# Interactive Navigation of Open Data Linkages

Erkang Zhu
University of Toronto
ekzhu@cs.toronto.edu

Ken Q. Pu
UOIT
ken.pu@uoit.ca

Fatemeh Nargesian
University of Toronto
fnargesian@cs.toronto.edu

Renée J. Miller
University of Toronto
miller@cs.toronto.edu

## ABSTRACT

We developed Toronto Open Data Search to support the *ad hoc*, interactive discovery of connections or *linkages* between datasets. It can be used to efficiently navigate through the open data cloud. Our system consists of three parts: a user-interface provided by a Web application; a scalable backend infrastructure that supports navigational queries; and a dynamic repository of open data tables. Our system uses LSH Ensemble, an efficient index structure, to compute linkages (attributes in two datasets with high containment score) in real time at Internet scale. Our application allows users to navigate along these linkages by joining datasets.

LSH Ensemble is scalable, providing millisecond response times for linkage discovery queries even over millions of datasets. Our system offers users a highly interactive experience making unrelated (and unlinked) dynamic collections of datasets appear as a richly connected cloud of data that can be navigated and combined easily in real time.

## 1. INTRODUCTION

Toronto Open Data Search (our system) supports the *ad hoc*, real time discovery of connections or *linkages* between datasets. Consider a data scientist exploring a dataset of interest. With a click of a button, our system will let her search a massive repository datasets to find other datasets that join with her dataset. Our system provides interactive response time even if the scientist's dataset is massive and even if the repository is dynamic (new datasets can be added and searched in real time).

Our demonstration will let VLDB participants browse an open data repository by topic; select a dataset of interest; perform basic data management operations on the dataset (for example, projecting out attributes or selecting a join attribute); and interactively find linkages with other datasets. Once a linkage is selected, a user can navigate new datasets using joins. Joined datasets can provide new insights for data scientists who may want to share their insights with collaborators. Our system provides sharable links to all discovered multi-dataset resources. Our system is integrated with existing data analysis tools to provide an efficient data discovery and data analysis experience for users.

In Section 2, we overview the technical innovations that enable us to compute linkages dynamically at interactive speeds. In Section 3, we discuss the features and architecture of our system. We conclude with some interesting case studies that illustrate the features of our system and a discussion of the demonstration experience.


*Proceedings of the VLDB Endowment,* Vol. 10, No. 12


## 2. TECHNICAL OVERVIEW

With the rapidly growing volume of open data published on the Web, the problem of searching and finding related datasets has become an important research problem. When available, metadata (attribute names) can be used to find related datasets [2, 6]. Alternatively, if our goal is to be able to join datasets, we can build a graph of *linkages* (pairs of attributes with high overlap in values or high containment score). Previous work on linkage discovery has shown how to efficiently build a static linkage graph over large collections of data sources (such as DBpedia and Freebase) [3].

In the light of the daily growth in available online data, and the lack of consistent and reliable metadata, our interest is to support *ad hoc* open data search. We do not assume any domain knowledge of how attribute values can be transformed or matched, nor the existence of common metadata names. Instead we search in real time for attributes with high containment score. We refer to this variation of data search or linkage discovery as the *domain search problem* [8].

Our system uses LSH Ensemble [8] a distributed index structure based on locality sensitive hashing (LSH) of minwise hash signatures of the attributes in open data datasets. It is highly scalable and can support efficient search of the open data cloud at Internet scale. Using the Web Data Commons Web Table corpora [5], LSH Ensemble can discover linkages between a user-defined dataset and over 200 million Open Data relational attributes in under 3 seconds [8], with most search response times in the sub-second range.

The remainder of this section will provide an overview of the *domain search problem* and LSH Ensemble.

### 2.1 The Domain Search Problem

A *domain* $\text{dom}(A)$ is simply a collection of values from an attribute $A$ of some relational dataset $T$. Given two attributes $A$ and $B$, we measure their relevance based on the amount of overlap of their respective domains using the set containment score, defined as:

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

To support dataset extension, a user provides a dataset, $T_q$, and an attribute, $A_q$, in the dataset. Our system finds all attributes in the open data cloud that have high set containment score with $A_q$.

DEFINITION 1 (DOMAIN SEARCH PROBLEM). *Given a relational dataset $T_q$ and one of its attributes $A_q$ and a threshold $t^* \in [0, 1]$. Find a set of relevant attributes $\mathcal{A}$ among all the datasets in the open data cloud such that for each $A \in \mathcal{A}$:*

$$\text{containment}(A_q, A) \geq t^*$$

Given a query as a dataset and one of its attributes the answer set to the domain search problem is a collection of datasets and their attributes $\{(T_i, A_i) : 1 \leq i \leq k\}$. The linkages $(A_q, A_i)$ allow us to "navigate" the open data cloud via the relational join (or outerjoin) operator:

$$S = T_q \bowtie_{A_q = A_i} T_i$$

The extended dataset $S$ is an enriched version of $T_q$ with additional attributes from $T_i$. One can continue exploring by using any attributes in $S$ for the next domain search query.

## 2.2 LSH Ensemble

The heart of our system is a highly scalable index structure, LSH Ensemble [8], that allows us to perform domain search at Internet scale. In order to deal with millions of datasets and arbitrarily large domains, we chose to index a data sketch of the domains, rather than the actual data values in the domain. Our index structure uses minwise hash signatures (minhash) [1, 4] as the data sketch.

It is well known that Jaccard similarity, defined as

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

between two sets $A$ and $B$ can be accurately estimated by the minhash signatures of $A$ and $B$. Furthermore, Jaccard-based nearest neighbor queries of minhash signatures can be efficiently evaluated using a locality sensitive hashing (LSH) index of the minhash signatures [7]. LSH Ensemble uses LSH to solve the domain search problem. But due to the unique characteristics of open data, some important innovations were needed.

***Dynamic thresholding***: Jaccard similarity is not a suitable measure of relevance for attribute domains as it is heavily biased to smaller attributes. The domain search problem uses containment score which is better suited as a measure of relevance in open data search [8]. LSH Ensemble uses the relationship between Jaccard similarity and containment score to map the query threshold on containment score $t^*$ to a threshold on Jaccard similarity $j^*$.

$$\text{Jaccard}(A, B) \simeq \frac{\text{containment}(A, B)}{\frac{u}{|A|} + 1 - \text{containment}(A, B)} \quad (1)$$

where $u$ is the maximal cardinality on $B$.

It is worth noting that the dynamic thresholding is guaranteed to never introduce any false negatives. Thus, the overall recall of the search system will not be affected by dynamic thresholding.

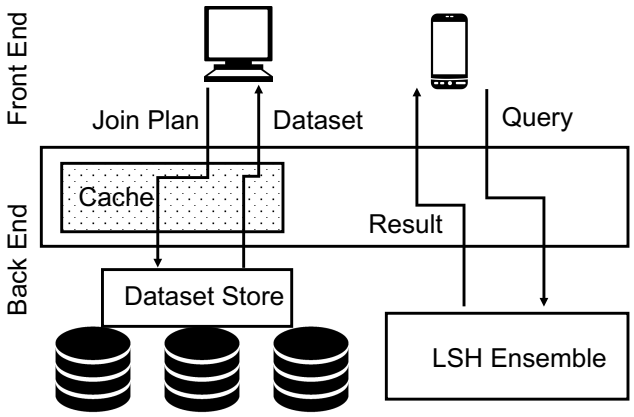


**Figure 1: Architecture Overview of the System**

***Data partitioning*** The distribution of domain sizes (set cardinality) in open data follows a highly skewed Zipfian distribution. This means that the maximal cardinality $u$ in Equation 1 is so large that the Jaccard threshold $j^*$ is too small for a normal LSH index to have any pruning power. Consequently, LSH alone does not adapt well to a highly skewed cardinality distributions, causing the hash look-up to degenerate to linear scan. LSH Ensemble overcomes the problem by partitioning the minhash signatures of domains into an ensemble of $n$ individual LSH indices according to the domain cardinality. Importantly, LSH Ensemble provides an optimal partitioning function that maximizes the overall pruning power of the index and is thus optimal for open data domain search [8].

# 3. THE SYSTEM

In this section, we introduce our system, TORONTO OPEN DATA SEARCH, which supports interactive navigation of Open Data through data linkages found in real time using LSH Ensemble. A demonstration of the system is already online using datasets from the Canadian Open Data[1].

## 3.1 System Overview

The system consists of three components: the front end, the back end, and the dataset store (Figure 1).

The front end runs directly inside the user's browser. It keeps the current state of navigation for the user. Since navigation is done by joining tables on linkages, a navigation state is the sequence of relational joins that leads to the current joined dataset that we call the *join plan*. The front end gets the result of the join plan from the back end and displays the result to the user.

Changes to the join plan can be made through adding additional datasets joined along a selected linkage. This is done by selecting an attribute $A$ and issuing a query request to the back end that returns a ranked list of all attributes that link to $A$. The user can then pick a dataset from the query result and add it to the current join plan, thereby making a new step in the navigation.

The back end is responsible for evaluating the join plan and executing queries using LSH Ensemble index. Its most important characteristic is that it is functional or stateless.

[1] *https://rjmillerlab.github.io/lshensembledemo/*

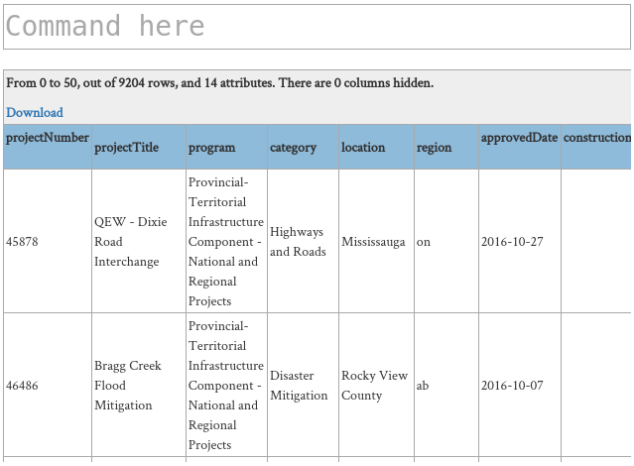


**Figure 2: Viewing the dataset table**

The back end it does not keep the user's current navigation state (join plan), which is solely the responsibility of the front end, and always attempts to evaluate the join plan completely. This allows the back end service to be deployed distributively on many servers. Optionally, the back end uses a memory cache to store the result of each join. This permits optimization of the evaluation if queries go to the same server and permits fast undo operators to recover a previous join plan. The use of memory cache does not affect the functional characteristic of the back end, since it can always evaluate a join plan from scratch.

The dataset store is a distributed disk-based key-value store responsible for retrieving datasets given their unique identifiers. Every attribute also has a unique identifier. MinHash signatures are computed for every attribute domain whenever a dataset is loaded. When available, we display metadata (free text descriptions of tables) to help users in selecting linkages.

Because all components can be deployed in a distributed environment, our system can scale out to handle massive numbers of open data datasets.

## 3.2 Features and User Experience

In this section, we present the features of our system. To begin, a user selects an example dataset to begin the navigation. The front end provides a list of datasets together with descriptions.

Once a dataset is selected, the front end displays the dataset as a table, as shown in Figure 2. By inspecting the data values, the user can gain direct insight into the data, that the metadata itself (such as column names which are not always present) cannot always provide.

Datasets from may contains many attributes. This can hinder the user from inspecting the table effectively. We provide a command line interface to allow the user to modify the table view. One such example is the column highlight command (`hi`), as shown in Figure 3. This command takes a set of keywords and highlights all columns whose name contains the keywords (using substring matching). Another example is the column filter command (`fi`) that allows the user to see only the columns selected.

To find linkages from an attribute, the user clicks the attribute header. The front end displays the search result in a

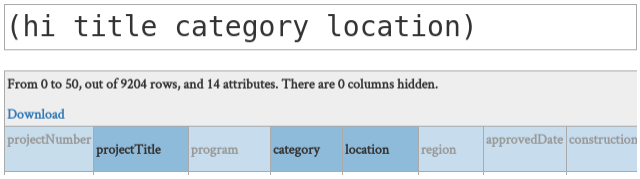


**Figure 3: Highlight columns**


**Figure 4: Searching for dataset linkages**

list, as shown in Figure 4. The list displays the names of the datasets and the attributes that are linked with the query attribute, ranked by their containment scores. By clicking any search result, the user can join the current dataset with the selected dataset on the selected attribute.

After joining with a new dataset, the user can perform further modification of the table view using `hi` or `fi` commands. As shown in Figure 5, the current dataset now contains attributes from two separate datasets, indicated using different colors in the headers. At this point, the user have successfully made his/her first step in the navigation. Further navigation steps follow a similar pattern, and the current state of navigation is displayed on a side panel, as shown in Figure 6.

Apart from navigation, our system also facilitates collaboration between different users through shareable URL and dataset export. A join plan can be serialized as part of a URL, so the user can choose to share the URL with a collaborator enabling her to see the same dataset. The collaborator can continue to navigate from the shared state (by joining new tables), while the original user may continue independently to explore the data in a different direction. The concurrent navigation of multiple users allows us to gather interesting usage data about the datasets that may be used for query recommendation. Lastly, every dataset view has a download option. This lets a user export the current view of the datasets as a CSV file that can be used for further analysis by importing it into a visualization or machine learning tool.

# 4. CASE STUDIES

We highlight the search functionality of Toronto Open Data Search through two scenarios in which given a dataset, our system can find joinable datasets in Canadian Open Data containing interesting and relevant attributes.

***Linkage between police and homicide:*** In the first scenario, the data scientist is working with a dataset, called

```
(fi title category location Total)
```

From 0 to 50, out of 6387 rows, and 39 attributes. There are 33 columns hidden.

Download

| projectTitle | category | location | totalEligibleCost | Total - All Returns/Total - Toutes les déclarations | Total Income - All Returns/Revenu total - Toutes les déclarations |
|---|---|---|---|---|---|
| Road Rehabilitation on Route 1 - Trans-Canada Highway and Route 2 - Pitts Memorial Drive | Highways and Roads | St. John's | 26733850.0 | 73590 | 4070114000 |
| Fort Amherst Sanitary Sewer Outfall Diversion | Wastewater | St. John's | 3173922.0 | 73590 | 4070114000 |
| Filter Cleaning Unit | Public Transit | St. John's | 90683.0 | 73590 | 4070114000 |
| Sewer Lining | Wastewater | St. John's | 1813670.0 | 73590 | 4070114000 |

**Figure 5: After joining with a new dataset**

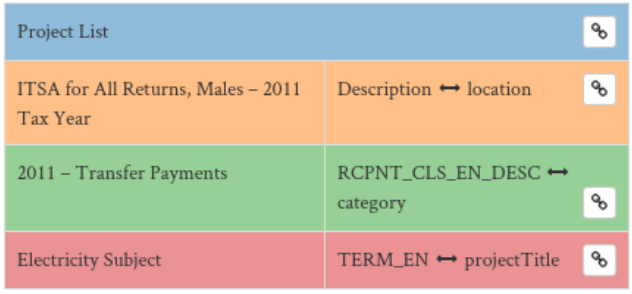


**Figure 6: Side panel showing the current state of navigation**

`Homicide survey`. This dataset provides yearly data on firearm-related homicides, including attributes that contain the name and numerical values of the indicators such as `Number of homicides` and `Percentage of homicides`. Performing search on the `year` attribute of this dataset, our system finds datasets that contain interesting attributes such as `Total number of officers per year` and `Population per officer per year`. Furthermore, searching on the attribute `name of indicators` results in other homicide survey datasets, such as `Number and percent of homicide victims, by sex and age group`, that contain attributes with more specific statistics. In the demonstration, we will highlight some interesting trends between the number of homicides and number of police officers that can only be identified by linking several open datasets.

***Multiple linkages from Canadian infrastructure to tax:*** In the second scenario, a data scientist is working with a dataset called `Infrastructure Canada Project` that contains a list of infrastructure projects across Canada. This dataset contains `title`, `category`, `program`, `location` and other attributes of projects. Attribute `location` refers to the region where the project takes place. Upon search for linkages on attribute `location`, our system returns dataset `ITSA for All Returns` that contains aggregated attributes on financial and tax related indicators, such as `total income` and `total tax return` of residents of each region. By joining such dataset with `Infrastructure Canada Project` dataset, the data scientist can investigate the correlation of the financial status of the residents of different regions where projects are located.

# 5. DEMONSTRATION PROPOSAL

We will demonstrate a fully functional implementation of Toronto Open Data Search as a (mobile friendly) Web application. We will show how Toronto Open Data Search helps a data scientist to interact with and effectively navigate data on the web.

Our demonstration shows the viability of computing linkages in real time over massive data. This is the first demonstration to show that instance-level linkages between attributes can be done accurately at interactive (real time) speeds. The demonstration will highlight the importance of the interactive speed in supporting curiosity-driven exploration of a large repositories of rich structure datasets.

We will show how our persistent URLs allow linked datasets to be shared and imported into visualization tools (plot.ly and others) to quickly plot new trends discovered through linkages. Our demonstration will also allow linkages to be filter by known metadata matching techniques (when attribute names are available) [2]. Of course, such metadata techniques need to be combined with our instance matching to ensure joined datasets contain meaningful results for data scientists.

Toronto Open Data Search is open source using LSH Ensemble as a powerful tool to access open data on the Web. This demonstration illustrates ways that LSH Ensemble can be applied to create an interactive and engaging user experience not before possible. The system is fully implemented. Live system, screencast video, and other information can be found on the project page:
https://rjmillerlab.github.io/lshensembledemo/

# 6. ACKNOWLEDGMENT

This work is partially funded by NSERC and Bell Graduate Scholarship.

# 7. REFERENCES

[1] A. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, pages 21–28, 1997.

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

[3] 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.

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

[5] 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.

[6] O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, and C. Bizer. The mannheim search join engine. In *WWW*, pages 159 – 166, 2015.

[7] A. Rajaraman and J. D. Ullman. *Mining of Massive Datasets*. 2011.

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