Spring
2022

ILLINOIS INSTITUTE OF TECHNOLOGY

Literature Review Report

TOPIC: DATASET DISCOVERY IN DATA LAKES

SUBMITTED BY:
GROUP - 2
AKSHAY, NIKHIL, KRUTHI

SUBMITTED TO:
BORIS GALVIC

Abstract

Data analytics can benefit from datasets that are collected without their conceptual relationships being explicitly known. Here, a solution was proposed to the problem of dataset discovery in data lakes that enables us to pull out datasets that might contribute to wrangling out a given target. In this approach they define similarity distances between features and to take those distances as measurements of relatedness with respect to the target table. And later comparing the approach with prior work, where pertinent, and showing significant improvements in all of precision, recall, target coverage, indexing and discovery times.

SECTION 1: INTRODUCTION

The number of external and internal datasets for organizations continues to grow, but it is difficult to discover the most useful ones for a given analytic task.

There is no consensus on the definition of a data lake. In this paper, we view open government data repositories as exemplar data lakes where we have schema for the datasets and some example records.

We go through several stages for analysis on data from a data lake. One of the stages in it is dataset discovery which we will be working on. Given a target, we try to find datasets from the data lake that are unionable and joinable with the target .It should be such that it should be populating as many target attributes as possible through this method.

In order to do so, we propose a solution which would be contributing to a novel approach for dataset discovery in data lakes. We extract five types of similarity features from the datasets and target:

- schema-level information : this feature will work on attribute - name similarity
- extent overlap : this feature would try to fetch common value from the columns and find similarity
- word embedding similarity : this feature would work when attributes are similar semantically but have different domains.
- format representation similarity : this feature would work with regular expressions to find similarity.
- domain distribution similarity : this feature would work for numeric value columns and find similarity based on domains. For example, age and height columns. They are both numbers but would have different domains.

SECTION 2: RELATED WORK

We take two already proposed methods as a base to build and measure performance of our method:

- Aurum
- TUS(Table Union Search)

Based on several similarity metrics, we build upon locality-sensitive hashing (LSH) and propose two hash functions with high probability of collision for inputs with high Jaccard similarity and high cosine similarity. For a small answer size, the LSH Forest extension is used and the LSH Ensemble indexing scheme is used.

SECTION 3: RELATEDNESS DISCOVERY

Define relatedness as the degree to which a dataset is relevant for populating a target dataset.

Given two datasets, if they are related to the same target attribute, the datasets are unionable. If they are also joinable, then the projection from their join result of the attributes related to some target attribute is potentially related to the target itself.

A: ATTRIBUTE RELATEDNESS - RELATEDNESS EVIDENCE

Identify related attributes in a data lake, quantify their relatedness, and find out if they store values for the same property type of the same real world entity.

Use five types of evidence for deciding on attribute relatedness, based on q-grams, tokens, formats, word-embeddings, and domain distributions. We also use finer-grained evidence for attribute names and values, which reduces the impact of dirty data.

- Transform an attribute name into a set of q-grams, and aim to measure the similarity between attribute names as a Jaccard distance between their qsets.
- Given an attribute value, we transform it into a set of tokens (tsets), which we can then use to represent its format (i.e., the regular, predictable structure of, e.g., email addresses, URIs, dates, etc.) by a set of regular expressions (rsets).
- Given an attribute value with textual content, we capture its context-aware semantics by using a word embedding model and use the cosine distance between words in that attribute value.
- We construe relatedness with a numeric attribute by comparing the Kolmogorov-Smirnov statistic.

B: ATTRIBUTE RELATEDNESS - DISTANCE COMPUTATION

Different types of evidence of relatedness can be used to compute distances between two attributes.

In order to avoid pairwise comparisons in the computation of distances between sets, we use MinHash / random projections to approximate the distances between sets based on their Jaccard/cosine similarity.

We build four LSH indexes to compute N -, V -, F -, and E - relatedness between attributes. We call these indexes IN, IV, IF, and IE, resp. And then Index Insertion Algo is Implemented.

C: ATTRIBUTE RELATEDNESS - THE NUMERIC CASE

Numeric attributes are a special case in our framework. We use the Kolmogorov-Smirnov (KS) statistic to decide whether two corresponding extents of two attributes are drawn from the same distribution, even though formatting is less indicative of conceptual equivalence for numbers.

We use evidence from the N and F indexes to decide whether we need to calculated D-related or not. And only compute KS, our measure of D-relatedness, when there is enough evidence that the indexes we already have are related.

D: DECIDING ON TABLE RELATEDNESS

Describe how to return, given a target table and exemplar tuples, the list of its k-most related datasets. We then group the results based on the dataset the attributes belong to.

In order to compute the relatedness distance between two data sets, we aggregate the distances between their related attributes using a weighted average.

To compute the weight of a given distance measure for a target attribute, we first retrieve the distance measure for every attribute of datasets in the lake that is related to the target attribute.

In order to compensate for a high number of weakly related attributes, each observed distance is treated as the probability that it is the smallest in a distribution of all observed distances between the target attribute and all other related attributes in the lake.

To derive a scalar value from a 5-dimensional vector, consider each dimensional dimension in a 5-dimensional space, and compute a combined distance of a point from another point.

Using a logistic regression classifier, we created a training set of related and unrelated pairs of the form (T,S) and used the coefficients to define the relative importance of each type of relatedness measure.

The intuition is that the classifier coefficients will minimize the distance between highly related datasets and maximize it between unrelated datasets.

EXTENDING RELATEDNESS THROUGH JOIN PATHS

The techniques described so far describe unionability factor but now here they discuss how to discover join opportunities between the top k related tables and non - top k tables.

Join datasets from the data lake to the target dataset to identify datasets that contribute to populating a target dataset.

Focus on joins based on inclusion dependencies and use a method to determine whether two datasets are joinable if two of the tests overlap in subjects and if at least one of those is a subject attribute.

We consider S and S' to be SA–joinable if

- there is IV-based evidence that the tsets T (a) and T (a'), where a and a' are attributes of S and S', resp., overlap, and
- at least one of a or a' is a subject attribute. Thus, we rely on IV to identify inclusion dependencies and, instead of the notion of candidate key, we use subject attributes

This is represented as graph traversal problem, and, in order to identify SA–join paths among the elements of S , we define an SA-join graph, GS = (S , I) over the entire data lake, where S is the node set and I the edge set defined using the two SA–joinability conditions from above: each edge from I connects two SA–joinable nodes Si and Sj .

Using Algorithm 3, we find the set of join paths from each vertex in an SA - join graph to all other vertices that are not in the set of most related datasets. Algorithm 3 returns a set of SA join paths for each dataset and a set of potential target attributes, which can be improved by considering the join opportunities.

SECTION 5: EVALUATION

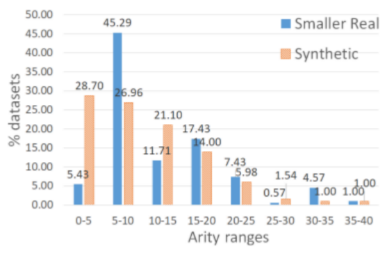
We compare the effectiveness of relatedness evidence in the D3L technique with TUS, Aurum, and synthetically derived tables and with real world information from the UK National Health Service. We also compare the impact of adding datasets that are joinable with tables in the top-k.

We use following repositories:

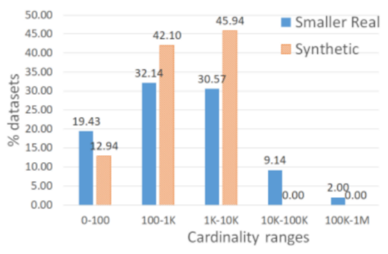
- synthetically derived tables: tables containing Canadian open government data using random projections and selections on the base tables.
- Smaller Real: real world information from the UK open government data, with information on domains such as business, health, transportation, public service, etc.
- Large Real: real world information from the UK National Health Service.

The arity, the cardinality and the percentage of numerical attributes influence the top-k ranking, target coverage, similarity estimation, and join path discovery of the two repositories.

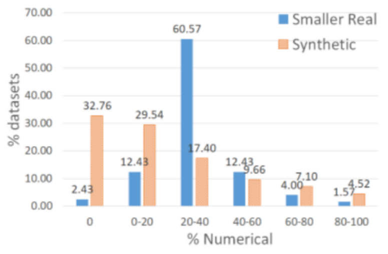
Figure below shows, analysis of the arity: number of attributes per dataset, cardinality : number of distinct values in an attribute and data types: numerical columns in Synthetic and Smaller real repositories.



(a) Arity



(b) Cardinality



(c) Data types

A: BASELINES AND REPORTED MEASURES

TUS is a unionability measuring framework which uses three types of evidence from instance values to measure unionability between datasets from different viewpoints.

In order to make it possible to discover the top-k datasets related to a given target, a two - step process profiles and indexes the data, creating a graph structure that can be used for key - word search, unionability, or joinability discovery.

To compare the result produced by the approaches we first find manually all the possible results to populate the target dataset using the selected repositories. And use them as ground truth.

For computing precision and recall, we define a true positive(TP), false positive(FP), and false negative(FN) for each table from repository R related to the ground truth.

- precision p = TP/(TP +FP)
- recall r = TP/(TP + FN)

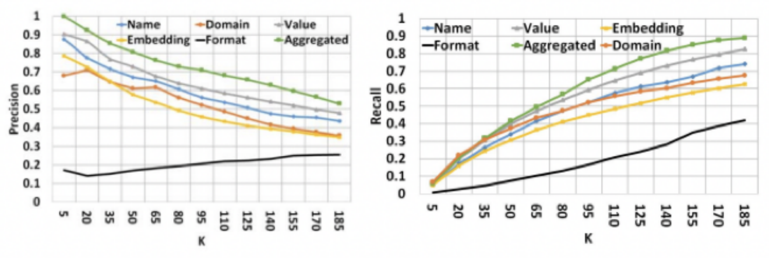
We considered the coverage of a target and computed the average of 100 randomly selected targets from the respective repository.

B: INDIVIDUAL EFFECTIVENESS

We first evaluate the effectiveness of evidence types using the Smaller Real repository. The results suggest that evidence types have a poor precision and recall in comparison with the combined approach.

There are many different attributes that represent different entities, and only embeddings and values yield high precision and recall.

Figure below, shows a graph for precision and recall for individual and aggregated relatedness evidence measure, with increasing size of the dataset. And we can observe that aggregating five different measures of relatedness results in a nearly constant increase in both precision and recall at k = 110, compared to the best individual evidence type value.



(a) Precision (b) Recall

Fig. 3: Individual Precision and Recall on *Smaller Real*

Performing the same experiment for non-numerical attributes results in an average decrease in precision and recall of less than 3.5% each.

C: COMPARATIVE EFFECTIVENESS

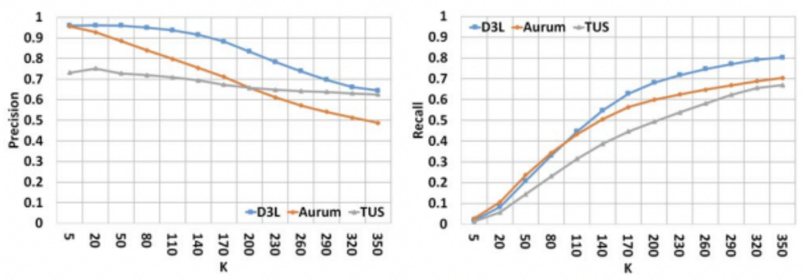
Using TUS, the most closely related datasets rank at the top, while the most unrelated datasets rank at the bottom.

D3L performs better than the baselines because it is able to balance high scores in one dimension with a score in another dimension to reduce the number of false positives.

D3L uses multi - evidence relatedness discovery to guard against too many misses, while TUS and Aurum are less dependable than content - based evidence.

D3L provides a better performance when identifying highly related datasets than TUS and Aurum, because the value - based similarity evidence used by TUS and Aurum expect equality between the instance values of similar attributes.

Figure below, shows a graph for precision and recall for relatedness evidence measure for Aurum, TUS and D3L. And we can observe that D3L can identify related datasets at an average answer size of 110 points, while TUS and Aurum can only identify around 55%.



(a) Precision (b) Recall

Fig. 4: Precision and Recall on *Synthetic*

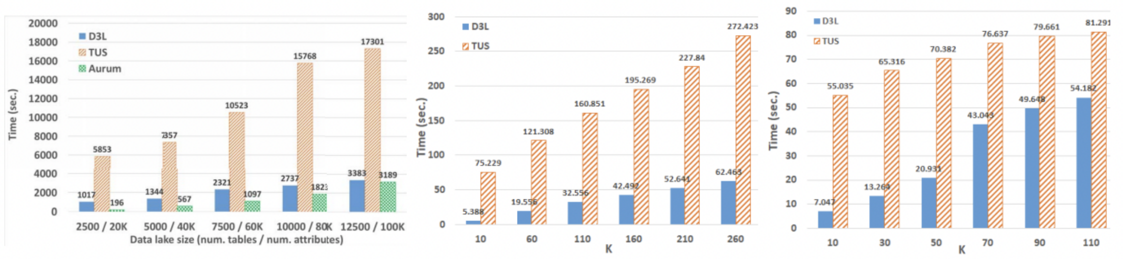
D: COMPARATIVE EFFICIENCY

We report performance data for D3L, TUS, and Aurum for dataset discovery when the size of the data lake increases, with the time to create and compute the top-k solution being reported for each system.

Comparing D3L and TUS, we have found that D3L performs better on small and medium sized lakes, while Aurum performs better on larger ones. The dominant task in all three systems is data pre-processing, while in Aurum it is graph structure creation.

The D3L and TUS query model is the most affected by the answer size parameter, but Aurum is not.

For Synthetic, D3L performs much better than TUS. The latter's reliance on YAGO proves to be a performance leakage point, and the former's distance-based approach means search returns plug directly into relatedness measurements.



(a) Indexing time

(b) *Synthetic* search time

(c) *Smaller real* search time

Fig. 6: Indexing and searching performance

Figure above shows that D3L outperforms TUS for queries where k > 50. TUS loses performance when k>50, but Aurum shows a performance boost for queries with k > 50.

TABLE II: Space overhead for different repositories.

	Synthetic	*Smaller Real*	*Larger Real (sample)*
D^3L	69%	33%	58%
TUS	56%	19%	32%
$Aurum$	55%	20%	29%

In the above table, we compared the space overhead of TUS indexes for synthetic, smaller real and larger real repositories, using Aurum 's graph data structure, profile store and LSH indexes.

D3L occupies less space than TUS and Aurum for both synthetic and real repositories. This is because D3L uses more relatedness evidence and has fewer indexes.

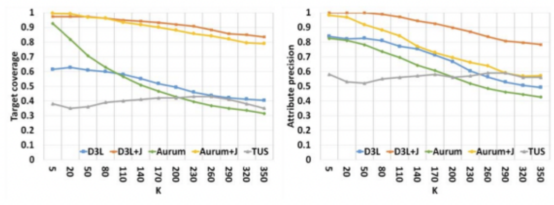
E: IMPACT ON JOIN OPPORTUNITIES

We take into account join paths to identify as many target attributes as possible in order to achieve as high a coverage as possible.

Let us define a measure for coverage on a dataset based on the relationships between the datasets. We define coverage as the ratio of related attributes in the datasets to the attributes in the target dataset. In comparing coverage for a target, we average the coverage of all join paths and use this average to calculate the coverage of the target for all join paths.

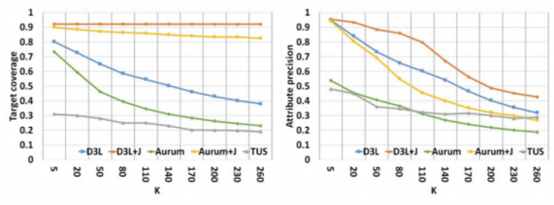
We can compute attribute precision for a dataset Si by finding alignments between attributes and target attributes, and reporting the average attribute precision for all Si in S k for various values of k.

We have augmented the top-k solution by adding datasets (from D3L + J and Aurum + J), but our hypothesis is that join paths allow us to identify relevant datasets that are not part of the initial ranked solution but can improve the target coverage.



(a) Target coverage (b) Attribute precision

Fig. 7: Coverage and precision on *Synthetic*



(a) Target coverage (b) Attribute precision

Fig. 8: Coverage and precision on *Smaller Real*

Figure above shows target coverage and attribute precision with increasing size of dataset for Synthetic and Small Real Dataset. We observe that,

- using join paths from the top-k datasets of D3L + J and Aurum + J leads to a higher coverage of the target, especially for small datasets. This is due to the use of finer grained features, as well as the multi-evidence similarity signals considered by D3L + J and Aurum + J.
- the top-k solution only covers approx. 25% of target attributes.
- D3L is better than TUS and Aurum at covering target attributes for the entire interval of k values, because it retrieves higher quality datasets from the lake.
- the datasets retrieved by TUS and Aurum are only relevant for populating the target attributes of at most 50% and 70%.

The precision of D3L is increased by the ability to identify attribute relatedness even when the format representation of values differs. It is preserved by joining tables from the join paths.

SECTION 6: CONCLUSIONS

We have used schema and instance-based features to construct hash-based indexes and can take hash values similarity as relatedness measurements.

We can use schema and instance - level features to identify related entities, which can be used to create a uniform distance space, and to discover join paths using LSH evidence. Also, the following points helped in increasing the effectiveness and efficiency of the proposed approach:

- the use of schema and instance–level fine–grained features that are more effective in identifying relatedness, especially when similar entities are inconsistently represented.
- the mapping of these features to a uniform distance space that offers an aggregated view on the notion of relatedness, to which each type of similarity evidence contributes
- the discovery of join paths using LSH evidence and subject–attributes that leads to an increased and precise target coverage

Thus, the proposed approach is more effective and more efficient in dataset discovery, when compared to TUS. And more effective in dataset discovery, but at the expense of efficiency, when compared to Aurum.