their names. He wants to look into other information about these companies that could potentially paint a more enlightening picture about why companies are receiving grant funding. In this case, the journalist does not know what specific information might be available and does not know what search keywords to use. The keyword search feature on *open.canada.ca*, and other similar Open Data portals, only allows the user to search for keywords against the accompanying metadata, which is typically human-curated and may not contain many (or any) of the data values in the data set. So the journalist cannot just use the company names as the keywords to find tables that contain those company names.

The journalist wants to find *joinable tables*, other tables that join with the table he has already created on company names. So how do we build a search engine for joinable tables? The problem can be formulated as a classic *set overlap search* problem by considering attributes as sets, and joinable values as overlap between sets. For example, Table 1 has an attribute "company name" which contains the set of values: `Bell Canada`, `Kruger Inc.`, `Pfizer Canada Inc.`, and so on. To find additional attributes about the companies the journalist wants to find tables that have an attribute that joins with these values.

2.1 Set Similarity Search

Set overlap search is an instance of the *set similarity search* problems that have been studied extensively [5, 13, 29, 42, 43]. However, these solutions focus on relatively small sets such as keywords, titles, or emails. The proposed algorithms typically use inverted indexes, and their runtimes are typically linear in the set sizes (both the query set and the result sets). However, Open Data has very large set sizes in comparison to keywords or titles: based on a crawl of 215,393 CSV tables from U.S., U.K., and Canadian Open Data portals as of March 2017, we found the average set size is 1,540 and the maximum size is 22,075,531. In comparison, typical data

[1]Natural Sciences and Eng. Research Council of Canada is the funding agency for sciences (including CS) in Canada.

[2]*https://open.canada.ca/data/en/dataset/c1b0f627-8c29-427c-ab73-33968ad9176e*

sets used in previous work for set similarity search [5, 43], have much smaller set sizes under 100. Thus, the existing search algorithms that work for keyword and string search may not be suitable for finding joinable tables in Open Data, because Open Data sets could contain thousands or even tens of thousands of values.

Most of the previous work in set similarity search proposed algorithms that find exact results. In addition to exact approaches, some research has considered solving the set similarity search problem approximately. Interestingly, research in approximate approaches pre-date that of the exact approaches we mentioned above, and started outside of the database research community, in web search engines [9]. The most popular recipes for approximate set similarity search use locality sensitive hashing (LSH) [3, 18, 23, 30, 40]. The reason for LSH's popularity is likely due to its simplicity: it is data-independent, requires only a few parameters, and can be implemented using off-the-shelf hash table code libraries. Most importantly, LSH also scales very well with respect to the size of sets, because each set is stored as a "sketch" – a small, fixed-size summary of the values (e.g., MinHash [8] or SimHash [11]).

The main justification for an approximate algorithm is that users are often willing to sacrifice a certain degree of accuracy in the search results to receive a significant speed-up in response time. Typically, an approximate algorithm returns much faster than an exact algorithm for the same query, and uses less resources (such as memory), because of the use of sketches instead of the actual data values. Thus, a search engine may respond quickly with approximate results first, and optionally provide the exact result only when required.

2.2 LSH Ensemble

We proposed LSH Ensemble, an LSH-based algorithm for approximate set overlap search, that can be used for finding joinable tables in Open Data [44]. Our work addresses two important research gaps in existing approximate set similarity search algorithms that were limiting their usefulness on Open Data. The first gap is in defining an appropriate similarity function. Most of the proposed LSH algorithms only support symmetric normalized similarity functions, such that $f(A, B) \to \mathbb{R}[0, 1]$, and $f(A, B) = f(B, A)$ regardless of the order of input. Examples of such similarity functions f are Jaccard and Cosine. The normalized version of set overlap is Containment similarity which in contrast is not symmetric:

$$Containment(Q, X) = \frac{|Q \cap X|}{|Q|} = \frac{Overlap(Q, X)}{|Q|} \to \mathbb{R}[0, 1] \tag{1}$$

where Q is the query set and X is a potential result set. Due to its asymmetry, containment similarity cannot be used by most existing LSH algorithms with the exception of Asymmetric Minwise Hashing (AMH) [40]. We used AMH on Open Data and observed that its accuracy deteriorates quickly when applied to data with a large skew in set cardinality size. We show the Zipfian distribution of attribute sizes in Open Data in Figure 1.

The large difference in set cardinalities is in fact the reason for choosing containment over other, better supported, similarity functions like Jaccard. When the query is large, the Jaccard similarity may be very small, even for sets that are fully contained in the query – sets we would want to return as fully joinable. The cardinality difference also leads to the second major gap in using LSH algorithms for searching Open Data: the accuracy of LSH decreases with skewness. AMH transforms Containment to Jaccard by making all sets the same size as the largest set. This approach pads small sets with added values that do not exist in any real sets and then build sketches on the padded sets. We have shown that in Open Data where the largest set can be much larger than the smallest sets, AMH is very inaccurate.[3] An analysis for this effect is provided in the full version of our paper [45]. In Open Data, the set size distribution is close to Zipfian (with massive numbers of small sets), so the vast majority of sets will have very low accuracy sketches by using padding, and this is unacceptable even for approximate search. Our work addresses both of these gaps. We convert Containment to Jaccard, which is a normalized symmetric similarity function, using a well-known transformation [14].

[3]It is possible to over-compensate by using very large sketches, however, this just defeats the purpose of using sketches.

Figure 1: Set size distribution of sets from Canada, US, and UK Open Data

$$Jaccard(Q, X) = \frac{|X \cap Q|}{|X \cup Q|} = \frac{Containment(Q, X)}{|X|/|Q| + 1 - Containment(Q, X)} \tag{2}$$

This makes it possible to adapt MinHash LSH using Jaccard to use Containment. However, the main problem is how to set the value of $|X|$, which is unknown at query time. Our approach uses an approximation, which goes hand-in-hand with reducing the effect of a skewed set size distribution that heavily impacts the accuracy of LSH algorithms. We partition all sets into disjoint buckets by their set size ranges – small sets will not be placed in the same buckets as large sets – and then build an LSH index for each bucket and use the smallest set size in each bucket as an approximation for $|X|$. The effect of partitioning is that as the cardinality difference within each bucket is smaller, the approximation is more accurate. Using this intuition, we developed an optimal partitioning strategy for Zipfian distributions of cardinality sizes [44]. Furthermore, since the query size $|Q|$ also affects the transformation from Containment to Jaccard similarity, and it is only given at run time, we use LSH Forest [3] so the LSH index can be tuned dynamically at run time for different query sizes.

2.3 Open Data Search Engine

Using LSH Ensemble, we built a search engine for Canadian Open Data [46], available at *demo.findopendata.com*.

Example 2.2: The journalist in our example could use this search engine to find tables that join with his query table in Table 1. As an example, he could find a table on contributions to political candidates[4] that lists every contribution made from companies to every political candidate from 2000 to 2004. By joining this table with the query table (and aggregating), the journalist could create Table 2.

company name	# of grants	total funding	total contribution
Bell Canada	26	4,505,988	22,385.22
Kruger Inc.	7	1,551,303	17,500
Pfizer Canada Inc.	2	464,725	11,100
Bombardier Inc.	1	120,000	5,000

Table 2: The result after joining the query table with a search result.

[4]https://open.canada.ca/data/en/dataset/af0b8907-8234-4fa3-b3be-f4406a9aacb8

This resulting table is a product of very low effort by our journalist, who otherwise would rely on serendipity, significant knowledge of available data, or labor intensive search. Using our tool, the journalist unlocks a level of government transparency that was otherwise lost in the sea of Open Data. The journalist's work might contribute to better public understanding of disbursement of government funds, enabling stronger participatory governance.

3 Finding Unionable Tables

Some data analytics tasks require building master lists of tuples by unioning relevant tables. For instance, to analyze the state of greenhouse gas emission, a data scientist might need to build a list of various geographical locations and their gas emission indicators. A simple keyword search for "greenhouse gas emission" in the Canadian Open Data portal returns an overwhelming number of data sets which need to be manually examined to understand if the information can be meaningfully unioned. Automated solutions for more effectively finding unionable tables from Open Data would make this a less laborious task for data scientists.

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	

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...	

Table 3: Unionable tables from London (top) and Washington State (bottom).

3.1 Attribute Unionability

Defining when two attributes are unionable is not as straight-forward as defining when two attributes are joinable.

Example 3.1: Consider the datasets in Table 3 containing greenhouse gas emission statistics one from UK Open Data for the London area[5] and one built based on the US Open Data for the state of Washington[6]. Clearly, it seems reasonable to consider attributes with a large value overlap (like "Date Year" and "Year") to be *unionable*, but an important question is how much overlap is sufficient? In addition, value overlap is often not required. Attributes (like "Borough" and "County") that have semantic overlap may also be considered unionable. Sometimes an ontology containing both entities and classes can be used to discover that values that do not overlap actually come from the same semantic class (such as a class "Region of Country"). However, ontologies (especially public ontologies) are rarely complete and may have some, but not all values in an attribute. As with value overlap, an important question is how many values need to map to the same semantic class for two attributes to be declared unionable? Consider attributes "Fuel" and "Commodity Type" which are intuitively unionable. The former uses standard generic fuel types, while the latter uses user-defined text descriptions of fuels. It is possible, though unlikely, that a sufficient number of these values are in an ontology but even if we do not have an ontology with sufficient coverage, the words used in these two attributes have a semantic closeness that can be measured using natural language techniques. Intuitively, from our understanding of natural language, we

[5] https://data.london.gov.uk/dataset/leggi
[6] https://catalog.data.gov/dataset/2015-greenhouse-gas-report-data

recognize that the values in “Fuel” and “Commodity Type” appear frequently in the same context in natural language. Again, the challenge is to define a principled mathematical framework for justifying how close the words in two attributes need to be before the attributes are considered unionable.

This example illustrates some measures that can be used to quantify attribute unionability. Indeed, table union has been considered for web tables (meaning tables extracted from web pages [10]), using set overlap or other similarity functions over attribute values (Jaccard, tf-idf, etc.) or by mapping values to ontologies [10, 27, 39] These heuristic approaches also leverage attribute names which can be very reliable in web tables. However, to really help our data scientist, we would like a principled way of turning a similarity measure into a hypothesis test: how likely is it that these two attributes contain values draw from the same common domain?

3.2 Table Unionability

Once we have a principled way of understanding how likely it is that two attributes are unionable, we need a way of evaluating how likely it is for two tables to be unionable.

Example 3.2: A data scientist who uses one of the datasets in Table 3 as a query table will receive in response a ranked list of tables that are unionable with her query by just using attribute unionability. However, if she uses a different query table, perhaps one containing only French words as values, over a corpus of English Language tables (where only an English ontology and word embeddings are used), it is not meaningful to return the nearest (closest) table if is too dissimilar to be unioned. So a similarity measure alone is not enough. Moreover, if her query is a table like *London Energy and Greenhouse Gas Inventory* containing a subset of strongly unionable attributes with a candidate table *State of Washington Greenhouse Gas Report* in the repository (like “Date Year” with “Year”, “Borough” with “County”, and “Fuel” with “Commodity Type” of Figure 3) as well as non-unionable attributes (like “Sector”), we would want the search engine to return unionable tables with a maximal alignment indicating which attributes can be unioned.

From this example, we can add to our requirements for a search engine that it automatically selects which measure is most suitable for estimating the unionability of a specific pair of attributes. In addition, we would like a search engine that not only finds tables that are unionable on all their attributes, but rather one that retrieves tables unionable on a subset of attributes. To do this, we also need to be able to pick a maximal set of attribute pairs that can be meaningfully aligned (unioned) in two tables.

3.3 Table Union Search

Our solution estimates the likelihood that two attributes contain values that are drawn from the same domain - a problem we call *attribute unionability* [35]. We defined three types of domains for attributes and three statistical tests (*set-unionability* for overlap based on values, *sem-unionability* for overlap based on mappings into an ontology, and *NL-unionability* which is natural-language based). These tests evaluate the hypothesis that two attributes come from the same domain. Specifically, in set-unionability and sem-unionability, we argue that the size of the syntactic or semantic overlap of attributes follows a hypergeometric distribution. This allows us to compute the likelihood of two attributes being drawn from the same domain as the Cumulative Distribution Function of the hypergeometric distribution. Given the cardinalities of A and B, n_a and n_b, the intersection size t, and the size of their domain n_D, the likelihood of set-unionability is defined as follows.

$$\mathbf{U}_{\text{set}}(A, B) = \sum_{0 \leq s \leq t} p(s | n_a, n_b, n_D) \tag{3}$$

To compute set-unionability, we need to know the size of the domain, n_D. In the context of Open Data, it is impractical or impossible to know the true domain. Thus, we make a practical assumption that the domain is approximately the disjoint union of A and B, and therefore $n_D \simeq n_a + n_b$. Similarly, we define the likelihood of the semantic unionability of A and B based on the size of intersection of the ontology classes that their values are mapped to.

In NL-unionability, we model an attribute with a multivariate normal distribution on the word embedding vectors ($\vec{v}$) of its values centered around μ_A with some covariance matrix Σ_A.

$$v \in A \implies \vec{v} \in \mathcal{N}(\mu_A, \Sigma_A) \tag{4}$$

To build such a distribution, we leverage pre-trained word embedding vectors [25]. We call μ_A the *topic vector* of an attribute. Two attributes are NL-unionable if they are sampled from the same distribution. The distance between the topic vectors of attributes (the estimated mean of the set of embedding vectors) is an indicator of whether they are drawn from the same domain. We apply Hotelling's two-sample statistics (T^2) to undertake tests of the distances between the topic vectors of attributes. The distance of the mean of multivariant distributions is inversely proportional to T^2 statistics and follows an F-distribution. This allows us to define NL-unionability as the Cumulative Distribution Function (**F**) of the distance between topic vectors.

$$\mathbf{U}_{\text{nl}}(A, B) = 1 - \mathbf{F}(T^2, n_a + n_b) \tag{5}$$

To automatically select the best measure for each pair of attributes in a repository, we also developed a new distribution-aware measure, *ensemble attribute unionability*. We defined the notion of *goodness* for a unionability score generated by each measure. Intuitively, given a measure and its corresponding unionability score for a query and a candidate attribute, goodness estimates the likelihood of not finding another attribute that has higher unionability with the query in a given corpus. This allows the selection of the measure that provides the strongest signal, or goodness, within a repository. We developed LSH-based indexes that permit us to efficiently retrieve an approximate set of candidate unionable attributes. These indexes allow our search engine to scale to large repositories of Open Data.

To perform search, we need to define a unionability score for table pairs. Ideally, the most unionable tables share a large number of highly unionable attributes with a query table. However, in reality, candidate tables might share a large number of weakly unionable attributes or a few strongly unionable attributes with a query. We defined an *alignment* of two tables as a one-to-one mapping between subsets of their attributes. Assuming the independence of unionability of different attribute pairs in an alignment, we defined the probability of unionability of two tables as the product of their attribute unionability. Note that table unionability is not monotonic with respect to the number of attributes in a mapping in an alignment. Thus, we need to define a meaningful way of comparing the unionability between sets of attributes of different sizes. We presented the *goodness* of an alignment as the likelihood of the alignment being the most unionable of the alignments of a specific size in a repository. Through this distribution-aware algorithm, we were able to find the optimal number of attributes in two tables that can be unioned [35].

Since there is no benchmark available for table union search, to evaluate accuracy, we built and publicly shared an Open Data table union benchmark. We believe table union search is an important tool enabling cross-publisher analyses by letting data scientists quickly find candidate data sets from different publishers that may have been gathered using similar protocols.

4 Data-driven Organization of Open Data

While the volume of Open Data continues to grow, Open Data repositories lack connectivity and a unified organization. Thus, the world's Open Data sets are typically hosted in isolated silos, each with locally and manually curated organizations. In this section, we present some research directions towards creating federated organizations over distributed Open Data repositories. Such organizations can be valuable in helping data scientists navigate collections of Open Data gather from different publishers.

4.1 Connectivity (Linkage) Graph

We have presented data-driven methods to infer linkages (similarity based on joinability or unionability) between data sets. This allows users to search, discover, and combinie data sets from different repositories. The Open Data search engine (Section 2.3) allows interactive navigation of repositories by join linkages [46]. During navigation, joinable data sets can be integrated and the result of join used to create new join linkages. In addition to such dynamic (query-driven) connectivity discovery, a user may want a navigation plaform that lets her view the entire repository abstractly and navigate progressively into parts of the repository of interest.

4.2 Hierarchical Organization

Good publishing practice dictates that Open Data should be published with metadata. Typical metadata files contains unstructure or semi-structured information that might include the authoring institution of the data set, the date-time of the creation of the data set, and a set of *tags* or *keywords*. Open Data portals[7] currently use these tags for search and navigation. Since the metadata is manually created by data content authors, it is often incomplete, inconsistent, and may exhibit different levels of semantic granularity for different data sets.

Example 4.1: Consider our journalist studying the state of industrial research in Canada from Example 2.1. He starts his research with *NSERC industry partner awards*[8], government awards given to industry-university collaborations. The schema of the data set consists of university institutions, departments, names of the researchers, and industry company names. Here is an incomplete list of the tags of this data set in the metadata: `NSERC`, `award`, `fellowship`, `training`, `postgraduate`, `postdoctoral`, `university`, `college`, `innovation`, `industry`, `science`, `engineering`, `technology`. These tags represent different aspects of the data set (like `science` and `training`) and semantics at different levels of granularity (like `NSERC`, a specific award-agency and `award`, a more abstract notion). A search engine on metadata could use a set similarity measure such as Jaccard similarity over tags to find related data sets. For example, a data set with tags *University and college; award and expenditures*[9] shares some similarity with the tags of the NSERC industry partner awards mentioned above. Interestingly, another data set on academic grants and expenditures has only *one* tag: `Summary tables`. One would not be able to discover the connection between such a data set and the data set on industry partner awards by relying only their respective metadata.

While metadata tags alone may be insufficient to organize repositories, we believe these tags, together with connectivity graphs, can be leveraged to create hierarchical organizations. Traditionally, taxonomy induction and ontology construction approaches extract labels (classes in ontologies) from text documents and use hand-constructed or automatically discovered textual patterns, as well as label co-occurrences to infer taxonomies on extracted classes [26, 41]. In the data set organization problem, the challenge is to construct taxonomies over groups of data sets by leveraging metadata and connectivity graphs.

5 Open Problems

Data discovery has been studied mostly within the enterprise, over corporate data lakes, or over web data (for example, web tables), with little research on Open Data repositories. We briefly review some of this work and highlight implications for Open Data management and open problems that remain.

[7]https://open.canada.ca/en and https://ckan.org/

[8]https://open.canada.ca/data/en/dataset/c1b0f627-8c29-427c-ab73-33968ad9176e

[9]https://open.canada.ca/data/en/dataset/66def988-48ca-4cc5-8e0b-19d9b73f6266

5.1 Finding Connections between Data Sets

Keyword-based search. To find relevant data sets to keyword queries, OCTOPUS performs web document-style search on the content and context of web tables [10]. These tables are then clustered into groups of unionable tables using common similarity measures. 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 an attribute of the query [37]. Keyword-based table search algorithms take time linear to the size of attribute values and have been shown to perform well on web tables. It is an open question if this technique could be made to scale over Open Data. For example, the WDC Web Table Corpus is massive (over 50 Million tables), but the average attribute size is 10 and the largest attribute has only 17,030 values [28], in contrast, in the Open Data we have reported on in this paper, the average set size is 1,540 and the maximum size is 22,075,531.

Attribute-based search. Existing work on the problem of finding linkage points among data sources on the web propose lexical analyzers and similarity functions accompanied by a set of search algorithms to discover joinable attributes [20]. As with our work, linkage point discovery does not require the existence of schemas, but it is unclear the performance over data with the massive sizes and skewed distributions of Open Data. Ilyas et al. [22] learn syntactic patterns of an attribute and uses this to perform attribute search. To search repositories, they adapt MinHash LSH to use these patterns. The system has only been evaluated over repositories of thousands of attributes, but it is a very interesting open question to see if some of the ideas of LSH Ensemble [44] could be used to made this approach feasible on massive repositories, where individual sets can also be massive.

Schema and table instance-based search. Data Civilizer, proposed after LSH Ensemble, also employs LSH to build a linkage graph between data sources in a data lake [12]. Extending ideas from enterprise-level data integration systems which compute possible join paths within large data sources (large relational or semi-structured databases) to suggest mappings between data sources [15], Data Civilizer computes possible join paths (foreign key paths) between data sources that can be used in queries [12]. Unlike our work which supports computation of linkages at interactive speed (the search problem), this work materializes a linkage graph. The system provides a declarative source retrieval query language that enables search over this materialized graph [32]. Very recent related work adds semantic linkages using word embeddings on attribute names in schemas (which are rich in enterprise data) [17] and dynamically maintains the materialized graph [16]. Other systems for data discovery over data lakes include Kayak [31] (which also uses materialized statistical profiles to discover or suggest data sets).

Das Sarma et al. [39] define joinability and unionability for web tables by assuming that each table has a subject column that contains entities that the table is about. Their entity-complement approach uses an ontology to identify entities and the accuracy depends on the coverage of this ontology. We have shown that in Open Data the coverage of open ontologies like YAGO can be very low (on our corpus less than 13% of string values can be mapped to the ontology) [35].

A lot of research has been done on scalable schema matching and ontology alignment, where the problem is to match pairs (or small sets) of large schemas (ontologies). However, schema matching and ontology alignment have not generally been studied as search problems over massive collections of Open Data sets. Despite the rich literature, there are many open problems. Here we suggest a few.

With the exception of set-unionability, the similarity measures used in table union work (ours and unioning of web tables) are not directly applicable to numerical values. Evaluating the unionability of numerical attributes requires first understanding what each attribute quantifies using which measurement unit. For instance, using the header names, attributes *Total Emissions (MT CO2e)* and *ktCO2* might be recognized as unionable and can be consolidated after unit conversion. To make sense of quantities in web tables, Ibrahim et al. [21] represent <measure, value, unit> triples over taxonomies of measures (e.g., physical and monetary). This representation can be augmented with distribution properties of an attribute. Once numerical attributes are canonicalized, we need a way of computing the likelihood of numerical attributes being drawn from the same domain and then we could incorporate this idea into table union search.

In table union search, we used pre-trained embedding models [25] (trained on Wikipedia documents) to compute NL-unionability. This allows NL-unionability to take advantage of embeddings trained on external sources of knowledge and results in high accuracy of search. However, these embeddings are limited to the vocabulary of the training corpus. Training word embeddings on text documents and knowledge bases has been extensively studied [6, 34, 36]. With the abundance of Open Data, training domain-specific and generic embedding models on Open Data using metadata, schema and constraints is an interesting research problem which has applications beyond table union search.

5.2 Data Set Organization and Management

Goods organizes data sets by building data set profiles that leverage query logs and metadata [19]. These profiles are exposed to users through portals that allow faceted search on the metadata of data sets. Open Data does not come with query logs but providing faceted search over Open Data is a desirable goal. 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 [4]. 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.

The VADA project considers source selection (data discovery) using detailed models of user preferences [1]. Source selection is a classical data integration (and information retrieval) problem that has been studied for decades. Relevant data sets (sources) are identified (typically using keyword search or exploration) and selection is done based on a model of user preferences [1] or models of the cost and/or value of using a source [38]. Source selection usually is a task following data set search, but nonetheless something that will be important to consider over Open Data.

DataHub [7] provides efficient management of versioned data in a collaborative environment that has a community of users. In contrast, Open Data is published out in the wild typically without strict versioning. An interesting open problem is detecting near versions of data and providing an efficient way of on-boarding open data into a data collaboration platform like DataHub. DataHub supports IngestBase, a declarative data ingestion platform where enterprise users can create sophisticated data ingestion plans to ingest operational data from sensors or other external sources (financial feeds, social media, etc.) [24]. Open Data however requires a data-driven way of discovering good ingest plans as it is published in a way that basic elements (like the schema or number and type of attributes) may change.

6 Conclusion

We have taken a detailed look at data discovery over Open Data in a way that will be both efficient and effective given the characteristics of both individual data sets and collections of Open Data. We have presented new table join and table union search solutions that provide interactive search speed even over massive collections of millions of attributes with heavily skewed cardinality distributions. We have also presented a research vision for creating data-driven organizations of data repositories and extending the functionality and reach of our search algorithms. Though examples, we showcased how powerful search solutions make finding Open Data an easier, less daunting task – we hope making Open Data a bit more transparent and useable by the public.

Acknowledgments This work was partially supported by NSERC.

References

[1] E. Abel, J. A. Keane, N. W. Paton, A. A. A. Fernandes, M. Koehler, N. Konstantinou, J. C. C. Ríos, N. A. Azuan, and S. M. Embury. User driven multi-criteria source selection. *Inf. Sci.*, 430:179–199, 2018.

[2] J. Attard, F. Orlandi, S. Scerri, and S. Auer. A systematic review of open government data initiatives. *Government Information Quarterly*, 32(4):399–418, 2015.

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

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

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

[6] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[7] A. P. Bhardwaj, S. Bhattacherjee, A. Chavan, A. Deshpande, A. J. Elmore, S. Madden, & A. G. Parameswaran. Datahub: Collaborative data science & dataset version management at scale. In *CIDR*, 2015.

[8] A. Z. Broder. On the Resemblance and Containment of Documents. In *Compression and Complexity of Sequences*, pages 21–28, 1997.

[9] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.

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

[11] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.

[12] 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.

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

[14] S. Duan, A. Fokoue, O. Hassanzadeh, A. Kementsietsidis, K. Srinivas, and M. J. Ward. Instance-based matching of large ontologies using locality-sensitive hashing. In *ISWC*, pages 49–64, 2012.

[15] 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.

[16] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. Aurum: A data discovery system. In *ICDE*, 2018. To appear.

[17] R. C. Fernandez, E. Mansour, A. Qahtan, A. Elmagarmid, I. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Seeping semantics: Linking datasets using word embeddings for data discovery. In *ICDE*, 2018. To appear.

[18] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *PVLDB*, pages 518–529, 1999.

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

[20] 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.

[21] Y. Ibrahim, M. Riedewald, and G. Weikum. Making sense of entities and quantities in web tables. In *CIKM*, pages 1703–1712, 2016.

[22] A. Ilyas, J. M. F. da Trindade, R. C. Fernandez, and S. Madden. Extracting syntactic patterns from databases. *CoRR*, abs/1710.11528, 2017.

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

[24] A. Jindal, J. Quiané-Ruiz, and S. Madden. INGESTBASE: A declarative data ingestion system. *CoRR*, abs/1701.06093, 2017.

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

[26] J. Lehmann and P. Hitzler. Concept learning in description logics using refinement operators. *Journal of Machine Learning*, 78–203(1), 2009.

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

[28] 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.

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

[30] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[31] A. Maccioni and R. Torlone. Crossing the finish line faster when paddling the data lake with kayak. *PVLDB System Demonstration*, 10(12):1853–1856, 2017.

[32] E. Mansour, A. A. Qahtan, D. Deng, R. C. Fernandez, W. Tao, Z. Abedjan, A. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Building data civilizer pipelines with an advanced workflow engine. In *ICDE System Demonstration*, 2018. To appear.

[33] J. Manyika, M. Chui, P. Groves, D. Farrell, S. V. Kuiken, and E. A. Doshi. Open data: Unlocking innovation and performance with liquid information. McKinsey Global Institute, 2013.

[34] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NISP*, pages 3111–3119, 2013.

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

[36] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

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

[38] T. Rekatsinas, A. Deshpande, X. L. Dong, L. Getoor, and D. Srivastava. Sourcesight: Enabling effective source selection. In *SIGMOD*, pages 2157–2160, 2016.

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

[40] A. Shrivastava and P. Li. Asymmetric minwise hashing for indexing binary inner products and set containment. In *WWW*, pages 981–991, 2015.

[41] R. Snow, D. Jurafsky, and A. Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *ACL*, pages 801–808, 2006.

[42] 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.

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

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

[45] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller. LSH ensemble: Internet scale domain search. *CoRR*, abs/1603.07410, 2016.

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