Data-Driven Domain Discovery for Structured Datasets

Masayo Ota[1] Heiko Müller[1] Juliana Freire[1] Divesh Srivastava[2]
[1]New York University [2]AT&T Labs-Research
{masayo.ota,heiko.mueller,juliana.freire}@nyu.edu; divesh@research.att.com

ABSTRACT

The growing number of open datasets has created new opportunities to derive insights and address important societal problems. These data, however, often come with little or no metadata, in particular about the *types* of their attributes, thus greatly limiting their utility. In this paper, we address the problem of *domain discovery*: given a collection of tables, we aim to identify sets of terms that represent instances of a semantic concept or domain. Knowledge of attribute domains not only enables a richer set of queries over dataset collections, but it can also help in data integration. We propose a data-driven approach that leverages value co-occurrence information across a large number of dataset columns to derive robust context signatures and infer domains. We discuss the results of a detailed experimental evaluation, using real urban dataset collections, which show that our approach is robust and outperforms state-of-the-art methods in the presence of incomplete columns, heterogeneous or erroneous data, and scales to datasets with several million distinct terms.

PVLDB Reference Format:
Masayo Ota, Heiko Müller, Juliana Freire, Divesh Srivastava. Data-Driven Domain Discovery for Structured Datasets. *PVLDB*, 13(7): 953-965, 2020.
DOI: https://doi.org/10.14778/3384345.3384346

1. INTRODUCTION

The trend towards transparency has led to an explosion in the number of open datasets available via data portals for countries, states, and cities [23, 20, 19, 21, 22]. By integrating and analyzing these data, many new opportunities for innovation, economic growth, and societal benefit are created. The NYC OPEN DATA collection published by the City of New York (NYC) [21], for example, consists of over 1,900 datasets covering a wide range of areas, such as requests and complaints to city services, performance indicators for public schools, building permits, and parking violations. These have been used by City agencies to make their processes more efficient, by policy makers to design data-driven policies, and by community leaders and residents in efforts to improve their neighborhoods [18].

A key challenge in using open datasets is that they are often published with little or no metadata. The datasets in NYC OPEN DATA are published as CSV or JSON files which list only attribute names and values, with associated syntactic types (*e.g.,* string, numeric). In the absence of details about the semantic types (or domains) of attributes, it is difficult to understand what the data actually entails, to discover data that is suitable for a given application, or infer connections across disparate datasets.

By inspecting columns individually, one is likely to obtain incomplete information about a data collection. A given column may contain only a subset of the possible values for a domain or include values from multiple domains. Consider the discovery of tables in a data lake that can be *unioned* [17]. Techniques based solely on column contents may lead to false negatives and miss columns whose values have a small (or no) overlap. For example, the NYC Department of Finance publishes several datasets that are horizontally partitioned based on different boroughs. Because their values for the borough column are disjoint, instance-based methods would fail to identify these as unionable. On the other hand, false positives may be derived for heterogeneous columns or when the overlap consists of ambiguous terms that belong to multiple domains, *e.g.,* `BLACK` and `WHITE` are values for color and ethnicity.

Domain Discovery: Challenges and Opportunities. We explore the problem of *domain discovery*: how to automate the process of inferring semantic types present in a dataset collection. Semantic types are crucial for many important information retrieval and integration tasks, including the creation of mediated schemas for traditional data integration applications [8] and the derivation of edges in knowledge graphs that encode datasets relationships in a data lake [7, 16]. In addition, semantic types provide a summary of the collection contents.

We view domains as sets of terms where each term is an instance of the concept represented by the domain. For example, the terms {`BRONX`, `BROOKLYN`, `MANHATTAN`, `QUEENS`, `STATEN ISLAND`} form a domain in NYC OPEN DATA that represents the five boroughs of New York City.

Challenge: Handling Collection-Specific Domains. Open data domains can be classified into two main categories: (i) *generic domains*, which are common to a wide range of datasets and application areas, *e.g.,* addresses and person names, and (ii) *collection-specific domains*, whose contents

Proceedings of the VLDB Endowment, Vol. 13, No. 7
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3384345.3384346

dvv_color
BEIGE
BLACK
BLUE
BROWN
GOLD
GRAY
GREEN
KHAKI
ORANGE
PINK
RED
WHITE

dvt_color
BEIGE
BLACK
BLUE
BROWN
GRAY
GREEN
KHAKI
MAROON
OLIVE
PINK
RED
YELLOW

color
BLUE
BROWN
GOLD
GREEN
MAROON
OLIVE
ORANGE
PINK
RED
WHITE
YELLOW

category$_2$
ASIAN
BLACK
ELL
EP
FEMALE
FORMER ELL
HISPANIC
MALE
NOT SWD
SWD
WHITE

dvt_make
4WD
BLACK
BLUE
BMW
BUICK
CHEVY
GRAY
GREEN
HONDA
TOYOTA
WHITE
... (87 more)

bor
BRONX
BROOKLYN
MANHATTAN
QUEENS
STATEN IS
UNKNOWN

borough
BRONX
BROOKLYN
MANHATTAN
QUEENS
STATEN IS

city$_1$
MANHATTAN
QUEENS
STATEN IS

neighborhood$_1$
AVERNE
ASTORIA
BAYSIDE
BRIARWOOD
CORONA
ELMHURST
UNKNOWN
... (41 more)

neighborhood$_2$
ASTORIA
BAYSIDE
BELLEROSE
CORONA
ELMHURST
FAR ROCKAWAY
FLUSHING
... (42 more)

city$_2$
ASTORIA
BAY RIDGE
BRONX
BROOKLYN
BUSHWICK
CHELSEA
CORONA
MANHATTAN
N/A
STATEN IS
... (121 more)

category$_1$
ASIAN
BLACK
HISPANIC
WHITE

ethnicity
ASIAN
BLACK
HISPANIC
WHITE

demographic
ASIAN
BLACK
HISPANIC
N/A
WHITE

boroname
BRONX
BROOKLYN
MANHATTAN
QUEENS
STATEN IS

borough_name
BRONX
BROOKLYN
MANHATTAN
QUEENS

rental_nbh
AVERNE
ASTORIA
BAYSIDE
JAMAICA
... (48 more)

$\mathbf{D}_{\text{boro}}$ = {BRONX, BROOKLYN, MANHATTAN, QUEENS, STATEN IS}
$\mathbf{D}_{\text{colo}}$ = {BEIGE, BLACK, BLUE, BROWN, GOLD, GRAY, GREEN, KHAKI, MAROON, OLIVE, ORANGE, PINK, RED, WHITE, YELLOW}
$\mathbf{D}_{\text{ethn}}$ = {ASIAN, BLACK, HISPANIC, WHITE}
$\mathbf{D}_{\text{qns}}$ = {AVERNE, ASTORIA, BAYSIDE, BELLEROSE, BRIARWOOD, CORONA, ELMHURST, FAR ROCKAWAY, FLUSHING, JAMAICA, ...}

Figure 1: Example columns from NYC OPEN DATA containing terms from four different domains.

are specific to a given area (or application), *e.g.,* boroughs and school codes, which vary for different cities. Generic domains can be handled through a number of strategies, including the use of regular expressions, ontologies, and dictionaries (e.g., [3, 12, 36]). The same is not true for collection-specific domains, for which it is hard to find well-established tools or master data to assist in domain discovery. In fact, we computed the overlap between the datasets in NYC OPEN DATA and both `Yago` [28] and word vectors trained with `GloVe` (using Wikipedia 2014 and Gigaword 5, containing a total of 6 billion tokens) [24]. We found that `Yago` covers only 10% of the terms and `GloVe` covers only 8%.

Opportunity: Leveraging Large Collections. While the volume of data and number of distinct datasets present a challenge for domain discovery, they also create an opportunity: the redundancy present in a large collection of tabular data containing multiple columns belonging to the same semantic type can be used to *construct* domains. However, to do so, we need to overcome several obstacles, which we discuss below using a concrete example.

Example. Figure 1 shows 17 columns extracted from NYC OPEN DATA, which contain a total of 259 terms. Each column contains terms from at least one of four semantic types: boroughs ($\mathbf{D}_{\text{boro}}$), colors ($\mathbf{D}_{\text{colo}}$), ethnicity ($\mathbf{D}_{\text{ethn}}$), and neighborhoods in the borough of Queens ($\mathbf{D}_{\text{qns}}$).

Incomplete Columns. For a given large domain D, it is unlikely for any column to contain all terms in D. Instead, columns often contain different subsets of D. In the example, columns `dvv_color`, `dvt_color`, and `color` all contain a subset of $\mathbf{D}_{\text{colo}}$; `borough_name` and `city`$_1$ are subsets of $\mathbf{D}_{\text{boro}}$; and `city`$_2$, `neighborhood`$_1$, `neighborhood`$_2$, and `rental_nbh` contain subsets of $\mathbf{D}_{\text{qns}}$. To construct domains, we need to combine information from multiple columns.

Heterogeneous Columns. A given column may contain terms that belong to different domains. For example, `city`$_2$ contains terms from domains $\mathbf{D}_{\text{boro}}$ and $\mathbf{D}_{\text{qns}}$ as well as names of neighborhoods in Manhattan and Brooklyn. Column `category`$_2$ contains terms from $\mathbf{D}_{\text{ethn}}$ and terms that categorize students by their gender and English language proficiency. If not properly handled, heterogeneity can lead to mistakes in inferred domains. Also note that column names can be heterogeneous: columns containing information about boroughs have names *city*, *bor*, *boro_name*. Thus, we cannot rely on the name of a column to infer its domain.

Ambiguous Terms. In the NYC OPEN DATA collection, it is common for terms to belong to more than one domain. For example, calendar months `APRIL`, `MAY`, and `JUNE` are also frequent person first names, and colors like `BLACK`, `GREEN`, and `WHITE` also represent last names. In our example, `BLACK` and `WHITE` belong to two domains: $\mathbf{D}_{\text{colo}}$ and $\mathbf{D}_{\text{ethn}}$.

Approach Overview. To address these challenges, we propose `D`^{4}, a data-driven domain discovery approach for collections of related tabular (structured) datasets. Given collection of datasets, `D`^{4} outputs a set of domains discovered from the collection in a holistic fashion, by taking all the data into account. Similar to word embedding methods such as `word2vec` [15], `D`^{4} gathers contextual information for terms. But unlike these methods which aim to build context for terms in unstructured text, we aim to capture context for terms within columns in a set of tables. The intuition is that terms from the same domain frequently occur together in columns or at least with similar sets of terms.

`D`^{4} has three main components: signature generation, column expansion, and domain discovery. The first component creates signatures that capture the context of terms taking into account term co-occurrence information over all columns in the collection. These signatures are made robust to noise, heterogeneity, and ambiguity. We propose a series of *robustification* strategies which trade off precision and recall for the discovered domains, and thus can be applied to a range of tasks, depending on the specific requirements and data characteristics. *Robust signatures* capture the context of related terms while blending out noise. They are essential to our approach in addressing the challenges of incomplete columns (through column expansion) and term ambiguity in heterogeneous and noisy data.

Column expansion addresses the challenge of incomplete columns. A straightforward approach to expansion is to use co-occurrence information, and iteratively expand a set of terms T using terms that co-occur with the terms in T. However, this can introduce noise and lead to cascading errors. Consider for example the term `N/A` that occurs with terms from $\mathbf{D}_{\text{ethn}}$, $\mathbf{D}_{\text{boro}}$, and $\mathbf{D}_{\text{qns}}$ in columns `demographic` and `city`$_2$ in Figure 1. By adding all terms that co-occur with N/A to these columns, the result will contain terms belonging to many distinct domains. $\mathtt{D}^4$ avoids this problem by using robust signatures to expand columns: it adds a term only if it has sufficient support from the robust signatures of terms in the column. This leads to high accuracy in domain discovery.

The *domain discovery* component works in two stages. In the first stage, it derives from each column a set of domain candidates, called *local domains*. Local domains are clusters of terms in an (expanded) column that are likely to belong to the same type. Clustering here is defined as a graph partitioning problem. Nodes in the graph are terms and edges represent references to terms in their respective robust signatures. Performing local domain discovery using graph partitioning based on robust signatures has the benefit that we are unlikely to cluster terms that occur in the same column but belong to different semantic types (*i.e.,* in heterogeneous columns). Clustering columns independently, along with column-based robustification, helps $\mathtt{D}^4$ handle ambiguous terms. In the second stage, we apply a data-driven approach to narrow down the set of local domains and create a smaller set of *strong domains* to be presented to the user. We also propose an alternative procedure to compute strong domains from local domains with a human-in-the-loop (Section 5.3). In this setting, the user provides a set of seed values, and an enhanced, more complete domain is computed. This scenario is related to set expansion and user-guided data discovery, which we discuss in Section 2. It is important to note that for the human-in-the-loop setting, we do not necessarily expect the user to have a-priori knowledge of the domains in a given collection. Instead, we view domain enhancement as part of an exploration phase where the user inspects the set of local domains discovered by our approach. The selection of seed values could, for example, be guided by the terms in the local domains that a user inspects, or by some incomplete knowledge that the user has about the domains in the collection.

The first data-driven approach proposed to discover semantic types in collections of tables in an enterprise was the $\mathtt{C}^4$ methodology [14]. $\mathtt{C}^4$ uses co-occurrence information to build a concept hierarchy and selects from the hierarchy nodes that are likely to represent important concepts (or semantic types). As we discuss in Section 2 and evaluate in Section 5, the effectiveness of $\mathtt{C}^4$ is compromised in the presence of ambiguous terms, incomplete, and heterogeneous columns. In addition, since $\mathtt{C}^4$ requires the computation of the similarity between every pair of terms to construct clusters, its applicability to datasets like NYC OPEN DATA which consists of millions of terms is limited.

Our work is orthogonal to and can be combined with approaches that use external data sources, including knowledge bases and ontologies. At the same time, domains that $\mathtt{D}^4$ discovers can be used to build or enhance existing knowledge bases or ontologies.

Contributions. Our main contributions in this paper can be summarized as follows:

- We present an end-to-end, scalable solution for the problem of domain discovery in collections that contain incomplete and heterogeneous columns, and ambiguous terms.
- We introduce the notion of robust context signatures and present different robustification algorithms that explore the trade off between precision and recall for the derived domains (Section 3).
- We propose a column-based clustering approach to identify domain candidates that addresses the challenge of incomplete columns through column expansion, and of heterogeneous columns and ambiguous terms through the use of robust signatures (Section 4).
- We perform a detailed experimental evaluation using four large and diverse real-world data collections. and show that: our approach for domain discovery is effective and scalable, and outperforms existing domain discovery approaches for collections where columns are heterogeneous, incomplete, and include ambiguous terms; and our human-in-the-loop domain enhancement strategy performs better than previous works on set expansion for domains obtained from heterogeneous columns, and has comparable performance for domains obtained from homogeneous columns (Section 5).

2. RELATED WORK

Domain discovery for structured data is a largely unexplored topic. To the best of our knowledge, $\mathtt{C}^4$ [14] is the only prior work that uses a data-driven approach to the problem. $\mathtt{C}^4$ was designed to build concept hierarchies using collections of enterprise spreadsheets. It computes, for each pair of terms, the Jaccard similarity between sets of columns they occur in. This similarity captures a measure for the co-occurrence of term pairs and naturally induces a graph in which nodes are terms and each edge is weighted by the corresponding Jaccard similarity. Related values can be grouped by deleting edges whose weights are below a given threshold and keeping the resulting connected components. Varying this threshold leads to different sets of connected components that form a tree of clusters. A cluster is chosen as a concept (or domain) if there exists a column in the dataset that has a high Jaccard similarity with that cluster. Thus, an important assumption to reliably extract meaningful concepts is that the original dataset contains homogeneous columns which largely cover the associated domain. However, this assumption fails to hold for data in the wild. As we discuss in Section 5, for such collections, $\mathtt{C}^4$ derives concepts that are incomplete or heterogeneous, because they have high similarity with incomplete and heterogeneous columns. In addition, the presence of ambiguous terms which are connected to terms across multiple domains, lead to large, low-precision clusters.

Our human-in-the-loop approach to compute strong domains is related to set expansion techniques that expand a small set of seed terms to obtain a more *complete* set by discovering terms that belong to the same concept. Some approaches to this problem rely on search engines and customized information extraction [31, 5]. SEAL [31] models terms as nodes in a graph and applies random walk to find terms that are close in the graph – terms that are closest to the seeds are used in expansion. Lyretail [5] improves upon SEAL by training page-specific information extractors. Wang et al. [30] use the concept name to retrieve relevant web tables in addition to a set of seed entities as inputs, and proposes a holistic, probabilistic model to rank the entities and tables. Other approaches exploit context features (*e.g.,* Wikipedia lists, free-text patterns) and ontologies to compute similarity between entities within text corpora [25, 26, 34]. In contrast, $\mathtt{D}^4$ is data driven: the only context it uses is the co-occurrence of terms within columns in a dataset. Nonetheless, the use of diverse sets of external resources (*e.g.,* Web tables, Web pages, knowledge graphs) is a topic that we plan to explore in future work as a means to improve both domain discovery and expansion. More closely related to our approach is `SEISA` [11], which uses collections of terms extracted from Web lists and query logs. `SEISA` takes an iterative approach to expand a seed with relevant terms. Relevance is measured as an aggregated similarity function based on Jaccard similarity between the set of collections the terms occur in. In contrast, our approach uses the seed to identify a set of clusters that combined have high F-score with respect to the seed. We include a comparison with `SEISA` in our experiments in Section 5.3.

Our work is also related to the problem of user-guided data discovery. For example, DataXformer [2] and InfoGather [33] take a target schema as input and find a set of values for the attributes in the schema using web tables. In contrast, we consider a setting where the input is a set of seed values for a domain of interest, and derive additional values in this domain. The Aurum system [7] addresses a different data discovery problem: it uses a knowledge graph to represent relationships among datasets, and uses this graph to help users find relevant data.

$\mathtt{D}^4$ was motivated in part by approaches that aim to construct context for terms, in particular, word embeddings. These techniques have enabled the creation of meaningful context-aware representations of words [15]. They typically rely on the training of a neural network to model the co-occurrences of words in a text and to extract representations of these words in the form of the vectors of constant dimension. Remarkably, these vectors capture semantic similarity between words and can be used to answer queries such as "a king is to a man as what is to a woman?". Such methods have been successfully used for different tasks over unstructured data. For example, word embeddings are used for query expansion to improve retrieval performance [13, 35]. While word embeddings present a promising direction for knowledge discovery, they also have limitations. Similar to ontologies, pre-trained word embeddings often have limited coverage for structured data as mentioned in Section 1. An alternative is to train word embeddings directly on a structured data collection. However, existing training algorithms have been devised for textual documents and it is not immediately clear how these can be applied to tabular data. Bordawekar and Shmueli [4] trained word embeddings on tabular data based on row co-occurrences by treating each row as a sentence. These embeddings can be used to capture semantic relationships within a row but ignore column context, which is key to domain discovery. For domain discovery, we need to capture co-occurrence information among all terms in a column, not just the ones in the nearby rows. Thus, whether word embeddings can be effectively used for domain discovery is an open research question.

Domain discovery is an important challenge for open data and data lakes [16]. Domain information is a useful piece of metadata. It not only enables queries that search for related tables (*e.g.,* find all tables that mention NYC schools), but it also has the potential to increase the coverage for union-search queries [17]. While systems have been proposed to manage and derive metadata for dataset collections [1, 9, 10, 27], they have not tackled the problem of domain discovery.

3. ROBUST CONTEXT SIGNATURES

In this section, we (i) introduce notations used throughout the paper, (ii) define context signatures that form the basis for domain discovery, and (iii) present algorithms for making context signatures robust. Robust signatures serve as input to the domain discovery algorithm (Section 4).

B_1	`WHITE`	0.7500	$\Delta = 0.1785$
B_2	`ASIAN, HISPANIC`	0.5714	$\Delta = 0.1429$
B_3	`GRAY`	0.4286	
	`BLUE, BROWN, GREEN, RED`	0.3750	
	`BEIGE, KHAKI`	0.2857	
	`PINK`	0.2500	$\Delta = 0.1071$
B_4	`4WD, BMW, BUICK, CHEVEY, HONDA, TOYOTA, ...` (92 more)	0.1429	$\Delta = 0.0179$
B_5	`GOLD, MAROON, N/A, OLIVE, ORANGE, YELLOW`	0.1250	

Figure 2: Context signature for term `BLACK` using σ_{ji}. The horizontal lines denote the most significant differences in similarity for consecutive elements in the context signature.

Preliminaries and Problem Definition. A *dataset* $\mathcal{D}$ consists of a set of columns $\{C_1, \ldots, C_n\}$. Each *column* $C \in \mathcal{D}$ is a set of terms $C \subseteq T$ from a *universal domain* $T = \{t_1, \ldots, t_k\}$. A *semantic type* (or *domain*) is a set of terms that represents a well-defined concept. We assume w.l.o.g. that each term in T occurs in at least one column, *i.e.,* $T = \bigcup_{C \in \mathcal{D}} C$. The set of columns that a term occurs in is denoted as $cols(t) \subseteq \mathcal{D}$.

Definition 3.1 (Domain Discovery). *Given dataset $\mathcal{D}$,* domain discovery *aims to find a set of domains $\mathcal{S} = \{S_1, \ldots, S_m\}$ representing all semantic types in $\mathcal{D}$.*

In the following, we use normal font upper case, *e.g.,* $C, S \subseteq T$, to denote sets of terms and calligraphic font, *e.g.,* $\mathcal{C}, \mathcal{S} \subseteq \mathcal{P}(T)$, to denote sets of term sets. We represent vectors of elements as $\vec{\cdot}$, *e.g.,* $\vec{s}$. The *i-th* position of $\vec{s}$ is denoted by $\vec{s}[i]$, and the number of elements in a vector or a set is represented by $|\cdot|$.

3.1 Context Signatures

We define the context of a term t as the co-occurrence information between t and all other terms $t' \in T$. We quantify *co-occurrence* for a pair of terms t_1, t_2 using similarity functions $\sigma : T \times T \to \mathbb{R}$. We refer to the values $\sigma(t_1, t_2)$ as the (*co-occurrence*) *similarity* of t_1 and t_2. Note that our algorithms are independent of the similarity function used. In our experiments, we evaluated several measures and found that the Jaccard similarity coefficient between the sets of columns the terms occur in provides the overall best results, and we use it for all experiments reported in this paper: $\sigma_{\text{ji}}(t_1, t_2) = \frac{|cols(t_1) \cap cols(t_2)|}{|cols(t_1) \cup cols(t_2)|}$.

Definition 3.2 (Context Signature). *Given a similarity function σ, the* context signature *for t is a vector $\vec{s}(t)$ of tuples $(t', \sigma(t, t'))$ containing exactly one entry for each element $t' \in T \setminus \{t\}$ such that $\sigma(t, t') > 0$. The elements in $\vec{s}(t)$ are sorted in decreasing order of σ.*

In the following we use $\vec{s}(t)|t'$ and $\vec{s}(t)|\sigma$ to denote the vectors that only contain the list of terms and co-occurrence similarities from a context signature, respectively.

Example. Figure 2 shows the context signature $\vec{s}$(`BLACK`). In the interest of space, we show terms having equal similarity with `BLACK` in a single line (column 2). Using the Jaccard similarity values (column 3) we can see that the term `BLACK` is most similar to `WHITE`, followed by `ASIAN` and `HISPANIC`, which are exactly the three other terms in domain $\mathbf{D}_{\text{ethn}}$. As this example illustrates, the context of terms can serve as a building block to construct domains.

In an ideal dataset, a term t only occurs in columns where all terms belong to the same domain. In most real world datasets, however, due to heterogeneous columns, NULL values, and erroneous data, it is common for a pair of terms t_1, t_2 to occur in the same column even if both terms are from different domains. We refer to terms in $\vec{s}(t)$ that have no domain in common with t as *noise*.

Example. The context signature for BLACK in our example contains 115 terms. Many of these terms do not share a domain with BLACK. They are the result of BLACK (erroneously) occurring in column `dvt_make`. In NYC OPEN DATA it is common for columns that contain the make of derelict vehicles to also contain popular car colors.

An important challenge that we must thus overcome is how to make D^4 robust in the presence of noise. To do so, we propose to prune context signatures such that each signature $\vec{s}(t)$ retains only those entries that are likely to share a domain with t. We refer to the result as a *robust signature.* In what follows, we introduce different pruning strategies to derive robust signatures.

3.2 Signature Blocks

A simple approach for defining the robust signature is to select a threshold and include in the signature only terms with similarity above the threshold. Choosing a threshold, however, is difficult and a fixed threshold is unlikely to work well for all terms. A high threshold may exclude relevant terms, while a low threshold may add noise.

Example. In our experiments, we use three dataset collections that we obtained from NYC OPEN DATA. We refer to these datasets as EDUCATION, FINANCE, and SERVICES. Details about these collections are given in Section 5 and a summary of their contents is presented in Table 1. The pairwise similarities for terms in $\mathbf{D}_{\text{boro}}$ range between 0.593 and 0.500 (in dataset FINANCE). For terms in $\mathbf{D}_{\text{ethn}}$, pairwise similarities are between 0.957 and 0.880 (EDUCATION), and for terms in $\mathbf{D}_{\text{colo}}$, between 0.667 and 0.067 (SERVICES). These examples show that using a single threshold for signature pruning is likely to result in robust signatures with a significant number of missing or noisy elements.

To avoid having to define a single threshold, we propose a pruning strategy that takes advantage of the structural properties of context signatures. We observed that terms belonging to the same domain as t usually form subsequences (intervals) within $\vec{s}(t)$. Terms within each interval have similarity with t that does not vary significantly. Individual intervals, on the other hand, are divided by a difference in similarity that is larger than the differences of similarities within the interval. Using this observation, we divide the context signature for each term t into intervals based on differences in similarity between consecutive elements in $\vec{s}(t)$. We refer to the terms in these intervals as *signature blocks.*

To generate signature blocks, we start by identifying the index position k, where $1 \leq k < |\vec{s}(t)|$, for which the value $\vec{s}(t)|\sigma[k] - \vec{s}(t)|\sigma[k+1]$ is maximal. Terms in $\bigcup_{1 \leq i \leq k} \vec{s}(t)|t'[i]$ form the first block. We then repeat this procedure starting at position $k+1$ until $k \geq |\vec{s}(t)|$. We use function DROP-INDEX (pseudocode is omitted due to space restrictions) to identify signature block boundaries. DROP-INDEX takes as arguments a vector $\vec{v}$ of similarity weights (*i.e.,* $\vec{s}(t)|\sigma$) and a start index. It returns the index of the element to the right of the largest difference between subsequent elements (*i.e.,* the steepest drop) in $\vec{v}$ past the given start index.

DEFINITION 3.3 (SIGNATURE BLOCKS). *Let $k_1, \dots, k_m$ denote a list of indices for $\vec{s}(t)$ with $k_1 = 1, k_i < k_{i+1}$, and $k_m = |\vec{s}(t)| + 1$. Indices $k_i, 2 \leq i \leq m$ are the result of invoking* DROP-INDEX *with k_{i-1} as start index. We use $\vec{b}(t) = (B_1, ..., B_{m-1})$, with $B_i = \bigcup_{k_i \leq j < k_{i+1}} \vec{s}(t)|t'[j]$, to denote the vector of signature blocks for t.*

Example. Figure 2 shows the five blocks in the context signature for BLACK using σ_{ji}. The first block contains WHITE and the second block contains the remaining terms from $\mathbf{D}_{\text{ethn}}$. The next block is a subset of $\mathbf{D}_{\text{colo}}$. Block B_4 contains a mix of terms resulting from columns `category`$_2$ and `dvt_make`. Block B_4 is the largest block with 98 terms. The final block B_5 contains less frequent terms from $\mathbf{D}_{\text{colo}}$ and the NULL value N/A.

3.3 Pruning Signature Blocks

After dividing a context signature into blocks, the next step is to decide which blocks to include in the robust signature. Note that we use context signatures that are vectors of weighted terms to first generate blocks (*i.e.,* sets) of terms. The robust signature for a term t is then the union of one or more signature blocks in $\vec{b}(t)$.

We have experimented with several strategies to identify relevant signature blocks. Below, we present two strategies, referred to as CONSERVATIVE and LIBERAL, according to how they prioritize precision and recall. We assess the effectiveness of these strategies in Section 5.

Conservative. The CONSERVATIVE pruning approach returns the first block $\vec{b}(t)[1]$. The reasoning behind this strategy is that the first block contains the terms that t is most similar to. In many cases, these terms together form a single domain. Using CONSERVATIVE in D^4 will normally *result in discovered domains with high precision* since the risk of including noisy data in the robust signature is reduced. The strategy, however, comes at the cost of lowering the recall.

Example. Using the CONSERVATIVE approach, the robust signature for BLACK is $\{$WHITE$\}$, the robust signature for WHITE is $\mathbf{D}_{\text{ethn}} \setminus \{$WHITE$\}$, and the robust signatures for ASIAN and HISPANIC contain the terms HISPANIC and ASIAN, respectively. Using conservative robust signatures, our algorithm discovers a domain that is equal to $\mathbf{D}_{\text{ethn}}$. On the other hand, the robust signatures for BLACK and WHITE do not contain any other colors, and the other terms in $\mathbf{D}_{\text{colo}}$ do not contain BLACK or WHITE in their conservative robust signature. Thus, our algorithm does not discover a domain with all the terms in $\mathbf{D}_{\text{colo}}$ using conservative signatures.

Liberal. The LIBERAL pruning strategy retains all blocks up until one *noisy block* that is likely to contain terms that are the result of noise in the dataset. We use the number of terms in each block as the criteria to identify the noisy block. The reasoning is that terms that occur due to noise will normally have low similarity with t. These terms are likely to be grouped into one large block (often towards the end of $\vec{b}(t)$ but not necessarily the last block). If we include anything before the largest block in the robust signature there is a high chance that we include most of the relevant terms and eliminate the majority of the noise.

Example. B_4 is the largest block in Figure 2. The resulting liberal robust signature for BLACK is $B_1 \cup B_2 \cup B_3$. This signature is a mix of terms from $\mathbf{D}_{\text{ethn}}$ and $\mathbf{D}_{\text{colo}}$. We are more likely to discover domains $\mathbf{D}_{\text{ethn}}$ and $\mathbf{D}_{\text{colo}}$ using liberal signatures. The downside of the LIBERAL approach is that

Algorithm 1 Column-specific Robust Signatures

```
1: function PRUNE-CENTRIST($t, C$)
2:     $\vec{x} \leftarrow [(\emptyset, 0)]$
3:     for $B \in$ LIBERAL-BLOCKS($\vec{b}(t)$) do
4:         $\vec{x}.append((B, \frac{|B \cap C|}{|B|}))$
5:     end for
6:     $sort(\vec{x})$
7:     $drop \leftarrow$ DROP-INDEX($(\vec{x}|\omega), 1$)
8:     return $\bigcup_{1 \leq i < drop} \vec{x}|B[i]$
9: end function
```

its lax pruning may lead to noise in the robust signature, which can negatively impact precision. We discuss this in more detail in Section 5.

3.4 Column-Specific Robust Signatures

Robust signatures generated by the CONSERVATIVE or LIBERAL approach are *global* in that they are the same for all columns in a dataset. In the following, we present another approach, which we call CENTRIST, that generates *local* robust signatures for each column independently.

Centrist. The CENTRIST approach aims to bridge the gap between the CONSERVATIVE approach that discovers domains of high precision but with lower recall and the LIBERAL approach that has higher recall but lower precision.

Example. Block B_2 in Figure 2 contains terms from $\mathbf{D}_{\text{ethn}}$, while B_3 contains terms from $\mathbf{D}_{\text{colo}}$. If we consider column `dvt_color`, none of the terms in B_2 occurs in the column but all of the terms in block B_3 do. Thus, with respect to column `dvt_color`, B_3 appears relevant while B_2 does not.

The CENTRIST pruning strategy is detailed in Algorithm 1. Given a term t and column C, it starts by computing a relevance score for each signature block with respect to C (lines 3-5). Initially, the same block pruning strategy as the LIBERAL one (denoted by function LIBERAL-BLOCKS) is used. That is, it only considers blocks before the largest block. To eliminate blocks with insufficient relevance, CENTRIST again uses DROP-INDEX to find the steepest drop in the sorted list of weights for $\vec{x}$ (lines 6-7). Similar to context signatures, we use $\vec{x}|B$ and $\vec{x}|\omega$ to denote the projection of $\vec{x}$ onto the block elements and weights, respectively. Using DROP-INDEX has the advantage that we do not need to define a single threshold for all columns. The result is the union of blocks for elements in $\vec{x}$ that come before the drop index.

Note that we initialize $\vec{x}$ with an empty set having relevance score 0 (line 2). This is motivated by the observation that, for some terms, all signature blocks have high relevance for a given column. In these cases we want to avoid pruning any of the signature blocks.

Example. For column `ethnicity`, blocks B_1 and B_2 in the context signature for `BLACK` have relevance 1 and B_3 has relevance 0. B_4 is not part of the result of LIBERAL-BLOCKS. Using the CENTRIST approach, the robust signature for `BLACK` with respect to `ethnicity` is $B_1 \cup B_2$. For column `dvt_color`, B_3 has relevance 1 and B_1, B_2 have relevance 0. Thus, the robust signature for `BLACK` with respect to `dvt_color` is B_3.

4. DOMAIN DISCOVERY

The `D`^{4} algorithm for domain discovery consists of three steps. First, to address the challenge of incomplete columns, it expands each column with terms that are likely to belong to the same domain as the terms in the column (Section 4.1). Then, for each column it discovers *local domains* (Section 4.2) which are clusters of terms that reference each other in their robust signatures. The final step identifies *strong domains* (Section 4.3), *i.e.,* local domains that have strong support from columns in the dataset.

Algorithm 2 Column Expansion

```
1: function COLUMN-EXPAND($C, \tau_{sup}, \delta_{dec}$)
2:     $C^+ \leftarrow C$, $\tau_{col} \leftarrow \tau_{sup}$
3:     while $\tau_{col} > 0$ do
4:         $C' \leftarrow \{\}$
5:         for $t' \in \bigcup_{t \in C^+}$ ROBSIG($t, C^+$) $\setminus C^+$ do
6:             $S \leftarrow \{t | t \in C^+ \wedge t' \in$ ROBSIG($t, C^+$)$\}$
7:             if $\frac{|S|}{|C^+|} > \tau_{sup} \wedge \frac{|S \cap C|}{|C|} > \tau_{col}$ then
8:                 $C' \leftarrow C' \cup \{t'\}$
9:             end if
10:        end for
11:        if $|C'| > 0$ then
12:            $C^+ \leftarrow C^+ \cup C'$, $\tau_{col} \leftarrow \tau_{col} - \delta_{dec}$
13:        else
14:            return $C^+$
15:        end if
16:    end while
17:    return $C^+$
18: end function
```

Note that `D`^{4} is agnostic to the pruning strategy used for signature blocks. We use ROBSIG(t, C) to denote the robust signature of t with respect to C. For CONSERVATIVE and LIBERAL, the robust signature for t is the same for all columns, while for CENTRIST, signatures are column specific.

4.1 Column Expansion

Incomplete columns yield incomplete domains. Consider for example columns `borough_name` and `city`$_1$. For these columns we can only discover a local domain that is a proper subset of $\mathbf{D}_{\text{boro}}$. To overcome this problem, we propose to expand each column $C \in \mathcal{D}$ with terms $t' \notin C$ that have sufficient support from the robust signatures of terms $t \in C$. Given that each robust signature contains terms that are likely to belong to the same domain as t, if term t' appears in many robust signatures in C, it is likely part of a domain for C and therefore can be added to C for the purpose of domain discovery. Thus, column expansion reduces the number of incomplete domains discovered.

We take an iterative approach to column expansion. The idea is that adding a term to a column may provide additional support for other terms to be added as well. In round one, we identify the expansion set $C' \subseteq T \setminus C$ of terms that have sufficient support from the terms in C. Round two uses the expanded column $C^+ = C \cup C'$ to identify the expansion set $C'' \subseteq T \setminus C^+$ and so on. Algorithm 2 outlines our strategy for column expansion.

Iterative expansion carries the risk of *concept drift.*[1] If we erroneously add terms that do not belong to any domain in C, we may increase support for other terms from domains not in C. To avoid drift, for expansion, we only consider terms that are supported by the original terms in C.

The function COLUMN-EXPAND computes, in each round, the support set S for terms t' that are not in C^+ but are present in at least one robust signature for terms $t \in C^+$ (line 6). We ensure that the support for t' in C^+ is greater than τ_{sup} and that the support in C is greater than τ_{col} (line 7). We decrease τ_{col} after each iteration by a constant δ_{dec}.

[1]See `https://en.wikipedia.org/wiki/Concept_drift`.

The algorithm terminates if (i) in any round we do not add new terms to C, or (ii) if $\tau_{col} \leq 0$.

Decreasing τ_{col} instead of using a fixed threshold provides tighter bounds for the addition of new terms in earlier rounds, but allows for terms with low support in C to be added should they be supported by terms that were added over multiple rounds of expansion. Consider a term t' with low support in C. If t' has support in C^+ above τ_{sup} after the first round of expansion we cannot be certain whether this is due to terms that were added to C^+ erroneously to the column or not. However, if t' still has sufficient support in C^+ after multiple rounds of expansion, we gain more confidence that t' should be added to C^+.

Example. Using CONSERVATIVE and parameters $\tau_{sup} = 0.25$ and $\delta_{dec} = 0.05$, seven columns are expanded: `borough_name` and `city_1` contain all terms in $\mathbf{D}_{\text{boro}}$, `GRAY` is added to `color`, and a total of 21 terms are added to columns `city`$_2$, `neighborhood`$_1$, `neighborhood`$_2$, and `rental_nbh`. Using the CENTRIST approach with the same parameters expands nine columns: `color`, `dvt_color`, and `dvv_color` contain all terms in $\mathbf{D}_{\text{colo}}$, `rental_nbh` contains all terms from $\mathbf{D}_{\text{qns}}$, while the other three columns from $\mathbf{D}_{\text{qns}}$ remain incomplete.

4.2 Domain Discovery for Columns

For each column C, the domain discovery algorithm operates on the expanded column C^+. D^4 first partitions C^+ into disjoint clusters $\{L_1, L_2, \ldots\}$ which we refer to as *local domains*. For columns that are homogeneous and complete (after expansion), we expect to generate a single cluster. For columns that are heterogeneous or noisy, the result will contain multiple clusters. We model the domain discovery problem as graph partitioning. We use undirected graphs where the nodes are the terms in C^+ and there is an edge between two nodes $t_1, t_2 \in C^+$ if at least one of them contains the other in its robust signature, *i.e.,* $t_1 \in \text{ROBSIG}(t_2, C^+) \vee t_2 \in \text{ROBSIG}(t_1, C^+)$. The set of local domains is given by the set of connected components in the undirected graph.

Given a partitioning $\{L_1, L_2, \ldots\}$, we aim to discover sets of semantically related terms based on their co-occurrence. Therefore, we eliminate clusters that contain a single term. In our experiments, we observed that NULL values, generic terms, and outliers tend to form single-term clusters. Furthermore, we remove all clusters where $L_i \cap C = \emptyset$. This provides an additional quality check for column expansion: if a cluster only contains terms that are not in the original column C, it is likely that column expansion added terms erroneously. The remaining clusters form the set of local domains for C.

Example. In Figure 1, CONSERVATIVE local domain discovery yields twelve clusters. Domains $\mathbf{D}_{\text{boro}}$ and $\mathbf{D}_{\text{ethn}}$ are the only complete domains that are discovered. Other local domains are subsets of $\mathbf{D}_{\text{colo}}$ and $\mathbf{D}_{\text{qns}}$. For column `dvv_color`, we discover local domains {`BLACK`, `WHITE`} and {`BEIGE`, `BLUE`, `BROWN`, `GOLD`, `GRAY`, `GREEN`, `KHAKI`, `ORANGE`, `PINK`, `RED`} due to the discussed limitations of conservative signatures. The CENTRIST approach discovers nine different local domains including $\mathbf{D}_{\text{boro}}$ $\mathbf{D}_{\text{ethn}}$, $\mathbf{D}_{\text{colo}}$ and $\mathbf{D}_{\text{qns}}$.

4.3 Strong Domains

The total number of local domains discovered for a dataset can be large. One reason is noise. For example, in one column from our experimental datasets, we discover a local domain $\mathbf{D}_{\text{boro}}$ but also one that contains the misspelled terms `BORNX`, `BROOKKYN`, `MANHTTAN`, `QEENS`, and `STATE ISLAND`. Another reason are incomplete domains. Columns that contain only a small subset of terms from a larger domain may not be completed by column expansion. This is especially true for heterogeneous columns where the overall support for additional terms is low. As a result, we frequently discover local domains that are similar to but not exactly the same subsets of a larger domain.

In the final step of our domain discovery algorithm, we prune local domains to retain those domains for which we have high confidence that they represent (part of) a meaningful semantic type. We use frequency to quantify the confidence that we have in a discovered local domain. Simply counting the number of columns that yield the exact same domain, however, can produce misleading results due to similar but non-identical incomplete local domains. Instead, we introduce the notion of a *strong domain* to denote a local domain that has support from a sufficiently large number of other local domains. We use thresholds τ_{dsim}, τ_{est}, and τ_{strong} to define strong domains in a data-driven way. Let $ji(L_1, L_2) = \frac{|L_1 \cap L_2|}{|L_1 \cup L_2|}$ be the Jaccard similarity for a pair of local domains L_1, L_2.

DEFINITION 4.1 (DOMAIN SUPPORT). *Let $\mathcal{C}$ denote the set of local domains for column C. Given a threshold τ, we say that a local domain $L \in \mathcal{C}$ has support from column $C' \in \mathcal{D} \setminus \{C\}$ if there exists a local domain $L' \in \mathcal{C}'$ such that $ji(L, L') > \tau$. The domain support $sup(L, \tau)$ is the number of columns that support L for threshold τ.*

A local domain L is supported by all other local domains that are similar to L above τ. Note that if identical local domains L_1, L_2, L_3 are discovered in three different columns, each of them has support from at least two local domains. There may be additional support from similar local domains discovered in other columns.

We next define what is meant by having support from a sufficient number of other local domains. A simple solution would be to use a value n and require each local domain to have support from at least n other local domains. The problem with this approach is that some semantic types are more frequent in a data collection than others. For example, $\mathbf{D}_{\text{boro}}$ occurs in over 300 columns in NYC OPEN DATA, while $\mathbf{D}_{\text{colo}}$ occurs in less than 20 columns. For a local domain that represents a less frequent type, we would want to use a smaller value of n, while for local domains that represent more common type, we would want n to be large.

To avoid having to choose different values for n for all local domains, we define *sufficient support* based on an estimate of the frequency of the semantic type that a local domain represents. Given a local domain L, we use domain support and threshold τ_{est} to identify columns that contain local domains that likely represent the same semantic type as L.

DEFINITION 4.2 (TYPE FREQUENCY). *Given local domain $L \in \mathcal{C}$, we estimate the frequency for the semantic type of L in $\mathcal{D}$ as the support $sup(L, \tau_{est})$.*

The idea is that (1) if a semantic type is contained in a column, we are likely to discover a local domain for that column containing terms from the type, and (2) all local domains that represent the type are likely to have some overlap.

To have confidence that L represents a meaningful type, we require it to have support from a large fraction of columns that contain local domains likely belonging to the same semantic type. That is, we consider L to be a strong domain if it has support τ_{dsim}, with $\tau_{dsim} > \tau_{est}$, from a large fraction (τ_{strong}) of columns that have been estimated to contain a local domain of the same type as L.

DEFINITION 4.3 (STRONG DOMAIN). *A local domain L is considered strong if $\frac{sup(L,\tau_{dsim})}{sup(L,\tau_{est})} > \tau_{strong}$.*

Example. From the ten local domains discovered using the CENTRIST approach, six are considered strong for thresholds $\tau_{dsim} = 0.5$, $\tau_{strong} = 0.25$, and $\tau_{est} = 0.1$. The set of strong domains contains $\mathbf{D}_{\text{boro}}$, $\mathbf{D}_{\text{colo}}$, $\mathbf{D}_{\text{ethn}}$, and $\mathbf{D}_{\text{qns}}$. The other two strong domains are subsets of 52 terms each of $\mathbf{D}_{\text{qns}}$.

4.4 Domain Discovery at Scale

Generating context signatures requires us to compute the intersection $cols(t_1) \cap cols(t_2)$ for every pair of terms t_1 and t_2. For large datasets the number of terms can easily exceed several hundred millions. In such cases, the quadratic complexity in the number of terms for context computation becomes a bottleneck. To enable D^4 to scale to large datasets, we make use of the following observation: most datasets that we encountered in practice contain a large number of terms that always occur together in the same set of columns. We group such terms into sets and refer to them as an *equivalence class*. This grouping makes sense in our scenario since we consider domains as sets of terms that frequently co-occur in columns. If every column that contains term t_1 also contains t_2, and vice versa, we have high confidence that t_1 and t_2 belong to the same domain.

Example. In Figure 1, the terms `BLUE`, `BROWN`, `GREEN`, and `RED` all occur in the same set of columns {`color`, `dvt_color`, `dvv_color`, `dvt_make`}. For our algorithm it makes sense to consider the set of terms that always occur in the same columns as an atomic component for the domain discovery process, and by doing so, we obtain a drastic reduction in the number of term pairs that need to be considered.

DEFINITION 4.4 (EQUIVALENCE CLASS (EQ)). *Given a set of terms $E \subseteq T$, we refer to E as an* equivalence class *if (i) all terms in E occur in the same set of columns,* i.e., $\forall_{t_1,t_2 \in E} cols(t_1) = cols(t_2.)$, *and (ii) no term $t' \in T \setminus E$ exists such that $cols(t') = cols(t)$ for any $t \in E$.*

The second condition in Def. 4.4 ensures that EQs are maximal and disjoint. It follows that each term belongs to exactly one EQ. We partition T into a set of EQs, denoted by $\mathcal{E} = \{E_1, \ldots, E_m\}$. The total number of EQs in our example dataset is 27. This number is almost ten times less than the number of 257 distinct terms. In practice, the reduction is even more significant. Table 1 shows statistics for the datasets that we use in our experiments. In all cases, the number of EQs is $< 2\%$ of the number of distinct terms. The majority of EQs usually contains only a single term. In particular, terms that occur in many columns are likely to be in an EQ on their own. The reduction comes from a few EQs that can contain hundreds of thousands of terms that occur together in a small set of columns.

Implementation. Our implementation uses EQs instead of terms as the atomic components. The main benefit is that computing pairwise co-occurrence information becomes feasible even for very large datasets when the set of EQs is significantly smaller than the set of terms. We replace each column $C \in \mathcal{D}$ with the set of EQs $\{E | E \in \mathcal{E} \wedge E \subseteq C\}$. Each term $t \in C$ belongs to exactly one EQ, and if $t \in C$ and $t \in E$ then $E \subseteq C$. We extend the definition of $cols(\cdot)$ to EQs. The set of columns that an EQ E occurs in is equal to the set of columns that each term in E occurs in, *i.e.,* $\forall_{t \in E} cols(t) = cols(E)$. In Algorithms 1 and 2, when we compute the size of a set of terms we now use the size of the union of all EQs in the set. All other parts of the algorithms remain the same.

Table 1: Dataset statistics.

Category	Tables	Rows	Cols	Terms	EQs
`Education`	201	3,208,141	1,401	350,231	5,370
`Finance`	176	12,447,106	1,594	5,713,755	21,187
`Services`	692	275,254,542	3,566	36,923,781	130,165
`Utah Open`	1,766	644,663,858	11,351	194,032,923	235,366

Our implementation takes advantage of multithreading. For signature blocks generation and strong domain discovery, we use parallel threads to compute similarity between EQs and local domains, respectively. For column expansion and local domain discovery, we use multiple threads to process individual columns in parallel.

5. EXPERIMENTS

We carried out a detailed experimental evaluation to: (1) compare D^4 with other state-of-the-art methods for domain discovery and domain enhancement; (2) assess the effectiveness of D^4 in the presence of column incompleteness, heterogeneity, and ambiguous terms; and (3) evaluate different components and configurations of D^4.

Our first set of experiments evaluates the effectiveness of D^4 for the problem of domain discovery. We compare D^4 against C^4 [14]. To the best of our knowledge, C^4 is the only prior work that tackles the problem of domain discovery in structured data in a holistic fashion. Our experimental and evaluation protocol closely follows that of [14].

As discussed in Section 1, our approach to constructing strong domains can be carried out with a human-in-the loop. More precisely, the user provides a set of seed values for a domain of interest, and an enhanced, more complete set of terms for the domain is computed. Since the *human-in-the-loop setting* is similar to the problem of set expansion, in Section 5.3, we include a comparison with `SEISA` [11], which has been shown to be more effective than previously-proposed approaches for noisy data and does not rely on external data sources.

The framework we propose can be configured in different ways, *e.g.,* different signature pruning strategies and thresholds can be used. In the last set of experiments, we evaluate our design decisions and study the strengths and weaknesses of different D^4 configurations.

5.1 Experimental Setting

Data. We use two repositories–NYC OPEN DATA and State of Utah Open data (UTAH OPEN), downloaded using the Socrata Open Data API [29] on Nov. 22nd 2016 and on Sep. 27th 2019, respectively. NYC OPEN DATA consists of over 1,100 tables from NYC agencies such as Department of Education and Department of Finance. Each table is labeled using 13 different labels. Based on these labels, we obtained three datasets: (a) EDUCATION, (b) FINANCE (using labels *economy* and *finance*) and (c) SERVICES which includes all tables that are not in (a) or (b). For our experiments, we consider columns where the majority of distinct terms is text since numerical values can be assigned to a separate domain. Table 1 shows statistics for the four datasets.

Ground Truth Domains. The challenges present in open data such as column incompleteness, heterogeneity, and er-

Table 2: The Results of Domain Discovery. The optimized thresholds for $\mathtt{C}^4$ are 0.7, 0.5, 0.7 and 0.4 for the datasets of EDUCATION, FINANCE, SERVICES, and UTAH OPEN respectively.

Dataset	Domain	Domain Size	$\mathtt{C}^4$ w/ THR = 0		$\mathtt{C}^4$ w/ OPT. THR		$\mathtt{D}^4$	
			Num of Clusters	F-score	Num of Clusters	F-Score	Num of Clusters	F-Score
NYC OPEN DATA EDUCATION	School Name (SNAME)	2865	2943	0.718	409	0.718	328	**0.819**
	District Borough Number (DBN)	2673		**0.882**		0.718		0.728
	School Number (SNUM)	2845		0.637		0.637		**0.703**
	Student Grade (GRADE)	5		1.0		1.0		1.0
	Ethnicity (ETHN)	4		1.0		1.0		1.0
NYC OPEN DATA FINANCE	Bronx Neighborhoods (BX)	97	8493	0.566	627	0.566	422	**0.635**
	Brooklyn Neighborhoods (BK)	103		0.742		0.742		**0.839**
	Manhattan Neighborhoods (MH)	91		0.710		0.710		0.710
	Queens Neighborhoods (QN)	137		0.341		0.341		**0.654**
	Staten Island Neighborhoods (SI)	44		**0.603**		**0.603**		0.581
	Boroughs in NYC (BORO)	5		1.0		1.0		1.0
	Building Permit Type (PERMIT)	22		0.483		0.483		**0.722**
	Rental Building Class (RENTAL)	17		**1.0**		**1.0**		0.903
NYC OPEN DATA SERVICES	City Agency (AGENCY)	74	36453	0.797	1065	0.797	935	**0.846**
	City Agency Abbreviation (ABBR)	190		0.361		0.361		**0.591**
	Car Plate Type (PLATE)	87		0.863		0.863		**0.955**
	US States (US)	50		0.939		0.939		**0.980**
	Other States (OTHER)	17		0.034		0.034		**0.395**
	Month (MONTH)	12		0.737		0.737		**1.0**
	Color (COLOR)	20		0.417		0.0		**0.947**
UTAH OPEN	County (COUNTY)	29	92503	1.0	3559	1.0	2477	1.0
	Incident Type (INCIDENT)	208		0.604		0.604		**0.732**
	Hospital Name (HOSPITAL)	137		0.330		0.299		**0.503**
	Medication Name (MED)	282		**1.0**		**1.0**		0.932
	Fund Name (FUND)	676		0.941		0.941		**0.981**

roneous data make it difficult to obtain ground truth domains (GTDs). This poses an obstacle for evaluation and comparison of different domain discovery methods. To tackle this challenge, we adopted the following procedure from [14] to approximate ground truth domains in our data:

- Select domains of interest. For each domain, a set of seed terms is created. We obtain the seed terms from Web sources such as the Wikipedia and the web site for NYC OPEN DATA. For instance, we scraped a list of neighborhoods in Manhattan from Wikipedia [32].
- Expand the seed set with terms from the data. The seed terms are unlikely to contain all the terms in this domain that are present in the data (*e.g.,* the seed may include `FLATIRON DISTRICT` but not `FLATIRON`, both of which occur in NYC OPEN DATA columns containing Manhattan neighborhoods). Thus, to expand the seed set, we obtain a set of columns that have a large overlap with the seed. For each column we *manually* inspect the terms outside of the intersection and decide whether they should be added to the ground truth domain (GTD) or not.

For our experiments we selected 25 representative GTDs of different sizes from the four datasets. These GTDs are chosen so that corresponding columns are sufficiently diverse. In particular, we aimed to include homogeneous, heterogeneous, complete and incomplete columns, as well as, columns containing ambiguous terms (see Table 2). Note that the signatures of each term is derived using *all* the columns in each of the four table collections.

Baselines and Configurations. We use CENTRIST signature pruning as the default setting. For column expansion, we use parameters $\tau_{sup} = 0.25$ and $\delta_{dec} = 0.05$. To obtain strong domains, we set thresholds $\tau_{dsim} = 0.5$, $\tau_{est} = 0.1$ and $\tau_{strong} = 0.25$ so that $\mathtt{D}^4$ produces a reasonable number of clusters to present to the user and to compare with $\mathtt{C}^4$.

Recall that $\mathtt{C}^4$ requires a Jaccard similarity threshold to select concepts from the candidate clusters. Selecting this threshold is challenging: while a high threshold leads to higher quality, it may lead to low recall; in contrast, a low threshold results in a large number of clusters. To benefit $\mathtt{C}^4$ with respect to recall, we set this threshold to zero, and consider the super-set of clusters it produces when we select the best cluster for each domain. Note that, even under this idealized scenario, $\mathtt{C}^4$ outperforms $\mathtt{D}^4$ only for 4 domains (see Table 2). In practice, however, using a threshold of 0 is difficult since the number of resulting clusters would be too large for a user to inspect, *e.g.,* between 2,943 and 92,503 for the datasets we consider.

We also identify the best threshold for $\mathtt{C}^4$ across several different values of height constraint [14]. While in practice, in the absence of gold data, it is not possible to select such thresholds, we followed this procedure to ensure that we obtained the best possible configuration for $\mathtt{C}^4$. In particular, for height restrictions 1, 3, 5, and 7, we consider similarity thresholds $0.1, 0.2, \ldots, 0.9$, and select the best one in terms of the average F-score. This procedure is performed for each dataset independently. Note that we stop at height restriction 7 since there is no change in the resulting clusters after 5. To obtain the best results, we search over all clusters using the thresholds we find (without the height constraint). The optimal thresholds are 0.7, 0.5, 0.7, and 0.4 for EDUCATION, FINANCE, SERVICES, and UTAH OPEN, respectively.

We run both $\mathtt{D}^4$ and $\mathtt{C}^4$ using equivalence classes (EQs): without EQs, the computation does not complete in a reasonable time. Note that using EQs does not affect results of $\mathtt{D}^4$ and $\mathtt{C}^4$ since we used Jaccard as the similarity measure.

5.2 Domain Discovery

Comparing $\mathtt{D}^4$ and $\mathtt{C}^4$. We compared $\mathtt{D}^4$ and $\mathtt{C}^4$ for domain discovery, and evaluated the resulting clusters using the setup of [14]: for each GTD and each cluster that overlaps with the GTD, we compute precision, recall and F-score; then, for each GTD, we select the cluster with the highest F-score. The results are summarized in Table 2. $\mathtt{D}^4$ compares favorably with $\mathtt{C}^4$, attaining higher or equal F-scores for 22 out of the 25 GTDs. In some GTDs, such as Queens Neighborhoods, Color, Month, Building Permit Type, City Agency Abbreviation and Hospital Name, F-scores of $\mathtt{D}^4$ are significantly higher.

Column incompleteness and heterogeneity are the main reasons of the poor performance of $\texttt{C}^4$ on Queens Neighborhoods. In general, all neighborhood `GTDs` have the problem of incomplete columns, *i.e.,* there is no column that contains all `GTD` terms. There is, however, a set of common neighborhoods for each borough that are contained in several columns. This is why $\texttt{D}^4$ and $\texttt{C}^4$ discover clusters with high precision but lower recall for neighborhood `GTDs`. $\texttt{D}^4$ is able to achieve higher recall for four of the five boroughs due to column expansion. On the other hand, there are several heterogeneous columns that contain Queens neighborhoods, boroughs and other cities. This results in some of the terms being similar not only to other Queens neighborhoods but also to cities. For instance, `BEECHHURST` has its highest similarity to any other Queens neighborhood at 0.57. Due to heterogeneity, `BEECHHURST` is also similar to several other cities such as `NASHVILLE` and `UNION CITY` at similarity of 0.57 or higher. Thus, candidate clusters of $\texttt{C}^4$ at threshold ≤ 0.55 will include Queens neighborhoods and cities leading to higher recall but lower precision and F-score.

Other examples where $\texttt{D}^4$ outperforms $\texttt{C}^4$ are the `GTDs` Color and Month. Here, the difference can be explained by the presence of ambiguous terms. Many Color terms (*e.g.,* `GREEN`, `BROWN`) also appear in Last Name columns. Terms `BLACK` and `WHITE` also occur frequently in Ethnicity columns (which appear in all the three datasets). The same is true for Month terms such as `MAY` and `AUGUST` that also appear in First Name and Last Name columns. The Jaccard similarity between ambiguous and other terms in their domains is usually lower due to the large number of different columns they occur in for their different types. $\texttt{C}^4$ only discovers small subsets of domains with ambiguous terms for higher thresholds but combines terms from different domains for lower thresholds. For example, any cluster in $\texttt{C}^4$ that includes `BLACK`, `WHITE`, and `GREEN` also includes `ASIAN` and `HISPANIC` as well as last names like `LOPEZ` and `SMITH`. On the other hand, $\texttt{D}^4$ handles this case correctly, *e.g.,* `BLACK` and `WHITE` are included not only in the cluster for Color but also in the clusters for Ethnicity and Last Name.[2] Column expansion is a major contributor to the superior performance of $\texttt{D}^4$ for other domains such as Building Permit type, City Agency Abbreviation, and Hospital Name that have similar problems of column incompleteness as discussed above.

On the other hand, $\texttt{D}^4$ attains a lower F-score than $\texttt{C}^4$ for Rental Building Class, Staten Island Neighborhoods, and Medication Name. There are 101 columns for the `GTD` of Rental Building Class: 80% of these columns contain the majority of terms, while the remaining columns contain only small portion (*i.e.,* 4 out of the 17 terms). The two sets of columns overlap only with one term (`D4-ELEVATOR`) in exactly two columns. $\texttt{D}^4$ only discovers a cluster for the larger subset of terms in the `GTD`. On the other hand, clusters produced by $\texttt{C}^4$ cover the whole `GTD`. This is due to the fact that $\texttt{C}^4$ creates an edge to connect the two subsets at Jaccard similarity of 0.0 (the very last threshold that is used to obtain connected components). Typically, clusters that are connected by such an edge are very large and contain a lot of noise. However, because terms in this `GTD` do not co-occur with terms outside of the `GTD`, the resulting cluster turns out to be accurate. Similarly for Staten Island Neighborhoods and Medication Name, some terms of these `GTDs` only occur in one column. This leads to the same issues as with Rental Building Class. Since $\texttt{D}^4$ is data-driven and based on co-occurrences of terms, this is inevitable and, therefore, a limitation of our approach.

[2]Recall that generic domains such as Last Name are not our focus in this work. Therefore, the results for these domains are omitted.

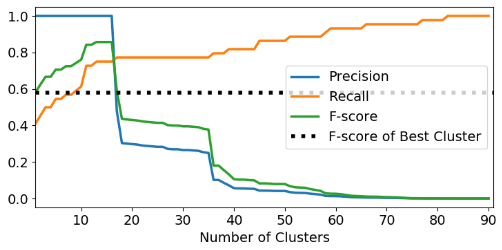

Figure 3: Precision and recall of clusters and F-score as a function of the number of clusters combined for the `GTD` of Staten Island Neighborhoods. The F-score of $\texttt{D}^4$ from Section 5.2 is represented by a dotted line.

5.3 User-Aided Domain Enhancement

In this section, we consider a setting in which the user provides a set of seed terms for a particular domain that she wishes to explore. For example, a user may know a few abbreviations for NYC agencies such as `FDNY`, `NYCHA`, and `NYPD` and is interested in the full set of agency abbreviations. For a given set of seed terms, the goal is to identify a set of local domains that together represent a domain of interest with respect to the seed.

Our approach in this setting is motivated by the observation that for many `GTDs`, the set of local domains contains several clusters with high precision but lower recall. By combining these high-precision clusters, we are able to generate domains with higher recall and F-score for the `GTD`. Consider the example shown in Figure 3. Here we compute precision and recall for each local domain in FINANCE for the `GTD` of Staten Island Neighborhoods. We then sort the local domains by decreasing precision and take the union of the first i entries in the sorted list. The plot shows precision, recall, and F-score of the resulting union as a function of the number of combined clusters. For the first 16 clusters precision stays at 1.0 while recall and F-score ascend rapidly. For $i > 16$, precision and F-score decline sharply while recall only increases gradually. These results suggest that by combining local domains we can achieve a much higher F-score than the best single cluster (shown as a dotted line).

In practice, the `GTD` is unknown and we consider user-provided seeds as a surrogate instead. The seed terms may be provided by a user based on her partial knowledge of a domain, be derived by $\texttt{D}^4$, or be obtained from an external source. Given a seed set, we proceed as before: (1) compute precision for each local domain using the seed as ground truth, (2) sort local domains by decreasing precision, (3) combine the first i entries in the sorted list. The result is the union of local domains that achieves the highest F-score with respect to the seed.

In our experiment, we use random samples of varying size from a `GTD` as seed terms. For statistical significance, sampling is repeated 100 times. The results are presented in Figure 4(a), which shows average F-scores with respect to the full `GTD` as a function of sample size. The results show that we are able to achieve F-scores which are higher than the ones from Section 5.2 using seeds of only 10%–30% of

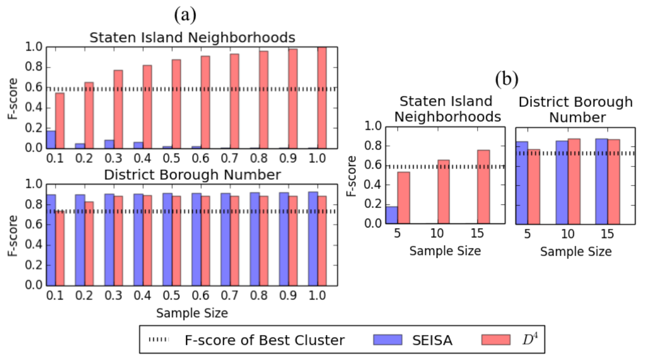

Figure 4: Average F-scores of SEISA and D^4 for various sample sizes: (a) 10%, 20%, ... of GTDs; (b) Sets with 5, 10 and 15 seeds. The best F-scores of D^4 from Section 5.2 are shown as a dotted line.

ground truth terms. Due to the space limitations we only present the results for two GTDs. Results for most of the other relatively large GTDs (*e.g.,* > 20 terms) are similar. Note that our approach in this setting is particularly useful for larger domains, since for small domains it can be easy to find the terms through manual inspection.

Comparing D^4 and SEISA. The idea of using seeds to compute a more complete set of terms corresponding to a particular semantic type has been explored in prior work on set expansion [31, 5, 30, 25, 26, 11]. While set expansion is not the main focus of D^4, we include a comparison with the dynamic thresholding algorithm from SEISA [11] in Figure 4(a). The results suggest that, by using robust signatures, expansion can lead to more accurate results in the presence of heterogeneity, missing values, and ambiguous terms. In general, SEISA is more sensitive to heterogeneity. For GTDs like Staten Island Neighborhoods which appears in heterogeneous columns, the majority of the expanded sets derived by SEISA are very large resulting in low precision (F-score between 0 and 0.2). For GTDs like District Borough Number which appears mainly in homogeneous columns, SEISA produces results that are stable regardless of seed sizes. Overall, D^4 outperforms or achieves comparable results to SEISA.

Small sized seeds. Our experiments show that using only 10-30% of ground-truth terms is sufficient to obtain a seed that leads to an optimal performance of D^4. This is a realistic assumption for smaller domains (*e.g.,* Staten Island Neighborhoods) or domains for which the seed can be obtained from an external source. However, when the user only provides a small seed (*e.g.,* 5–15 terms), intermediate results of D^4 can be used to derive a larger sample from the input seed. Using D^4, this can be done as follows. We remove local domains that are subsets of the seed and sort the remaining domains by F-score with respect to the seed in descending order. Next, we create a collection $\mathcal{L}$ of domains by traversing the sorted list. The first local domain is always included in $\mathcal{L}$. Each subsequent domain is included if and only if it covers terms in the seed that are not covered by the domains in $\mathcal{L}$. The expanded seed is the union of all the local domains in $\mathcal{L}$. We use random samples of 5, 10, and 15 terms from a GTD as a seed, and repeat sampling 10 times. The results in Figure 4(b) show that D^4 attains F-scores that are comparable to the ones obtained with medium-sized seeds, and higher than the ones from Section 5.2, using seeds with only 5-10 ground-truth terms.

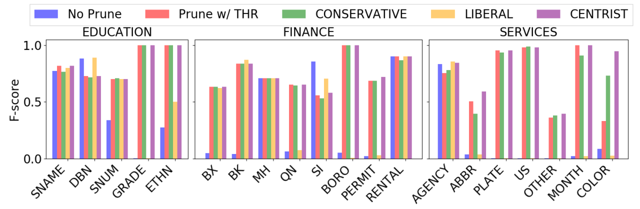

Figure 5: F-scores of D^4 with no signature pruning, pruning with fixed threshold (=0.5), CONSERVATIVE, LIBERAL, and CENTRIST.

5.4 Evaluation of Different D^4 Components

We evaluated the different components and configurations of D^4 to assess their contribution to the overall approach. We use the default settings (see Section 5.1) and alter one parameter while fixing everything else to study the effects of the variations on the F-scores in domain discovery.

5.4.1 *Signature Pruning*

We propose three strategies for signature pruning (Sections 3.3 and 3.4) which are based on signature blocks generated from context signatures using steepest drops in similarity. We compare CONSERVATIVE, CENTRIST, and LIBERAL with two alternatives: (1) no signature pruning, *i.e.,* we use the whole context signature as the robust signature, and (2) pruning elements of the context signature using a fixed threshold (> 0.5). Figure 5 shows the F-scores for the GTDs from the three datasets. CONSERVATIVE and CENTRIST perform well among the five methods. CONSERVATIVE aims for high precision and results in a lower recall, while CENTRIST attains high precision while maintaining high recall.

For *no signature pruning* and LIBERAL, the ranking of the F-scores fluctuates considerably. Even for small GTDs, like Ethnicity and Borough, both methods perform poorly due to increased noise in the signatures resulting in low precision and F-score. F-scores for *signature pruning with a fixed threshold* are comparable to the F-scores obtained with CENTRIST. For GTD Color, however, using a fixed threshold results in low recall and thus low F-score. Terms with low similarity due to ambiguity lead the threshold method to ignore relevant terms whose similarity score is below the threshold. Therefore, we use CENTRIST pruning as default as it avoids signatures that are too restrictive or noisy.

5.4.2 *Column Expansion*

To study the impact of column expansion, for a given GTD, we identify columns that either contain more than 50% of terms in the GTD or for which more than 50% of terms belong to the GTD. We *sparsify* each column by removing x fraction of terms that belong to the GTD, e.g., if a column contains the all 5 boroughs and $x = 0.4$, we remove two randomly-chosen boroughs from the column to make the column incomplete. We perform domain discovery on the resulting dataset.

We compare the following column expansion options: (1) no expansion, (2) single expansion with $\tau_{sup} = 0.25$, and (3) iterative expansion with $\tau_{sup} = 0.25$ and $\delta_{dec} = 0.05$. Note that we use $\tau_{sup} = 0.25$ since it leads to better results than a higher threshold (*e.g.,* $\tau_{sup} = 0.5$) in our experiments. We experimented with different values for x (0, 0.2, 0.3, and 0.4) and for each x we repeat the random experiment 5 times. The resulting average F-scores are shown in Figure 6. We observe that as we increase the sparsity, F-scores without expansion drop significantly. On the other hand, F-scores

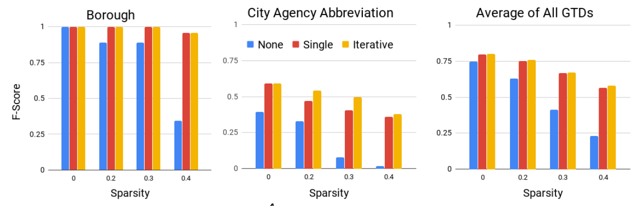

Figure 6: F-scores of D^4 with no expansion, single expansion and iterative expansion on the sparsified datasets for the GTDs Borough and City Agency Abbreviation and Average F-scores of all the GTDs.

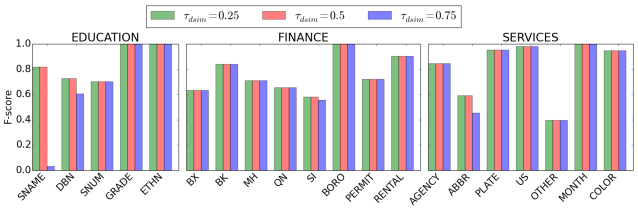

Figure 7: F-scores of D^4 varying domain support threshold.

for both iterative and single expansion are not affected very much by the sparsified datasets.

5.4.3 *Varying Parameters for Strong Domains*

To obtain strong domains from local domains, we use three thresholds, τ_{dsim}, τ_{est}, and τ_{strong} (Section 4.3). We use $\tau_{est} = 0.1$. Our experiments show that varying this parameter does not affect results significantly. However, setting it too low may lead to overestimation of type frequency.

Domain Support Threshold. We first study the effect of local domain support threshold τ_{dsim} at 0.25, 0.5 (default setting) and 0.75. The results are presented in Figure 7. F-scores with $\tau_{dsim} = 0.75$ are lower than those for the other thresholds in four out of 20 GTDs. Only School Names show a significant difference in F-score. Lower F-scores for higher thresholds are not unexpected since a set of strong domains with a higher threshold is a subset of a set of strong domains with a lower threshold. However, there is little difference between 0.25 and 0.5, so we can choose the value that produces a reasonable number of meaningful clusters.

Strong Domain Threshold. We next study the effects of varying the strong domain threshold τ_{strong} for values 0.1, 0.25 (default setting) and 0.5 (detailed results omitted due to space limitations). Similar to domain support threshold, when using $\tau_{strong} = 0.5$, the F-score is lower for three GTDs, most notably for Color with a drop of about 0.8 and School Names (about 0.2 lower). This threshold of 0.5 may be too restrictive and we can pick a threshold between 0.1 and 0.25.

These results suggest that our approach is robust to these thresholds and that it is easy to select effective values.

5.4.4 *Performance and Scalablility*

We use the full NYC OPEN DATA containing over 42 million terms and 212,766 EQs to evaluate runtime performance of different D^4 components. Starting with a random sample of 10% of EQs, we randomly increase the sample size in steps of 10% until we reach the full dataset. Experiments were run on HPC cluster nodes with Intel$^{\mathrm{TM}}$IvyBridge (3.00GHz) CPUs having 20 Cores. For our experiments we request 10 or 20 cores and 48GB memory. All numbers shown are averaged over five runs.

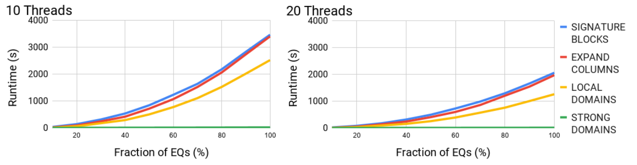

Figure 8: Runtime performance of D^4 for increasing dataset sizes using 10 and 20 parallel threads.

Figure 8 shows running times for different D^4 components. The running times for signature blocks generation increase with the data size in non-linear fashion due to the quadratic time complexity of computing pairwise similarity between EQs. However, by doubling the number of parallel threads from 10 to 20, we can decrease running times by $40 - 45\%$. For column expansion, the number of iterations and the pruning method have a large impact on running times. For CONSERVATIVE, running times are within seconds in all settings we tested. For CENTRIST and LIBERAL, running time increases with data size and the number of iterations. The CENTRIST pruning method has the additional need to compute similarity between each signature block and the columns for all EQs in a column. Running time for local domain discovery is primarily dependent on the size of the (expanded) columns. Both the column expansion step and the local domain discovery step behave similar to signature blocks generation for increasing data size and different numbers of threads. Running time for strong domain discovery is negligible compared to the overall running time for D^4.

6. CONCLUSIONS

We proposed D^4, a data-driven approach to domain discovery. Our framework and algorithms address several challenges that are common in open data collections. We (1) introduced the notion of robust context signatures that lie at the core of D^4 and contribute to its effectiveness at discovering domains in the presence of heterogeneous columns and erroneous data; (2) proposed column-specific signatures as a means to handle ambiguous terms; (3) showed that, by expanding columns we can address the problem of incomplete columns; and (4) by using equivalence classes, D^4 can handle datasets that contain millions of distinct values. Our experimental evaluation which used real, open datasets, showed that D^4 outperforms the state-of-the-art methods for domain discovery and domain enhancement (Section 5.2 and 5.3). We also analyzed how the different components and design decisions affect the precision and recall for the discovered domains, and argued that CENTRIST signature pruning with iterative column expansion provides the best performance and at the same time is robust. In the absence of an accepted benchmark, our experimental evaluation using large and diverse collections provides compelling evidence of the effectiveness of the approach we propose. Showing theoretical properties is an interesting research problem that can be explored in future work. Our code and data are available [6]. We hope that the data can serve as a starting point for a community-based effort to build a benchmark for the problem of domain discovery.

Acknowledgements. This work was partially supported by the DARPA D3M program, NSF award OAC-1640864, and the NYU Moore Sloan Data Science Environment.

7. REFERENCES

[1] Z. Abedjan, L. Golab, and F. Naumann. Profiling Relational Data: A Survey. *VLDB Journal*, 24(4):557–581, 2015.

[2] Z. Abedjan, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, and M. Stonebraker. DataXFormer: A Robust Transformation Discovery System. In *ICDE*, pages 1134–1145, 2016.

[3] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao. Inference of Regular Expressions for Text Extraction from Examples. *TKDE*, 28(5):1217–1230, 2016.

[4] R. Bordawekar and O. Shmueli. Using Word Embedding to Enable Semantic Queries in Relational Databases. In *DEEM*, pages 5:1–5:4, 2017.

[5] Z. Chen, M. Cafarella, and H. V. Jagadish. Long-tail Vocabulary Dictionary Extraction from the Web. In *WSDM*, pages 625–634, 2016.

[6] Data-Driven Domain Discovery (D4). `https://github.com/VIDA-NYU/domain-discovery-d4`.

[7] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. Aurum: A Data Discovery System. In *ICDE*, pages 1001–1012, 2018.

[8] B. Golshan, A. Y. Halevy, G. A. Mihaila, and W. Tan. Data Integration: After the Teenage Years. In *PODS*, pages 101–106, 2017.

[9] R. Hai, S. Geisler, and C. Quix. Constance: An Intelligent Data Lake System. In *SIGMOD*, pages 2097–2100, 2016.

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

[11] Y. He and D. Xin. SEISA: Set Expansion by Iterative Similarity Aggregation. In *WWW*, pages 427–436, 2011.

[12] A. Ilyas, J. M. F. da Trindade, R. Castro Fernandez, and S. Madden. Extracting Syntactical Patterns from Databases. In *ICDE*, pages 41–52, 2018.

[13] S. Kuzi, A. Shtok, and O. Kurland. Query Expansion Using Word Embeddings. In *CIKM*, pages 1929–1932. ACM, 2016.

[14] K. Li, Y. He, and K. Ganjam. Discovering Enterprise Concepts Using Spreadsheet Tables. In *KDD*, pages 1873–1882. ACM, 2017.

[15] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781, 2013.

[16] F. Nargesian, E. Zhu, R. Miller, K. Pu, and P. Arocena. Data Lake Management: Challenges and Opportunities. *PVLDB*, 12(12):1986–1989, 2019.

[17] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller. Table Union Search on Open Data. *VLDB Journal*, 11(7):813–825, 2018.

[18] Open Data for All – 2017 Progress Report. `https://moda-nyc.github.io/2017-Open-Data-Report`, 2017.

[19] Australian Government Open Data. `https://data.gov.au/`.

[20] EU Open Data Portal. `https://data.europa.eu`.

[21] New York City Open Data. `https://opendata.cityofnewyork.us`.

[22] Prefeitura de São Paulo: Dados Abertos. `http://dados.prefeitura.sp.gov.br`.

[23] U.S. Government's Open Data. `https://www.data.gov`.

[24] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *EMNLP*, pages 1532–1543, 2014.

[25] X. Rong, Z. Chen, Q. Mei, and E. Adar. EgoSet: Exploiting Word Ego-Networks and User-Generated Ontology for Multifaceted Set Expansion. In *WSDM*, pages 645–654, 2016.

[26] J. Shen, Z. Wu, D. Lei, J. Shang, X. Ren, and J. Han. SetExpan: Corpus-Based Set Expansion via Context Feature Selection and Rank Ensemble. In *ECML PKDD*, pages 288–304. Springer, 2017.

[27] T. J. Skluzacek, R. Kumar, R. Chard, G. Harrison, P. Beckman, K. Chard, and I. Foster. Skluma: An Extensible Metadata Extraction Pipeline for Disorganized Data. In *eScience*, pages 256–266, 2018.

[28] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *WWW*, pages 697–706, 2007.

[29] Tyler Technologies. *Socrata Open Data API*. `http://api.us.socrata.com/api/catalog/v1`.

[30] C. Wang, K. Chakrabarti, Y. He, K. Ganjam, Z. Chen, and P. A. Bernstein. Concept Expansion Using Web Tables. In *WWW*, pages 1198–1208, 2015.

[31] R. C. Wang and W. W. Cohen. Iterative Set Expansion of Named Entities Using the Web. In *ICDM*, pages 1091–1096, 2008.

[32] Wikipedia. *List of Manhattan neighborhoods*. `https://en.wikipedia.org/wiki/List_of_Manhattan_neighborhoods`.

[33] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: Entity Augmentation and Attribute Discovery by Holistic Matching With Web Tables. In *SIGMOD*, pages 97–108, 2012.

[34] P. Yu, Z. Huang, R. Rahimi, and J. Allan. Corpus-Based Set Expansion with Lexical Features and Distributed Representations. In *SIGIR*, pages 1153–1156, 2019.

[35] H. Zamani and W. B. Croft. Estimating Embedding Vectors for Queries. In *ICTIR*, pages 123–132, 2016.

[36] Y. Zhang, A. Ogletree, J. Greenberg, and C. Rowell. Controlled Vocabularies for Scientific Data: Users and Desired Functionalities. In *ASIS&T*, pages 1–8, 2015.