DS-Prox: Dataset Proximity Mining for Governing the Data Lake

Ayman Alserafi[1,2], Toon Calders[2,3], Alberto Abelló[1], Oscar Romero[1]

[1] Universitat Politècnica de Catalunya - BarcelonaTech, Barcelona, Catalunya, Spain
{alserafi,aabello,oromero}@essi.upc.edu
[2] Université Libre de Bruxelles (ULB), Brussels, Belgium
{aalseraf,toon.calders}@ulb.ac.be
[3] Universiteit Antwerpen (UAntwerp), Antwerp, Belgium
toon.calders@uantwerp.be

Abstract. With the arrival of Data Lakes (DL) there is an increasing need for efficient dataset classification to support data analysis and information retrieval. Our goal is to use meta-features describing datasets to detect whether they are similar. We utilise a novel proximity mining approach to assess the similarity of datasets. The proximity scores are used as an efficient first step, where pairs of datasets with high proximity are selected for further time-consuming schema matching and deduplication. The proposed approach helps in early-pruning unnecessary computations, thus improving the efficiency of similar-schema search. We evaluate our approach in experiments using the OpenML online DL, which shows significant efficiency gains above 25% compared to matching without early-pruning, and recall rates reaching higher than 90% under certain scenarios.

1 Introduction

Data Lakes (DL) [1] are huge data repositories covering a wide range of heterogeneous topics and business domains. Such repositories need to be effectively governed to gain value from them; they require the application of data governance techniques for extracting information and knowledge to support data analysis and to prevent them from becoming an unusable *data swamp* [1]. This involves the organised and automated extraction of metadata describing the structure of information stored [15], which is the main focus of this paper.

The main challenge for data governance posed by DLs is related to information retrieval: identify related datasets to be analysed together as well as duplicated information to avoid repeating analysis efforts. To handle this challenge it was previously proposed in [2] to utilise schema matching techniques which can identify similarities between attributes of different datasets. Most techniques proposed by the research community [4] are designed for 1-to-1 schema matching applications that do not scale up to large-scale applications like DLs prone to gather thousands of datasets.

To facilitate such holistic schema matching and to deal with the sheer size of the DL, [4] proposed to utilise the strategy of early pruning which limits the number of comparisons of pairs of datasets. We apply this approach in this paper by proposing a technique which approximates the proximities of pairs of datasets using similarity-comparisons of their meta-features. More specifically, we use a supervised machine learning approach to model topic-wise related classification of datasets. We then utilise this model in assigning proximities between new datasets and those already in the DL, and then predicting whether those pairs should be compared using schema matching (i.e., have related information) or not. We implement this technique in the datasets-proximity (DS-Prox) approach presented in this paper. Our focus is on early-pruning of unnecessary dataset comparisons prior to applying state-of-the-art schema matching and deduplication (the interested reader is referred to [4, 13] for more details on such techniques).

Our contributions include the following: 1. a novel proximity mining approach for calculating the similarity of datasets (Section 4), 2. applying our new technique to the problem of early-pruning in holistic schema matching and deduplication within different scenarios for maintaining the DL (Sections 2, 3), and finally, 3. testing the proposed proximity mining approach on a real-world DL to demonstrate its effectiveness and efficiency in early-pruning (Section 5).

2 Problem Statement

Our goal is to automate information profiling, defined in [2], which aims at efficiently finding relationships between datasets in large heterogeneous repositories of *flat semi-structured data* (i.e., tabular data like CSV, web tables, spreadsheets, etc.). Those repositories usually include datasets uploaded multiple times with the same data but with different transformed attributes. Such datasets are structured as groups of *instances* describing real-world entities, where each instance is expressed as a set of *attributes* describing the properties of the entity. We formally define a dataset D as a set of instances $D = \{I_1, I_2, ...I_n\}$. The dataset has a schema of attributes $S = \{A_1, A_2, ...A_m\}$, where each attribute A_i has a fixed type, and every instance has a value of the right type for each attribute. We focus on two types of attributes: continuous numeric attributes and categorical nominal attributes, and two types of relationships for pairs of datasets $[D_1, D_2]$:

- $Rel(D_1, D_2)$: Related pairs of datasets describe similar real-world objects or concepts from the same domain of interest. These datasets store similar information in (some of) their attributes. Typically, the information contained in such attributes partially overlap. An example would be a pair of datasets describing different human diseases, like one for diabetes patients and another for hypertension patients. The datasets will have similar attributes (partially) overlapping their information like the patient's age, gender, and some common lab tests like blood samples.
- $Dup(D_1, D_2)$: Duplicate pairs of datasets describe the same concepts. They convey the same information in *most* of their attributes, but such information

can be stored using differences in data. For example, two attributes can describe the weight of an object but one is normalised between 0 and 1 and the other holds the raw data in kilograms. Both attributes are identified to be representing similar information although their data are not identical.

Examples. We scrutinise the relationship between two pairs of datasets in Fig. 1. Each dataset has a set of attributes. An arrow links similar attributes between two datasets. For example, attributes 'A1' from D_2 and D_3 are nominal attributes with two unique values, making them similar. A numeric attribute like 'A2' in D_2 holds similar data as attributes 'A3' and 'A4' from D_3, as expressed by the intersecting numeric ranges. In our approach we extract meta-features from the datasets (for this example, the number of distinct values and means respectively) to assess the similarity between attributes of a given pair of datasets. The *Rel* and *Dup* properties are then used to express datasets similarities. For example, $Dup(D_1, D_2)$ returns '1' because they have similar information in most attributes (even though 'A5' and 'A3' do not match). Based on these two properties, our proposed approach will indicate whether two datasets are possibly related (e.g., $Rel(D_2, D_3)$ ='1') and should be considered for further scrutinising by schema matching, or if they are possibly duplicated (e.g., $Dup(D_1, D_2)$ ='1') and should be considered for deduplication efforts.

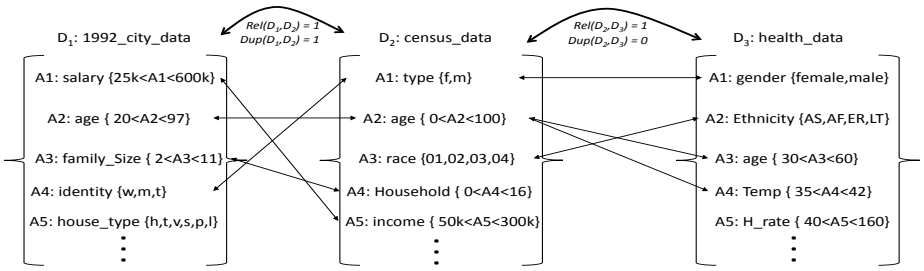

Fig. 1: Similarity relationships between two pairs of datasets

Scenarios. We aim at governing the DL by maintaining the *Rel* and *Dup* relationships between the datasets it contains. We consider two typical scenarios. In ***scenario (a)***, we want to dredge a data swamp which we don't know any relationships for, thus, for all pairs in the DL we need to find if they are related or duplicated. In ***scenario (b)***, we have an existing DL for which we know all relationships between the datasets. However, given the dynamic nature of DLs new datasets are frequently ingested. Thus, we need to compare this dataset against the datasets already in the DL to find its relationships with them.

3 Related Work

As described in [15], metadata describing the information stored in datasets need to be collected to effectively govern big data repositories. Such metadata

are usually automatically collected across multiple datasets using data profiling techniques like schema matching [11], which seeks to identify schematic overlaps between datasets. This involves detecting related objects (instances or attributes) and matching instances between two different schemata [4]. The main line of research in this field is focused around improving the efficiency of matching techniques for two very large schemata. In our research, however, we focus on matching attributes between *multiple large amounts of schemata*, closely related to the field of holistic schema matching [4, 13]. A more restrictive case of schema matching involves *deduplication* [13]; finding highly overlapping instances [6]. Similar to our special requirements for schema matching, we also seek to detect *duplicated schemata* instead of instances. This is when schemata have similar overlapping attributes, not necessarily the same instances.

As described in [4], it is recommended to utilise early-pruning mechanisms for holistic schema matching, which filters out unnecessary matching efforts using less complex techniques. This is commonly done using similarity search techniques which seek to eliminate unnecessary comparisons of datasets [12]. Several techniques for *instance-based matching* were proposed including techniques like clustering [3, 6, 8], hashing [12], and indexing [10, 12]. Alternatively, we propose to focus on *attribute-based* matching across multiple-schemata for governing the DL which needs new and efficient techniques. This field was not sufficiently studied before, with only preliminary results in [14]. We propose a new approach utilising a novel technique of computationally cheaper meta-features proximity comparisons. We seek to prevent unnecessary and expensive schema matching computations in further steps. We propose a machine learning approach for early-pruning that is based on metadata collected from datasets. Such learning techniques were proposed for future research in similarity search [5] where they use a supervised machine learning model based on SVM to find similar strings for deduplication. [5] shows that using machine learning leads to more accurate similarity search from different domains of knowledge.

4 The DS-Prox Approach

We propose a proximity computation based on overall meta-features extracted from the datasets, which we call DS-Prox. We are seeking to have approximate similarity comparisons of pairs of datasets for the early-pruning task. Here we apply cheap computation steps for the overall similarity search, to prevent further expensive detailed analysis of the content of datasets which are estimated to be dissimilar. Similar to our previous work in [2], we seek to profile the datasets ingested in the DL by extracting some *meta-features* describing the overall content and attributes in the datasets. We compute distances between each of the meta-features as proximity metrics. We take a sample of pairs of datasets which are analysed by a data analyst and annotated whether they hold *related* or *duplicate* data by means of the *Rel* and *Dup* properties. *Rel* and *Dup* are boolean functions retrieving either 1 (similar/duplicate respectively) or 0 (dissimilar/not duplicate). We then use machine learning techniques over the proximity metrics

to create two independent models which can classify pairs of datasets according to $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$ respectively. The classification models are used to *score* pairs of datasets with a similarity measure $Sim(D_1, D_2)$. The similarity score 'Sim' is defined independently for each of the relationships $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$ as a number between 0 and 1, where 0 means dissimilar and 1 means most similar: $Sim(D_1, D_2) \in [0, 1]$.

4.1 The Meta-Features Distance Measures

Table 1: DS-Prox meta-features

Type	Meta-feature	Description
General	Number of Instances	The number of instances in the dataset
	Number of Attributes	The number of attributes in the dataset
	Dimensionality	The ratio of number of attributes to number of instances
Attributes by Type	Number per Type	The number of attributes per type (Nominal or Numerical)
	Percentage per Type	The percentage of attributes per type (Nominal or Numerical)
Nominal Attributes	Average Number of Values	The average number of distinct values per nominal attribute
	Standard Deviation of Number of Values	The standard deviation in the number of distinct values per nominal attribute
	Minimum/Maximum Number of Values	The minimum and maximum number of distinct values per nominal attribute
Numeric Attributes	Average Numeric Mean	The average of the means of all numeric attributes
	Standard Deviation of the Numeric Mean	The standard deviation of the means of the numeric attributes
	Minimum/Maximum Numeric Mean	The minimum and maximum mean of numeric attributes
Missing Values	Missing Attribute Count	The number of attributes with missing values
	Missing Attribute Percentage	The percentage of attributes with missing values
	Minimum/Maximum Number of Missing Values	The minimum and maximum number of instances with missing values per attribute
	Minimum/Maximum Missing Values Percentage	The minimum and maximum percentage of instances with missing values per attribute
	Mean Number of Missing Values	The mean number of missing values from each attribute
	Mean Percentage of Missing Values	The mean percentage of missing values from each attribute

For each dataset in the DL, we extract meta-features using data profiling techniques. This includes general statistics about the dataset and its attributes as described in Table 1. Our purpose for those meta-features is to describe the general structure and content of the datasets for an approximate comparison using our proximity metric and classification models. We compute distances for each meta-feature m_i from Table 1 between each pair of datasets $[D_1, D_2]$ using equation 1 which gives the relative difference as a number between 0 and 1. Those distances we feed to the supervised machine learning algorithm in our approach.

$$dist_{m_i}(D_1, D_2) = \frac{\max\{m_i(D_1), m_i(D_2)\} - \min\{m_i(D_1), m_i(D_2)\}}{\max\{m_i(D_1), m_i(D_2)\}} \quad (1)$$

4.2 The Approach

The approach proposed for early-pruning depends on classical machine learning which is divided into two phases: *Supervised Learning* Phase and *Scoring and Classification* Phase. In the first phase, which can be seen in Fig. 2, we build a classification model for each of the properties *Rel* and *Dup* using supervised

learning techniques. First, for each dataset we extract its meta-features from Table 1 which returns its data profile (In Fig. 2 we see a sample of two meta-features: number of attributes 'nAttr', and number of instances 'nIns'). Then, for each dataset, we generate all pairs with each of the other datasets and compute the distances between their meta-features using Equation 1. We also present the pairs of datasets to a human-annotator who manually decides whether they satisfy (assign '1') or not satisfy (assign '0') $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$. Any pair annotated as a match for $Dup(D_1, D_2)$ must also be annotated as a match for $Rel(D_1, D_2)$ (i.e., all duplicate pairs of datasets are also related). We feed both the annotated pairs of datasets with their distances as training examples to a learner which creates two classifiers: M_{rel} and M_{dup}.

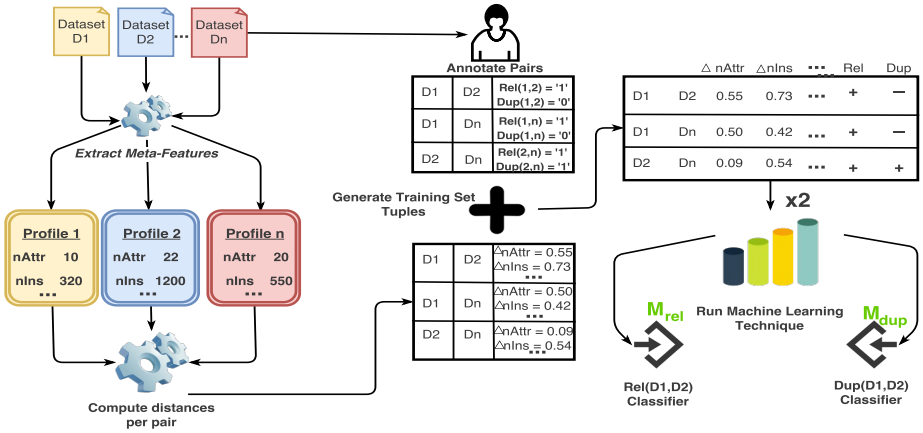

Fig. 2: DS-Prox: supervised machine learning

In the second phase, we apply the classifiers to the scenarios discussed in Section 2, to score each new pair of previously unseen datasets. In ***scenario (a)***, we have a setting where there are two DLs. DL_1 has a group of datasets which have previously known annotations of all their $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$ relationships between all pairs of datasets. On the other hand, DL_2 is without any annotations of such relationships and is therefore a data swamp we would like to dredge. Therefore, we need to learn the models for $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$ from DL_1 and apply them to DL_2 which has different datasets. In ***scenario (b)***, we have an existing DL for which we know all relationships between the datasets. We need to deal with a new dataset as it arrives in this DL. We learn the models from the DL, and we apply them to each new dataset D_i ingested within the same DL. The models should identify all datasets in the DL which are related or duplicate of D_i.

When applying the classifiers, we compute for each pair of datasets the similarity score of $Sim_{rel}(D_1, D_2)$ and $Sim_{dup}(D_1, D_2)$ using the classifiers extracted in the previous phase. The Sim score is the positive-class distribution value generated by each classifier. The predicted distribution-value achieved for the 'true'

class from each classifier is checked against a minimum threshold to indicate whether the pair of datasets are overall related or duplicates. In our approach, pairs of datasets are evaluated first if they match the $Dup(D_1, D_2)$ relationship (indicating that it also matches $Rel(D_1, D_2)$). If it fails this duplicate test, then we evaluate if the pair still satisfies $Rel(D_1, D_2)$. The output classifiers can classify in the future any new pairs of datasets as either related or duplicate according to two matching approaches: *1-to-1 matching* or *cluster matching*.

1-to-1 matching: all pairs satisfying $Rel(D_1, D_2)$ and $Dup(D_1, D_2)$ need to be selected for further schema matching and deduplication. The calculations are performed under the assumption that each and every pair of matching datasets should be correctly identified using our models.

Cluster-based matching: It is common to use clustering based approaches for the matching process [3, 4, 8]. Groups of datasets with close proximity are segmented into clusters. In our case, the relationship *Rel* can be used to cluster the datasets in the DL, after all relationships are discovered. We therefore relax our requirements for the second phase so that a new dataset should match with *any single* dataset in the same cluster in order to consider it a positive match. Therefore, if a dataset matches one or more dataset(s) from a cluster, we consider all pairs of datasets in this cluster as positively matching pairs (even if the classifier did not indicate a positive match for some of those pairs separately). The rationale behind this approach is that in a real holistic schema matching setting, a new dataset ingested should be compared to *all* the datasets in a cluster it matches to. Clustering can take place after schema matching identifies the relationships between datasets (which is outside the scope of this paper, but the reader can refer to [3, 8] for such clustering in instance-based matching).

We illustrate our general approach with a toy example in Fig. 3. Suppose we have two meta-features $nIns$ and $nAttr$ for each dataset. To classify a pair $[(nIns_1, nAttr_1)\ , (nIns_2, nAttr_2)]$ we compute the relative differences. In Fig. 3(a) we have plotted $(\Delta nIns, \Delta nAttr)$ for all pairs in the training data. '+' indicates a matching pair, '-' a non-matching pair. Based on this data we learn a classifier, for instance a separating hyperplane as shown in Fig. 3(a) by the red line. Here, for simplification, we show pairs of datasets plotted based on the distances of only two meta-features ($nIns$ and $nAttr$). The actual approach would consider all meta-features in Table 1.

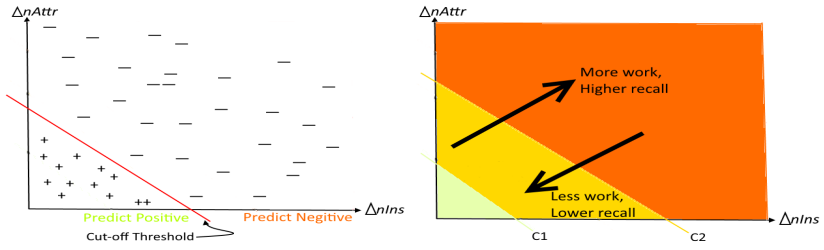

Fig. 3: DS-Prox cut-off thresholds tuning

Most classification models produce a score instead of a binary output. In the example of the separating hyperplane the obtained distance to the hyperplane can be used as a score. This score can be compared against different cut-off thresholds to decide on the final classification '+' or '-'. The threshold can be chosen to lead to different results, as seen in Fig. 3(b). If we choose the cut-off threshold 'C1' we restrict the classifier to return less pairs of high proximity (i.e., low distance), leading to lower recall but less work. Alternatively, if we alter the cut-off threshold to 'C2', we relax the classifier to return pairs of lower proximity. This leads to more pairs (i.e., more work) returned by the classifier as positive matches and higher recall of positive cases, but, with more pairs marked incorrectly as matching. Therefore, the cut-off threshold can be tweaked by the data scientist according to practical requirements in order to increase recall at the expense of more work or vice versa. This is the trade-off which we seek to optimise in our experiments when selecting different thresholds. We can use different thresholds 'c_{rel}' and 'c_{dup}' for each of the classifiers evaluated. This means that we consider a positive match if the classifier scores a new pair of datasets with a score greater than the threshold as in Equations (2) and (3).

$$Rel(D_1, D_2) = \begin{cases} 1, & Sim_{rel}(D_1, D_2) > c_{rel} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

$$Dup(D_1, D_2) = \begin{cases} 1, & Sim_{dup}(D1, D_2) > c_{dup} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The complexity of our approach is quadratic in the number of datasets, however, it applies the cheapest computational steps for early-pruning (just computing distances in Equation 1 and the classifier scoring model on each pair). This way, we save unnecessary expensive schema matching processing in later steps.

5 Experimental Evaluation

We tested an implementation of the DS-Prox approach on OpenML[1], which can be considered an online DL. It consists of different datasets covering heterogeneous topics, each having a name and a description.

5.1 Datasets

The main challenge is to create the ground-truth which we use to evaluate our approach. To achieve this, we created an experimental environment where we extracted the following independent sets of datasets from OpenML:

– **Restricted-topics sample**: First, we extract some datasets by topic using 11 keywords-search over OpenML, e.g., "Disease", "Cars","Flights", "Sports", etc. This restricted sample consists of 130 datasets and we consider them to be similar if they belong to the same topic.

[1] http://www.openml.org

– **All-topics sample**: This is an independent set of other datasets collected from OpenML. To collect this sample, we scraped the OpenML repository to extract all datasets not included in the restricted-topics sample and having a description of more than 500 characters. Out of the 514 datasets retrieved we selected 213 with descriptive descriptions (i.e., excluding datasets whose descriptions do not allow to interpret its content and to assign a topic).

Therefore, we created two new groups of datasets from OpenML for our experiments, each having its own independent set of datasets without any overlap. Having two independent sets strengthens our results and allows us to generalise our conclusions. A domain expert and one of the authors collaborated to manually label the pairs of datasets with the same topic as duplicated and / or related. The interested reader can download the two annotated datasets from GitHub[2]. The details of each sample is summarised in Table 2, which lists the number of datasets, the number of topics, top topics by the number of datasets, and the number of related and duplicated pairs per sample.

Table 2: A description of the OpenML samples collected

Sample	Datasets	Topics	Top Topics	Rel($\mathbf{D_1}$, $\mathbf{D_2}$)	Dup($\mathbf{D_1}$, $\mathbf{D_2}$)
Restricted-topics	130	29	Diseases (45), Health (31), Cars (13), Academic Courses (6), Sports (5)	1205	72
All-topics	213	79	computer software defects (17), citizens census data (12), digit handwriting recognition (12), Diseases (11)	570	128

Table 3: An example of pairs of datasets from the all-topics sample from OpenML

No.	DID 1	Dataset 1	DID 2	Dataset 2	Topic	Relationship
1	23	cmc	179	adult	Census Data	related
2	14	mfeat-fourier	1038	gina_agnostic	Digit Handwriting Recognition	related
3	55	hepatitis	171	primary-tumor	Disease	related
4	189	kin8nm	308	puma32H	Robot Motion Sensing	duplicate
5	1514	micro-mass	1515	micro-mass	Mass Spectrometry Data	duplicate

Some of the pairs from the all-topics sample can be seen in Table 3. Dataset with ID 23 should match all datasets falling under the topic of 'census data' like dataset 179. Both datasets have data about citizens from a population census. In rows 4 and 5 we can see examples of duplicated datasets, which have highly intersecting data in their attributes. Duplicate pairs in row 4 have the same number of instances, but described with different number of attributes, which are overlapping. The duplicate pairs in row 5 have identical number of attributes, yet, the attributes are transformed using pre-processing techniques and there are different number of instances between both datasets, so in essence the second dataset is a transformed and cleaned version of the first. We aim to detect such kind of scenarios using our DS-Prox approach.

5.2 Experimental Setup

In order to evaluate our approach, we create an experimental setup where we have two sets of datasets for each experiment: 1. Training set and 2. Test set. The

[2] https://github.com/AymanUPC/datasets_proximity_openml

training set is used in the supervised learning phase to create the classification models. The classification models are then evaluated using the test set. We use the restricted-topics sample as a training set, and we use both the restricted-topics and the all-topics samples in the scoring phase as test sets to evaluate our approach. We describe how we used those samples to create the training and test sets within our experiments for the two scenarios from Section 2:

- **Scenario (a)** from Section 2: We evaluate our approach by using the restricted-topics sample as the training set and the all-topics sample as the test set. In this case the testing set is an independent collection of datasets. We evaluate both of the 1-to-1 matching and cluster matching approaches for $Rel(D_1, D_2)$. We also evaluate the 1-to-1 matching with $Dup(D_1, D_2)$.
- **Scenario (b)** from Section 2: We evaluate our approach using a leave-one-out (LOO) variant evaluation method and the restricted-topics sample. Here we remove a dataset and all its pairs from the original training set and we use those pairs for evaluation of the output classifiers as a separate test set. We also remove all duplicate pairs of this dataset from the training set to guarantee independence between the training and evaluation environments. We repeat this for every dataset in the input training set. We use the 1-to-1 matching approach in our evaluation.

To execute our experiments, we profile the datasets to extract their meta-features. We use the training set of annotated datasets with the WEKA[3] tool to create the classification models using different supervised techniques: **Bayesian** (Bayesian Network with K2 search, Naïve Bayes) , **Regression** (LogitBoost) , **Support Vector Machines** (Sequential Minimal Optimization) , and **Decision Trees** (Random Forest). We also use **Ensemble Learners** [7]: AdaBoost (with Decision Stump classifier), Classification Via Regression (with M5 Tree classifier), and Random Subspace (with Regression Tree classifier). We tested different techniques because it was suggested by [7] that some individual techniques can outperform the ensemble learners in classification problems. We evaluate the classifiers with 10 different cut-off thresholds for 'c_{rel}' and 'c_{dup}' from Equations (2) and (3), in order to cover a wide range of values. We benchmark the techniques against the decision table technique [9] which simply assigns the majority class based on matching the features to a table of learned examples.

5.3 Results

We evaluate the effectiveness of our approach using the recall, precision, and efficiency-gain measurements, as described in Equations (4),(5) and (6) respectively. Here, TP means true-positives which are the pairs of datasets correctly classified by the classifier. FN are false negatives, FP are false-positives, TN are true-negatives, and N indicates the total number of possible pairs of datasets. The efficiency gain measures the amount of reduction in work required, in terms of number of pairs of datasets eliminated by the classifier.

[3] https://weka.wikispaces.com/Use+WEKA+in+your+Java+code

$$recall = \frac{TP}{TP + FN} \quad (4) \quad precision = \frac{TP}{TP + FP} \quad (5) \quad efficiency - gain = \frac{TN + FN}{N} \quad (6)$$

We change the cut-off thresholds, and we aim to maximize this as much as possible while maintaining the highest recall possible. The effectiveness of our approach is evaluated by recall and precision. By applying our approach with the different scenarios and relationships, we conduct 4 sets of experiments as in Table 4. The results are depicted in the graphs in Fig. 4.

The measures shown in the graphs are all averages from all datasets involved in the test sets for a specific data mining technique and a certain cut-off threshold for the proximity score (darker points have higher cut-off values). The common measure for all graphs, which is the recall plotted on the y-axis, is highlighted by having some of its main values labelled on each graph. Graphs (a) and (b) are the same graphs as (c) and (d) respectively but for the 1-to-1 matching approach applied with the different scenarios. We select for the experiments certain target results, which are minimum expected values for each measure. All area above those values are shaded as follows: **Min. recall**: 0.75 for 1-to-1 matching & 0.9 for cluster matching, **Min. efficiency gain**: 0.33 for 1-to-1 matching & 0.25 for cluster matching, **Min. precision**: 0.25 for all approaches. This means that we were targeting at least 75% recall rate for 1-to-1 matching and 90% recall rate for the cluster based matching (which improves our previous results in [2]). We aimed for at least 25% efficiency gains with the cluster matching approach, which exceeds those achieved in [3]. However, we acknowledge that their approach applied to instances-matching within the same dataset, not cross-schema attribute-matching as in our case. For experiment 4 for $Dup(D_1, D_2)$, we aim for a min. recall of 0.9, min. efficiency gain of 0.75, and min. precision of 0.33. In real-world applications, the data scientist can choose different minimum thresholds for each measure according to practical requirements.

5.4 Discussion

General trend. From the results depicted in Fig. 4, the optimum technique and cut-off threshold is the one in the top-right quadrant of each graph, optimising both measures plotted. The recall-precision and recall-efficiency plots follow the general trend expected which indicate the trade-off between both measures in each plot, yet, more optimised solutions are possible for balancing recall-efficiency, as seen by the classifiers performing in the top-right quadrant. As the cut-off thresholds increase, there is a drop in recall against an increasing efficiency gain. Still, the top mining techniques and thresholds can be used to achieve high efficiency gain and recall. This is discussed for each property below. As the precision rates are generally low, we conclude that our approach can only be used as an early-pruning step, and should be followed by other more expensive and more detailed matching steps. Yet, a good compromise can still achieve high recall and efficiency gains; efficiency gains up to 0.5 for *Rel* and 0.8 for *Dup*. Such efficiency gains can make an important difference for computationally-expensive applications of holistic schema matching in the DL environment.

Table 4: A description of the experiments conducted

Experiment	Graphs in Fig.4	Matching Approach	Scenario	Relationship
1	row 1: (a) and (b)	1-To-1	(b)	$Rel(D_1, D_2)$
2	row 2: (c) and (d)	1-To-1	(a)	$Rel(D_1, D_2)$
3	row 3: (e) and (f)	Cluster-based	(a)	$Rel(D_1, D_2)$
4	row 4: (g) and (h)	1-To-1	(a)	$Dup(D_1, D_2)$

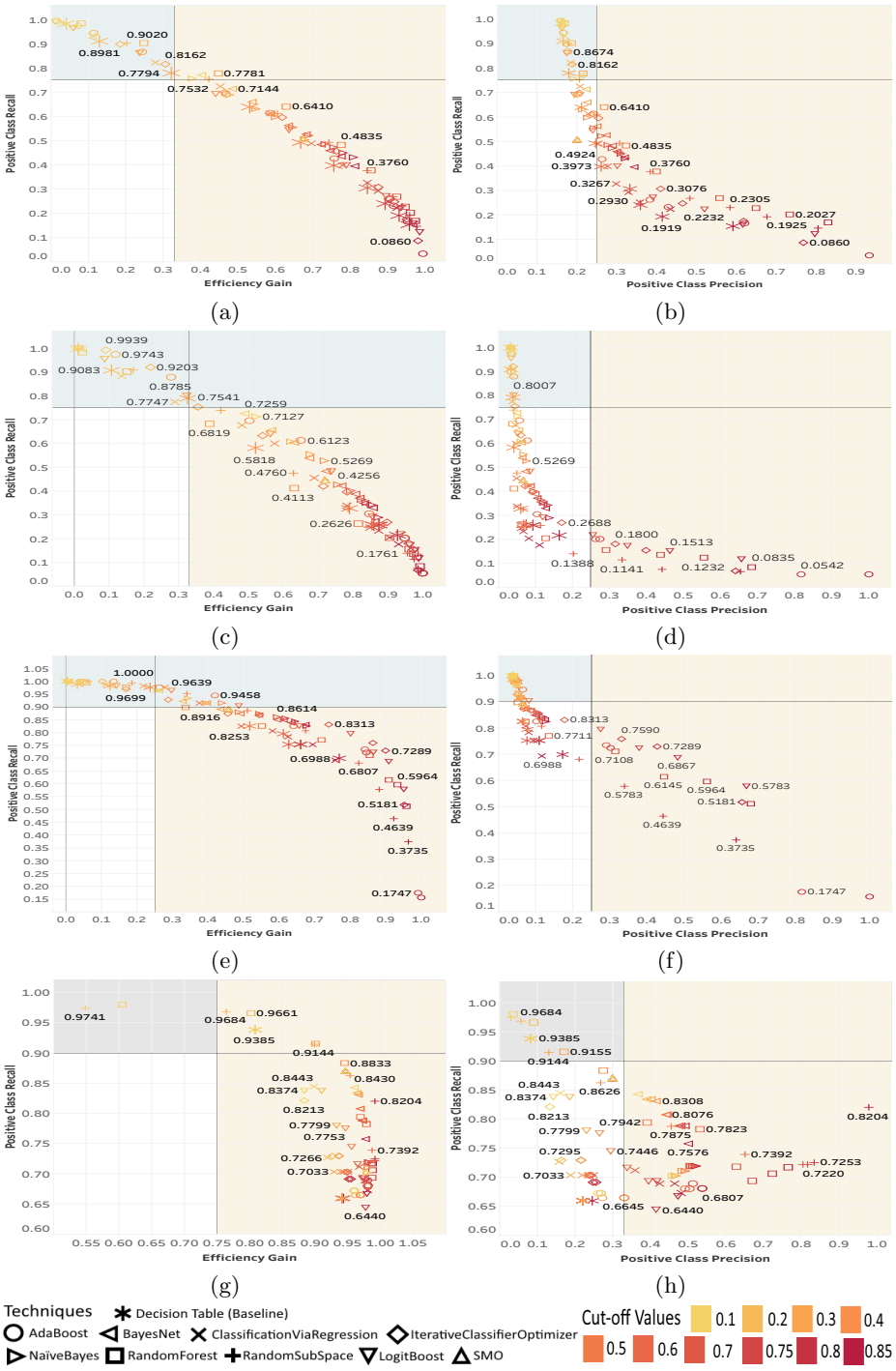

Fig. 4: Recall-efficiency plots (left column) and recall-precision plots (right column) for experiments 1,2,3 and 4 in each row

Rel evaluation. The recall-efficiency plots indicated that it was possible to achieve an optimum technique and threshold in the top-right quadrant, which represent the compromise of not sharply losing recall with higher efficiency gain. For example, from Fig. 4 (e) for experiment 3, using the AdaBoost technique at a threshold of 0.5 can lead to 0.42 efficiency gains while still maintaining 0.95 recall. If a recall of 1.0 is required, then this can be achieved by the cut-off threshold of 0.3 for the same technique, but only 0.13 efficiency gain is achieved. The data scientist will have to decide if this efficiency gain is sufficient and whether a recall rate of 100% is critical in their application, else, a 0.05 drop in recall should be allowed to achieve much higher efficiency gain using the techniques and thresholds in the top quadrant. For the 1-to-1 matching in Fig. 4 (c), we can achieve 0.75 recall and 0.35 efficiency gain. There is a drop in recall, as would be expected, because the classifier has more challenges in matching all possible 'related' datasets, while in the cluster matching approach, a single match to a dataset in a cluster acts like a pivot which results in matching all the required related datasets in the same cluster. The cluster-matching approach shows an improved performance over the 1-to-1 matching approach, therefore it is recommended to use DS-Prox with the clustering-based approach.

Dup evaluation. For the results in Fig. 4 (g) and (h), the top performing techniques were Random subspace and Random Forest at 0.2 cut-off thresholds. This achieved about 0.97 recall and 0.76-0.8 efficiency gain. The baseline method was not able to differentiate at different cut-off thresholds, and had best recall of 0.65, except for the lowest cut-off of 0.1 where it achieved a jump to 0.94 recall. Since the recall was very high for our target efficiency gain using the 1-to-1 approach, the cluster-based approach did not yield any better results.

Baseline comparisons. Different techniques can yield better results than the baseline for several of our experiments. There is not one single technique which is best, yet, ensemble learners tend to perform better than their counterparts. However, simple techniques like logistic regression and Naïve Bayes can still have good performance as seen in the graph (e) top-right quadrant. The baseline technique was never in the top-quadrant of graph (e) and many techniques outperformed it. In the 1-to-1 matching in graphs (a) and (c), the baseline classifier was comparable with the other techniques. The top techniques include the iterative optimiser in graph (c) with 0.75 recall and 0.35 efficiency gain. Nearly 1.0 recall was possible using the same classifier at a lower threshold, yet with only 0.09 efficiency gain. For experiment 1, Random Forest and Random Subspace outperformed the baseline with 0.3 cut-off thresholds.

Generalizability. Although our approach is generic and does not apply to a specific domain only, we note that we do not claim that the classifiers for one type of data or of a certain domain will have the same guaranteed effectiveness when applied in another setting. The approach might need to be adjusted and retrained within other settings. Albeit, our results from experiments 2 and 3 show a positive indicator of the possibility to train the model on specific domains, independent of those used in the test set (or real-world setting), and still be effective. We think that this needs further experimentation in the future.

6 Conclusion and Future Work

This paper presented a novel approach of similarity search within a DL based on a proximity mining technique for early-pruning in holistic dataset schema matching and deduplication applications. The approach uses supervised machine learning techniques based on meta-features describing semi-structured datasets. Experiments on a real-life DL demonstrate the effectiveness in achieving high recall rates and efficiency gains. Proposed techniques support data governance in the DL by identifying relationships between datasets. The drawback of our approach, however, is that it needs some manual effort to annotate training examples for the classifiers. In the future, we will test the generalizability of applying the same classifier to different data sources. We plan to experiment with more detailed meta-features which might lead to improved results. We will also test our approach on other kinds of semi-structured data (like RDF or XML).

Acknowledgement. This research has been funded by the European Commission through the Erasmus Mundus Joint Doctorate (IT4BI-DC).

References

1. Abelló, A.: Big Data Design. In: Proceedings of ACM DOLAP. pp. 35–38 (2015)
2. Alserafi, A., Abelló, A., Romero, O., Calders, T.: Towards Information Profiling: Data Lake Content Metadata Management. In: DINA Workshop, ICDM (2016)
3. Ares, L.G., Brisaboa, N.R., Ordoñez, A., Pedreira, O.: Efficient Similarity Search in Metric Spaces with Cluster Reduction. In: SISAP. pp. 70–84. Springer (2012)
4. Bernstein, P.a., Madhavan, J., Rahm, E.: Generic Schema Matching , Ten Years Later. Proceedings of the VLDB Endowment 4(11), 695–701 (2011)
5. Bilenko, M., Mooney, R.J.: Adaptive Duplicate Detection Using Learnable String Similarity Measures. In: ACM SIGKDD. pp. 39–48 (2003)
6. Cruz, J.A.C., Garza, S.E., Schaeffer, S.E.: Entity Recognition for Duplicate Filtering. In: SISAP. pp. 253–264. Springer (2014)
7. Džeroski, S., Ženko, B.: Is Combining Classifiers with Stacking Better than Selecting the Best One ? Machine learning 54(3), 255–273 (2004)
8. Figueroa, K., Paredes, R.: List of Clustered Permutations for Proximity Searching. In: SISAP. pp. 50–58. Springer (2013)
9. Kohavi, R.: The Power of Decision Tables. In: ECML. pp. 174–189 (1995)
10. Lokoc, J., Cech, P., Novak, J., Skopal, T.: Cut-Region : A Compact Building Block for Hierarchical Metric Indexing. In: SISAP. pp. 85–100. Springer (2012)
11. Naumann, F.: Data profiling revisited. ACM SIGMOD Record 42(4), 40–49 (2014)
12. Patella, M., Ciaccia, P.: Approximate similarity search : A multi-faceted problem. Journal of Discrete Algorithms 7(1), 36–48 (2009)
13. Rahm, E.: The Case for Holistic Data Integration. In: ADBIS. pp. 11–27 (2016)
14. Stonebraker, M., et al.: Data Curation at Scale: The Data Tamer System. In: 6th Biennial Conference on Innovative Data Systems Research (CIDR) (2013)
15. Varga, J., Romero, O., Pedersen, T.B.: Towards Next Generation BI Systems : The Analytical Metadata Challenge. Data Warehousing and Knowledge Discovery - Lecture Notes in Computer Science 8646, 89–101 (2014)