Organizing Data Lakes for Navigation

Fatemeh Nargesian
University of Rochester
f.nargesian@rochester.edu

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

Erkang Zhu
Microsoft Research
ekzhu@microsoft.com

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

Renée J. Miller
Northeastern University
miller@northeastern.edu

ABSTRACT

We consider the problem of creating an effective navigation structure over a data lake. We define an *organization* as a navigation graph that contains nodes representing sets of attributes within a data lake and edges indicating subset relationships among nodes. We propose the *data lake organization problem* as the problem offi nding an organization that allows a user to most effectively navigate a data lake. We present a new probabilistic model of how users interact with an organization and propose an approximate algorithm for the data lake organization problem. We show the effectiveness of the algorithm on both a real data lake containing data from open data portals and on a benchmark that contains rich metadata emulating the observed characteristics of real data lakes. Through a formal user study, we show that navigation can help usersfi nd relevant tables that cannot be found by keyword search.

CCS CONCEPTS

• **Information systems → Data management systems**;

KEYWORDS

data lakes; dataset discovery and search; structure learning

ACM Reference Format:
Fatemeh Nargesian, Ken Q. Pu, Erkang Zhu, Bahar Ghadiri Bashardoost, and Renée J. Miller. 2020. Organizing Data Lakes for Navigation. In *2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3318464.3380605

1 INTRODUCTION

The popularity 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 tofi nd tables relevant to the keywords [3, 36]. Alternatively, the query can be a table and the problem is tofi nd other tables that are close to the query table [5]. If the input is a query table, then the output may be tables that join or union with query table [7, 34, 45, 47, 48].

A complementary alternative to search is navigation. In this paradigm, a user navigates through an organizational structure tofi nd 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 [6]. Even today, hierarchical organizations of Web content (especially entities like videos or products) is still used by companies such as `Youtube.com` and `Amazon.com`. Hierarchical navigation allows a user to browse available entities going from more general concepts to more specific concepts using existing ontologies or structures automatically created using taxonomy induction [26, 40, 44]. When entities have known features, we can apply faceted-search over entities [25, 41, 46]. Taxonomy induction looks for *is-a* relationships between entities (e.g., `student is-a person`), faceted-search applies predicates (e.g., `model = volvo`) to filter entity collections [1]. In contrast to hierarchies over entities, in data lakes tables contain attributes that may mention many different types of entities and relationships between them. There may be *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 (perhaps using entity-mentions in attribute values), the *is-a* relationships between class labels could provide an organization on tables. However, recent studies show the difference in size and coverage of domains in various public cross-domain knowledge bases and highlight that no single public knowledge base sufficiently covers all domains and attribute values represented in heterogeneous corpuses such

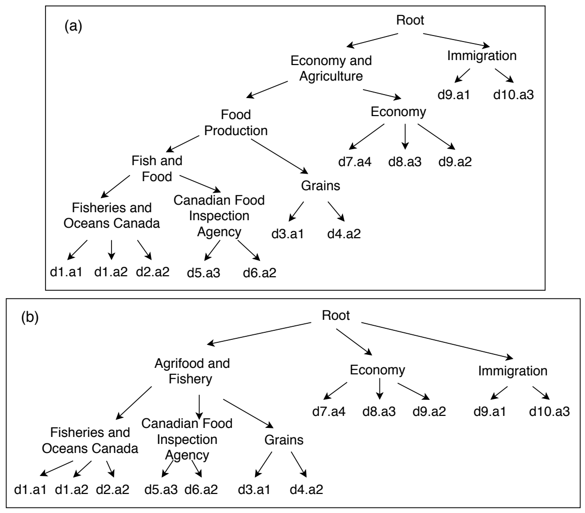


Figure 1: (a) Deep and (b) Effective Organization.

as data lakes [16, 34, 39]. Knowledge bases are also not designed to provide effective navigation. We propose instead to build an organization that is designed to best support navigation and exploration over heterogeneous 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.

1.1 Organizations

Dataset or table search is often done using attributes by finding similar, joinable, or unionable attributes [7, 11, 12, 34, 47, 48]. We follow a similar approach and 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 the attribute values themselves or from metadata when available. A table can be associated to all nodes containing one or more of its attributes. An edge in this DAG indicates that the attributes in a parent node is a superset of the attributes in a child node. A userfi nds 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 tofi nd an organization that allows a user to most efficientlyfi nd tables. We describe user navigation 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 in an organization indicates the transition from the current state to a next state. Due to the subset property of edges, each transitionfi lters out some attributes until the navigation reaches attributes of interest.

An organization is effective if certain properties hold. At each step, a user should have only a reasonable number of choices. We call the maximum number of choices the *branching factor*. The choices should be distinct (as dissimilar as possible) to make it easier for a user to choose the most relevant one. Also, the number of choices she needs to make (the *length of the discovery path*) should not be large. Hence, give the typical skew of topics in real data lakes, a data lake organization must be able to automatically determine over which portions of the data, more organizational structure is required, and where a shallow structure is sufficient.

Example1. *Consider a collection of tables, $d1, \ldots, d10$ from Canadian Open Data. They contain data on various topics such as agrifood,fi sheries, the economy, and immigration. One way to expose this data to a user is through afl at structure of attributes of these tables. A user can browse the data (and any associated metadata) 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, she has to choose between only two nodes. Herfi rst two choices seem clear.* Economy and Agriculture *and* Food Production *seem more relevant to her 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 in the organization below the* Fish and Food*. The organization in Figure 1(b) addresses this problem by organizing attributes of* Grains, Food Inspection*, and* Fisheries *at the same level. Note that this organization has a higher branching factor, but the choices are more distinct at each node.*

1.2 Contributions

We propose a navigation model for the user experience during discovery using an organization. We define when an organization is optimal in that it is best suited to help a user navigate to an attribute of interest.

- We propose to model navigation as a Markov model, which computes the probability of discovering a table relevant to a topic of interest. We define the *data lake organization problem* as the problem offi nding an organization that maximizes the expected probability of discovering tables.
- We propose an algorithm that approximates an optimal organization by leveraging available metadata.
- Through a user-study comparing navigation to keyword search, we show that navigation can help usersfi nd a more diverse set of tables than keyword search. In addition, in our study, 42% preferred the use of navigation over

keyword search (the others preferred search), suggesting these are complementary and both useful modalities for dataset discovery in data lakes.

2 FOUNDATIONS

We describe a probabilistic model of how users navigate a data lake. 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. We then define the data lake organization problem as the problem offinding the most effective organization for a data lake.

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, attr(T). Let $\mathcal{A}$ be the set of all attributes, $\mathcal{A} = \bigcup\{\text{attr}(T) : T \in \mathcal{T}\}$ in a data lake. Each attribute A has a set of values that we call a domain and denote by dom(A). An organization $O = (V, E)$ is a DAG. Let ch(.) be the child relation mapping a node to its children, and par(.) be the parent relation mapping a node to its parents. A node $s \in V$ is a leaf if ch(s) $= \emptyset$, otherwise s is an interior node in O. Every leaf node s of 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 dom(s) which is dom(A_s) when s is a leaf node and $\bigcup_{A \in D_s}$ dom(A) otherwise.

2.2 Navigation Model

We model a user's experience during discovery using an organization O as a Markov model where states are nodes and transitions are edges. We will use the terms state and node interchangably. Because of the inclusion property, transitions from a statefi lter out some of the attributes of the state. Users select a transition at each step and discovery stops once they 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. In our definition of effectiveness, we assume the probability of a user's transition from s to $c \in$ ch(s) is determined by the similarity between X and the values of the attributes in D_c.

Example2. *Returning to Example 1, to evaluate (and compare) the effectiveness of the two organizations in Figure 1 for a specific user topic like X = "food inspection" we can compute the similarity of this topic to the values in the attributes of the various nodes in each organization.*

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 she chooses nodes that are closest to her (unknown to us) intended topic query. The concept of a user's query topic is used only to build a good organization. We define an optimization problem where we build an organization that maximizes the expected probability offinding any table in the data lake byfi nding 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 attributes in the lake.

2.3 Transition Model

We define the transition probability of $P(c|s, X, O)$ as the probability that the user will choose the next state as c if they are at the state s. The probability should be correlated to the similarity between dom(c) and X. Let $\kappa(c, X)$ be a similarity metric between dom(c) and X. The transition probability is as follows.

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

The constant γ is a hyper parameter of our model. It must be a strictly positive number. The term $|$ 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 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.

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

In this model, a user makes transition choices only based on the current state and the similarity of her 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 O while searching for X is as follows.

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

where $Paths(s)$ is the set of all discovery sequences in O that reach s from the root. Additionally, the probability of reaching a state can be evaluated incrementally.

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

Definition1. *The* discovery probability *of an attribute A in organization O is defined as $P(s|A,O)$, where s is a leaf node. We denote the discovery probability of A as $P(A|O)$.*

2.4 Data Lake Organization

A table T is discovered in an organization O by discovering any of its attributes.

Definition2. *For a single table T, we define the* discovery probability of a table *as*

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

For a set of tables $\mathcal{T}$ the organization effectiveness *is*

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

For a data lake, we define the data lake organization problem asfi nding an organization that has the highest organization effectiveness over all tables in the lake.

Definition3. Data Lake Organization Problem. *Given a set of tables $\mathcal{T}$ in a data lake, the organization problem is to find an organization $\hat{O}$ such that:*

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

We remark that in this model we do not assume the availability of any query workloads. Since our model uses a standard Markov model, we can apply existing incremental model estimation techniques [24] to maintain and update the transition probabilities as behavior logs and workload patterns become available through the use of an organization by users.

2.5 Multi-dimensional Organizations

Given the heterogeneity and massive size of data lakes, it may be advantagous to perform an initial grouping of tables and then build an organization on each group. If we have a grouping of $\mathcal{A}$ into k (possibly overlapping) groups, $G_1, \ldots, G_k$, then we can discover an organization over each and use them collectively for navigation. Given a grouping $G_1, \ldots, G_k$ of attributes, we define a k-dimensional organization $\mathcal{M}$ for a data lake $\mathcal{T}$ as a set of organizations $\{O_1, \ldots, O_k\}$, such that attributes of each table in $\mathcal{T}$ is organized in at least one organization and O_i is the most effective organization for G_i. The probability of discovering table T in $\mathcal{M}$ is the probability of discovering T in any of dimensions of $\mathcal{M}$.

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

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 justify our design choices using lessons learned from search-based table discovery. We then consider the metadata that is often available in data lakes, specifically table-level tags, and explain how this metadata (if available) can be exploited for navigation. Finally, we present a local search algorithm for building an approximate solution for the data lake organization problem.

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. Hence, to use numerical attributes one wouldfi rst need to understand their semantics. Work is emerging on how to do this [18–20]. We leave the exploitation of numerical attributes in navigation for future research as doing this well would only improve on our approach.

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 a set of relevant class labels from a knowledge base [18, 28, 42]. We have found that open knowledge bases have relatively low coverage on data lakes [34]. Hence, we follow an approach which has proven to be successful in search-based table discovery which is to represent attributes by their collective word embedding vectors [34].

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 an Euclidean or angular distance measure [31]. Attribute A can be represented by a *topic vector*, μ_A, which is the sample mean of the population $\{\vec{v}|v \in \mathrm{dom}(A)\}$ [34]. In organization construction, we also represent X by μ_X.

Definition4. *State s is represented with a* topic vector, *μ_s, which is the sample mean of the population $\{\vec{v}|v \in \mathrm{dom}(s)\}$.*

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 [21], 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. This is because $P(c|s, X, O)$ is normalized with all children of parent s.

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 [32]. In mass-collaboration datasets, like web tables, contextual information can be used to extract metadata [16].

State Construction. If metadata is available, it can be distilled into tags (e.g., keywords, concepts, or entities). We associate these table-level tags with every attribute in the table. 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. We call these parents *tag states*. 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.

Example3. *In the organization of Figure 1 (b) we have actually used real tags from open data to label the internal nodes (with the exception of* Agrifood and Fishery *which is our space saving representation of a node representing the three real tags:* Canadian Food Inspection Agency*,* Fisheries and Oceans Canada *and* Grains*). Instead of building an organization over the twelve attributes, we can build a smaller organization over thefi ve tags:* Canadian Food Inspection Agency*,* Fisheries and Oceans Canada*,* Grains*,* Economy*, and* Immigration.

Definition5. *Let* data(t) *be the relation mapping a tag t in a lake metadata to the set of its associated attributes. Suppose M_s is the set of tags in a tag state s. Thus, the set of attributes in s is $D_s = \bigcup_{t \in M_s} \text{data}(t)$. A tag state s is represented with a* topic vector*, μ_s, which is the sample mean of the population* $\{\vec{v} | v \in \text{dom}(A), A \in D_s\}$.

Flat Organization: A Baseline. In the organization, we consider the leaves to be states with one attribute. However, now the parent of a leaf node is associated to only one tag. This means that the last two levels of a hierarchy arefi xed and an organization is constructed over states with a single tag. If instead of discovery, 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 begins with an initial organization. At each step, the algorithm proposes a *modification* to the current organization O which leads to a new organization O'. 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 with a probability that is a ratio of the effectiveness, namely [13]:

$$min[1, \frac{P(\mathcal{T}|O')}{P(\mathcal{T}|O)}] \tag{9}$$

The algorithm terminates 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.

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

$$P(s|O) = \frac{1}{|A \in \mathcal{T}|} \sum_{A \in \mathcal{T}} P(s|A, O) \tag{10}$$

Starting from an initial organization, the search algorithm performs downward traversals from the root and proposes a modification on the organization for states in each level of the organization ordered from lowest reachability probability to highest. A state is in level l if the length of the shortest discovery paths from root to the state is l. We restrict our choices of a new organization at each search step to those created by two operations.

Operation I: 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, or that it is indistinguishable from a sibling. We can remedy either of these by adding a new parent. Suppose that the search algorithm chooses when considering a state s to modify the organization. Recall that Equation 4 indicates that the probability of reaching a state increases as it is connected to more parent states. Suppose s is at level l of an organization. The algorithmfi nds a state, called n, at level $l - 1$ of the organization such that it is not a parent of s and has the highest reachability probability among the states at level $l - 1$. To satisfy the inclusion property, we update node n and its ancestors to contain the attributes in s (that is, D_s). To avoid generating cycles in the organization, the algorithm makes sure that none of the children of s is chosen as a new parent. State n is added to the organization as a new parent of s. Note that `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 of the newly added parent.

EXAMPLE4. *For example, in Figure 1(b) the attribute* `d6.a2` *is actually from a table about* Canadian Food Inspection Agency (CFIA) Fish List. *This attribute is reachable from the node with the tag* Canadian Food Inspection Agency. *This attribute is also related to the node with the tag* Fisheries and Oceans Canada *which in this organization is not a parent. The algorithm can decide to add an edge from* Fisheries and Oceans Canada *to* `d6.a2` *and make this state its second parent.*

Operation II: Deleting Parent. Another reason a state can have low reachability is that its parent has low reachability probability and we should perhaps remove its parent. Reducing the length of paths from the root to state s is a second way to boost the reachability probability of this state. The operation eliminates the least reachable parent of s (which we called r). To reduce the height of O, the operation eliminates all siblings of r except siblings with one tag. Then, it connects the children of each eliminated state to its parents. This makes the length of paths to s smaller which boosts the reachability probability of this state. However, replacing the children of a state by all its grandchildren increases the branching factor of the state, thus, decreasing the transition probabilities from that state.

EXAMPLE5. *For example, the algorithm can decide to eliminate the state* Economy and Agriculture *in the organization of Figure 1(a) and connect its two children directly to the root.*

The initial organization may be any organization that satisfies the inclusion property of attributes of states. For example, the initial organization can be the DAG defined based on a hierarchical clustering of the tags of a data lake.

3.4 Scaling Organization Search

The local search algorithm makes local changes to an existing organization by applying operations. An operation is successful if it increases the effectiveness of an organization. The evaluation of the effectiveness (Equation 7) involves computing the discovery probability for all attributes which requires 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, wefirst 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.

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 this sub-graph rooted by the grandparent and only for attributes associated to the leaves of the sub-graph.

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 this state. Furthermore, the parents of s and their ancestors are updated to satisfy the inclusion property of states. Suppose a parent 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 compute the LCA of s and each of the parents of its parent-to-be. All states in the sub-graph of the LCA require re-evaluation.

Focusing on states that are affected by operations reduces the complexity of the exact evaluation of an organization. To further speed up search, we evaluate an organization on a small number of attribute representatives. Then we evaluate the effectiveness of an organization on this small set at each search iteration. The discovery probability of each representative approximates the discovery probability of its corresponding attributes. We assume a one-to-one mapping between representatives and a partitioning of attributes. Suppose ρ is a representative for a set of attributes $D_\rho = \{A_1, \ldots, A_m\}$. We approximate $P(A_i|O), A_i \in D_\rho$ with $P(\rho|O)$. The choice and the number of representatives impact the error of this approximation. Note that during the approximation we consider the states and the representatives corresponding to the attributes affected by the local change.

4 EVALUATION

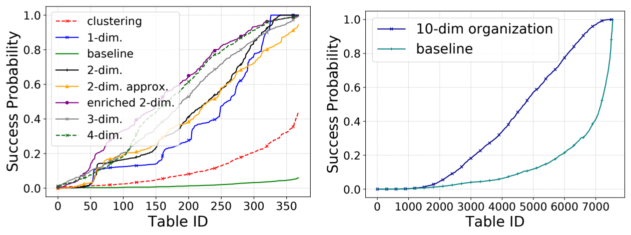
We now experimentally study several of our design decisions and the influence of using approximation for our approach. We do this using a small synthesized benchmark called `TagCloud` that is designed so that we know precisely the best tag per attribute. Next, using a large real open data lake, we quantify the benefits of our approach over aflat baseline where we ask what value is provided by the hierarchy we create over just grouping tables by tags. Finally, we present a user study in Section 4.4.

4.1 Datasets

Socrata Data Lakes - For our comparison studies, we used real open data. We crawled 7,553 tables with 11,083 tags 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. We have 264,199 attribute-tag associations. The distribution of tags per table and attributes per table of `Socrata` lake is skewed with two tables having over 100K tags and the majority of the tables having 25 or fewer. `Socrata-2` is a collection of 2,175 tables with 13,861 attributes and 345 tags from `Socrata` and `Socrata-3` is a collection of 2,061 tables with 16,075 attributes and 346 tags from `Socrata`. Note that `Socrata-2` and `Socrata-3` do not share any tags and are used in our user study.

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 to tags. Each attribute in the benchmark is associated to exactly one 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 1000). 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. Therefore, 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), 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.



(a) (b)

Figure 2: Success Probability of Organizations on (a) `TagCloud` Benchmark and (b) `Socrata` Lake.

4.2 Experimental Set-up

Evaluation Measure. For our initial studies evaluating our design decisions and comparing our approach to three others (Yago, a simple tag-based baseline, and an Enterprise Knowledge Graph), we do not have a *user*. We simulate a user by reporting the success probability offi nding 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 wouldfi nd 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 itfi nds tables with an attribute of or a similar attribute to the query's attributes. Wefi rst 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. The *success probability* of an attribute A is defined as $Success(A|O) = 1 - \prod_{A_i \in \mathcal{A} \wedge \kappa(A_i,A) \geq \theta}(1 - P(A_i|O))$. 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 $Success(T|O) = 1 - \prod_{A \in T}(1 - Success(A|O))$. We report success probability for every table in the data lake sorted from lowest to highest probability on the x-axis (see Figure 2 as an example).

4.3 Performance of Approximation

We evaluate the effectiveness and efficiency of our exact algorithms for the exact computation of discovery probabilities on the `TagCloud` benchmark and a real data lake.

4.3.1 Effectiveness. We constructed the `baseline` organization where each attribute has as parents the tag-state 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 but of course much cleaner than real data portals. We performed an agglomerative hierarchical clustering over this baseline to create a hierarchy with branching factor 2. 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 2 reports the success probability of each table in different organizations.

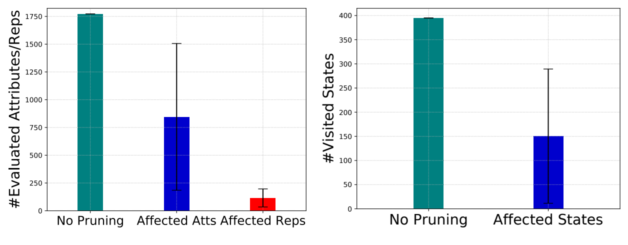
The `baseline` organization requires users to consider a large number of tags and select the best, hence the average success probability for tables in this organization is just 0.016. The `clustering` organization outperforms the `baseline` by ten times. This is because the smaller branching factor of this organization reduces the burden of choosing among so many tags as thefl at 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 0.426 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, increasing the number of dimensions in an organization improves the success probability, as shown in Figure 2. This is because each dimension is constructed on a smaller number of tags that are more similar

In Figure 2, 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 to only one tag. This makes these 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.

4.3.2 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 tofi nish optimizing all dimensions.

4.3.3 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



(a) **(b)**

Figure 3: Pruning (a) Domains and (b) States on `TagCloud` Benchmark.

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 3, 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 2, 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%) the construction time is 30.3 seconds. The remaining experiments report organizations (on much larger lakes) created using this approximation for scalability.

4.3.4 Comparison to a Baseline on Real Data. We constructed ten-dimensional organizations on the `Socrata` lake byfi rst partitioning its tags into ten groups using k-medoids clustering [23]. Then, we applied our organization algorithm 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 1 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.

Table 1: 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

Figure 2b shows the success probability of the ten-dimensional organization on the `Socrata` 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 to the tag. The distribution of attributes to tags depends on the metadata. Therefore, the organization 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.

4.4 User Study

We performed a formal user study to compare discovery using navigation to the major alternative of using keyword search. 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 table metadata (including attribute names and table tags). We use pretrained GloVe word vectors [35] to evaluate the similarity of words and identify similar terms. The search engine uses the Xapian library [43] to perform keyword search and supports BM25 [29] document search over metadata and data in tables. Users can optionally disable query expansion. We also created a prototype that enables participants to navigate our organizations. In this prototype, each node of an organization is labeled with a set of representative tags. We label leaf nodes with corresponding table names and penultimate nodes (where single tags are located) with the corresponding tags. Otherwise, a node is labeled with two tags which are thefi rst and the second most occurring tags, among its children's labels. If these tags belong to the label of the same child, we choose the third most occurring tag and so on. At each state, the user can navigate to a desired child node or backtrack to the parent of the current node.

Hypotheses. The goal of our user study is to test the following hypotheses: H1) given the same amount of time, participants would be able tofi nd more relevant tables with navigation than with keyword search; H2) given the same amount of time, participants who use navigation would be able tofi nd relevant tables that cannot be found by keyword search. For the second hypotheses, 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. The scenario used for `Socrata-2` asks participants tofi nd tables relevant to the scenario "suppose you are a journalist and you'd like tofi nd datasets published by governments on the topic of *smart city*". The scenario used for `Socrata-3` asks participants tofi nd tables relevant to the scenario "suppose you are a data scientist in a medical company and you would like tofi nd datasets about *clinical research*". During the study, we asked participants to use keyword search and navigation tofi nd a set of tables which they deemed relevant to given scenarios. For this study, we recruited 12 participants (details on participants is available in [33]) using convenience sampling [10, 14].

This study was a *within-subject* design. To avoid sources of invalidity (the order of using search techniques and the similarity of data lakes),fi rst, we made sure that `Socrata-2` and `Socrata-3` do not have overlapping tags or tables. Second, we used a balanced *latin square* design with 4 blocks (block 1: `Socrata-2`/navigationfi rst, block 2: `Socrata-3`/navigation first, etc.). We randomly assigned an 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. Wefi rst 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. 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.

This confirms ourfi rst hypothesis. The largest number of tables found by navigation and 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 technique, then the statistical significance of sets of were evaluated. 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. As some participants described, they were having a hard timefi nding 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. 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. One very interesting observation in this study is that although participantsfi nd a 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 navigation 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 [4] after each block. 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.

5 RELATED WORK

Data Repository Organization - Goods is Google's specialized dataset catalog for about billions of Google's internal datasets [15]. 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 [22]. Skluma [2] also extracts metadata graphs from afi le system of datasets. Many of these *metadata* approaches include the use of static or dynamic linkage graphs [8, 11, 12, 30], join graphs for adhoc navigation [49], or version graphs [17]. 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 [26] or structured data [37]. Our work is closes to concept learning, where entities are grouped into new concepts that are themselves organized in *is-a* hierarchies [27].

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 [9, 38, 46]. The large size and dynamic nature of data lakes makes the manual creation of a hierarchy infeasible. Moreover, since values in tables may not exist in external corpora [34], 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 [3]. 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.

6 CONCLUSION AND FUTURE WORK

We define the data lake organization problem as the problem offi nding an organization that maximizes the discovery probability of tables in a data lake and proposed an algorithm for creating good organizations. The effectiveness and efficiency of our algorithm are evaluated by experiments on synthetic and real world datasets. Our formal user study shows that our organizations offer good performance and complement keyword search in data exploration. Future work includes extending the organization to include numerical and text columns. In future work, we plan to compare organizations with existing taxonomies and to provide techniques for metadata enrichment. We also plan to perform a detailed scalability study of our technique with respect to the size of data lakes and to integrate keyword search and navigation as two interchangeable modalities in a unified framework.

Funding: NSERC Strategic Grant.

REFERENCES

[1] Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Kharlamov,Šarunas Marciuška, and Dmitriy Zheleznyakov. 2016. Faceted Search over RDF-based Knowledge Graphs. *Web Semantics* 37 (2016), 55–74.

[2] Paul Beckman, Tyler J. Skluzacek, Kyle Chard, and Ian T. Foster. 2017. Skluma: A Statistical Learning Pipeline for Taming Unkempt Data Repositories. In *Scientific and Statistical Database Management*. 41:1–41:4.

[3] 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.

[4] John Brooke. 2013. SUS: A Retrospective. *J. Usability Studies* 8, 2 (2013), 29–40.

[5] Michael J. Cafarella, Alon Y. Halevy, and Nodira Khoussainova. 2009. Data Integration for the Relational Web. *PVLDB* 2, 1 (2009), 1090–1101.

[6] Anne Callery. 1996. Yahoo! Cataloging the Web. misc.library.ucsb.edu/untangle/callery.html

[7] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Y. Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding related tables. In *SIGMOD*. 817–828.

[8] Dong Deng, Raul Castro Fernandez, Ziawasch Abedjan, Sibo Wang, Michael Stonebraker, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, and Nan Tang. 2017. The Data Civilizer System. In *CIDR*.

[9] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. Supporting Keyword Search in Product Database: A Probabilistic Approach. *PVLDB* 6, 14 (2013), 1786–1797.

[10] Ilker Etikan, Sulaiman Abubakar Musa, and Rukayya Sunusi Alkassim. 2016. Comparison of Convenience Sampling and Purposive Sampling. *American Journal of Theoretical and Applied Statistics* 5, 1 (2016), 1–4.

[11] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A Data Discovery System. In *ICDE*. IEEE, 1001–1012.

[12] Raul Castro Fernandez, Essam Mansour, Abdulhakim Ali Qahtan, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2018. Seeping Semantics: Linking Datasets Using Word Embeddings for Data Discovery. In *ICDE*. IEEE, 989–1000.

[13] Nir Friedman and Daphne Koller. 2003. Being Bayesian About Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning* 50, 1-2 (2003), 95–125.

[14] Lisa M Given (Ed.). 2008. *The Sage encyclopedia of qualitative research methods*. Sage Publications, Lox Angeles, Calif.

[15] Alon Halevy, Flip Korn, Natalya F. Noy, Christopher Olston, Neoklis Polyzotis, Sudip Roy, and Steven Euijong Whang. 2016. Goods: Organizing Google's Datasets. In *SIGMOD*. 795–806.

[16] Oktie Hassanzadeh, Michael J Ward, Mariano Rodriguez-Muro, and Kavitha Srinivas. 2015. 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*. 25–34.

[17] Joseph M. Hellerstein, Vikram Sreekanti, Joseph E. Gonzalez, James Dalton, Akon Dey, Sreyashi Nag, Krishna Ramachandran, Sudhanshu Arora, Arka Bhattacharyya, Shirshanka Das, Mark Donsky, Gabriel Fierro, Chang She, Carl Steinbach, Venkat Subramanian, and Eric Sun. 2017. Ground: A Data Context Service. In *CIDR*.

[18] Madelon Hulsebos, Kevin Zeng Hu, Michiel A. Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César A. Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *KDD*. 1500–1508.

[19] Yusra Ibrahim, Mirek Riedewald, and Gerhard Weikum. 2016. Making Sense of Entities and Quantities in Web Tables. In *CIKM*. 1703–1712.

[20] Yusra Ibrahim, Mirek Riedewald, Gerhard Weikum, and Demetrios Zeinalipour-Yazti. 2019. Bridging Quantities in Tables and Text. In *ICDE*. IEEE, 1010–1021.

[21] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. *ACL* (2017).

[22] Eser Kandogan, Mary Roth, Peter M. Schwarz, Joshua Hui, Ignacio G. Terrizzano, Christina Christodoulakis, and Renée J. Miller. 2015. LabBook: Metadata-driven social collaborative data analysis. In *IEEE Big Data*. 431–440.

[23] Leonard Kaufmann and Peter Rousseeuw. 1987. Clustering by Means of Medoids. *Data Analysis based on the L1-Norm and Related Methods* (1987).

[24] Wael Khreich, Eric Granger, Ali Miri, and Robert Sabourin. 2012. A survey of techniques for incremental learning of HMM parameters. *Inf. Sci.* 197 (2012), 105–130.

[25] Jonathan Koren, Yi Zhang, and Xue Liu. 2008. Personalized interactive faceted search. In *WWW*. 477–486.

[26] Zornitsa Kozareva and Eduard Hovy. 2010. A Semi-supervised Method to Learn and Construct Taxonomies Using the Web. In *EMNLP*. 1110–1118.

[27] Jens Lehmann. 2009. DL-Learner: Learning Concepts in Description Logics. *J. Mach. Learn. Res.* 10 (2009), 2639–2642.

[28] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and Searching Web Tables Using Entities, Types and Relationships. *PVLDB* 3, 1-2 (2010), 1338–1347.

[29] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.

[30] Essam Mansour, Dong Deng, Raul Castro Fernandez, Abdulhakim Ali Qahtan, Wenbo Tao, Ziawasch Abedjan, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2018. Building Data Civilizer Pipelines with an Advanced Workflow Engine. In *ICDE*. IEEE, 1593–1596.

[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.

[32] Renée J. Miller, Fatemeh Nargesian, Erkang Zhu, Christina Christodoulakis, Ken Q. Pu, and Periklis Andritsos. 2018. Making Open Data Transparent: Data Discovery on Open Data. *IEEE Data Eng. Bull.* 41, 2 (2018), 59–70.

[33] Fatemeh Nargesian, Ken Q. Pu, Bahar Ghadiri Bashardoost, Erkang Zhu, and Renée J. Miller. 2020. Data Lake Organization. arXiv:1812.07024.

[34] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *PVLDB* 11, 7 (2018), 813–825.

[35] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*. 1532–1543.

[36] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering Table Queries on the Web Using Column Keywords. *PVLDB* 5, 10 (2012), 908–919.

[37] Jeffrey Pound, Stelios Paparizos, and Panayiotis Tsaparas. 2011. Facet discovery for structured web search: a query-log mining approach. In *SIGMOD*. 169–180.

[38] Jeffrey Pound, Stelios Paparizos, and Panayiotis Tsaparas. 2011. Facet Discovery for Structured Web Search: A Query-log Mining Approach. In *SIGMOD*. 169–180.

[39] Dominique Ritze, Oliver Lehmberg, Yaser Oulabi, and Christian Bizer. 2016. Profiling the Potential of Web Tables for Augmenting Cross-domain Knowledge Bases. In *WWW*. 251–261.

[40] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic Taxonomy Induction from Heterogenous Evidence. In *International Conference on Computational Linguistics*. The Association for Computer Linguistics, 801–808.

[41] Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics* 39, 3 (2013), 665–707.

[42] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering semantics of tables on the web. *PVLDB* 4, 9 (2011), 528–538.

[43] Xapian. [n. d.]. Available online at xapian.org (Last accessed on Feb 29, 2020).

[44] Hui Yang and Jamie Callan. 2009. A Metric-based Framework for Automatic Taxonomy Induction. In *International Conference on Computational Linguistics*. The Association for Computer Linguistics, 271–279.

[45] Jianye Yang, Wenjie Zhang, Shiyu Yang, Ying Zhang, Xuemin Lin, and Long Yuan. 2018. Efficient set containment join. *VLDB J.* 27, 4 (2018), 471–495.

[46] Bweijunl Zheng, Wei Zhang, and Xiaoyu Fu Boqin Feng. 2013. A survey of faceted search. *Journal of Web engineering* 12, 1&2 (2013), 041–064.

[47] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. 2019. JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. In *SIGMOD*. 847–864.

[48] Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. 2016. LSH Ensemble: Internet-Scale Domain Search. *PVLDB* 9, 12 (2016), 1185–1196.

[49] Erkang Zhu, Ken Q. Pu, Fatemeh Nargesian, and Renée J. Miller. 2017. Interactive Navigation of Open Data Linkages. *PVLDB* 10, 12 (2017), 1837–1840.