RONIN: Data Lake Exploration

Paul Ouellette
University of Rochester
pouellet@u.rochester.edu

Aidan Sciortino
University of Rochester
asciorti@u.rochester.edu

Fatemeh Nargesian
University of Rochester
fnargesian@rochester.edu

Bahar Ghadiri Bashardoost
University of Toronto
ghadiri@cs.toronto.edu

Erkang Zhu
Microsoft Research
ekzhu@microsoft.com

Ken Q. Pu
Ontario Tech University
ken.pu@ontariotechu.ca

Renée J. Miller
Northeastern University
miller@northeastern.edu

ABSTRACT

Dataset discovery can be performed using search (with a query or keywords) to find relevant data. However, the result of this discovery can be overwhelming to explore. Existing navigation techniques mostly focus on linkage graphs that enable navigation from one data set to another based on similarity or joinability of attributes. However, users often do not know which data set to start the navigation from. RONIN proposes an alternative way to navigate by building a *hierarchical structure* on a collection of data sets: the user navigates between groups of data sets in a hierarchical manner to narrow down to the data of interest. We demonstrate RONIN, a tool that enables user exploration of a data lake by seamlessly integrating the two common modalities of discovery: data set search and navigation of a hierarchical structure. In RONIN, a user can perform a keyword search or joinability search over a data lake, then, navigate the result using a hierarchical structure, called an *organization*, that is created on the fly. While navigating an organization, the user may switch to the search mode, and back to navigation on an organization that is updated based on search. This integration of search and navigation provides great power in allowing users to find and explore interesting data in a data lake.

PVLDB Reference Format:
Paul Ouellette, Aidan Sciortino, Fatemeh Nargesian, Bahar Ghadiri Bashardoost, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. . PVLDB, 14(11): XXX-XXX, 2021.
doi:10.14778/3476311.3476364

1 INTRODUCTION

The sheer number of available data sets has fueled the need for data set discovery [1–3, 7, 8]. Yet, despite this research in data discovery and information retrieval, finding relevant data sets for a task is still a challenge. Consider the following scenario.

Example 1.1. A user wants to find data sets about `smart city`*. A search engine, like Google data set search [2], returns more than one hundred data sets for this query that the user has to go through and manually examine for relevance. Similar problems occur with join or union searches, a user may find too many data sets that are hard to explore manually.*

Data set search has been studied as a threshold-based or top-k nearest neighbor search problem. The result of this search is often presented to the user in a flat structure, potentially a long list of data sets. Some data portals may provide limited filtering functionalities on metadata fields. A well-known complementary approach to search is navigation in a linkage graph. The links/edges in a linkage graph indicate relevance (metadata similarity or joinability) of two data sets/attributes. Navigation is done by path traversal in materialized linkage graphs [3] or a chain of searches done on demand [9]. In this model of navigation, users navigate from one data set to another relevant data set. Yet, search and navigation on other forms of data such as products and video contents can be done using directory structures (e.g., Yahoo! and Amazon).

In RONIN, we demo an alternative way to navigate by integrating navigation into search. Moreover, RONIN uses *organizations*, where navigation is done in a hierarchical manner to guide the user to data sets of interest. The root of an organization is the most common and generic topic in a search (query) result. The data sets are located at the most granular levels. Navigating from one topic to a more granular topic means that the user filters some less interesting data sets from the result. This hierarchy is first modeled as a DAG where nodes are sets of data sets and edges indicate the inclusion of data sets of a child in its parents. The nodes are then labeled automatically with relevant topics. Unlike other approaches that provide a single static hierarchy for a data lake [5], RONIN generates organizations on the fly based on user searches. Data set search techniques make sure that the result is relevant to the query (a data set, an attribute, or keywords), and the on-the-fly navigation helps users explore the result, even if it is large.

Example 1.2. The effectiveness of keyword search relies heavily on the selected keywords. However, the user may not be familiar with all topics related to `smart city`*, therefore, may not choose an optimal set of query keywords for her needs. For example, data sets about* green energy *may be found relevant to a* smart city *query, but the user may not know that. A user searching for* smart city *would therefore miss the* green energy *data sets.*

In RONIN, navigation and search are integrated. An organization provides a hierarchy on the data sets retrieved by a query. Using RONIN, at any time during the navigation of an organization, the user can switch back to the search mode. During navigates, the

Proceedings of the VLDB Endowment, Vol. 14, No. 11 ISSN 2150-8097.
doi:10.14778/3476311.3476364

Table 1: Percentage of Data Sets with Metadata Fields.

Name	Descrip.	Attribution	Categories	Tags	Cat. or Tags
100.0%	82.44%	69.2%	86.02%	70.33%	88.9%

user can pick a data set and issue a query using its data (for example, to find joinable data sets) or use its metadata to refine query keywords. This is only possible via a set of optimizations to speed up organization construction on the fly.

We demonstrate RONIN [1], a data lake exploration tool that enables integrated search and navigation. The main component of RONIN is an algorithm for constructing an organization on a set of data sets. An effective organization maximizes the expected probability of finding data sets by navigating this organization. RONIN models navigation as a Markov model. The key idea is to view the organization problem as combinatorial optimization where we want to find the DAG that maximizes effectiveness. We represent the organization using metadata of data sets and adapt our prior work [4] to perform an efficient local search on the space of organizations.

In our demonstration scenario, the attendees impersonate a data scientist. They are free to specify their own query topics. The participants observe first-hand how RONIN efficiently builds a directory structure on their search result and combines navigation and search. We proceed to discuss related work and how RONIN differs from the prior art (§ 2), provide a solution sketch (§ 3), and conclude with a detailed outline of our demonstration (§ 4).

2 RELATED WORK

Our work relates to data set discovery systems that exploit data values [6, 8], metadata [2] or both [1, 3, 7]. Unlike existing navigation work that provides a linkage graph where links indicate one-to-one similarity or joinability of data sets [3, 9], RONIN enables navigation in a hierarchical structure. Unlike navigating a linkage graph, the user does not need to know or find a data set as the start point of navigation. Navigation in RONIN is exploratory and starts from a broad topic which is the root node. The hierarchical structure of RONIN is closest to the hand-curated directory structures of Yahoo! on web pages and amazon on products. Previous work studies the generation of a static organization on a data lake of thousands of data sets [4]. RONIN applies a similar organization and navigation model, from which RONIN differs in three aspects. First, unlike a data lake organization, where nodes are represented with attribute values, RONIN exclusively relies on metadata which is available and clean and additionally results in faster construction time. Second, RONIN organizes the result collection of a search; this collection gets updated as the user performs various searches. Finally, RONIN builds and updates organizations on the fly, as the user discovers data sets. The dynamic and efficient generation of an organization facilitates seamless interaction of search and navigation.

3 SOLUTION SKETCH

Figure 1 depicts RONIN's architecture.

Preprocessing RONIN pre-processes a data lake of 40,431 relational data sets with 780,921 attributes crawled from the Socrata open data portal[2] by first generating the minhash data sketches of attribute values using `LSH Ensemble`[3]. Then, it extracts metadata fields from data sets. The metadata is often available as values for fields: `name`, `description`, `categories`, `attribution`, and `tags`. Table 1 shows the ratio of data sets that have values for each metadata field. A total of 88.9% of the data sets are accompanied by at least one value in `category` or `tag`. Figure 2 shows the distribution of the data sets with values for these two fields. This source of metadata gives us the opportunity to use metadata for search and navigation. RONIN represents each data set with a *semantic vector*. Given a set of metadata fields $\mathcal{M} = \{m_1, \ldots, m_k\}$ for data set D_i, the semantic vector of D_i is computed by averaging the word embedding vectors of tokens in m_j's. RONIN uses the pre-trained vectors of `fasttext`[4] which are trained on the English corpus of Wikipedia. The data sketches and metadata of data sets are stored into a `SQLite` database.

Data Set Search RONIN supports two existing variations of search: keyword search over metadata and search for joinable data sets. Here, we briefly explain these variations. Similar to the idea of semantic vectors, RONIN represents a keyword query with the average vector of the word embedding of query tokens. Then, it returns top-k data sets with semantic vectors similar to the query vector. To speed up the search, upon server startup, we index vectors in a *faiss index*[5] that we later use to pose top-k similarity queries. The profile of a data set enables searching for joinable data sets with any of its attributes. RONIN incorporates `LSH Ensemble` which uses set containment as the measure of joinability and enables high precision containment search over attributes [8].

Keyword search in metadata [2] or joinability/unionability search [6, 8], particularly in a threshold-based paradigm, can result in an arbitrary number of data sets. For a given collection of data sets, $\mathcal{L} = \{D_1, \ldots, D_n\}$, RONIN constructs a hierarchical structure for effective navigation. This structure is constructed using the previous model proposed for navigating data lake attributes [4].

3.1 Navigation in an Organization

An organization is a DAG with nodes that represent sets of data sets. An edge indicates that the data sets in a parent node form a superset of the data sets in a child node. A user finds a data set by traversing a path from a root node to the leaf that contains the data set. To make an organization readable for users, each node is annotated with descriptive text. Each data set $D_i \in \mathcal{L}$ has a set of metadata values that we call a domain and denote by $\text{dom}(D_i)$. For an organization $\mathcal{O} = (V, E)$, let ch(.) be the child relation mapping a node to its children, and par(.) be the parent relation mapping a node to its parents. Each node s corresponds to a set of data sets $\mathbb{D}_s \subseteq \mathcal{L}$. For a leaf node s, the cardinality of $\mathbb{D}_s$ is one. If $c \in ch(s)$, then $\mathbb{D}_c \subseteq \mathbb{D}_s$, we call this the *inclusion* property, and $\mathbb{D}_s = \bigcup_{c \in ch(s)} D_c$. We denote the domain of a node s by dom(s) which is dom($\mathbb{D}_s$) when s is a leaf node and $\bigcup_{D_i \in \mathbb{D}_s} \text{dom}(D_i)$ otherwise.

RONIN represents the domain of a data set with its semantic vector. The domain of a node is similarly computed by aggregating the embedding vectors of the metadata fields of its data sets. Alternatively, a data set (and a node) can be represented by the set of word embedding vectors instead of an aggregated vector. The similarity

[1]http://ronin.cs.rochester.edu

[2]https://dev.socrata.com

[3]https://github.com/ekzhu/lshensemble/

[4]https://github.com/facebookresearch/fastText

[5]https://github.com/facebookresearch/faiss

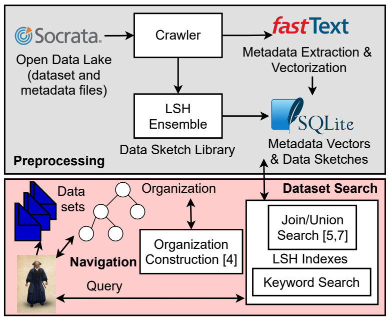


Figure 1: The RONIN Architecture.

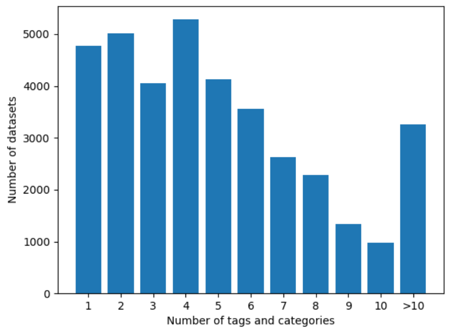


Figure 2: Metadata Distribution in Data Lake.

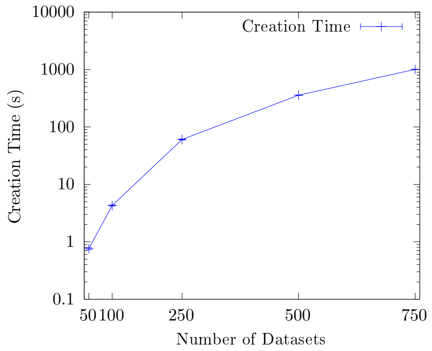


Figure 3: Construction Time.

can then be calculated as the average similarity of pairs of vectors. In this demo, we implement the former representation.

Navigation Model Previous work models a user's experience during navigation in an organization as a Markov model where states are nodes and transitions are edges [4]. Suppose the user has a query topic X in mind and, for now, suppose there is a way of modeling X with a semantic vector. Starting at the root node, a user navigates through sets of data sets by following edges. Because of the inclusion property, each transition filters some of the data sets. The navigation stops once the user reaches a leaf node. The probability of a user's transition from state s to $c \in \text{ch}(s)$ is determined by the similarity between X and the state c. Since semantic vectors are constructed on word embedding vectors, we use the common cosine similarity measure. The transition probability function from s to c is $P(c|s,X,O) = \frac{e^{\frac{\gamma}{|\text{ch}(s)|}.\text{cosine}(c,X)}}{\sum_{t\in ch(s)} e^{\frac{\gamma}{|\text{ch}(s)|}.\text{cosine}(t,X)}}$.

The constant γ is hyper parameter with a strictly positive number. The term $|\,\text{ch}(s)|$ is to penalize having nodes with too many children. Plugging in this transition funtion into markov model, the probability of reaching state s_k through a path $r = s_1, \ldots, s_k$, while searching for X is $P(s_k|r,X,O) = \prod_{i=1}^{k} P(s_i|s_{i-1},X,O)$. Since O is a DAG, the probability of reaching a state s in O while searching for X is $P(s|X,O) = \sum_{r\in paths(s)} P(s|r,X,O)$, where $paths(s)$ is the set of all paths in O that reach s from the root. Note that this model naturally penalizes long paths. A data set is found when the navigation reaches the leaf node that contains that data set.

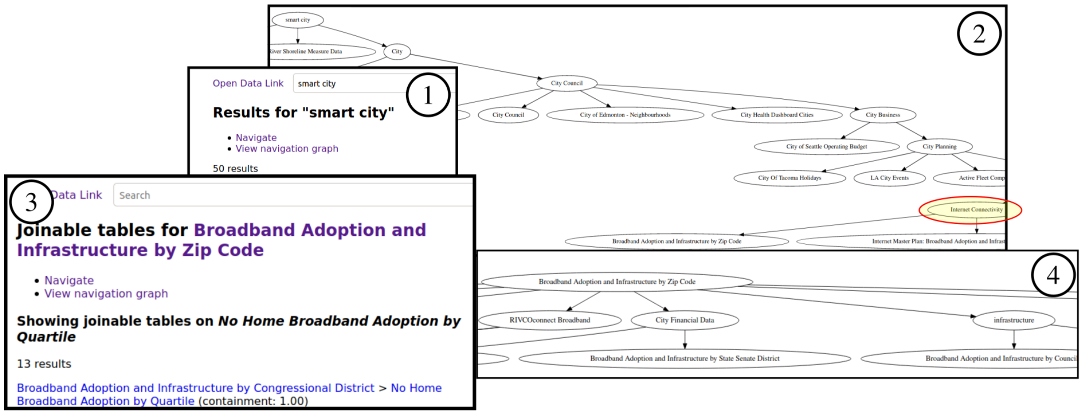
Given an organization O defined on $\mathcal{L}$, the *effectiveness* of O is the expected probability of finding a data set in $\mathcal{L}$ while navigating O, defined as $\text{eff}(O) = \sum_{D_i\in\mathcal{L}} q_i \,.\, P(D_i|O)$, where $P(D_i|O)$ is the probability of reaching the leaf node containing data set D_i while searching for D_i, i.e. $X = D_i$, and q_i is the probability or significance of D_i. Currently, RONIN assumes a uniform distribution for q. As the user search history becomes available through the use of RONIN, we can learn q. Given $\mathcal{L}$, our goal is to find an organization with the highest effectiveness.

3.2 Organization Construction

Finding an organization with optimal effectiveness requires a search in the powerset lattice of all data sets, while computing the effectiveness of each candidate organization. Note that calculating the effectiveness requires traversing all paths and calculating the reachability probability for each data set; this has complexity in the size of graph. An organization on n data sets can have as many as $2^n - 1$ nodes. To solve the combinatorial optimization of finding the most effective organization, previous work proposes a local search algorithm that starts from an initial DAG with nodes as subsets of data sets that satisfies the inclusion property. And, it iteratively enhances the effectiveness by changing the structure of the DAG [4].

For an initial organization, RONIN generates an agglomerative hierarchical clustering on the semantic vectors of data sets. This is a deep tree structure with a branching factor of two. The algorithm starts by defining a node for each data set. At each iteration, the algorithm finds the most similar pair of nodes and creates a new node as their parent. This guarantees that the final tree satisfies the inclusion property. The semantic vector of a node is computed by aggregating its children. The number of similarity computations to build an initial organization is in the order of $O(n^2)$. RONIN leverages the nearest neighbor search index of faiss to speed up finding the next pair of nodes to merge.

Then, RONIN iteratively changes the organization by intelligently removing nodes and adding edges. RONIN traverses the DAG in a bottom-up fashion and at each level (depth of the DAG) picks a node with the smallest reachability probability. The reachability of the node can be improved by either adding an edge between an upper level node and the node (increasing the number of paths leading to the node) or eliminating the node and assigning its children to its parents (decreasing path lengths in the DAG). RONIN makes these changes while guaranteeing the inclusion property and updating the node vectors. Both changes may or may not increase the effectiveness. For instance, adding a new parent increases the path lengths, and eliminating a node increases the branching factors, both have the potential of reducing the effectiveness. RONIN only accepts the change if it leads to a DAG with higher effectiveness. The complexity of the calculation of effectiveness is in the size of DAG. For future work, we plan to implement an incremental way of evaluating the effectiveness by exclusively recomputing reachability probability for the subset of nodes affected by the change. In our implementation, this process is terminated when the effectiveness of the organization has plateaued (changed less than 1e−9) for $\beta.|\mathcal{L}|$ or the algorithm has made $\delta.|\mathcal{L}|$ updates, where $\beta = 14$ and $\delta = 2000$, i.e. stricter condition for larger organizations. Our implementation is in gonum library for the vast majority of computations, including its implementation of the BLAS specification for vector computations, as well as graph traversal.


2
smart city
River Shoreline Measure Data
City
City Council
City Council
City of Edmonton - Neighbourhoods
City Health Dashboard Cities
City Business
City of Seattle Operating Budget
City Planning
City Of Tacoma Holidays
LA City Events
Active Fleet Comp
Internet Connectivity
Broadband Adoption and Infrastructure by Zip Code
Internet Master Plan: Broadband Adoption and Infras
1
3
4
Broadband Adoption and Infrastructure by Zip Code
RIVCOconnect Broadband
City Financial Data
infrastructure
Broadband Adoption and Infrastructure by State Senate District
Broadband Adoption and Infrastructure by Counci

Open Data Link smart city

Results for "smart city"

- Navigate
- View navigation graph

50 results

Data Link Search

Joinable tables for Broadband Adoption and Infrastructure by Zip Code

- Navigate
- View navigation graph

Showing joinable tables on *No Home Broadband Adoption by Quartile*

13 results

Broadband Adoption and Infrastructure by Congressional District > No Home Broadband Adoption by Quartile (containment: 1.00)

Figure 4: The RONIN demo: 1) the user performs a keyword search, 2) the user navigates an organization of the results to reach a data set of interest, 3) the user performs a similarity search using this data set as a query, selects another data set, and finds joinable data sets with it, 4) the user navigates the updated organization on joinable search results.

Figure 3 shows the average construction time of organizations with various number of data sets. This plot considers 30 random queries. For each value n in the x-axis (number of data sets), we consider n similar data sets returned by RONIN for each query. Note that the x-axis is the number of leaf nodes. In fact, the number of nodes of the initial organization is exponential in n. RONIN is currently interactive for result sets containing 100 data sets, which indicates a large enough result set presented by existing search engines [2]. This experiment is conducted on an Intel Xeon @ 2.30 GHz with 512Gb of memory.

To make an organization human-readable, RONIN annotates each node with a descriptive label. RONIN considers the collection of all values for the `categories` metadata field as possible labels. The number of considered labels is 978. As shown in Table 1, the categories are prevalent in the data lake. Each label is expressed with a semantic vector. RONIN traverses an organization in breadth-first manner and annotates a node with the non-duplicate label that has the highest cosine similarity to the node. To perform an efficient similar label search, in our implementation, we leverage the *faiss* index on the vector of all labels. It is also possible to consider other sources of labels such as Wikipedia categories. An alternative way of annotating a node is topic modeling of the data sets in the node.

4 DEMONSTRATION DESCRIPTION

We demonstrate RONIN on a crawl of approximately 40K data sets from Socrata API. We describe user activities in four steps (Figure 4).
Step 1: keyword search The user performs a keyword search. RONIN displays all of the relevant data sets to the keyword and a link to an organization that is constructed on the result on the fly. In our screenshot, the user first enters the keyword `smart city`.
Step 2: navigation in organization Instead of manually examining the list of result data sets, the user starts the navigation from the root node of the organization until reaching data sets of interest. In our screenshot, the user starts from root `smart city` and navigates to the node `city infrastructure`. Since the user is tech savvy, the user chooses to finally navigate to the node `internet connectivity` that contains data sets `internet master plan` and `broadband adoption and infrastructure by zip code`.
Step 3: data set search At any time during navigation, the user can switch to the search mode using a data set or a keyword as a query. For example, the user may choose to perform a search using the new keyword `internet connectivity` or perform a similarity search on the data sets of a node. In our screenshot, the user requests RONIN for similar data sets to `broadband adoption by zip code`. Among the result, the user finds `broadband adoption and infrastructure by council district`. The search continues by a request for joinable data sets with one of the attributes of this data set. This is particularly interesting as the user may now find data sets that were not in the results of the original search.
Step 4: updating organization The user can now navigate the organization of the joinability search result or choose a new keyword to search with from.
Demonstration engagement In addition to our guided demonstration, participants can request for results of their data set search. Moreover, they can navigate the organizations of their search result.
Acknowledgement This research is supported in part by NSF 1717877 and 2107050.

REFERENCES

[1] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *ICDE*. 709–720.
[2] Dan Brickley, Matthew Burgess, and Natasha F. Noy. 2019. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *WWW*. 1365–1375.
[3] Raul Castro Fernandez, Dong Deng, Essam Mansour, Abdulhakim Ali Qahtan, Wenbo Tao, Ziawasch Abedjan, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2017. A Demo of the Data Civilizer System. In *SIGMOD*. 1639–1642.
[4] Fatemeh Nargesian, Ken Q. Pu, Erkang Zhu, Bahar Ghadiri Bashardoost, and Renée J. Miller. 2020. Organizing Data Lakes for Navigation. In *SIGMOD*. 1939–1950.
[5] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. 2019. Data Lake Management: Challenges and Opportunities. *PVLDB* 12, 12 (2019), 1986–1989.
[6] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *PVLDB* 11, 7 (2018), 813–825.
[7] Yi Zhang and Zachary G. Ives. 2020. Finding Related Tables in Data Lakes for Interactive Data Science. In *SIGMOD*. 1951–1966.
[8] Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. 2016. LSH Ensemble: Internet-Scale Domain Search. *PVLDB* 9, 12 (2016), 1185–1196.
[9] Erkang Zhu, Ken Q. Pu, Fatemeh Nargesian, and Renée J. Miller. 2017. Interactive Navigation of Open Data Linkages. *PVLDB* 10, 12 (2017), 1837–1840.