Data Lake Organization

Fatemeh Nargesian, Ken Pu, Bahar Ghadiri-Bashardoost, Erkang Zhu, Renée J. Miller,

Abstract—We consider the problem of building an organizational directory of data lakes to support effective user navigation. The organization directory is defined as an acyclic graph that contains nodes representing sets of attributes and edges indicating subset relationships between nodes. A probabilistic model is constructed to model user navigational behaviour. The model also predicts the likelihood of users finding relevant tables in a data lake given an organization. We formulate the data lake organization problem as an optimization over the organizational structure in order to maximize the expected likelihood of discovering tables by navigating. An approximation algorithm is proposed with an analysis of its error bound. The effectiveness and efficiency of the algorithm are evaluated on both synthetic and real data lakes. Our experiments show that our algorithm constructs organizations that outperform many existing organizations including an existing hand-curated taxonomy, a linkage graph, and a common baseline organization. We have also conducted a formal user study which shows that navigation can help users discover relevant tables that are not easily accessible by keyword search queries. This suggests that keyword search and navigation using an organization are complementary modalities for data discovery in data lakes.

Index Terms—Data Lake, Dataset Discovery, Taxonomy, Structure Learning.

1 Introduction

The popularity and growth of data lakes is fueling interest in dataset discovery. Dataset discovery is normally formulated as a search problem. In one version of the problem, the query is a set of keywords and the goal is to find tables relevant to the keywords [1], [2]. Alternatively, the query can be a table and the problem is to find relevant, joinable, or unionable tables [3], [4], [5], [6], [7], [8], [9], [10]. A complementary alternative to search is navigation. In this paradigm, a user navigates through an organizational structure to find tables of interest. In the early days of Web search, navigation was the dominant discovery method for Web pages. Yahoo!, a mostly hand-curated directory structure, was the most significant internet gateway for Web page discovery [11]. Even today, hierarchical organizations of Web content are still used by `Youtube.com` and `Amazon.com`. Hierarchical navigation over entities allows a user to browse available entities going from more general concepts to more specific concepts using existing taxonomies or structures automatically created using taxonomy induction [12], [13], [14]. When entities have known features, we can apply faceted-search over entities [15], [16], [17]. Taxonomy induction looks for *is-a* relationships between entities (e.g., `student is-a person`), while faceted-search applies predicates (e.g., `model = Volvo`) to filter entity collections [18]. In contrast to hierarchies over entities, in data lakes, tables contain attributes whose domains may mention many different types of entities. There may be no *is-a* relationships between tables, or their attributes, and no easily defined facets for grouping tables. If tables are annotated with class labels of a knowledge base, the *is-a* relationships between class labels could provide an organization on tables. However, as we will show in our experiments, knowledge bases are not designed to provide effective navigation. We propose instead to build an organization that is designed to best support navigation and exploration over data lakes. Our goal is not to compete with or replace search, but rather to provide an alternative discovery option for users with only a vague notion of what data exists in a lake.

We define an *organization* as a Directed Acyclic Graph (DAG) with nodes that represent sets of attributes from a data lake. A node may have a label created from attribute values or metadata. A table can be associated with all nodes containing any of its attributes. An edge in this DAG indicates that the set of attributes in a parent node is a superset of the attributes in a child node. A user finds a table by traversing a path from a root of an organization to any leaf node that contains any of its attributes.

We propose the *data lake organization problem* where the goal is to find an organization that allows a user to most efficiently find tables. We describe the user navigation of an organization using a Markov model. In this model, each node in an organization is equivalent to a state in the navigation model. At each state, a user is provided with a set of next states (the children of the current state). An edge indicates the transition from the current state to a next state. Due to the subset property of edges, each transition filters out some attributes until the navigation reaches attributes of interest. An organization is effective if certain properties hold. At each step, a user should be presented with only a reasonable number of choices (*branching factor*). The choices should be distinct to make it easier for a user to choose the most relevant one. The transition probability function

- *F. Nargesian is with the Department of Computer Science, University of Rochester. E-mail: fnargesian@rochester.edu*
- *K. Pu is with the Department of Computer Science, University of Ontario Institute of Technology. E-mail: ken.pu@uoit.ca*
- *B. Ghadiri-Bashardoost is with the Department of Computer Science, University of Toronto. E-mail: ghadiri@cs.toronto.edu*
- *E. Zhu is with Microsoft Research. E-mail: ekzhu@microsoft.com*
- *R. J. Miller is with the Khoury College of Computer Sciences, Northeastern University. E-mail: miller@northeastern.edu*

Manuscript received April, 2021.

of our model assumes users choose the next state that has the highest similarity to the topic query they have in mind. Also, the number of choices they need to make (the *length of the discovery path*) should not be large.

TABLE 1: Sample Tables from Open Data.

Id	Table Name
d1	*Surveys Data for Olympia Oysters, Ostrea lurida, in BC*
d2	*Sustainability Survey for Fisheries*
d3	*Grain deliveries at prairie points 2015-16*
d4	*Circa 1995 Landcover of the Prairies*
d5	*Mandatory Food Inspection List*
d6	*Canadian Food Inspection Agency (CFIA) Fish List*
d7	*Wholesale trade, sales by trade group*
d8	*Historical releases of merchandise imports and exports*
d9	*Immigrant income by period of immigration, Canada*
d10	*Historical statistics, population and immigrant arrivals*

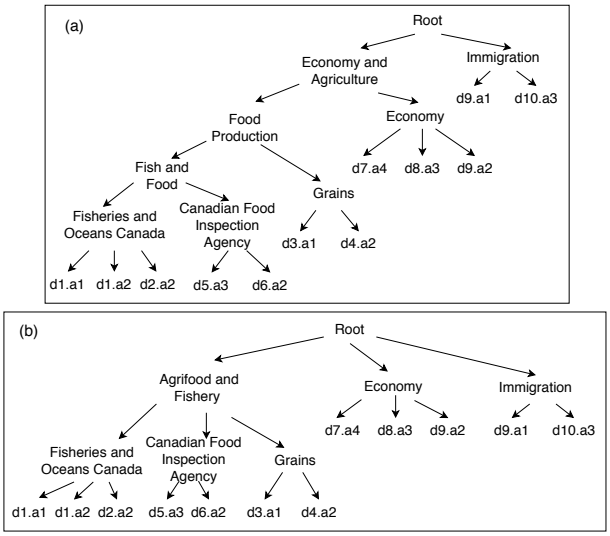


Fig. 1: (a) Deep and (b) Effective Organization.

Example 1. *Consider the (albeit small) collection of tables from an open data lake (Table 1). A table can be multi-faceted and attributes in a table can be about different topics. One way to expose this data to a user is through a flat structure of attributes of these tables. A user can browse the data and select data of interest. If the number of tables and attributes is large, it would be more efficient to provide an organization over attributes. Suppose the tables are organized in the DAG of Figure 1(a). The label of a non-leaf node in this organization summarizes the content of the attributes in the subgraph of the node. Suppose a user is interested in the topic of* food inspection*. Using this organization, at each step of navigation, they have to choose between only two nodes. The first two choices seem clear,* Economy and Agriculture *and* Food Production *seem more relevant to the user than their alternatives (*Immigration *and* Economy*). However, having a small branching factor in this organization results in nodes that may be misleading. For example, it may not be clear if there is any inspection data under the node* Fish and Food *or under* Grains*. This is due to the large heterogeneity of attributes (like* Oysters *and* Grain Elevators*) in the organization below the* Fish and Food *node. The organization in Figure 1(b) addresses this problem by organizing some of the attributes of* Grains*,* Food Inspection*, and* Fisheries *at the same level.*

This paper is an extension of prior work [19] that introduced the *data lake organization problem* as the problem of finding an organization that maximizes the expected probability of discovering lake tables. Navigation was modeled as a Markov process that computes the probability of discovering a table that is relevant to a topic of interest. A formal user-study, showed that navigation can help users find a more diverse set of tables than keyword search. In this paper, we make the following additional contributions.

- We revisit our approximation algorithm for solving the data lake organization problem and prove an upper bound for the error of the approximation.
- We consider data lakes that do not have sufficient metadata to leverage for creating organizations. We propose a metadata enrichment algorithm that effectively bootstraps any existing metadata. We provide an empirical evaluation that shows that metadata can be transferred across data lakes.
- We provide two important new empirical evaluations. The first compares using data lake organizations with a rich general-purpose taxonomy (Yago) [20] to navigate and find tables in an organization. The second compares data lake organizations with a common linkage graph [21] when both are used to navigate a real data lake. Our results quantify the benefits of using an organization tailored to the distributions of a real data lake to enable effective navigation of the lake.

2 FOUNDATIONS

We envision scenarios in which users interactively navigate through topics in order to locate relevant tables. The topics are derived and organized based on attributes of tables. Our model captures user elements such as the cognitive effort associated with multiple choices and the increasing complexity of prolonged navigational steps.

2.1 Organization

Let $\mathcal{T}$ be the set of all tables in a data lake. Each table $T \in \mathcal{T}$ consists of a set of attributes, $\mathrm{attr}(T)$. Let $\mathcal{A}$ be the set of all attributes, $\mathcal{A} = \bigcup\{\mathrm{attr}(T) : T \in \mathcal{T}\}$ in a data lake. Each attribute A has a set of values that we call a domain and denote by $\mathrm{dom}(A)$. An organization $\mathcal{O} = (V, E)$ is a DAG. Let $\mathrm{ch}(.)$ be the child relation mapping a node to its children, and $\mathrm{par}(.)$ be the parent relation mapping a node to its parents. A node $s \in V$ is a leaf if $\mathrm{ch}(s) = \emptyset$, otherwise s is an interior node in $\mathcal{O}$. Every leaf node s of $\mathcal{O}$ corresponds to a distinct attribute $A_s \in \mathcal{A}$. Each interior node s corresponds to a set of attributes $D_s \subseteq \mathcal{A}$. If $c \in ch(s)$, then $D_c \subseteq D_s$, we call this the *inclusion* property, and $D_s = \bigcup_{c \in ch(s)} D_c$. We denote the domain of a state s by $\mathrm{dom}(s)$ which is $\mathrm{dom}(A_s)$ when s is a leaf node and $\bigcup_{A \in D_s} \mathrm{dom}(A)$ otherwise.

2.2 Navigation Model

We model a user's experience during discovery using an organization $\mathcal{O}$ as a Markov chain model where states are nodes and transitions are edges. We will use the terms state and node interchangeably. Users select a transition at each step and because of the inclusion property, the transition from filters out some of the attributes of the state. The

discovery stops once users reach a leaf node. To define the effectiveness of an organization, we define a user's intent using a query topic X modeled as a set of values. Starting at the root node, a user navigates through sets of attributes (states) ideally finding attributes of interest.

Note that when an organization is being used, we do not know X. Rather we are assuming that a user performs navigation in a way that they choose nodes that are closest to their intended topic query. The concept of a user's query topic is used only to build an organization. We build an organization that maximizes the expected probability of finding any table in the data lake by finding any of its attributes (assuming a user could potentially *have in mind* any attribute in the lake). In other words, the set of query topics we optimize for is the set of lake attributes and tables.

Given an organization, we define the transition probability of $P(c|s, X, \mathcal{O})$ as the probability that the user who searches for topic X will choose c as the next state if they are at the state s. The probability should be correlated to the similarity between $\mathrm{dom}(c)$ and X. Let $\kappa(c, X)$ be a similarity metric between $\mathrm{dom}(c)$ and X. The transition probability is defined as follows.

$$P(c|s, X, \mathcal{O}) = \frac{e^{\frac{\gamma}{|\mathrm{ch}(s)|}\cdot\kappa(c,X)}}{\sum_{t\in ch(s)} e^{\frac{\gamma}{|\mathrm{ch}(s)|}\cdot\kappa(t,X)}} \tag{1}$$

The constant γ is a hyper parameter and must be a strictly positive number. The term $|\mathrm{ch}(s)|$ is a penalty factor to avoid having nodes with too many children. The impact of the high similarity of a state to X diminishes when a state has a large branching factor.

A discovery sequence is a path, $r = s_1, \ldots, s_k$ where $s_i \in ch(s_{i-1})$ for $1 < i \leq k$. A state in $\mathcal{O}$ is reached through a discovery sequence. The Markov property says that the probability of transitioning to a state is only dependent on its parent. Thus, the probability of reaching state s_k through a discovery sequence $r = s_1, \ldots, s_k$, while searching for X is defined as follows.

$$P(s_k|r, X, \mathcal{O}) = \prod_{i=1}^{k} P(s_i|s_{i-1}, X, \mathcal{O}) \tag{2}$$

In this model, a user makes transition choices only based on the current state and the similarity of their query topic X to each of the child states. Note that the model naturally penalizes long sequences. Since an organization is a DAG, a state can be reached by multiple discovery sequences. The probability of reaching a state s in $\mathcal{O}$ while searching for X is as follows.

$$P(s|X, \mathcal{O}) = \sum_{r\in Paths(s)} P(s|r, X, \mathcal{O}) \tag{3}$$

where $Paths(s)$ is the set of all discovery sequences in $\mathcal{O}$ that reach s from the root. Additionally, the probability of reaching a state can be evaluated incrementally. Recall that $\mathrm{par}(.)$ is the parent relation mapping a node to its parents.

$$P(s|X, \mathcal{O}) = \sum_{p\in par(s)} P(s|p, X, \mathcal{O})P(p|X, \mathcal{O}) \tag{4}$$

Definition 1. *The* discovery probability *of an attribute A in organization $\mathcal{O}$ is defined as $P(s|A, \mathcal{O})$, where s is a leaf node. We denote the discovery probability of A as $P(A|\mathcal{O})$.*

2.3 Organization Discovery Problem

A table T is discovered in an organization $\mathcal{O}$ by discovering any of its attributes. Here, we make an independence assumption for the discovery of attributes. Modeling the correlation between attributes is an interesting direction for future work.

Definition 2. *For a single table T, we define the* discovery probability of a table *as follows.*

$$P(T|\mathcal{O}) = 1 - \prod_{A\in T} (1 - P(A|\mathcal{O})) \tag{5}$$

For a set of tables $\mathcal{T}$, the organization effectiveness *is the expected probability of finding tables.*

$$P(\mathcal{T}|\mathcal{O}) = \frac{1}{|\mathcal{T}|} \sum_{T\in\mathcal{T}} P(T|\mathcal{O}) \tag{6}$$

The data lake organization problem is to find an organization that has the highest effectiveness.

Definition 3. Data Lake Organization Problem. *Given a set of tables $\mathcal{T}$ in a data lake, the organization problem is to find an organization $\hat{\mathcal{O}}$ such that:*

$$\hat{\mathcal{O}} = \arg\max_{\mathcal{O}} P(\mathcal{T}|\mathcal{O}) \tag{7}$$

2.4 Multi-dimensional Organizations

Given the heterogeneity and massive size of data lakes, it may be advantageous to perform an initial grouping of tables and then build an organization on each group. For example, for the data lake of Example 1, we would define groups of tables, perhaps a group on `Immigration` data and another on `Environmental` data over which we may be able to build more effective organizations.

If we have a grouping of $\mathcal{A}$ into k (possibly overlapping) groups or dimensions, $G_1, \ldots, G_k$, then we can discover an organization over each and use them collectively for navigation. A k-dimensional organization $\mathcal{M}$ for a data lake $\mathcal{T}$ is defined as a set of organizations $\{\mathcal{O}_1, \ldots, \mathcal{O}_k\}$, such that $\mathcal{O}_i$ is the most effective organization for G_i. For each dimension, the organization is constructed independently, Therefore, the probability of finding an attribute in a dimension is independent of other organizations. We define the probability of discovering table T in $\mathcal{M}$, as the probability of discovering T in any of dimensions of $\mathcal{M}$.

$$P(T|\mathcal{M}) = 1 - \prod_{\mathcal{O}_i\in\mathcal{M}} (1 - P(T|\mathcal{O}_i)) \tag{8}$$

3 Constructing Organizations

Given the abstract model of the previous section, we now present a specific instantiation of the model that is suited for real data lakes. We then consider the metadata that is often available in data lakes, specifically table-level tags, and explain how this metadata can be exploited for navigation. We present a local search algorithm for building an approximate solution for the data lake organization problem. Finally, we formally analyze the algorithm and provide an upper bound on the error in the approximation.

3.1 Attribute and State Representation

We have chosen to construct organizations over the text attributes of data lakes. This is based on the observation that although a small percentage of attributes in a lake are text attributes (26% for a Socrata open data lake that we use in our experiments), the majority of tables (92%) have at least one text attribute. We have found that similarity between numerical attributes (measured by set overlap or Jaccard) can be very misleading as attributes that are semantically unrelated can be very similar and semantically related attributes can be very dissimilar. Therefore, to use numerical attributes one would first need to understand their semantics. Work is emerging on how to do this [22], [23], [24].

Since an organization is used for the exploration of heterogeneous lakes, we are interested in the semantic similarity of values. To capture semantics, a domain can be canonicalized by its collective word embedding vectors [3]. Each data value v is represented by an embedding vector $\vec{v}$ such that values that are more likely to share the same context have embedding vectors that are close in embedding space according to a Euclidean or angular distance measure [25]. Attribute A can be represented by a *topic vector*, denoted μ_A, which is the sample mean of the population $\{\vec{v}|v \in \text{dom}(A)\}$ [3]. In organization construction, we also represent X by μ_X. We represent state s with a *topic vector*, μ_s, which is the sample mean of the population $\{\vec{v}|v \in \text{dom}(s)\}$.

If a sufficient number of values have word embedding representatives, the topic vectors of attributes are good indicators of attributes' topics. We use the word embeddings of fastText database [26], which on average covers 70% of the values in the text attributes in the datasets used in our experiments. In organization construction, to evaluate the transition probability of navigating to state c we choose $\kappa(c, X)$ to be the Cosine similarity between μ_C and μ_X. Since a parent subsumes the attributes in its children, the Cosine similarity satisfies a monotonicity property of $\kappa(X, c) > \kappa(X, s)$, where $c \in ch(s)$. However, the monotonicity property does not necessarily hold for the transition probabilities.

3.2 State Space Construction

Metadata in Data Lakes Tables in lakes are sometimes accompanied by metadata hand-curated by the publishers and creators of data. In enterprise lakes, metadata may come from design documentation from which tags or topics can be extracted. For open data, the standard APIs used to publish data include tags or keywords [27]. In mass-collaboration datasets, like web tables, contextual information can be used to extract metadata [28].

State Construction If metadata is available, it can be distilled into tags (e.g., keywords, concepts, or entities). In an organization, the leaves still have a single attribute, but the immediate parent of a leaf is a state containing all attributes with a given (single) tag. Building organizations on tags reduces the number of possible states and the size of the organization while still having meaningful nodes that represent a set of attributes with a common tag. Suppose D_s is the set of attributes in state s. Note the topic vector of state s, μ_s, is the sample mean of the population $\{\vec{v}|v \in \text{dom}(A), A \in D_s\}$.

ALGORITHM 1: Algorithm `Organize`

Input: Data Lake $\mathcal{T}$
Output: Organization $\mathcal{O}$
1 mod_ops = [DELETE_PARENT, ADD_PARENT]
2 $\mathcal{O} \leftarrow$ INIT_ORG($\mathcal{T}$), $p \leftarrow$ EVAL($\mathcal{O}$)
3 **while** $\neg termination\_cond$ **do**
4 $state \leftarrow$ STATE_TO_MODIFY($\mathcal{O}$)
5 $\mathcal{O}' \leftarrow$ CHOOSE_APPLY_OP($\mathcal{O}, state, mod\_ops$)
6 **if** ACCEPT(EVAL($\mathcal{O}'$), p) **then**
7 $\mathcal{O} \leftarrow \mathcal{O}', p \leftarrow p'$
8 **end**
9 **end**

Flat Organization: A Baseline Using metadata, the parent of a leaf node is associated with only one tag. This means that the last two levels of a hierarchy are fixed and an organization is constructed over states with a single tag. If we place a single root node over such states, we get a *flat organization* that we can use as a baseline. This is a reasonable baseline as it is conceptually, the navigation structure supported by many open data APIs that permit programmatic access to open data resources. These APIs permit retrieval of tables by tag.

3.3 Local Search Algorithm

Our local search algorithm, `Organize`, is outlined in Algorithm 1. It begins with an initial organization (Line 2). The initial organization may be any organization that satisfies the inclusion property of nodes. For example, the initial organization can be the DAG defined based on a hierarchical clustering of the tags of a data lake. To estimate the posterior probability, at each step, the algorithm takes a sample from the space of organization graphs by applying a *modification* to the current organization $\mathcal{O}$ which leads to a new organization $\mathcal{O}'$ (Line 5). If the new organization is closer to a solution for the *Data Lake Organization Problem* (Definition 3), it is accepted as the new organization, otherwise it is accepted (Line 6) with a probability that is a ratio of the effectiveness [29]: $min[1, \frac{P(\mathcal{T}|\mathcal{O}')}{P(\mathcal{T}|\mathcal{O})}]$. The algorithm terminates (Line 3) once the effectiveness of an organization reaches a plateau. In our experiments, we terminate when the expected probability has not improved significantly for the last 50 iterations. It is a common practice to perform several local searches with different initializations and choose the result with the highest local optimum among different restart runs. Here, we describe how one run is performed.

The algorithm strategically tries to maximize the effectiveness of an organization by making its states highly reachable. We use Equation 4 to evaluate the probability to reach a state when searching for attribute A and define the overall *reachability probability* of a state as follows.

$$P(s|\mathcal{O}) = \frac{1}{|A \in \mathcal{T}|} \sum_{A \in \mathcal{T}} P(s|A, \mathcal{O}) \tag{9}$$

Starting from an initial organization, the search algorithm performs downward traversals from the root and modifies

the organization for states in each level of the organization ordered from lowest reachability probability to highest (Line 4 and 5). A state is in level l if the length of the shortest discovery paths from the root to the state is l. At each iteration, a set of operations are applied on a node, each creating an organization, then the most effective organization is selected. We restrict our choices of a new organization at each search step to those created by the following operations.

Adding Parent Given a state s with low reachability probability, one reason for this may be that it is one child amongst many of its current parent. We can remedy this by adding a new parent for s. Suppose that the search algorithm chooses to modify the organization with respect to state s. Recall that Equation 4 indicates that the probability to reach a state increases as it is connected to more parent states. Suppose s is at level l of the organization $\mathcal{O}$. The algorithm finds the state, called n, at level $l-1$ of $\mathcal{O}$ such that it is not a parent of s and has the highest reachability probability among the states at $l-1$. To satisfy the inclusion property, we update node n and its ancestors to contain the attributes in s, D_s. State n is added to the organization as a new parent of s. ADD_PARENT potentially increases the reachability probability of a state by adding more discovery paths ending at that state, at the cost of increasing the branching factor.

Deleting Parent Another reason a state can have low reachability is that its parent has low reachability probability and we should perhaps remove a parent. Reducing the length of paths from the root to state s is a second way to boost the reachability probability of s. The operation eliminates the least reachable parent of s, called r. To reduce the height of $\mathcal{O}$, the operation eliminates all siblings of r except the ones with one tag. Then, it connects the children of each eliminated state to its parents. This operation does not change the number of paths in the graph that lead to the node and due to the inclusion property, it does not change the representation of the grandparent nodes either. In fact, it only makes the length of paths to s smaller which boosts the reachability probability of s. However, the branching factor of the grandparent increases which decreases the transition probabilities from that state.

In Algorithm 1, STATE_TO_MODIFY (Line 4) orders states by level starting at level 1 and within a level by reachability (lowest to highest) and returns the next state each time it is called. CHOOSE_APPLY_OP picks the operator that when applied creates the most effective organization $\mathcal{O}'$. Every time a new organization is chosen, STATE_TO_MODIFY may need to reorder states appropriately.

4 Scaling Organization Search

The evaluation of the effectiveness (Equation 7) involves computing the discovery probability for all attributes and evaluating the probability of reaching the states along the paths to an attribute (Equation 4). The organization graph can have a large number of states, especially at the initialization phase. To improve the search efficiency, we first identify the subset of states and attributes whose discovery probabilities may be changed by an operation and second we approximate the new discovery probabilities using a set of attribute representatives.

4.1 Affected States

At each search iteration, we only re-evaluate the discovery probability of the states and attributes which are affected by the local change. Upon applying DELETE_PARENT on a state, the transition probabilities from its grandparent to its grandchildren are changed and consequently all states reachable from the grandparent. However, the discovery probability of states that are not reachable from the grandparent remain intact. Therefore, for DELETE_PARENT, we only re-evaluate the discovery probability of the states in the subgraph rooted by the grandparent and only for attributes associated with the leaves of the subgraph.

The ADD_PARENT operation impacts the organization more broadly. Adding a new parent to a state s changes the discovery probability of s and all states that are reachable from s. Furthermore, the parent state and consequently its ancestors are updated to satisfy the inclusion property of states. Suppose the parent itself has only one parent. The change of states propagates to all states up to the lowest common ancestor (LCA) of s and its parent-to-be before adding the transition to the organization. If the parent-to-be has multiple parents the change needs to be propagated to other subgraphs. To identify the part of the organization that requires re-evaluation, we iteratively compute the LCA of s and each of the parents of its parent-to-be. All states in the subgraph of the LCA require re-evaluation.

4.2 Approximating Discovery Probability

To further speed up the search, we evaluate an organization on a small number of attribute representatives that each summarizes a set of attributes. The discovery probability of each representative approximates the discovery probability of its corresponding attributes. We assume a one-to-one mapping between representatives and partitions of attributes. Suppose ρ is a representative for a set of attributes $D_\rho = \{A_1, \ldots, A_m\}$. We approximate $P(A_i|\mathcal{O})$, $A_i \in D_\rho$ with $P(\rho|\mathcal{O})$. The choice and the number of representatives impact the error of this approximation.

Theorem 1. *The approximation error of the discovery probability of attribute A using its representative ρ in organization $\mathcal{O}$ is:*

$$\epsilon_r \leq (\prod_{i=1}^{k} P(s_i|s_{i-1}, A, \mathcal{O})).(1 - \frac{1}{e^{\gamma'(1-\kappa(\rho,A))}})^k. \quad (10)$$

where A is reachable with the discovery path $r = s_1, \ldots, s_k$.

Proof. Recall that the discovery probability of a leaf state is the product of transition probabilities along the path from the root to the state. To determine the error that a representative introduces to the discovery probability of an attribute, we first define an upper bound on the error incurred by using representatives in transition probabilities. We show that the error of transition probability from m to s is bounded by a fraction of the actual transition probability which is inversely correlated with the similarity of the

representative to the attribute. For brevity, in Equation 1, we assume $\gamma' = \frac{\gamma}{|ch(m)|}$.

$$P(s_i|m, A, \mathcal{O}) = \frac{e^{\gamma' \kappa(A,s_i)}}{\sum_{s_j \in ch(m)} e^{\gamma' \kappa(A,s_j)}} \tag{11}$$

Suppose δ is the distance defined based on κ, which is $\delta(a, b) = 1 - \kappa(a, b)$. From the triangle property, it follows that:

$$\delta(s_i, \rho_k) \leq \delta(s_i, A) + \delta(A, \rho) \tag{12}$$

Evaluating $P(s_i|m, A, \mathcal{O})$ and $P(s_i|m, \rho, \mathcal{O})$ requires computing $\kappa(A, s_j)$ and $\kappa(\rho, s_j)$ on $\mathrm{ch}(m)$. We rewrite the triangle property as follows:

$$1 - \kappa(s_i, \rho) \leq 1 - \kappa(s_i, A) + 1 - \kappa(A, \rho) \tag{13}$$

Therefore, the upper bound of $\kappa(s_i, A)$ is defined as follows.

$$\kappa(s_i, A) \leq \kappa(s_i, \rho) - \kappa(A, \rho) + 1 \tag{14}$$

We also have the following.

$$0 \leq \kappa(s_i, A) - \kappa(s_i, \rho) \leq 1 - \kappa(A, \rho) \tag{15}$$

Let $\Delta_i = \kappa(s_i, A) - \kappa(s_i, \rho)$. Without loss of generality, we assume $\kappa(s_i, A) > \kappa(s_i, \rho)$, thus Δ_i is a positive number. Now, we can rewrite $\kappa(s_i, \rho) = \kappa(s_i, A) - \Delta_i$. From Equation 1, we know that $P(s_i|m, A, \mathcal{O})$ and $P(s_i|m, \rho, \mathcal{O})$ are monotonically increasing with $\kappa(s, A)$ and $\kappa(s_i, \rho)$, respectively. Therefore, the error of using the representative ρ in approximating the probability of transition from m to s_i when looking for A is as follows:

$$\epsilon = P(s_i|m, A, \mathcal{O}) - P(s_i|m, \rho, \mathcal{O}) \tag{16}$$

It follows from the monotonicity property that $\epsilon \geq 0$. By applying Equation 11 to ϵ, we have the following:

$$\epsilon = \frac{e^{\gamma' \kappa(A,s_i)}}{\sum_{s_j \in ch(m)} e^{\gamma' \kappa(A,s_j)}} - \frac{e^{\gamma' \kappa(\rho,s_i)}}{\sum_{s_j \in ch(m)} e^{\gamma' \kappa(\rho,s_j)}} \tag{17}$$

We rewrite the error by replacing $\kappa(s_i, \rho)$ with $\kappa(s_i, A) - \Delta_i$.

$$\epsilon = \frac{e^{\gamma' \kappa(A,s_i)}}{\sum_{s_j \in ch(m)} e^{\gamma' \kappa(A,s_j)}} - \frac{e^{\gamma' \kappa(s_i,A)}.e^{-\gamma' \Delta_i}}{\sum_{s_j \in ch(m)} e^{\gamma' \kappa(s_j,\rho)}.e^{-\gamma' \Delta_j}} \tag{18}$$

Following from Equation 15, we have:

$$\frac{1}{e^{\gamma'(1-\kappa(A,\rho))}} \leq e^{-\gamma' \Delta_i} \leq 1 \tag{19}$$

The upper bound of the approximation error of the transition probability is:

$$\epsilon \leq \frac{e^{\gamma' \kappa(A,s_i)}}{\sum_{s_j \in ch(m)} e^{\gamma' \kappa(A,s_j)}} - \frac{e^{\gamma' \kappa(A,s_i)}}{\sum_{s_j \in ch(m)} e^{\gamma' \kappa(A,s_j)}}. \frac{1}{e^{\gamma'(1-\kappa(\rho,A))}} \tag{20}$$

The error can be written in terms of the transition probability to a state given an attribute:

$$\epsilon \leq P(s_i|m, A, \mathcal{O}).(1 - \frac{1}{e^{\gamma'(1-\kappa(\rho,A))}}) \tag{21}$$

Since $e^{\gamma'(1-\kappa(\rho,A))} \geq 1$, the error is bounded.

Assuming the discovery path $r = s_1, \ldots, s_k$, the bound of the error of approximating $P(A_i|\mathcal{O})$ using ρ is as follows:

$$\epsilon_r \leq (\prod_{i=1}^{k} P(s_i|s_{i-1}, A, \mathcal{O})).(1 - \frac{1}{e^{\gamma'(1-\kappa(\rho,A))}})^k \tag{22}$$

□

To minimize the error of approximating $P(s|m, A, \mathcal{O})$ considering ρ instead of A, we want to choose ρ's that have high similarity to the attributes they represent, while keeping the number of ρ's relatively small. In our experiments, we apply a practical way of creating representatives. Attributes are partitioned using k-means clustering and the centroid of each cluster is considered as the representative of attributes in the cluster. Indeed, the problem of finding the optimal set of representatives that minimizes the approximation error for all attributes is an interesting problem that we would like to investigate going forward.

4.3 Structural and Semantic Processing of Metadata

Tags provide an initial grouping on attributes and reduce organization discovery cost. However, tags may be incomplete (some attributes may have no tags). Moreover, the schema and vocabulary of metadata across data originating from different sources may be inconsistent which can lead to disconnected organizations.

We propose to transfer tags across data lakes such that data lakes with no (or little) metadata are augmented with the tags from other data lakes. Annotating attributes with existing tags is a multi-label multi-class classification problem, which we choose to solve by training binary classifiers, one per tag, which predict the association of attributes to the corresponding tags. The classifiers are trained on the topic vector, μ_A, of attributes.

The metadata in data lakes is created and published by different sources, as a result, it does not follow a standard format. To start, we normalize the existing metadata following a series of syntactic, semantic, and frequency-driven approaches. Let $\mathcal{K}$ be the set of tags, assigned to the collection of attributes $\mathcal{A}$. Let $r(t, A)$ denote the fact that tag $t \in K$ is assigned to attribute $A \in \mathcal{A}$. Since our objective is to generate a user-efficient hierarchical organization for $\mathcal{A}$ using the tags, we want to identify redundant tags and prune them as potential candidates. For syntactic pruning, we utilize a stemmer to compute a normalized string representation of the tags. All the tags that share the same normal form are collapsed into a single *meta tag*. We further identify semantically equivalent tags by comparing the word embedding vectors and their strings. Any group of tags $\{t_1, t_2, \ldots, t_k\}$, such that their pair-wise cosine similarity is close to one are merged into a single meta tag, that is, $\forall i, j \leq k, sim(emb(t_i), emb(t_j)) \geq 1 - \epsilon$.

Finally, we also consider the co-occurrences of tags. Let $D(t) = \{A \in \mathcal{A} : r(t, A)\}$. Given two tags t_0 and t_1, if one is subsumed by the other, that is $D(t_0) \subset D(t_1)$, we should merge t_0 in t_1 to form a meta tag and eliminate t_0. The concept of tag subsumption can be generalized to meta tags including a larger number of tags. We define k-subsumption as the situation where t_0 is subsumed by k other tags $\{t_1, t_2, \ldots, t_k\}$ where $D(t_0) \subset \cup_{i=1}^{k} D(t_i)$.

The *support* of a tag t is the number of documents tagged by t, denoted by $|D(t)|$. In our experiments, we consider tags to be *redundant* if they are 1-subsumed or 2-subsumed, and have a support ≤ 20. As an example, suppose that a tag t_0 is 2-subsumed by (t_1, t_2), we would remove t_0 from the organization since the user can discover the documents tagged by t_0 using either t_1 and t_2.

5 Evaluation

We first seek to quantify and understand some of the design decisions, the efficiency, and the influence of using approximation for our approach. We do this using a small synthesized benchmark called `TagCloud` that is designed so that we know precisely the best tag per attribute. Next, using real open data, in Section 5.4, we quantify the benefits of our approach over 1) an existing ontology (Yago [20]) for navigation; 2) a flat baseline; and 3) a table linkage graph, called an Enterprise Knowledge Graph [30], to navigate. We also illustrate that tags from a real open data lake can be easily transferred to a different data lake with no metadata using a simple classifier (Section 4.3). Finally, we present a user study in Section 5.6.

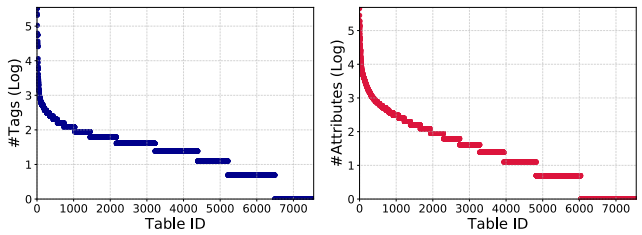
5.1 Datasets

We begin by describing our real and synthetic datasets which are summarized in Table 2.

TABLE 2: Experimental Datasets

Name	#Tables	#Attr	#Tags
Socrata	7,553	50,879	11,083
Socrata-1	1,000	5,511	3,002
Socrata-2	2,175	13,861	345
Socrata-3	2,061	16,075	346
CKAN	1,000	7,327	0
TagCloud	369	2,651	365
YagoCloud	370	2,364	500

Socrata and CKAN Data Lakes - For our comparison studies, we used real open data. We crawled 7,553 tables with 11,083 tags from the Socrata API. We call this lake `Socrata`. It contains 50,879 attributes containing words that have a word embedding. In this dataset, a table may be associated with many tags and attributes inherit the tags of their table. Particularly, all tables have at least a `name` as metadata and 88.9% are accompanied by at least one `category` or `tag` metadata fields, both of which we consider as tags. This metadata is created by the publisher and indeed can be incomplete. In Section 4.3, we propose a way of further enriching the metadata by annotating each attribute with the existing tags in the data lakes which leads to an improved organization. We have 264,199 attribute-tag associations. The distribution of tags per table and attributes per table of `Socrata` lake is plotted in Figure 2. The distribution is skewed with two tables having over 100K tags and the majority of the tables having 25 or fewer. `Socrata-1` is a random collection of 1,000 tables and 3,002 tags from `Socrata` lake that we use in our comparison with the Enterprise Knowledge graph. `Socrata-2` is a collection of 2,061 tables and 345 tags from `Socrata` and `Socrata-3` is a collection of 2,175 tables and 346 tags from `Socrata`.



(a) (b)

Fig. 2: Distribution of (a) Tags and (b) Attributes per Table in `Socrata`.

Note that `Socrata-2` and `Socrata-3` do not share any tags and are used in our user study. The `CKAN` data lake is a separate collection of 1,000 tables and 7,327 attributes from the CKAN API.

TagCloud benchmark - To study the impact of the density of metadata and the usefulness of multi-dimensional organizations, we synthesized a dataset where we know exactly the most relevant tag for an attribute. Note that in the real open data, tags may be incomplete or inconsistent (data can be mislabeled). We create only a single tag per attribute which is actually a disadvantage to our approach that benefits from more metadata. The benchmark is small so we can report both accuracy and speed for the non-approximate version of our algorithm in comparison to the approximate version that computes discovery probabilities using attribute representatives. We synthesized a collection of 369 tables with 2,651 attributes. First, we generate tags by choosing a sample of 365 words from the fastText database that are not very close according to Cosine similarity. The word embeddings of these words are then used to generate attributes associated with tags. Each attribute in the benchmark is associated with exactly one tag. The values of an attribute are samples of a normal distribution centered around the word embedding of a tag. To sample from the distribution of a tag, we selected the k most similar words, based on Cosine similarity, to the tag, where k is the number of values in the attribute (a random number between 10 and 1,000). This guarantees that the distribution of the word embedding of attribute values has small variance and the topic vector of attributes are close to their tags. This artificially guarantees that the states that contain the tag of an attribute are similar to the attribute and likely have high transition probabilities.

To emulate the metadata distribution of real data lakes (where the number of tags per table and number of attributes per table follow Zipfian distributions (Figure 2)), we generated tables so that the number of tags (and therefore attributes) also follows a Zipfian distribution. In the benchmark the number of attributes per table is sampled from [1, 50] following a Zipfian distribution.

YagoCloud benchmark - To have a fair comparison with YAGO, we cannot use our real open data lakes because YAGO has low coverage over this data. Hence, we synthesis a data lake where YAGO has full coverage and therefore the class hierarchy of YAGO would be a reasonable alternative for navigation. `YagoCloud` is a collection of 370 tables with 2,364 attributes that can be organized using the taxonomy

of YAGO. Like `TagCloud` benchmark, the distribution of attributes in tables and tags per table in this benchmark is created to emulate the observed characteristics of our crawl of open data portals (Figure 2). In YAGO, each entity is associated with one or more *types*. We consider 500 random leaf types that contain the words `agri`, `food`, or `farm` in their labels. These types are equivalent to tags. Each attribute in the benchmark is associated with exactly one tag. For each attribute of a table, we randomly sample its values from the set of entities associated with its tag. This guarantees that the attribute is about the type (tag). We tokenize the sampled entities into words and the word embeddings of these words are then used to generate the topic vectors of attributes. The number of values in an attribute is a random number between 10 and 1000. and the number of attributes per table is sampled from [1, 50] following a Zipfian distribution.

The subgraph of the YAGO taxonomy that covers all ancestor classes of the benchmark types is considered as the taxonomy defined on benchmark attributes. This taxonomy is a connected and acyclic graph. Each class in the taxonomy is equivalent to a non-leaf state of navigation. To guarantee the inclusion property in the taxonomy, each non-leaf state consists of the tags corresponding to its descendant types. The topic vector of an interior state is generated by aggregating the topic vectors of its descendant attributes.

5.2 Experimental Set-up

Evaluation Measure: For our evaluation, we do not have a *user*. We simulate a user by reporting the success probability of finding each table in the lake. Conceptually what this means is that if a user *had in mind* a table that is in the lake and makes navigation decisions that favor picking states that are closest to attributes and tags of that table, we report the probability that they would find that table using our organization.

We therefore report for our experiments a measure we call *success probability* that considers a navigation to be successful if it finds tables with an attribute of or a similar attribute to the query's attributes. We first define the success probability for attributes. Specifically, let κ be a similarity measure between two attributes and let $0 < \theta \leq 1$ be a similarity threshold.

Definition 4. *The* success probability *of an attribute A is defined as*

$$Success(A|\mathcal{O}) = 1 - \prod_{A_i \in \mathcal{A} \wedge \kappa(A_i, A) \geq \theta} (1 - P(A_i|\mathcal{O})) \quad (23)$$

We use the Cosine similarity on the topic vectors of attributes for κ and a threshold θ of 0.9. Based on attribute success probabilities, we can compute table success probability as $Success(T|\mathcal{O}) = 1 - \prod_{A \in T}(1 - Success(A|\mathcal{O}))$. We report success probability for every table in the data lake sorted from lowest to highest probability on the x-axis (see Figure 3 as an example).

Implementation: Our implementation is in Python and uses scikit-learn library for creating initial organizations. Our experiments were run on a 4-core Intel Xeon 2.6 GHz server with 128 GB memory. To speed up the evaluation of an organization, we cache the similarity scores of attribute pairs as well as attribute and state pairs as states are updated during search.

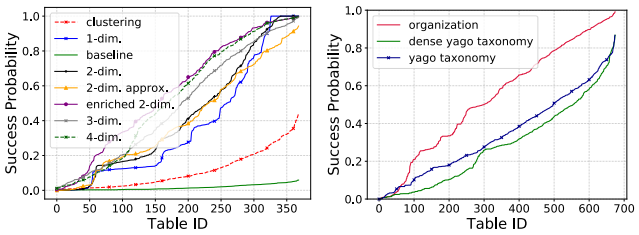


Fig. 3: Success Probability of Organizations (a) on `TagCloud` Benchmark and (b) `YagoCloud` Benchmark.

5.3 Performance of Approximation

We evaluate the effectiveness and efficiency of our exact algorithm (using exact computation of discovery probabilities, not the approximation discussed in Section 4.2) on the `TagCloud` benchmark.

5.3.1 Effectiveness

We constructed the `baseline` organization where each attribute has as parents the states consisting of tags of the attribute. Recall in this benchmark each attribute has a single, accurate tag. This organization is similar to the organization of open data portals. We performed an agglomerative hierarchical clustering over this baseline to create a hierarchy with branching factor two. This organization is called `clustering`. Then we used our algorithm to optimize the clustering organization to create N-dimensional ($N \in \{1, 2, 3, 4\}$) organizations (called `N-dim`). Figure 3 reports the success probability of each table in different organizations.

In the `baseline` organization, requires users to consider a large number of tags and select the best, hence the average success probability for tables in this organization is just 0.016. This `clustering` organization outperforms the `baseline` by ten times. This is because the smaller branching factor of this organization reduces the burden of choosing among so many tags as the flat organization and results in larger transition probabilities to states even along lengthy paths. Our `1-dim` optimization of the `clustering` organization improves the success probability of the `clustering` organization by more than three times.

To create the N-dimensional organization ($N > 1$), we clustered the tags into N clusters (using n-medoids) and built an organization on each cluster. The `2-dim` organization has an average success probability of 0.326 which is an improvement over the `baseline` by 40 times. Although the number of initial tags is invariant among 1-dimensional and multi-dimensional organizations since each dimension is constructed on a smaller number of tags that are more similar, increasing the number of dimensions in an organization improves the success probability, as shown in Figure 3.

In Figure 3, almost 47 tables of `TagCloud` have very low success probability in all organizations. We observed that almost 70% of these tables contain only one attribute each of which is associated with only one tag. This makes these

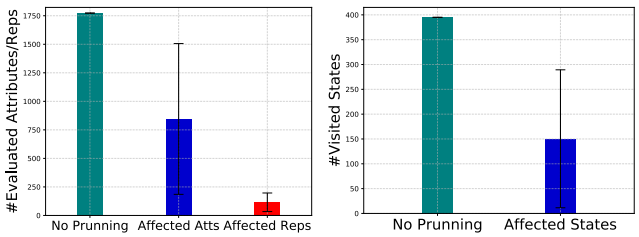


Fig. 4: Pruning (a) Domains and (b) States on `TagCloud` Benchmark.

tables less likely to be discovered in any organization. To investigate this further, we augmented `TagCloud` to associate each attribute with an additional tag (the closest tag to the attribute other than its existing tag). We built a two-dimensional organization on the enriched `TagCloud`, which we name `enriched 2-dim`. This organization proves to have higher success probability overall, and improves the success probability for the least discoverable tables.

5.3.2 Efficiency

The construction time of `clustering`, `1-dim`, `2-dim`, `3-dim`, `4-dim`, and `enriched 2-dim` organizations are 0.2, 231.3, 148.9, 113.5, 112.7, 217 seconds, respectively. Note that the `baseline` relies on the existing tags and requires no additional construction time. Since dimensions are optimized independently and in parallel, the reported construction times of the multi-dimensional organizations indicate the time it takes to finish optimizing all dimensions.

5.3.3 Approximation

The effectiveness and efficiency numbers we have reported so far are for the non-approximate version of our algorithm. To evaluate an organization during search we only examine the states and attributes that are affected by the change an operation has made. Thus, this pruning guarantees exact computation of success probabilities. Our experiments, shown in Figure 4, indicate that although local changes can potentially propagate to the whole organization, on average less than half of states and attributes are visited and evaluated for each search iteration. Furthermore, we considered approximating discovery probabilities using a representative set size of 10% of the attributes and only evaluated those representatives that correspond to the affected attributes. This reduces the number of discovery probability evaluations to only 6% of the attributes. As shown in Figure 3, named `2-dim approx`, this approximation has negligible impact on the success probabilities of tables in the constructed organization. The construction time of `2-dim` (without the approximation) is 148.9 seconds. For `2-dim approx` (using a representative size of 10%) is 30.3 seconds. The remaining experiments report organizations (on much larger lakes) created using this approximation for scalability.

5.4 Comparison of Organizations

We now compare our approach with (1) a hand-curated taxonomy on the `YagoCloud` benchmark, (2) the flat baseline on a large real data lake `Socrata`, and (3) an automatically generated enterprise knowledge graph (EKG) [21], [30] on a smaller sample of this lake `Socrata-1`.

5.4.1 Comparison to A Knowledge Base Taxonomy

We compare the effectiveness of using the existing YAGO taxonomy on `YagoCloud` tables for navigation with our organization. Figure 3b shows the success probability of tables in the benchmark when navigating the YAGO taxonomy and our data lake organization. The taxonomy consists of 1,171 nodes and 2,274 edges while our (1-dimensional) organization consists of 442 nodes and 565 edges. Each state in the optimized organization consists of a set of types that need to be further interpreted while each state in the taxonomy refers to a YAGO class label. However, the more compact representation of topics of the attributes leads to a more effective organization for navigation. To understand the influence of taxonomy/organization size on discovery effectiveness, we condensed the taxonomy. To create the condensed taxonomy, we have adopted a similar approach as *knowledge fragment selection approach [31]* to gather information which is most important to the task of navigation and eliminated the immediate level of nodes above the leaves unless they have other children. This leads to a taxonomy that has 444 nodes and 1,587 edges which is closer to the size of our organization. Condensing the taxonomy increases the branching factor of nodes while decreasing the length of discovery paths. This leads to a slight decrease in success probability compared to the original taxonomy. Knowledge base taxonomies are efficient for organizing the knowledge of entities. However, our organizations are able to optimize the navigation better based on the data lake distribution.

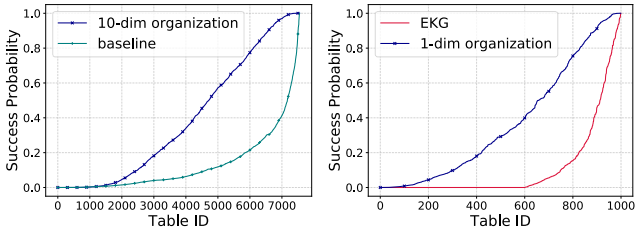
5.4.2 Comparison to a Baseline

We constructed ten organizations on the `Socrata` lake by first partitioning its tags into ten groups using k-medoids clustering [32]. We apply Algorithm 1 on each cluster to approximate an optimal organization. We use an agglomerative hierarchical clustering of tags in `Socrata` as the initial organization. In each iteration, we approximate the success probability of the organization using a representative set with a size that is 10% of the total number of attributes in the organization. Table 3 reports the number of representatives considered for this approximation in each organization along with other relevant statistics. Since the cluster sizes are skewed, the number of attributes reachable via each organization has a high variance. Recall that `Socrata` has just over 50K attributes and they might have multiple tags, so many are reachable in multiple organizations. It took 12 hours to construct the multi-dimensional organization.

Figure 5a shows the success probability of the ten organizations on the `Socrata` data lake. Using this organization, a table is likely to be discovered during navigation of the data lake with probability of 0.38, compared to the current state of navigation in data portals using only tags, which is 0.12. Recall that to evaluate the discovery probability of an attribute, we evaluate the probability of discovering the penultimate state that contains its tag and multiply it by the probability of selecting the attribute among the attributes associated with the tag. The distribution of attributes to tags depends on the metadata. Therefore, the organization

TABLE 3: Statistics of 10 Organizations of Socrata Lake.

Org	#Tags	#Atts	#Tables	#Reps
1	2,031	28,248	3,284	2,824
2	1,735	11,363	1,885	1,136
3	1,648	20,172	9,792	2,017
4	1,572	19,699	2,933	1,969
5	1,378	11,196	1,947	1,119
6	1,245	17,083	1,934	1,708
7	829	8,848	1,302	884
8	353	6,816	831	681
9	240	3,834	614	383
10	43	118	33	11



(a) (b)

Fig. 5: (a) Success Probability of Tables in Socrata Lake using Organization and Baseline. (b) Success Probability of Tables in Socrata-1 Lake using Organization and EKG.

algorithm does not have any control on the branching factor at the lowest level of the organization, which means that the optimal organization is likely to have a lower success probability than 1.

5.4.3 Comparison to Linkage Navigation

Automatically generated linkage graphs are an alternative navigation structure to an organization. An example of a linkage graph is the Enterprise Knowledge Graph (EKG) [21], [30]. Unlike organizations which facilitate exploratory navigation, in EKG, navigation starts with known data. A user writes queries on the graph to find related attributes through keyword search followed by similarity queries to find other attributes similar to these attributes.

To study the differences between linkage navigation and organization navigation, we compare the probability of discovering tables using each. We used a smaller lake (Socrata-1) for this comparison because the abundance of linkages between attributes in a large data lake makes our evaluation using large EKGs computationally expensive. In the EKG, each attribute in a data lake is a node of the graph and there are edges based on node similarity. A syntactic edge means the Jaccard similarity of attribute values [30] is above a threshold and a semantic edge means the combined semantic and syntactic similarity of the attribute names is above a threshold [21]. To use EKG for navigation, we adapt Equation 1 such that the probability of a user navigating from node m to an adjacent node s is proportional to the similarity of s to m and is penalized by the branching factor of m. We consider the similarity of s and m to be the maximum of their semantic similarity and syntactic similarity. Since the navigation can start from arbitrary nodes, to compute the success probability of a table in an EKG, we consider the average success probability of a table over up to 500 runs each starting from a random node. We use the threshold $\theta = 0.9$ for filtering the edges in EKG. This makes 3,989 nodes reachable from some node in the graph. The average and maximum branching factors of this EKG are 122.30 and 725, respectively. The average success probability of is 0.0056.

Figure 5b shows the success probability of navigation using an EKG and using an organization built on Socrata-1, when we limited the start nodes to be the ancestors of attributes of a table. Although the data lake organization has higher construction time (2.75 hours) than the EKG (1.3 hours), it outperforms EKG in effectiveness. The EKG is designed for discovering similar attributes to a known attribute. When EKG is used for exploration, the navigation can start from arbitrary nodes which results in long discovery paths and low success probability. Moreover, depending on the connectivity structure of an EKG, some attributes are not reachable in any navigation run (points in the left side of Figure 5b). In an EKG the number of navigation choices at each node depends on the distribution of attribute similarity. This causes some nodes to have high branching factor and leads to overall low success probability.

5.5 Analysis of Metadata Processing

For our experiments, we removed all tags from the CKAN lake and we transfer the tags of the Socrata data lake to attributes of CKAN. Figure 6a shows the distribution of deduplicated positive training samples per tag in Socrata. As part of the preprocessing, we perform syntactic, semantic, and structural normalization to the tags. For *syntactic normalization*, we perform word stemming[1]. Two tags that share the same stem are merged together. This process removed 1,186 tags. For *semantic normalization*, we merged tags that are semantically similar. For this, we rely on their word embedding vectors. Any pairs of tags that have cosine similarity between their embedding vectors greater than 0.9 are merged. This further reduced the unique tags by 1,156.

Finally, we perform *structural normalization* in which tags that have small support ($\leq$ 20 documents), and that are subsumed by at most two other tags are merged. Structural normalization reduced the tag count by over 8,000. This shows that, in practice, tag clouds have high degree of redundancy and that many outlier tags are not essential in generating navigational organizations. In the end, we have 1,156 meta tags to guide the organization.

We employed the distributed gradient boosting of XGBoost [33] to train classifiers on the tags with at least 10 positive training samples (866 tags). We considered the ratio of one to nine for positive and negative samples. The training algorithm performs grid search for hyper-parameter tuning of classifiers. Figure 6b demonstrates the accuracy of the 10-fold cross validation of the classifiers with top-100 F1-scores. Out of 866 tags of Socrata, 751 were associated with CKAN attributes and a total of 7,347 attributes got at least one tag. The most popular tag is *domaincategory_government*. Figure 7a shows the distribution of newly associated tags to CKAN attributes for the 20 most popular tags. Figure 7b reports the success probability of a 1-dimensional organization. More than half of the tables are now searchable

1. https://www.nltk.org/howto/stem.html

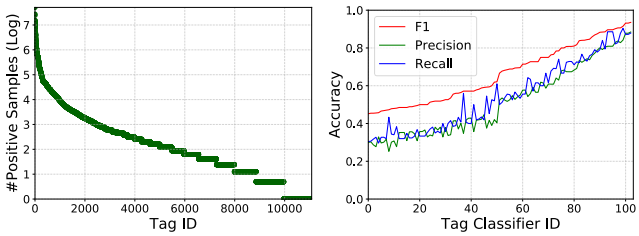


Fig. 6: (a) No. Positive Training Samples and (b) Accuracy of Tag Classifiers.

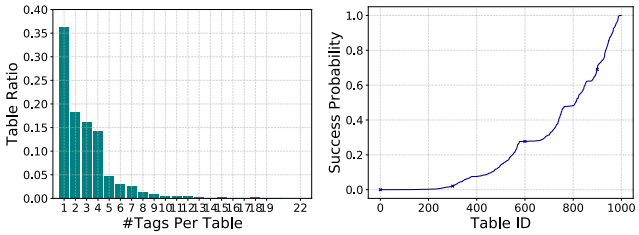


Fig. 7: (a) Distribution of Tags Added to CKAN (b) Success Prob. of CKAN after Metadata Enrichment.

through the organization that were otherwise unreachable in the lake.

5.6 User Study

We performed a formal user study to compare search by navigation to the major alternative of keyword search. Through a formal user study, we investigated how users perceive the usefulness of the two approaches.

To remain faithful to keyword search engines, we created a semantic search engine that supports keyword search over attribute values and tables metadata (including attribute names and table tags). The search engine performs query expansion with semantically similar terms. The search engine supports BM25 [34] document search and semantic keyword search using pre-trained GloVe word vectors [35]. Our implementation uses the library Xapian [2] to perform keyword search. Users can optionally enable query expansion by augmenting the keyword query with additional semantically similar terms. We also created a prototype that enables participants to navigate our organizations. In this prototype, each node is labeled with a set of representative tags. We label leaf nodes (where attributes are located) with corresponding table names and penultimate nodes (where single tags are located) with the corresponding tags. The remaining nodes are labeled with the two most frequent unseen tags, among its children's labels. Of course we may not always pick the best tag for a particular user. Hence, our system allows for the dynamic inclusion of additional tags at the interface level. A user may click to explore additional relevant tags to a node other than the ones used as labels. At each state, the user can navigate to a desired child node or backtrack to the parent of the current node.

2. https://xapian.org/

Hypotheses. The goal of our user study is to test the following hypotheses: H1) given the same amount of time, participants would be able to find as many relevant tables with navigation as with keyword search; H2) given the same amount of time, participants who use navigation would be able to find relevant tables that cannot be found by keyword search. We measure the disjointness of results with the symmetric difference of result sets normalized by the size of the union of results, i.e., for two result sets R and T, the disjointness is computed by $1 - \frac{|R \cap T|}{|R \cup T|}$.

Study Design. We considered Socrata-2 and Socrata-3 data lakes for our user study and defined an overview information need scenario for each data lake. We made sure that these scenarios are similar in difficulty by asking a number of domain experts who were familiar with the underlying data lakes to rate several candidate scenarios. Note that the statistically insignificant difference between the number of tabels found for each scenario by our participants provides evidence that our two scenarios were infact similar in difficulty. The scenario used for Socrata-2 asks participants to find tables relevant to the scenario "suppose you are a journalist and you'd like to find datasets published by governments on the topic of *smart city* ". The scenario used for Socrata-3 asks participants to find tables relevant to the scenario "Suppose you are a data scientist in a medical company and you would like to find datasets about *clinical research*". During the study, we asked participants to use keyword search and navigation to find a set of tables which they deemed relevant to given scenarios. For this study, we recruited 12 participants using convenience sampling [36], [37].

This study was a *within-subject* design. In our setting, we want to avoid two potential sources of invalidity. First, participants might become familiar with the underlying data lake during the first scenario which then might help them to search better in their second scenario. To address this problem, we made sure that Socrata-2 and Socrata-3 do not have overlapping tags and tables. Second, the sequence in which the participants use the two search approaches can be the source of confounding. To mitigate for this problem, we made sure that half of the participants first performed keyword search and the rest performed navigation first. In summary, we handled the effect of these two sources of invalidity using a balanced *Latin square* design with 4 blocks (block 1: Socrata-2/navigation first, block 2: Socrata-3/navigation first, etc.). We randomly assigned equal number of participants to each of these blocks. For each participant, the study starts with a short training session, after that, we gave the participants 20 minutes for each scenario.

Results. Because of our small sample size, we used the non-parametric Mann-Whitney test to determine the significance of the results and tested our two-tailed hypotheses. We found that there is no statistically significant difference between the number of relevant tables found using the organization and keyword search. We first asked two collaborators to eliminate irrelevant tables found. Since the number of irrelevant data was negligible (less than 1% for both approaches) we will not further report on this process. This confirms our first hypothesis. The maximum number of tables found by navigating an organization and performing

keyword search was 44 and 34, respectively. Moreover, a Mann-Whitney test indicated that the disjointness of results was greater for participants who used organization (Mdn = 0.985) than for participants who used keyword search (Mdn = 0.916), U = 612, p = 0.0019. This confirms our second hypothesis. Note that the disjointness was computed for each pair of participants who worked on the same scenario using the same approach, then the pairs generated for each technique were compared together. Based on our investigation, this difference might be because participants used very similar keywords, whereas the paths which were taken by each participant while navigating an organization were very different. In other words, As some participants described, they were having a hard time finding keywords that best described their interest since they did not know what was available, whereas with our organization, at each step, they could see what seemed more interesting to them and find their way based on their preferences. As one example, for the *smart city* overview scenario, everyone found tables tagged with the term `City` using search. But using organization, some users found traffic monitoring data, while others found crime detection data, while others found renewable energy plans. Of course, if they knew *a priori* this data was in the lake they could have formulated better keyword search queries, but navigation allowed them to conveniently discover these relevant tables without prior knowledge. One very interesting observation in this study is that although participants find similar number of tables, there is only around 5% intersection between tables found using keyword search and tables found using our approach. This suggests that organization can be a good complement to the keyword search and vice versa.

We evaluated the usability by asking each participant to fill out a standard post-experiment system usability scale (SUS) questionnaire [38] after each block. This questionnaire is designed to measure a user's judgment of a system's effectiveness and efficiency. We analyzed participants' rankings, and kept record of the number of questions for which they gave a higher rankings to each approach. Our results indicate that 58% of the participants preferred to use keyword search, we suspect in part due to familiarity. No participant had neutral preference. Still having 42% prefer navigation indicates a clear role for this second, complementary modality.

6 Related Work

Entity-based Querying - While traditional search engines are built for pages and keywords, in entity search, a user formulates queries to directly describe her target entities and the result is a collection of entities that match the query [39]. An example of a query is "database #`professor`", where `professor` is the target entity type and "database" is a descriptive keyword. Cheng et. al propose a ranking algorithm for the result of entity queries where a user's query is described by keywords that may appear in the context of desired entities [40].

Data Repository Organization - Goods is Google's specialized dataset catalog for about billions of Google's internal datasets [41]. The main focus of Goods is to collect and infer metadata for a large repository of datasets and make it searchable using keywords. Similarly, IBM's LabBook provides rich collaborative metadata graphs on enterprise data lakes [42]. Skluma [43] also extracts metadata graphs from a file system of datasets. Many of these *metadata* approaches include the use of static or dynamic linkage graphs [21], [30], [44], [45], join graphs for adhoc navigation [46], or version graphs [47]. These graphs allow navigation from dataset to dataset. However, none of these approaches learn new navigation structures optimized for dataset discovery.

Taxonomy Induction - The task of taxonomy induction creates hierarchies where edges represent *is-a* (or subclass) relations between classes. The *is-a* relation represents true abstraction, not just the *subset-of* relation as in our approach. Moreover, taxonomic relationship between two classes exists independent of the size and distribution of the data being organized. As a result, taxonomy induction relies on ontologies or semantics extracted from text [13] or structured data [48]. Our work is closes to concept learning, where entities are grouped into new concepts that are themselves organized in *is-a* hierarchies [49].

Faceted Search - Faceted search enables the exploration of entities by refining the search results based on some properties or facets. A facet consists of a predicate (e.g., `model`) and a set of possible terms (e.g., `Honda`, `Volvo`). The facets may or may not have a hierarchical relationship. Currently, most successful faceted search systems rely on term hierarchies that are either designed manually by domain experts or automatically created using methods similar to taxonomy induction [15], [50], [51]. The large size and dynamic nature of data lakes make the manual creation of a hierarchy infeasible. Moreover, since values in tables may not exist in external corpora [3], such taxonomy construction approaches of limited usefulness for the data lake organization problem.

Keyword Search - Google's dataset search uses keyword search over metadata and relies on dataset owners providing rich semantic metadata [1]. As shown in our user study this can help users who know what they are looking for, but has less value in serendipitous data discovery as a user tries to better understand what data is available in a lake.

7 Conclusion and Future Work

We defined the data lake organization problem of creating an optimal organization over tables in a data lake. We proposed a probabilistic framework that models navigation in data lakes on an organization graph. The data lake organization problem is framed as an optimization problem of finding an organization that maximizes the discovery probability of tables in a data lake and proposed an efficient approximation algorithm for creating good organizations. To build an organization, we use the attributes of tables together with any tags over the tables to combine table-level and instance-level features. We proposed a metadata enrichment technique for annotating attributes with tags, when the metadata is sparse. The metadata enrichment is a multi-label multi-class classification problem. As an application exercise, we modeled the problem as a set of binary classifiers for each tag. For future work, we plan to explore more elaborate algorithms for training a multi-class classifier of

all tags. The effectiveness and efficiency of our system are evaluated by benchmark experiments involving synthetic and real world datasets. We have also conducted a user study where participants use our system and a keyword search engine to perform the same set of tasks. It is shown that our system offers good performance, and complements keyword search engines in data exploration. Future work includes integrating keyword search and navigation as two interchangeable modalities in a unified data exploration framework. Based on the feedback and comments from the user study, we strongly believe that these extensions will further improve the user's ability to navigate in large data lakes.

REFERENCES

[1] D. Brickley, M. Burgess, and N. F. Noy, "Google dataset search: Building a search engine for datasets in an open web ecosystem," in *WWW*, 2019, pp. 1365–1375.

[2] R. Pimplikar and S. Sarawagi, "Answering table queries on the web using column keywords," *PVLDB*, vol. 5, no. 10, pp. 908–919, 2012.

[3] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller, "Table union search on open data," *PVLDB*, vol. 11, no. 7, pp. 813–825, 2018.

[4] A. Das Sarma, L. Fang, N. Gupta, A. Y. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu, "Finding related tables," in *SIGMOD*, 2012, pp. 817–828.

[5] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller, "LSH ensemble: Internet-scale domain search," *PVLDB*, vol. 9, no. 12, pp. 1185–1196, 2016.

[6] E. Zhu, D. Deng, F. Nargesian, and R. J. Miller, "JOSIE: overlap set similarity search for finding joinable tables in data lakes," in *SIGMOD*, 2019, pp. 847–864.

[7] J. Yang, W. Zhang, S. Yang, Y. Zhang, X. Lin, and L. Yuan, "Efficient set containment join," *VLDB J.*, vol. 27, no. 4, pp. 471–495, 2018.

[8] M. J. Cafarella, A. Y. Halevy, and N. Khoussainova, "Data integration for the relational web," *PVLDB*, vol. 2, no. 1, pp. 1090–1101, 2009.

[9] Y. Zhang and Z. G. Ives, "Finding related tables in data lakes for interactive data science," in *SIGMOD*, 2020, pp. 1951–1966.

[10] A. Bogatu, A. A. A. Fernandes, N. W. Paton, and N. Konstantinou, "Dataset discovery in data lakes," in *ICDE*, 2020, pp. 709–720.

[11] A. Callery, "Yahoo! Cataloging the Web," 1996. [Online]. Available: misc.library.ucsb.edu/untangle/callery.html

[12] R. Snow, D. Jurafsky, and A. Y. Ng, "Semantic taxonomy induction from heterogenous evidence," in *International Conference on Computational Linguistics*. The Association for Computer Linguistics, 2006, pp. 801–808.

[13] Z. Kozareva and E. Hovy, "A semi-supervised method to learn and construct taxonomies using the web," in *EMNLP*, 2010, pp. 1110–1118.

[14] H. Yang and J. Callan, "A metric-based framework for automatic taxonomy induction," in *International Conference on Computational Linguistics*. The Association for Computer Linguistics, 2009, pp. 271–279.

[15] B. Zheng, W. Zhang, and X. F. B. Feng, "A survey of faceted search," *Journal of Web engineering*, vol. 12, no. 1&2, pp. 041–064, 2013.

[16] J. Koren, Y. Zhang, and X. Liu, "Personalized interactive faceted search," in *WWW*, 2008, pp. 477–486.

[17] P. Velardi, S. Faralli, and R. Navigli, "Ontolearn reloaded: A graph-based algorithm for taxonomy induction," *Computational Linguistics*, vol. 39, no. 3, pp. 665–707, 2013.

[18] M. Arenas, B. Cuenca Grau, E. Kharlamov, v. Marciuška, and D. Zheleznyakov, "Faceted search over rdf-based knowledge graphs," *Web Semantics*, vol. 37, pp. 55–74, 2016.

[19] F. Nargesian, K. Q. Pu, E. Zhu, B. G. Bashardoost, and R. J. Miller, "Organizing data lakes for navigation," in *SIGMOD*. ACM, 2020, pp. 1939–1950.

[20] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *WWW*, 2007, pp. 697–706.

[21] R. C. Fernandez, E. Mansour, A. A. Qahtan, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang, "Seeping semantics: Linking datasets using word embeddings for data discovery," in *ICDE*. IEEE, 2018, pp. 989–1000.

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

[23] M. Hulsebos, K. Z. Hu, M. A. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp, and C. A. Hidalgo, "Sherlock: A deep learning approach to semantic data type detection," in *KDD*, 2019, pp. 1500–1508.

[24] Y. Ibrahim, M. Riedewald, G. Weikum, and D. Zeinalipour-Yazti, "Bridging quantities in tables and text," in *ICDE*. IEEE, 2019, pp. 1010–1021.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

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

[27] R. J. Miller, F. Nargesian, E. Zhu, C. Christodoulakis, K. Q. Pu, and P. Andritsos, "Making open data transparent: Data discovery on open data," *IEEE Data Eng. Bull.*, vol. 41, no. 2, pp. 59–70, 2018.

[28] O. Hassanzadeh, M. J. Ward, M. Rodriguez-Muro, and K. Srinivas, "Understanding a large corpus of web tables through matching with knowledge bases: an empirical study," in *Proceedings of the 10th Int. Workshop on Ontology Matching*, 2015, pp. 25–34.

[29] N. Friedman and D. Koller, "Being bayesian about network structure. A bayesian approach to structure discovery in bayesian networks," *Machine Learning*, vol. 50, no. 1-2, pp. 95–125, 2003.

[30] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker, "Aurum: A data discovery system," in *ICDE*. IEEE, 2018, pp. 1001–1012.

[31] S. Hellmann, J. Lehmann, and S. Auer, "Learning of owl class descriptions on very large knowledge bases," in *ISWC*, 2008, pp. 102–103.

[32] L. Kaufmann and P. Rousseeuw, "Clustering by means of medoids," *Data Analysis based on the L1-Norm and Related Methods*, 1987.

[33] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *SIGKDD*, 2016, pp. 785–794.

[34] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.

[35] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014, pp. 1532–1543.

[36] I. Etikan, S. A. Musa, and R. S. Alkassim, "Comparison of convenience sampling and purposive sampling," *American Journal of Theoretical and Applied Statistics*, vol. 5, no. 1, pp. 1–4, 2016.

[37] L. M. Given, Ed., *The Sage encyclopedia of qualitative research methods*. Lox Angeles, Calif: Sage Publications, 2008.

[38] J. Brooke, "Sus: A retrospective," *J. Usability Studies*, vol. 8, no. 2, pp. 29–40, 2013.

[39] S. Chakrabarti, K. Puniyani, and S. Das, "Optimizing scoring functions and indexes for proximity search in type-annotated corpora," in *WWW*, 2006, pp. 717–726.

[40] T. Cheng, X. Yan, and K. C.-C. Chang, "Entityrank: Searching entities directly and holistically," in *VLDB*, 2007, pp. 387–398.

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

[42] 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*, 2015, pp. 431–440.

[43] 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*, 2017, pp. 41:1–41:4.

[44] 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.

[45] E. Mansour, D. Deng, R. C. Fernandez, A. A. Qahtan, W. Tao, Z. Abedjan, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang, "Building data civilizer pipelines with an advanced workflow engine," in *ICDE*. IEEE, 2018, pp. 1593–1596.

[46] E. Zhu, K. Q. Pu, F. Nargesian, and R. J. Miller, "Interactive navigation of open data linkages," *PVLDB*, vol. 10, no. 12, pp. 1837–1840, 2017.

[47] J. M. Hellerstein, V. Sreekanti, J. E. Gonzalez, J. Dalton, A. Dey, S. Nag, K. Ramachandran, S. Arora, A. Bhattacharyya, S. Das, M. Donsky, G. Fierro, C. She, C. Steinbach, V. Subramanian, and E. Sun, "Ground: A data context service," in *CIDR*, 2017.
[48] J. Pound, S. Paparizos, and P. Tsaparas, "Facet discovery for structured web search: a query-log mining approach," in *SIGMOD*, 2011, pp. 169–180.
[49] J. Lehmann, "Dl-learner: Learning concepts in description logics," *J. Mach. Learn. Res.*, vol. 10, pp. 2639–2642, 2009.
[50] J. Pound, S. Paparizos, and P. Tsaparas, "Facet discovery for structured web search: A query-log mining approach," in *SIGMOD*, 2011, pp. 169–180.
[51] H. Duan, C. Zhai, J. Cheng, and A. Gattani, "Supporting keyword search in product database: A probabilistic approach," *PVLDB*, vol. 6, no. 14, pp. 1786–1797, 2013.

Fatemeh Nargesian is an assistant professor in computer science at the University of Rochester. She received her PhD from the University of Toronto in 2019.

Ken Q. Pu is an associate professor in Computer Science at Ontario Tech University. His research interests are in novel data processing techniques and applications of database systems.

Bahar Ghadiri Bashardoost is a PhD candidate at the department of computer science, University of Toronto.

Erkang Zhu is a senior researcher at Microsoft Research, focusing on data science tools and platforms. He received his PhD from the University of Toronto in 2019.

Renée J. Miller is a University Distinguished Professor of Computer Science in the Khoury College of Computer Sciences at Northeastern University. She received her PhD in computer science from the University of Wisconsin, Madison and bachelors degrees in mathematics and cognitive science from MIT.