The final publication is available at ACM via http://dx.doi.org/10.1145/3388870

Keeping the Data Lake in Form: Proximity Mining for Pre-filtering Schema Matching

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

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

Abstract. Data Lakes (DLs) are large repositories of raw datasets from disparate sources. As more datasets are ingested into a DL, there is an increasing need for efficient techniques to profile them and to detect the relationships among their schemata, commonly known as *holistic schema matching*. Schema matching detects similarity between the information stored in the datasets to support information discovery and retrieval. Currently, this is computationally expensive with the volume of state-of-the-art DLs. To handle this challenge, we propose a novel early-pruning approach to improve efficiency, where we collect different types of *content metadata* and *schema metadata* about the datasets, and then use this metadata in early-pruning steps to pre-filter the *schema matching* comparisons. This involves computing proximities between datasets based on their metadata, discovering their relationships based on overall proximities and proposing similar dataset pairs for schema matching. We improve the effectiveness of this task by introducing a supervised mining approach for effectively detecting similar datasets which are proposed for further schema matching. We conduct extensive experiments on a real-world DL which proves the success of our approach in effectively detecting similar datasets for schema matching, with recall rates of more than 85% and efficiency improvements above 70%. We empirically show the computational cost saving in space and time by applying our approach in comparison to instance-based schema matching techniques.

Keywords: Data Lake Governance, Holistic Schema Matching, Content Metadata Management, Early-pruning, Dataset Similarity Mining

1 Introduction

Today, it is more and more common for data scientists to use Data Lakes (DLs) to store heterogeneous datasets coming from different sources in their raw format [35]. Such data repositories support the new era of data analytics where datasets are ingested in large amounts and are required to be analysed just-in-time [20]. However, it is a challenge for data wranglers [13,16,35] preparing the datasets

for analysis to understand their structure and *commonalities* for DL governance purposes [4]. They must extract those datasets which have related data to be used together in data analysis tasks [20,23]. This is commonly referred to as *schema matching*, where the aim is to find connection strengths between similar concepts from different pairs of datasets [8,10,21,27]. The large-scale application of such a task to big data repositories like DLs is referred to as *holistic schema matching* [5,23,27,29], where the goal is to match multiple datasets together considering them all in the matching task.

We focus on DLs having datasets storing data in flat tabular formats. Flat datasets are organised as attributes and instances, such as tabular data, comma separated values (CSV) files, hypertext markup language (HTML) tables, etc. (see Section 3). It is a challenge with such DLs to efficiently process the datasets to detect their common features, as schema matching tasks are generally expensive (involving huge amounts of string comparisons and mathematical calculations) [7,8,17]. In this paper, we propose novel techniques to reduce those comparisons using *pre-filtering* techniques that generate less comparisons. To illustrate this, consider the different types of comparisons in Fig. 1. Traditionally, schema matching makes many comparisons of data values of instances from different datasets (see the instance-based matching box). With the rise of DLs, previous research [5,12,7] recommended using early-pruning steps to facilitate the task using *schema matching pre-filtering*. Here, only datasets detected to be of relevant *similarity* are recommended for further fine-grained schema matching tasks, and dissimilar ones are filtered out from further comparisons. Fig. 1 reflects this stratified approach that filters by means of extracted metadata at the dataset and attribute level before going for expensive instance-based approaches. For example, consider a DL with 1000 datasets, with 15 attributes and 1000 instances each. Since schema matching techniques generate $\frac{n*(n-1)}{2}$ comparisons, it would result in 499500 comparisons at the dataset level, 113 million at the attribute level and about 500 billion at the instance level. Clearly, fine-grained comparisons do not scale and pre-filtering is necessary.

We propose a pre-filtering approach based on different levels of granularity, at which we collect metadata using data profiling techniques. Our approach collects metadata at two different levels: at the dataset and attribute level. This metadata is then used in a supervised learning model to estimate *proximity* among pairs of datasets. Such proximity is then used for pruning out pairs *less likely* to be related. This is illustrated in Fig. 1, where our goal is to filter candidate pairs of datasets before conducting computationally intensive instance-based schema matching techniques. The scope of this paper is the early-pruning at the top two granularity tiers. We refer the interested reader to previous research about classical instance-based schema matching which is outside the scope of this paper [7,8,21,30,31].

It is a challenge to compute dataset similarities for pre-filtering tasks due to the difficulty of finding adequate similarity metrics and features to use [5]. This paper is an extension of our previous work [5], and it presents a novel proximity[4]

[4] In this paper, we use proximity and similarity interchangeably.

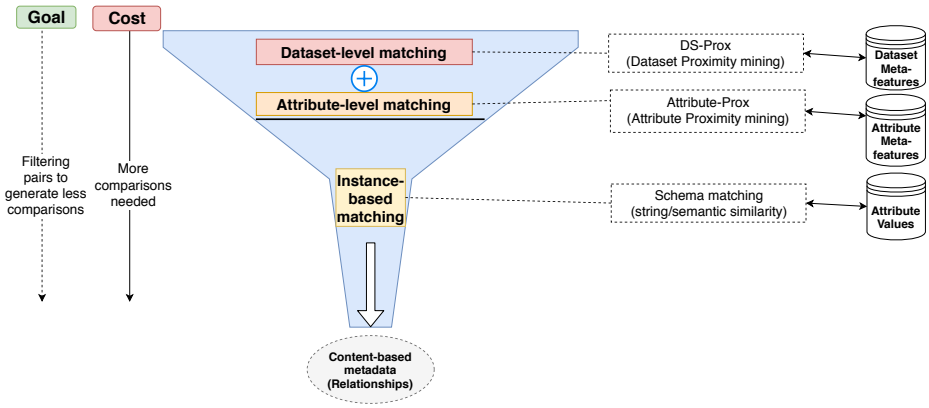

Fig. 1: The stratified holistic schema matching approach at different levels of granularity.

mining approach (see Section 4) between datasets. The similarity functions are based on automatically extracted metadata and can be effectively used for pre-filtering in a stratified approach (see the DS-Prox and Attribute-Prox boxes attached to the matching steps in Fig. 1). To our knowledge, no other approach uses automatically extracted metadata for this purpose.

To show the feasibility of our approach, we assess its performance by using it on a real-world DL. Our approach was able to filter the number of fine-grained comparisons by about 75% while maintaining a recall rate of at least 85% after filtration. Our early-pruning approach also saves computational costs in terms of space and time requirements by at least 2 orders of magnitude compared to instance-based matching.

Contributions. We present an approach for pre-filtering schema matching tasks. We propose techniques for detecting similar schemata based on meta-data at different levels of granularity. This supports in early-pruning of the raw-data instance-based schema matching tasks. We present an expanded and cross-validated experiment for the DS-Prox technique from our previous work [5] and comparisons against combining it with our new proposed attribute-level proximity metrics to find the most appropriate metrics to assign similarities between pairs of datasets. We demonstrate a detailed analysis of the different proximity metrics based on different types of meta-features (name-based and content-based). Our improvements outperform our previous work in terms of effectiveness measures like recall and lift-scores.

The paper is organised as follows: Section 2 presents the related work, Section 3 introduces the main concepts in our research, Section 4 presents our proximity mining based approach for early-pruning tasks of holistic schema matching, Section 5 presents our experimental evaluation, and finally, we conclude in Section 6.

2 Related Work

State-of-the-art schema matching techniques either use schema-level metadata (mainly names and data types of schema components) [10,11,25,27] or instance-level values of data objects [8,10,12,14,21,25,33]. Some others use a hybrid approach utilising schema metadata and data instances [7,28]. At the schema-level, these techniques usually use the syntactic similarity of the names of the schema components for the matching task. At the instance-level, values are usually compared using Jaccard similarity of intersecting exact values [23]. This can also be achieved by first matching duplicate instances and finding the correspondences between their schema components [8]. Further, these algorithms can be domain-specific or generic [21].

Schema matching is a computationally intensive task that requires large amounts of comparisons [4,5,7] because they typically generate a Cartesian product between the values to be compared. Moreover, other approaches also exploit the semantic or linguistic similarity of values, which requires further computations to translate data values (finding synonyms and hypernyms) or to map them to ontologies [21].

The current focus of the schema matching research community is to implement efficient holistic schema matching that improves performance by reducing the number of actual comparisons to conduct [23]. To handle this challenge, multiple techniques were proposed. For example, several approaches use clustering techniques as a pre-filter of datasets to match [3,6,27]. Only datasets falling in the same cluster are matched, or datasets from one cluster are only matched against representative datasets from other clusters. This is similar to the concept of "blocking" for record linkage [32], where items having equal data values in some or all of their attributes are placed in the same bucket for comparison. The work in [32] focuses on matching instances of data (rows in tabular data) rather than attributes in schemata (columns in tabular data). We propose in this paper a supervised learning technique that can classify dataset pairs (schemata) for a decision whether they are related (should be compared) or not related, rather than unsupervised techniques like blocking and clustering. In addition, we tackle the similar schemata search problem (i.e., schema matching) rather than similar records search (i.e., record linkage or entity resolution).

In [27], they cluster the schemata and attributes of datasets based on TF-IDF similarity scores of their textual descriptions. In [3], they exploit the structural properties of semi-structured XML documents, i.e., data elements embeddings and hierarchies, to cluster the datasets before applying instance-based schema matching. In [6], they use the textual descriptions and keywords of the datasets to cluster them using TF-IDF and WordNet. In this paper, we do not consider textual descriptions of datasets, which could be misleading or missing, but rely on metadata that can be automatically extracted from any dataset. Metadata describing datasets, their schemata, and the information stored in them can be collected using data profiling techniques [1,4,18,20,24]. Different types of metadata can also describe the information inside datasets at different levels of

granularity, e.g., overall dataset level [5] or attribute descriptions like we propose in this paper.

The pre-filtering of dataset pairs which are less-likely to have interrelated data before performing schema-matching is called early-pruning [4,12,7], and it was implemented in previous research on semi-structured datasets, like XML, by finding the similarity of hierarchical structures between named data objects [3]. Other works have investigated schema matching with semi-structured datasets like XML [21,25] and JSON [14]. In the web of data, previous research like [26] investigated recommendation of RDF datasets in the semantic web using pre-defined annotations such as the *sameAs* property. In this paper, we consider flat datasets without such a hierarchy of embedded data objects and without pre-defined semantic linkages.

To facilitate the early-pruning tasks for schema matching, we can apply the same approaches and concepts from collaborative filtering and adapt them to the holistic schema matching problem [2,15]. The goal is to use profiling information for comparison and recommendation, which was applied to multimedia in [2] and semi-structured documents in [14]. Content-based metadata was also used to predict schema labels [11]. They use minimum and maximum values for numeric attributes, and exact values for nominal attributes, including the format of values. We propose to apply similar techniques but at the dataset granularity level. Accordingly, we adapt such techniques and find the appropriate similarity metrics for tabular datasets.

Another line of research aims at optimising the schema matching process by using computational improvements [12,28]. This can be done using partitioning techniques that parallelise the schema matching comparison task [28]. Another approach uses efficient string matching comparisons. Such techniques are very useful in the case when schema components are properly named in the datasets. However, such techniques fail when the components are not properly named (e.g., internal conventionalism, like sequential ID numbering of the components). In [12], they introduce intelligent indexing techniques based on value-based signatures.

Schema matching can also be automated using data mining techniques [10,11,14]. In [10], they use hybrid name-based and value-based classification models to match dataset attributes to a mediated integration schema. Their approach is focused on one-to-one mediation between two schemata, while our approach targets all datasets in a DL by holistic schema matching using coarser meta-features. In [11], they use content-based meta-features in multi-value classification models to match schema labels of attributes across datasets. Decision trees were also used to profile semi-structured documents before schema matching [14]. In this paper, we also use data mining classification models, however the goal differs from [10], [11] and [14] as it tackles the early-pruning and pre-filtering task rather than instance-based schema matching.

We summarise the state-of-the-art in Table 1. This table gives a comparison of the most relevant techniques discussed with our approach based on the main features discussed in this section. As a result, we can see that we propose

an approach based not only on string matching, but also on content metadata matching involving statistics and profiles of data stored in the datasets. Content metadata are matched based on approximate similarities and not just exact value-matches at the instance-level [20,23]. We focus on proposing novel early-pruning techniques that use supervised learning to pre-filter irrelevant dataset pairs and to detect likely-to-be similar pairs. Finally, the table shows that our technique makes a novel contribution to the schema matching pre-filtering problem that is not achieved by other state-of-the-art techniques.

Table 1: Schema matching techniques state-of-the-art comparison

	COMA++ [28]	**PARIS [33]**	**LOD Data Linking [6]**	**XML Semantic-based Matching [17]**	**Ontology Clustering [3]**	**Proximity Mining [this paper]**
Type of Data	Tabular, semi-structured, Semantic OWL	RDF	Semantic RDF	Semi-structured XML	Semi-structured, Semantic OWL	Tabular
Instance-based	✓	✓	✓	✓	×	×
Metadata used	Attribute-level schema names	×	Ontology mappings, RDF schema names, Textual descriptions	Attribute-level schema names and structural metadata	Attribute-level structural metadata	Dataset-level content and name, Attribute-level content and name
Data Mining based	×	×	Clustering	×	Clustering	Supervised learning
Approximate Matching	×	×	✓	×	✓	✓

3 Preliminaries

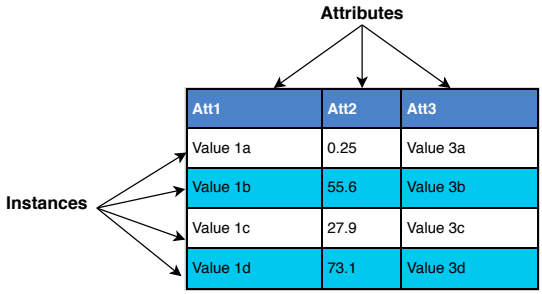

Fig. 2: A flat dataset with attributes and instances.

We consider DLs with datasets having tabular schemas that are structured as attributes and instances like Fig. 2. We formally define a dataset D as a set of instances $D = \{I_1, I_2, ...I_n\}$. Each dataset has a schema $S = \{A_1, A_2, ...A_m\}$, where each attribute A_i has a *data type* and describes a single property of the instances in the dataset. We focus on two types of attributes: ***numeric attributes*** (consisting of real numbers) and ***nominal attributes*** (consisting of discrete categorical values). We differentiate between those two types of attributes, similar to previous research like [10,11,25], because we collect different profiling metadata for them. The resulting statistics collected are called *content meta-features*, and are as follows:

- **Nominal attributes:** frequency distributions of their distinct values.
- **Numeric attributes:** aggregated statistical value distributions like mean, min, max, and standard deviations.

For pairs of datasets and attributes, we compute the functions in Table 2 and describe them in the rest of this section.

Table 2: Schema matching pre-filtering functions

Relationship	Function Type	Output	Object Type	Description
$Rel(A_i, A_j)$	Binary	$\mathbb{Z} \in \{0, 1\}$	Attribute pair	Related attributes storing data about the same real-life concept which contain overlapping information in their values. 1 means positively related and 0 means not.
$Sim(A_i, A_j)$	Continuous	$\mathbb{R} \in [0, 1]$	Attribute pair	A value $\mathbb{R}$ to measure the attribute similarity in the range [0,1].
$Rel(D_y, D_z)$	Binary	$\mathbb{Z} \in \{0, 1\}$	Dataset pair	Related datasets which contain information about the same real-life object. 1 means positively related and 0 means not.
$Sim(D_y, D_z)$	Continuous	$\mathbb{R} \in [0, 1]$	Dataset pair	A value $\mathbb{R}$ to measure the dataset similarity in the range [0,1].

We aim at finding the relationship between a pair of datasets $Rel(D_y, D_z)$. We determine such relationship directly using dataset-level meta-features and by computing the relationships between their pairs of attributes (A_i, A_j), being A_i from D_y and A_j from D_z, as could be seen in Fig. 3. The figure shows an overview of our proposed proximity mining approach.

The goals of the approach is to use efficient and effective techniques to accurately predict $Rel(D_y, D_z)$ for the pre-filtering task. As could be seen in Fig. 3, this can be done using metadata and similarity collected at the dataset level (right-side) or by using the attribute level $Sim(A_i, A_j)$ to predict it (left-side). The figure shows the steps required for combining attribute-level similarity with the dataset-level similarity to predict $Rel(D_y, D_z)$. Here, we only use $Sim(A_i, A_j)$ as an auxiliary step that supports us in the main task of predicting $Rel(D_y, D_z)$. This is possible because the attribute metadata are of finer granularity which can be aggregated to a single similarity score at the dataset-level

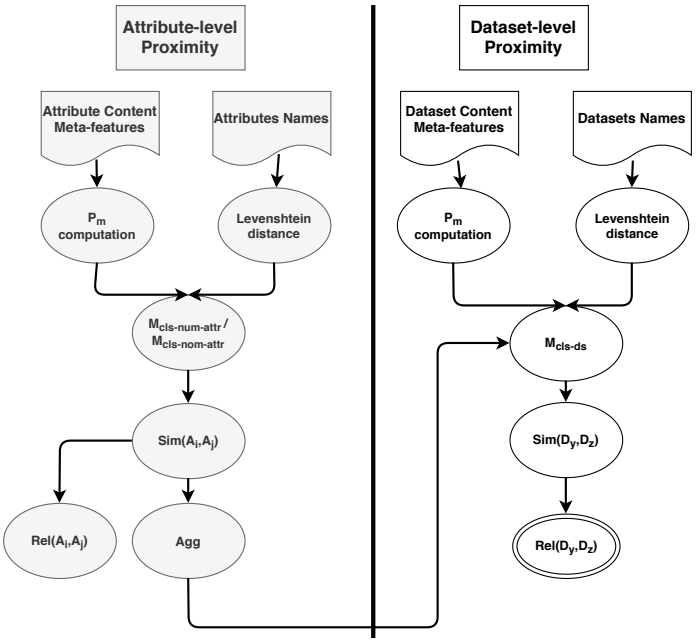

Fig. 3: The dependencies of components in the metadata-based proximity mining approach for pre-filtering schema matching.

with an aggregation function Agg like averaging $Sim(A_i, A_j)$ scores. When predicting $Rel(D_y, D_z)$, typically the dataset pair will have information contained in some of their attributes which are partially overlapping, satisfying $Rel(A_i, A_j)$, where $\exists A_i \in D_y \wedge A_j \in D_z \implies Rel(A_i, A_j) = 1$. An example would be a pair of datasets describing different human diseases, (e.g., diabetes and hypertension). The datasets will have similar attributes (partially) overlapping their information like the patient's age, gender, and some common lab tests like blood samples.

The *intermediate output* leading to $Rel(A_i, A_j)$ and $Rel(D_y, D_z)$ in our proposed proximity-based approach, seen in Fig. 3, is a similarity score consisting of a real number in the range of $[0, 1]$, which we indicate using $Sim(A_i, A_j)$ and $Sim(D_y, D_z)$ respectively.

The similarity scores are computed based on proximity models we construct using ensemble supervised learning techniques [34], which we denote as M_{cls-ds} for models handling dataset-level metadata and $M_{cls-num-attr}$ or $M_{cls-nom-attr}$ for models handling attribute-level metadata (depending on the attribute type, numerical or nominal respectively). The models take as input the *distance* between the meta-features describing content of each object pair, whether dataset pair or attribute pair for $Sim(D_y, D_z)$ and $Sim(A_i, A_j)$ respectively, and we call the distance in a specific meta-feature 'm' *a proximity metric* which is denoted as $P^D_m(D_y, D_z)$ for dataset pairs or $P^A_m(A_i, A_j)$ for attribute pairs. The names of

objects can also be compared using Levenshtein string distance comparison [22] to generate a distance score. The output from the models is a score we compute using the positive class distribution (see Section 4.2).

We convert the intermediate similarity scores to the *final output* consisting of a boolean value for the binary relationships using Equations (1) and (2). The *sim* score computed for each relationship type is checked against a minimum threshold in the range of $[0, 1]$ to indicate whether the pair involved is overall related '1' or unrelated '0', and therefore whether they should be proposed for expensive schema matching or otherwise filtered out. Using cut-off thresholds of similarity rankings for the collaborative filtering task and schema matching is a common practice [12,15,20]. We can use different thresholds 'c_d' and 'c_a' for each of the relationship evaluated at the dataset level and attribute level respectively. This means that we only consider a pair *similar* if their similarity score is greater than the threshold as in Equations (1) and (2).

$$Rel(D_y, D_z) = \begin{cases} 1, & Sim(D_y, D_z) > c_d \\ 0, & \text{otherwise} \end{cases} \quad (1) \qquad Rel(A_i, A_j) = \begin{cases} 1, & Sim(A_i, A_j) > c_a \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

To summarise Fig. 3, the hierarchy to compute the final output is: Rel is based on Sim similarity scores, which in turn are based on P_m proximity metrics of meta-features. To convert from P_m to Sim we use an ensemble supervised model M_{cls} which takes the P_m proximity metrics as input. The output Sim is compared against a minimum threshold, and those passing the threshold are positive cases for Rel to be considered for further detailed schema matching.

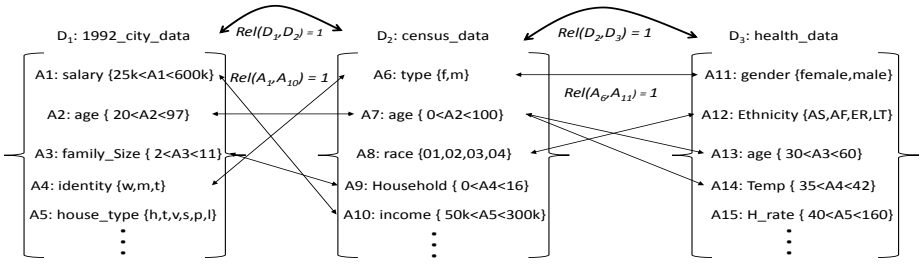

Fig. 4: Final output of our approach consisting of similarity relationships between two pairs of datasets.

Examples. Consider the relationships between the three datasets in Fig. 4 which presents the *final output* of our approach. Each dataset has a set of attributes. An arrow links attributes having similar data profiles. We label this as a $Rel(A_i, A_j) = 1$. For example, attributes 'A6' and 'A11' from D_2 and D_3 are nominal attributes with two unique values which we included as a meta-feature called *'number of unique values'*. The proximity metric is the distance (difference) in the meta-feature of number of unique values, which in this case

$P_m^A = 0$, because they are identical (i.e., $2 - 2 = 0$), thus making the attribute pair similar (in this case, by their number of distinct values). If we consider this proximity metric of 'number of unique values' alongside other collected content-based meta-features using an *ensemble supervised learning model* M_{cls}, we can compute a $Sim(A_6, A_{11})$ score based on the positive-class distribution (see Section 4.2). This can lead to $Sim(A_6, A_{11}) = 0.95$ and if we use a threshold of $c_a = 0.75$ then the final output for $Rel(A_6, A_{11}) = 1$. A numeric attribute like 'A7' in D_2 holds similar data as attributes 'A13' and 'A14' from D_3, as expressed by the intersecting numeric ranges. For such numeric attributes we can consider a meta-feature like *'mean value'*. On the other hand, attributes 'A1' and 'A7' have different data profiles (different numeric ranges) and therefore are not labelled with an arrow and do not satisfy the $Rel(A_1, A_7)$ relationship, as they will have large differences in their meta-features, leading to high proximity metric and a low similarity scores. In those examples, we collect attribute level meta-features from the datasets (in this case, the number of distinct values for nominal attributes and means for numeric attributes) to assess the similarity between attributes of a given pair of datasets. In our approach, we compute the similarity between attributes $Sim(A_i, A_j)$ using real number proximity metrics in the range of $[0, 1]$ and we use it to predict $Rel(D_y, D_z)$ instead of using the binary output of $Rel(A_i, A_j)$. We should *aggregate* the individual attribute pairs' similarities with an aggregation function agg to obtain a single value proximity metric for the overall dataset level similarity. We discuss this in the description of our approach in Section 4.

Furthermore, we extract higher-granularity dataset level meta-features (e.g., 'number of attributes per attribute type') from the datasets for the task of directly computing the $Sim(D_y, D_z)$ similarity relationships. For example, $Rel(D_2, D_3)$ returns '1' in the case we use $c_a = 0.67$ because they have 2 nominal and 3 numeric attributes each, so overall they can have $Sim(D_2, D_3) = 0.7$ passing the minimum threshold. Based on $Rel(D_2, D_3) =$'1', our approach indicates that these two datasets are possibly related and should be considered for further scrutinising by schema matching.

4 Approach: Metadata-based Proximity Mining for Pre-filtering Schema Matching

Our goal is to effectively apply early-pruning for holistic schema matching in a DL setting. Such pre-filtering is based on novel proximity mining techniques. Those techniques evaluate similarity among pairs of datasets using automatically extracted meta-features and utilising a data-mining ensemble supervised model to select highly similar pairs. We apply this using the stratified approach (Fig. 1). An overview of the approach is summarised in Fig. 3, which shows the steps required to compute the schema matching pre-filtering functions from Table 2. We explain how we build and apply those models in this section.

In the remaining subsections, we describe the details of our approach as follows: profile the datasets to extract the meta-features and pair-

wise proximity metrics in *Subsection 4.1*, use supervised proximity mining to build the ensemble models for $Rel(A_i, A_j)$ and $Rel(D_y, D_z)$ in *Subsection 4.2*, then apply the models on new pairs of attributes and datasets in the DL to compute their $Sim(A_i, A_j)$ and $Sim(D_y, D_z)$ scores in *Subsection 4.2*, and finally using the $Sim(D_y, D_z)$ to predict $Rel(D_y, D_z)$ for pairs of datasets and applying the pre-filtering step for schema matching in *Subsection 4.3*.

4.1 Proximity Metrics: Meta-features distances

Our approach gathers metadata at two levels of granularity: at the **I. dataset level** and **II. attribute level**. Further, at each of these levels, we gather **A. content-based** meta-features with profiling statistics and **B. name-based** meta-features with the naming of datasets and their attributes. The name-based techniques are the most commonly used metadata in previous research [14,30,31]. We propose other content-based meta-features at the two levels of granularity as follows:

- ***Dataset level (DS-Prox):*** We collect overall meta-features summarising the dataset content: overall statistics concerning all the attributes collectively, the attribute types found and the overall number of instances. The meta-features used are described in our previous work [5], which includes a detailed list of meta-features that proved to be effective in predicting related datasets for schema matching pre-filtering, e.g., number of instances, number of attributes per attribute type, dimensionality, number of missing values, etc.
- ***Attribute level (Attribute-Prox):*** The set of meta-features used for both types of attributes, nominal and numeric, is described in Table 3. For each attribute A_i in dataset D, we profile it based on its type by computing the appropriate features.

Table 3: Attribute level content meta-features

Attribute Type	Meta-feature	Description
All	distinct_values_cnt	The number of distinct values
All	distinct_values_pct	The percentage of the distinct values from number of instances
All	missing_values_pct	The percentage of missing values from number of instances
Nominal	val_size_avg	The average number of strings in values from the attribute
Nominal	val_size_min	The minimum number of strings in values from the attribute
Nominal	val_size_max	The maximum number of strings in values from the attribute
Nominal	val_size_std	The standard deviation of number of strings in values from the attribute
Nominal	val_pct_median	The median percentage of instances per each value of the attribute
Nominal	val_pct_min	The minimum percentage of instances per each value of the attribute
Nominal	val_pct_max	The maximum percentage of instances per each value of the attribute
Nominal	val_pct_std	The standard deviation of the percentage of instances per each value of the attribute
Numeric	mean	The mean numeric value of the attribute
Numeric	std	The standard deviation of the numeric value of the attribute
Numeric	min_val	The minimum numeric value of the attribute
Numeric	max_val	The maximum numeric value of the attribute
Numeric	range_val	The numeric range of the values of the attribute
Numeric	co_of_var	The numeric coefficient of variance of the attribute

Equation 3 shows the proximity metric computed for a pair of attributes (or datasets), denoted as O_i, O_j. Using the meta-features described, we compute the

z-score distance for each meta-feature m. The result is a real number, P_m. The z-score is a normalisation where we use the mean 'μ' and standard deviation 'σ' of each meta-feature considering its value from all datasets in the DL. A value of 0 is the most similar, while larger negative or positive number means more different. The z-score is used to standardise the comparisons of attributes in a holistic manner that considers all datasets and attributes in the DL. Most pairs of attributes and dataset will have a value falling in the range of $[-3, 3]$.

$$P_m = zscore\_distance(O_i, O_j) = \left| \frac{m(O_i) - \mu}{\sigma} - \frac{m(O_j) - \mu}{\sigma} \right| \tag{3}$$

For the name-based metadata we compute the proximity metric P_m with a Levenshtein string comparison function [22], as in Equation 4.

$$P_m = levenshtein\_distance(name(O_i), name(O_j)) \tag{4}$$

4.2 Supervised Proximity Mining

After the different proximity metrics of meta-features are generated by profiling the datasets, a *representative sample* (an adequate sample size should be similar to the sample in our experiments in Section 5) of dataset and attribute pairs should be selected by a human annotator and should be labelled whether they satisfy $Rel(D_y, A_z)$ and $Rel(A_i, A_j)$ respectively. The dataset and attribute pairs with their proximity metrics and labels are fed to a supervised learning algorithm to create a proximity scoring model. We propose supervised ensemble models based on different dataset level and attribute level proximity metrics for computing overall similarity between pairs of datasets. The models decide on the number of attributes to consider in order to evaluate a pair of datasets as 'related' by using different aggregation functions *agg* for the attribute level metrics, giving different weights to a different number of attribute linkages of different similarity ranks. This will be explained in detail in this section.

Our approach builds supervised ensemble models M_{cls-ds} for $Rel(D_y, D_z)$, and $M_{cls-nom-attr}$ & $M_{cls-num-attr}$ for $Rel(A_i, A_j)$ whether the attribute type is nominal or numerical respectively. Model-based learning for pre-filtering has been applied before in the collaborative filtering field [2]. In such scenarios, item pairs are recommended or filtered out using model-based learning algorithms where a learnt model is used to assign the similarities and rankings of item pairs based on previously annotated examples. We give details of how we learn the models and how we use them in our approach in the subsections below.

Building the models from annotated samples An overview process for building the supervised models in our approach can be seen in Fig. 5. In the ***build*** phase, we take the pairs of datasets and profile them by computing their dataset level and attribute level meta-features, followed by computing the proximity metrics for those extracted features. We take different dataset pair samples for building the attribute level models and the dataset level models as seen in the split into samples OML01 and OML02 (how to build such samples is given in

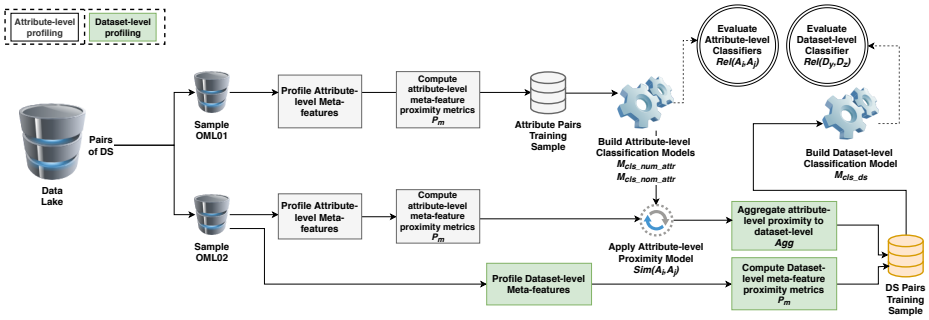

Fig. 5: An overview of the process to ***build*** the supervised ensemble models in our proposed datasets proximity mining approach using previously manually annotated dataset pairs.

Section 5.1). The pairs in sample OML01 should be already annotated to indicate whether their attributes match (i.e., $Rel(A_i, A_j)$ for attribute level models) or whether the datasets are relevant for schema matching or not in sample OML02 (i.e., $Rel(D_y, D_z)$ for dataset level models). Initially, we start with the attribute level supervised learning procedure as it is only an auxiliary subcomponent used for the dataset level, where an aggregation step is used to compute dataset level proximities. First, we divide the pairs into training and test sets, we train a supervised learning ensemble model for each attribute type (nominal and numeric types) using the training sample, and we test the performance of the model on the test set (evaluation distinguished by dotted lines and circles in the figure). We conduct this test to guarantee that the models generated are accurate in detecting $Rel(A_i, A_j)$. Similarly, we do the same with the dataset level supervised models which generate $Rel(D_y, D_z)$. We use the dataset level proximity metrics and the attribute level aggregated proximity metrics together to train a supervised model using a training sub-sample of dataset pairs from OML02. Finally, we evaluate the generated dataset level supervised models to guarantee their accuracy in detecting $Rel(D_y, D_z)$.

Supervised learning. To build the models, we use classical supervised learning to create the proximity models. The meta-features are used as input to the models as seen in Fig. 6, where an object could be an attribute for attribute-level models or a dataset for dataset-level models. First, for each object we extract its meta-features (i.e., 'm1', 'm2', ...). Then, for each object, we generate all pairs with each of the other objects and compute the proximity metrics between their meta-features using either Equation 3 for content-based meta-features or Equation 4 for the name-based comparison. We then take a sample of pairs of objects which are analysed by a data analyst; a human-annotator who manually decides whether the pairs of objects satisfy (assign '1') or not (assign '0') the *Rel* properties (see Section 3). This can be achieved by simply labelling the objects with their respective subject-areas and those falling under the same one are annotated as positively matching '1', otherwise all others are

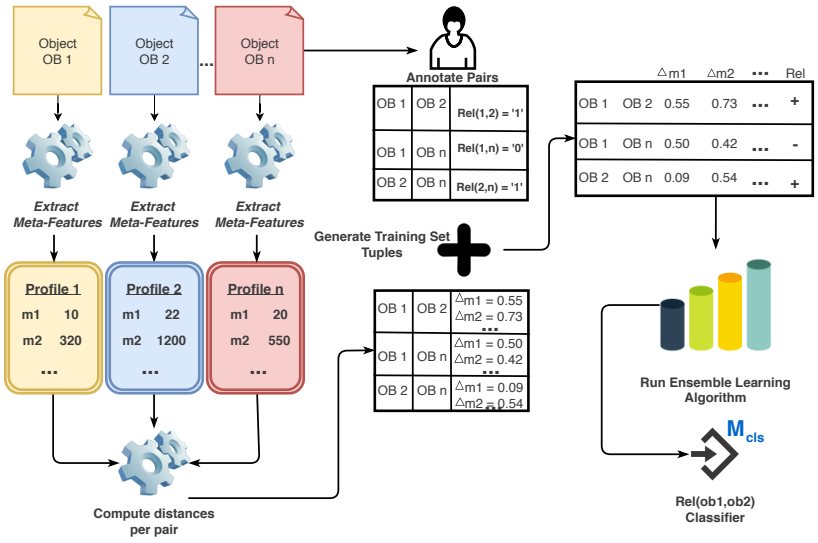

Fig. 6: Proximity Mining: supervised machine learning for predicting related data objects.

labelled with '0' (see Section 5.1). We then use supervised learning techniques and 10-fold cross-validation over the proximity metrics to create two types of independent models which can classify pairs of datasets or pairs of attributes according to $Rel(D_y, D_z)$ and $Rel(A_i, A_j)$ respectively. This is the final output consisting of the two auxiliary supervised models $M_{cls-nom-attr}$, $M_{cls-num-attr}$ for $Rel(A_i, A_j)$ and the main dataset level model M_{cls-ds} for $Rel(D_y, D_z)$. The positive-class distribution from the generated models is used to *score* new pairs of objects (unseen in the training process) with a similarity score $Sim(D_y, D_z)$ using M_{cls-ds}, and $Sim(A_i, A_j)$ using $M_{cls-nom-attr}$ or $M_{cls-num-attr}$.

We use a random forest ensemble algorithm [9,34] to train the supervised models in predicting related attribute and dataset pairs as it is one of the most successful supervised learning techniques. The algorithm generates a similarity score based on the positive-class distribution (i.e., the predicted probability of the positive-class based on weighted averages of votes for the positive class from all the sub-models in the ensemble model) to generate a score in $[0, 1]$ for Sim. For example, if Random Forest generates 1000 decision trees, and for a pair of datasets $[D_y, D_z]$ we get 900 trees vote positive for $Rel(D_y, D_z)$ then we get $\frac{900}{1000} = 0.9$ for $Sim(D_y, D_z)$ score.

We feed the supervised learning algorithm the normalised proximity metrics of the meta-features for pairs of datasets $[D_y, D_z]$. For attribute level meta-features, we feed the M_{cls-ds} model with all the different aggregations of the meta-features after computing their normalised proximity metrics (i.e., after applying Equation 7, which we describe later in this section).

ALGORITHM 1: Attribute Level Top-Similarity Matching

Input: Sets of the attribute meta-features of each type $Att_{nominal}$ and $Att_{numeric}$ containing the proximity metrics of the meta-features of each attribute in the pair $\{A_i, A_j\}$ for each dataset in the pair $\{D_y, D_z\}$, the model $M_{cls-nom-attr}$ for nominal attributes, the model $M_{cls-num-attr}$ for numeric attributes, an aggregation function Agg for aggregating attribute links to compute dataset level proximity

Output: The partially ordered set SP of proximity metrics $P_m^D(D_y, D_z)$ for each pair of $\{D_y, D_z\}$

$SP_{dataset} \leftarrow \emptyset$;
$SP_{attribute} \leftarrow \emptyset$;
$SP_{top\_attribute} \leftarrow \emptyset$;
foreach $\{D_y, D_z\} \subset DL$ *and* $y \neq z$ **do**
  foreach $\{A_i, A_j\} \subset Att_{nominal}$ *and* $A_i \in D_y$ *and* $A_j \in D_z$ **do**
    $Sim(A_i, A_j) = M_{cls-nom-attr}(A_i, A_j)$;
    $SP_{attribute} \leftarrow SP_{attribute} \cup \{[A_i, A_j, Sim(A_i, A_j)]\}$;
  end
  foreach $\{A_i, A_j\} \subset Att_{numeric}$ *and* $A_i \in D_y$ *and* $A_j \in D_z$ **do**
    $Sim(A_i, A_j) = M_{cls-num-attr}(A_i, A_j)$;
    $SP_{attribute} \leftarrow SP_{attribute} \cup \{[A_i, A_j, Sim(A_i, A_j)]\}$;
  end
  \\Iterate on the set of attribute pairs $SP_{attribute}$ to find top matching pairs
  while *more attribute pairs* $\{A_i, A_j\}$ *can be picked* **do**
    Pick pair $\{A_i, A_j\}$ from $SP_{attribute}$ that maximises $Sim(A_i, A_j)$ where A_i and A_j were not picked before;
    $SP_{top\_attribute} \leftarrow SP_{top\_attribute} \cup \{[A_i, A_j, Sim(A_i, A_j)]\}$;
  end
  $P_m^D(D_y, D_z) \leftarrow Agg(SP_{top\_attribute})$;
  $SP_{dataset} \leftarrow SP_{dataset} \cup \{[D_y, D_z, P_m^D(D_y, D_z)]\}$;
end

Attribute-level proximity. To compute the overall proximity of datasets using their attribute level meta-features we use Algorithm 1, which first compares the proximity metrics from the meta-features of each attribute of a specific type against all other attributes of the same type in the other dataset using $M_{cls-nom-attr}$ for nominal attributes and $M_{cls-num-attr}$ for numeric attributes. The algorithm then finds top matching attribute pairs where we match each attribute to the most similar attribute in the other dataset using a *greedy approach* [19]. For each pair of datasets, we match each attribute only once (we do not allow many-to-many matching). We rank attribute pairs by proximity top-to-least, then we assign matching pairs on the top of the list where each attribute did not appear in a previous higher ranking pair (i.e., both attributes need to be unmatched by any higher ranking pair in the list, otherwise the algorithm skips to the next pair until all the list of pairs is traversed). Finally, in order to summarise the attribute linkages to predict the overall proximity of the dataset pairs, we compute an *aggregation* of the top-matching attribute linkages found between a pair of datasets using an *function Agg* to convert the multiple $Sim(A_i, A_j)$ scores to a single proximity metric P_m^D for their dataset pair. We use different types of attribute level aggregation functions. Those functions assign different weights 'W' (which is an indicator of relevance, a bigger weight means more relevant) to the attribute links to consider. The different aggregations should have the goal of giving less weight to attribute links which could be considered as *noise*; i.e., those pairs which are too strongly correlated without

any meaning (e.g., discrete ID numbers) or those attribute pairs with too low proximity to be significant.

Thus, the top-matching pairs of attributes are sorted by proximity weights and are fed to the aggregation function which allocates a weight between $[0, 1]$ for aggregation in the summation of weights. The total sum of weights should add up to 1.0. The different aggregations we use are as follows:

- ***Minimum:*** we allocate all the weight (i.e., $W = 1.0$) to the single attribute pair link with the minimum similarity, and we consider this as the overall proximity between the dataset pair. Therefore, all top-matching attribute pair links need to have a high similarity score to result into a high proximity for a dataset pair.
- ***Maximum:*** we allocate all the weight (i.e., $W = 1.0$) to the single attribute pair link with the maximum similarity, and we consider this as the overall proximity between the dataset pair. Therefore, only one top-matching attribute pair link needs to have a high similarity score to result into a high proximity for a dataset pair.
- ***Euclidean:*** a Euclidean aggregation of the similarities Sim of all matching pairs of attributes without any weighting as in Equation 5. Here we consider all the attribute pair links in the aggregation and we assign equal weights to all the links.

$$P_m^D = \sqrt{\sum_{i=1,j=2}^{n} [Sim(A_i, A_j)]^2} \tag{5}$$

- ***Average:*** a standard averaging aggregation of the pairs of attributes without any weighting, where all attribute links are equally weighted in the average.
- ***Weighted function:*** a normal distribution function to assign different proximity weights W for all attribute linkages found, and then summing up all the weighted similarities as the overall proximity as in Equation 6.

$$P_m^D = \sum_{i=1,j=2}^{n} [W_i * Sim(A_i, A_j)] \tag{6}$$

This is visualised in Fig. 7. Here the weight $0.0 \leq W \leq 1.0$ for each top-matching attribute linkage is assigned based on ordering the linkages top-to-least in terms of their similarity scores, and the weight allocated varies according to a normal distribution. We use different p-parameters (probability of success) of $\{0.1, 0.25, 0.5, 0.75, 0.9\}$, where a parameter of 0.5 leads to a standard normal distribution of weights allocated for the sorted pairs of attributes. A lower parameter value leads to skewness to the left, allocating more weight to highly related pairs, and a higher parameter leads to skewness to the right, allocating higher weights to pairs with lower ranked relationships. This means that with lower p we expect similar datasets to have a few very similar attributes and a higher p value means we expect most of the attributes to be strongly similar.

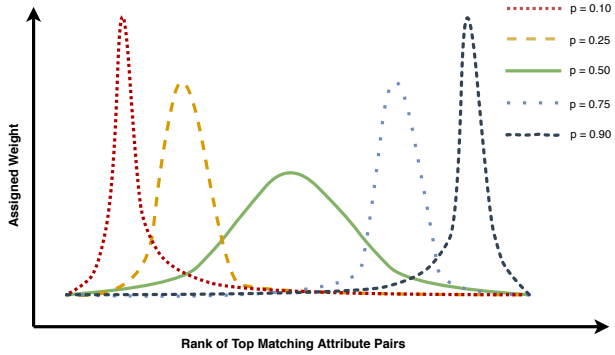

Fig. 7: Different normal distributions for assigning weights to ranked attribute linkages.

As can be seen from their descriptions, each aggregation leads to a different meaning of similarity based on the number of attribute linkages to consider and which attribute linkages are considered more important (having higher weights assigned). All the dataset proximity metrics generated by the different aggregations listed above are finally normalised using Equation 7. The proximity metric P_m^D for two datasets D_y and D_z is computed by multiplying the number of matching attributes found (N), and divided by the minimum number of attributes of both datasets ($Min(|Attr_y|, |Attr_z|)$). This is done to prevent an inaccurate similarity score for two datasets having few very similar attributes of a single type, and many other attributes of different types. For example, if dataset D_1 has 1 nominal attribute and 10 numeric attributes and D_2 just has 8 nominal attributes, then if the single nominal attribute in D_1 is highly similar to a nominal attribute in D_2 (e.g., $Sim(A_1, A_2) = 0.9$) then the overall outcome without normalisation will be a high proximity metric between both datasets although they have many disjoint attribute types. The resulting proximity metric after normalisation for the datasets would be calculated as follows: $P\prime_m^D = 0.9 * \frac{1}{9} = 0.1$, so overall they will have a low proximity compensating for all the unmatched attributes without corresponding types.

$$P\prime_m^D = P_m^D * \frac{N}{Min(|Attr_y|, |Attr_z|)} \tag{7}$$

Applying the models on the DL In the second phase, after building the ensemble models, we apply them to each new pair of previously unseen datasets to achieve a measure of the similarity score. When applying the models, we compute for each pair of datasets the similarity score of $Sim(D_y, D_z)$ and for each attribute pair $Sim(A_i, A_j)$ using the supervised models extracted in the previous phase. The Sim score is the positive-class distribution value generated by each ensemble model [34]. For the attribute level scoring, we complete the proximity mining task by aggregating the sim scores between pairs of datasets

(as seen in the last steps of Algorithm 1). To compare dataset pairs, we use Algorithm 2, and the M_{cls-ds} model generated by the previous build phase.

ALGORITHM 2: Dataset level Matching

Input: Dataset-level proximity metrics for each pair of datasets $\{D_y, D_z\}$, the model M_{cls-ds}
Output: The set SP of similarity score $[D_y, D_z, Sim(D_y, D_z)]$ for each pair of $\{D_y, D_z\}$
$SP \leftarrow \emptyset$;
foreach $\{D_y, D_z\} \subset DL$ *and* $i \neq j$ **do**
    $Sim(D_y, D_z) = M_{cls-ds}(D_y, D_z)$;
    $SP \leftarrow SP \cup \{[D_y, D_z, Sim(D_y, D_z)]\}$;
end

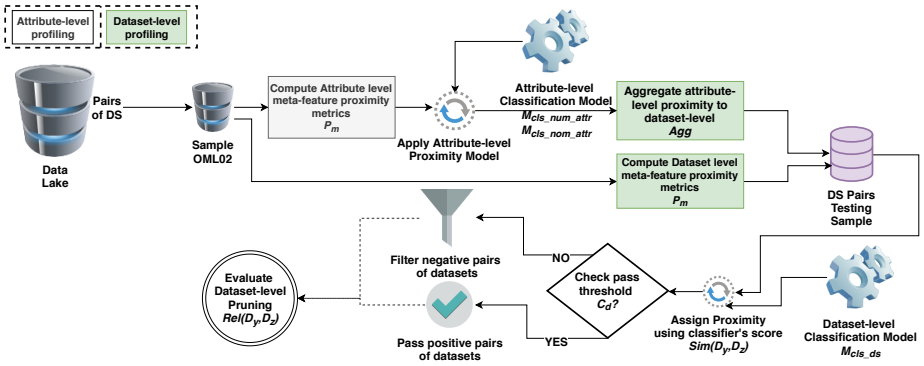

Fig. 8: An overview of the process to ***apply*** the learnt supervised models in our approach for pre-filtering previously unseen dataset pairs independent of the build process.

In the ***apply*** phase visualised in Fig. 8, we take the pairs of datasets from sample OML02 which have not been used in the build phase and we compute the proximity metrics for the dataset level and attribute level meta-features. First, we profile the attribute level meta-features from this new dataset pairs sample. Then, we apply the attribute level supervised models resulting from the previous sub-process to score the attribute pairs similarities from the different dataset pairs. Then, we aggregate the resulting attribute pairs similarities to the dataset level using the aggregation functions. Once we have the dataset level proximity metrics generated from dataset level and attribute level meta-features, we feed them all to the dataset level supervised models from the build phase to unseen testing set pairs, not used in the training of the models, which assigns a proximity score to the pairs. If a pair exceeds a certain proximity threshold, we consider that pair as a positive match to propose for further schema matching, otherwise the pair is considered as a negative match and is pruned out from

further schema matching tasks (we evaluate this by pruning effectiveness metrics in Section 5.2). This is described in the next subsection.

4.3 Pre-filtering Dataset Pairs for Schema Matching

For the final step of pre-filtering pairs of datasets before applying detailed instance-based schema matching, we check whether the pairs of datasets are overall related or not, and therefore whether they should be filtered out or proposed for expensive schema matching. We analyses the final $Sim(D_y, D_z)$ score generated by the model M_{cls-ds} for each dataset pair in the DL to decide whether they satisfy the $Rel(D_y, D_z)$ or not. We consider the relationship of each dataset with each of the other datasets existing in the DL. Each dataset must pass the similarity threshold c_d with each individual dataset to be proposed for detailed schema matching (as in Equation 1).

If we choose a high cut-off threshold we restrict the supervised model to return less pairs of high proximity, leading to lower recall but also less comparisons, thus helping to reduce the computational time at the expense of possibly missing some misclassified pairs. Alternatively, if we choose a lower cut-off threshold, we relax our model to return pairs of lower proximity. This leads to more pairs (i.e., more work for further schema matching tasks) yielding positive matches and higher recall of positive cases, but, with more pairs marked incorrectly as matching. We propose how to select an appropriate threshold that optimises this trade-off empirically in Section 5.

The complexity of our approach is quadratic in the number of objects (attributes or datasets) compared, and therefore runs in polynomial time, however, it applies the cheapest computational steps for early-pruning (just computing distances in Equations 3 and 4 and applying the model to score each pair). This way, we save unnecessary expensive schema matching processing per each value instance of the attributes in later steps, reducing the computational workload at the detailed granularity schema matching level by pre-filtering the matching tasks. We demonstrate this empirically in Section 5.5.

5 Experimental Evaluation

In this section, we present the experiments which evaluate our approach by using a prototype implementation. We evaluate the following components of our approach in predicting $Rel(D_y, D_z)$ for pre-filtering schema matching:

- **Proximity metrics:** we evaluate the different individual dataset level and aggregated attribute level meta-features.
- **Supervised models:** we also evaluate the ensemble supervised models, which consume the proximity metrics, in the pre-filtering task.

In addition, we evaluate the sub-components of our approach which include the attribute level models $M_{cls-nom-attr}$ and $M_{cls-num-attr}$ in predicting

$Rel(A_i, A_j)$. We test the attribute level model in experiment 1, the dataset level pruning effectiveness against the ground-truth in experiment 2, and the computational performance in experiment 3.

In experiments 2 and 3, we compare the performance of our proposed proximity mining models against traditional instance-based schema matching techniques. Those are the most expensive techniques which compare values from instances in the datasets to for computing schema similarity. We benchmark our results against a naïve averaging of attribute similarity from a prototype called Probabilistic Alignment of Relations, Instances, and Schema (PARIS), which is one of the most cited schema matching tools [33]. PARIS was found to be best performing with large datasets when compared against other tools [19] and does not need collection of extra metadata (see Table 1). PARIS does exact value-string matching based on value-frequency inverse functionality [33]. We implement a prototype [4] which compares pairs of attributes from different datasets using PARIS and generates an overall score for $Sim(D_y, D_z)$ by averaging $Sim(A_i, A_j)$ generated by PARIS from the top-matching attribute-pairs (similar to Algorithm 1, where PARIS replaces the supervised models). It converts tabular datasets to RDF triples, and executes a probabilistic matching algorithm for identifying overlapping instances and attributes. We selected PARIS because of its simplicity and ease of integration with Java-based APIs and its high performance in previous research [19]. We parametrised the prototype with the top performing settings from experiments in [4] sampling 700 instances per dataset, 10 iterations comparisons, with identity and shingling value strings matching. This will be a baseline pre-filtering heuristic approach we shall compare against in the experiments.

The rest of this section describes the datasets used in the experiments, the evaluation metrics used and the different experiments implemented. We present the results from our experiments and discuss their implications.

5.1 Datasets

We use the OpenML DL[5] in our experiments [36], which has more than 20,000 datasets intended for analytics from different subject areas. OpenML is a web-based data repository that allows data scientists to upload different datasets, which can be used in data mining experiments. OpenML stores datasets in the ARFF tabular format which consist of diverse raw data loaded without any specific integration schema. This allows us to evaluate our approach in a real-life setting where datasets come from heterogeneous domains.

We use two subsets of manually annotated datasets from OpenML as our ground-truth (gold standard) for our experiments. Those two subsets have been generated using two different independent processes, and therefore provide independently generated ground truths that do not overlap. As the research community is lacking appropriate benchmarking gold standards for approximate (non-equijoins) dataset and attribute similarity search [23], we published those

[5] https://www.openml.org

datasets online to support in future benchmarking tasks[6]. The experimental datasets are described as follows:

– **OML01 - The attribute level annotated 15 DS:** consists of 15 datasets from different domains as described in Table 4. The total number of attributes is 126 (61 nominal and 65 numeric), and the average number of attributes per dataset is 8. There is a total of 3468 pairs of attributes to be matched (1575 nominal pairs and 1892 numeric pairs). All the pairs of attributes in this subset were manually scrutinised by 5 annotators consisting of post-graduates with an average age of 28, where 4 are pharmacists and 1 is a computer scientist. They checked the attributes in the datasets and annotated all the pairs of attributes from different datasets with related data, $Rel(A_i, A_j)$. It took on average 3 hours by each annotator to complete the task. Annotators assign a single value from $\{0, 1\}$, where '1' means a related attribute, and the majority vote is taken for each pair, where the average Kappa coefficient for the inter-rater agreement is 0.59, the maximum is 0.80 and the minimum 0.37. Annotators were given the following to judge if the attribute pair is related: attribute name, OpenML dataset description, top 10 values, and the mean and standard deviation for numeric attributes. We didn't give instructions on how to use the provided information to judge, but we described that "related attributes should store data related to similar real-world properties, e.g., car prices, specific body size measurements like height, etc., and should contain similar data". Examples of the annotations can be seen in Table 5. There are only 56 positively matching pairs (19 nominal and 37 numeric). This subset is used in training the attribute level models for computing the similarity between attributes from different datasets and predicting related attributes, $Rel(A_i, A_j)$.

Table 4: Description of the OML01 datasets

Domain	Datasets IDs	Datasets
Vehicles	21,455,967,1092	car,cars,cars,Crash
Business	223,549,841	Stock,strikes,stock
Sports	214	baskball
Health	13,15,37	breast-cancer,breast-w,diabetes
Others	48,50,61,969	tae,tic-tac-toe,Iris,Iris

Table 5: Example Cross-dataset Attribute Relationships from OML01

No.	Dataset 1	Dataset 2	Attribute 1	Attribute 2	Relationship
1	37 (diabetes)	214 (baskball)	age	age	related
2	455 (cars)	549 (strikes)	model.year	year	related
3	455 (cars)	967 (cars)	all	all	duplicate
4	455 (cars)	1092 (Crash)	name	model	related
5	455 (cars)	1092 (Crash)	weight	Wt	related

[6] https://github.com/AymanUPC/all_prox_openml

– **OML02 - The dataset level annotated 203 DS:** consists of 203 datasets different from those in the OML01 subset. To collect this sample, we scraped the OpenML repository to extract all datasets not included in the OML01 sample and having a description of more than 500 characters. Out of the 514 datasets retrieved, we selected 203 with meaningful descriptions (i.e., excluding datasets whose descriptions do not allow to interpret the content and to assign a topic). The datasets have a total of 10,971 attributes (2,834 nominal, 8,137 numeric). There are 19,931 pairs of datasets with about 35 million attribute pairs to match. According to Algorithm 1, there are 3.7 million comparisons for nominal attributes (leading to 59,570 top matching pairs) and 31.5 million numeric attribute pairs (leading to 167,882 top matching pairs). We try to prevent the value-based schema matching on all possible pairs of values between datasets, where there are 216,330 values which would lead to 23.4 billion comparisons at the value level. A domain expert with a background in pharmaceutical studies and one of the authors collaborated to manually label the datasets[7]. They used the textual descriptions of the datasets to extract their topics, which is common experimental practice in dataset matching assessment, similar to the experimental setup in [6]. The annotators sat together in the same room and discussed each dataset with its description and decided on its appropriate real-life subject-area (e.g., car engines, computer hardware, etc.). To group similar datasets in the same subject-area grouping, annotators had to discuss and agree together on a single annotation to give to a dataset. This was done by discussing the specific real-world concept which the dataset describes, e.g., "animal profiles", "motion sensing", etc. The annotators were only allowed to scrutinise the textual descriptions of the datasets and did not receive the underlying data stored in their attributes to prevent any bias towards our proposed algorithms. It took the annotators about 15 hours in total to annotate the datasets. Pairs of datasets falling under the same subject-area were positively annotated for $Rel(D_y, D_z)$. The sample consists of 543 positive pairs from the 20,503 total number of pairs. The details of the sample is summarised in Table 6, which lists the number of datasets, the number of topics, top topics by the number of datasets, and the number of related pairs. Some of the pairs from the sample can be seen in Table 7. We can see, for example, that 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 row 4, we can see an example of duplicated datasets having highly intersecting data in their attributes. Duplicate pairs like those in row 4 have the same number of instances, but described with different number of attributes, which are overlapping. We consider all duplicate pairs of datasets as related pairs. We aim to detect and recommend such kind

[7] Those dataset annotations were reviewed by 5 independent judges, and the results of this validation are published online at:
https://github.com/AymanUPC/all_prox_openml/blob/master/OML02/oml02_revalidation_results.pdf

of similar dataset pairs as those in Table 7 for schema matching using our proximity mining approach.

Table 6: Description of the OML02 datasets

Datasets	**Topics**	**Top Topics**	**Rel($\mathbf{D_y}$, $\mathbf{D_z}$)**
203	74	computer software defects (16), health measurements (13), digit handwriting recognition (12), robot motion sensing (11), plant and fungi measurements (9), citizens census data (8), diseases (8)	543

Table 7: An example of pairs of datasets from the OML02 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

5.2 Evaluation Metrics

We use different evaluation metrics to assess the effectiveness of our approach. We use the traditional recommendation and information retrieval evaluation metrics similar to other research [15,22], including precision, recall and ROC measurements. For the supervised models, we use traditional data mining classification effectiveness metrics [34]. We evaluate the computational costs of our approach vs. traditional schema matching for baseline comparison. Those metrics are categorised per the experiment types and granularities:

- Classification effectiveness
  - Granularity: Attribute level $Rel(A_i, A_j)$ and Dataset level $Rel(D_y, D_z)$
  - Models evaluated: $M_{cls-nom-attr}$, $M_{cls-num-attr}$, M_{cls-ds}
  - Classification measures: Classification accuracy, Recall, Precision, ROC, Kappa
- Pre-filtering (pruning) effectiveness
  - Granularity: Dataset level $Rel(D_y, D_z)$
  - Model evaluated: M_{cls-ds}, $PARIS$
  - Retrieval measures: Recall, Precision, Efficiency Gain, Lift Score
- Computational performance
  - Granularity: Attribute level $Rel(A_i, A_j)$ and Dataset level $Rel(D_y, D_z)$
  - Model evaluated: $M_{cls-nom-attr}$, $M_{cls-num-attr}$, M_{cls-ds}, $PARIS$
  - Computational measures: computational processing time (milliseconds), metadata size (megabytes)

Table 8: The significance of the Kappa statistic

Kappa	Significance
<0	Disagreement
0.0 - 0.10	No significance
0.0 - 0.20	Slight
0.21 - 0.40	Fair
0.41 - 0.60	Moderate
0.61 - 0.80	High
0.81 - 1.0	Excellent

Table 9: The significance of the ROC statistic

ROC	Significance
<0.5	Disagreement
0.5 - 0.6	No significance
0.6 - 0.7	Slight
0.7 - 0.8	Moderate
0.8 - 0.9	High
0.9 - 1.0	Excellent

For the classification effectiveness measures, the classification accuracy is given in our results as a percentage. The recall and precision rate are also percentages. For the ROC (area under the curve) and Kappa statistic, they are a real value between 0 and 1, where the value significance is evaluated in our results according to Tables 8-9.

For the pruning effectiveness measures, we evaluate our approach using the measurements described in Equations (8),(9), (10) and (11). Here, TP means true-positives which are the pairs of datasets correctly classified by the models. FN are false negatives, FP are false-positives, TN are true-negatives, and N indicates the total number of possible pairs of datasets (which is a sum of all pairs TP + FP + TN + FN). The efficiency gain measures the amount of reduction in work required, in terms of number of pairs of datasets eliminated by our models. The lift score measures the capability of the model in filtering out more pairs than randomly removing pairs for the recall rate achieved. A higher amount is better, where a value of 3.0 would mean that the model is capable of retrieving 3 times more positive pairs than the expected amount of positive pairs from a random sample without using the model.

$$recall = \frac{TP}{TP + FN} \quad (8) \qquad precision = \frac{TP}{TP + FP} \quad (9) \qquad efficiency\text{-}gain = \frac{TN + FN}{N} \quad (10)$$

$$lift\text{-}score = \frac{recall}{(1.0 - efficiency\text{-}gain)} \quad (11)$$

5.3 Experiment 1: Attribute-level Models

Our goal in this experiment is to evaluate the supervised models we build for detecting the relationship between related attributes $Rel(A_i, A_j)$ using attribute level content meta-features as follows:

- **Dataset**: OML01
- **Evaluation metrics**: Classification effectiveness
- **Relationship evaluated**: $Rel(A_i, A_j)$
- **Input**: the attribute level meta-features matching for pairs of attributes.
- **Output**: a supervised model to predict related attributes per type.
- **Goal**: select the most appropriate models for predicting related attributes by evaluating their effectiveness for each type (nominal or numerical).

– **Description**: we take two subsets of attribute pairs and their meta-features, depending on the attribute type: nominal attributes and numeric attributes. The subsets are annotated by a human to decide whether $Rel(A_i, A_j)$ is 1 or 0. We build a proximity model using a supervised learning algorithm.

Experimental Setup we evaluate the model using the leave-one-out approach (where we exclude one pair from training in each run, and use it to test the output model, therefore having a cross-validation where the number of folds is equal to the number of pairs). As the number of positive pairs to negative pairs are imbalanced, we create a balanced training set for each type, nominal or numeric, which consists of all the positive pairs of attribute matches and an equal number of negative unmatching pairs. To make the training set representative of all the different negative cases, we cluster the negative cases using the Expectation Maximisation (EM) algorithm [34] and we select a representative sample of negative cases from each cluster.

Results The attribute level models were evaluated for both nominal attribute pairs and numeric attribute pairs. We evaluate the $M_{cls-nom-attr}$ and $M_{cls-num-attr}$ models which assign the $Sim(A_i, A_j)$ for attribute pairs. As could be seen in Table 10, we created two supervised models; one for each type of attribute pairs. Both models achieved excellent ROC performance and highly significant results on the Kappa statistic (see Tables 8-9 results significance). The models had good accuracy, recall, and precision rates. This is important because the dataset pairs pre-filtering step depends on this attribute proximity step, so we have to achieve a good performance at this level to minimise accumulation of errors for the following tasks.

Table 10: Performance evaluation of attribute pairs proximity models

Model	ROC	Kappa	Accuracy	Positive Recall	Positive Precision
Nominal	0.957	0.65	82.5%	89.5%	77.3%
Numeric	0.915	0.7	84.8%	89.2%	80.5%

5.4 Experiment 2: Dataset-level Models

In this experiment, we evaluate the effectiveness of the dataset level models in pre-filtering dataset pairs for further schema matching. Our goal is to evaluate how good is our approach in retrieving related datasets $Rel(D_y, D_z)$ and filtering out unrelated datasets from the schema matching process. We evaluate the effectiveness of correctly proposing related datasets for schema matching using the different types of models we describe later in this section. The evaluation is as follows:

– **Dataset**: OML02

- **Evaluation metrics**: Classification effectiveness, Pre-filtering effectiveness
- **Relationship evaluated**: $Rel(D_y, D_z)$
- **Input**: Pairs of datasets with different types of meta-features matching [dataset level names, dataset level content meta-features, attribute level names, attribute level meta-features, all meta-features].
- **Output**: a supervised model M_{cls-ds} to predict related datasets based on proximity mining and PARIS baseline.
- **Goal**: select the best proximity model to predict related datasets and the proximity threshold c_d to use with that model.
- **Description**: we take annotated pairs of datasets and their meta-features' normalised metrics. The pairs are annotated by a human annotator to decide whether $Rel(D_y, D_z)$ is 1 or 0. We build a proximity model using a supervised learning algorithm.

We create different types of dataset pairs ensemble models to score $Sim(D_y, D_z)$ by using different combination of meta-feature types. We create different models depending on the meta-feature type(s) used as input (namely those are dataset content meta-features, dataset name similarity, attribute content meta-features, and attribute name similarity). We can combine the meta-feature types used to build the model or use each type separately to lead to the following model types depending on which meta-features are used:

- **DS-Prox-Content:** uses the dataset level content meta-features, without considering the dataset name distance.
- **DS-Prox-Name:** uses the dataset name Levenshtein distance as the only predictor of dataset pairs similarity.
- **Attribute-Prox-Content:** uses the attribute level content meta-features, not considering the attribute name meta-features.
- **Attribute-Prox-Name:** uses the attribute level name meta-features only, not considering the attribute content meta-features.
- **Name-Prox:** uses dataset level and attribute level name-based meta-features only.
- **Content-Prox:** uses dataset level and attribute level content-based meta-features only.
- **All-Prox:** uses all the dataset level and attribute level meta-features, including both name-based and content-based meta-features.

We differentiate between the meta-feature types in our experiments so we can test if a specific subset of meta-features is better in predicting $Rel(D_y, D_z)$ or whether it is necessary to use all of them together to build an effective proximity mining model for the pre-filtering task. We also investigate if there is a difference in performance with regards to the types of meta-features extracted: classical name-based meta-features vs. the newly proposed content-based meta-features, and whether using both types together in combination leads to better results. We also separate the types so we can distinguish if purely content-based meta-features can be used as an alternative to name-based meta-features, especially in the case when the datasets and their attributes are not properly named. The most

comprehensive of all models is the *All-Prox* model which uses all the possible meta-features we collect from the data profiling step. The *DS-Prox-Name* is the most generic of all models as it just considers a single meta-feature at the most abstract-level, therefore it will be used as our baseline for our performance comparisons in the experiments.

Experimental Setup To evaluate our approach and models, we consider in our experiments a 10-fold cross-validation experimental setup using different subsets of datasets from a real-world DL. The purpose of a cross-validation setup is to select the best supervised model for the pre-filtering task by evaluating the different models on test-sets separate from the training-sets, which is commonly used in recommendation assessment experiments [2,15] and schema matching tasks [10]. Such an experimental setup increases the validity and generalisability of our experiments and approach. This is only achieved if the training set is representative of the cases found in the real-life population.

We make sure that the folds do not include intersecting dataset pairs. We create a balanced training set with all the positive cases and an equal number of negative cases similar to experiment 1. As the number of negative cases is much higher in the OML02 subset too, we also follow the clustering of negative cases approach to select a representative sample from each cluster (see Section 5.3). We iterate 10 times using an alternating fold as the test set, and the remaining folds as the training set. We evaluate the models in accurately predicting $Rel(D_y, D_z)$ with 9 different cut-off thresholds in [0.1-0.9] for 'c_d' from Equation (1) in order to cover a wide range of values. Finally, we evaluate the performance of each model by averaging the evaluation metrics from all 10 iterations. We also compute standard deviations in performance between different folds to evaluate the stability and consistency of the models evaluated. For the PARIS baseline implementation, it does not need to train any models, so we simply run it on all pairs of datasets and compare its results to our approach.

Results Classification effectiveness. First, we created the models for the dataset pairs which assign $Sim(D_y, D_z)$ and check if they satisfy $Rel(D_y, D_z)$ by passing the minimum threshold. We evaluated the classification effectiveness measures for each proximity model after the 10-fold cross-validation. The results are summarised in Figures 9-11. The figures show a plot of results (from 10 folds) and interquartile ranges of accuracy, kappa statistic and ROC statistic for each model type. The distribution between folds can also be seen to assess the stability of the models. For our comparison, we use the name-based models, which are common in previous research, as our baseline comparison. As could be seen, all models were stable with very close values for the different evaluation metrics, indicating the versatility of our approach. However, still the All-Prox and Attribute-Prox models consistently had slightly better stability (lower deviations) than Name-Prox and other models. It can be seen from the results that the All-Prox and Attribute-Prox models are consistently performing better than the name-based model in terms of accuracy, ROC and Kappa statistic. This indi-

cates that our proposed content-based and name-based combined meta-features models perform best with schema matching pre-filtering.

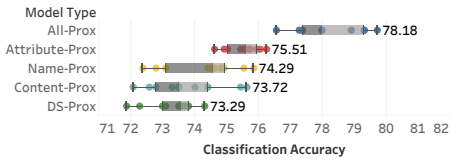

Fig. 9: Classification accuracy from 10-fold cross-validation of dataset pairs pre-filtering models.

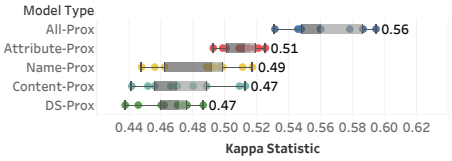

Fig. 10: Kappa statistic from 10-fold cross-validation of dataset pairs pre-filtering models.

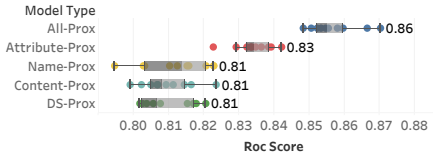

Fig. 11: ROC statistic from 10-fold cross-validation of dataset pairs pre-filtering models.

The different models used for the schema matching pre-filtering task achieve different results because the meta-features used in the different models are not correlated, therefore contain different information about the datasets leading to the different performance of each model. We evaluated the Spearman rank correlation [15] between the different types of meta-features, which is presented in Table 11. The Spearman rank correlation ranks the dataset pairs according to the proximity metrics of the meta-features. If the dataset pairs have the same identical rankings between two different meta-features then we get a perfect correlation. If the rankings produced in descending order by the two proximity metrics are different (e.g., a dataset pair can be ranked in the 100th position by one meta-feature and in the 9th position by the other, which have a difference of 81 ranks) then we get a lower correlation, with completely uncorrelated meta-features. We evaluated the average, standard deviation, minimum, and maximum of the correlation between the meta-features falling under the different types of meta-features. Recall that each type will have multiple meta-features (see Section 4.1), like attribute content will include all the meta-features in Table 3 with all their different proximity metrics according to the aggregations described in Section 4.2. We calculate the correlation between each individual meta-feature pair and we calculate aggregates per type. As can be seen in Table 11, all the correlation values are low.

Table 11: Spearman rank correlation for the different meta-features. We aggregate minimum (Min.), average (Avg.), maximum (Max.), & standard deviation (Std. Dev.) for different meta-feature types.

Type 1	Type 2	Min. Correlation	Avg. Correlation	Max. Correlation	Std. Dev. Correlation
Attribute Name	Attribute Content	-0.12	-0.01	0.19	0.04
Attribute Name	Dataset Content	0.02	0.04	0.10	0.02
Attribute Name	Dataset Name	0.06	0.07	0.13	0.02
Dataset Content	Attribute Content	-0.01	0.09	0.15	0.04
Dataset Content	Dataset Name	-0.02	0.00	0.02	0.02
Dataset Name	Attribute Content	0.00	0.01	0.04	0.01

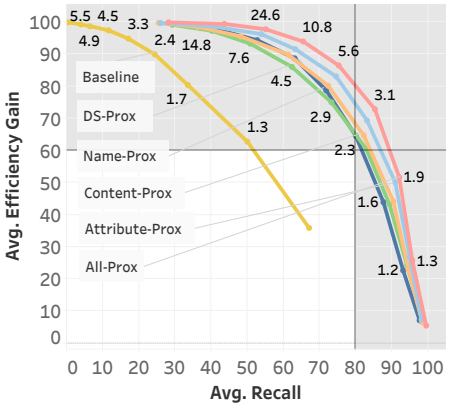

Fig. 12: Recall against efficiency gain for the different supervised models.

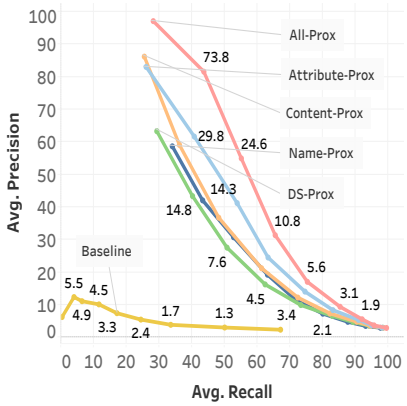

Fig. 13: Recall against precision for the different supervised models.

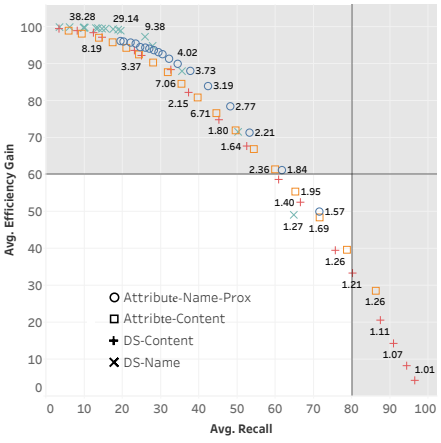

Fig. 14: Recall against efficiency gain for the different metric types.

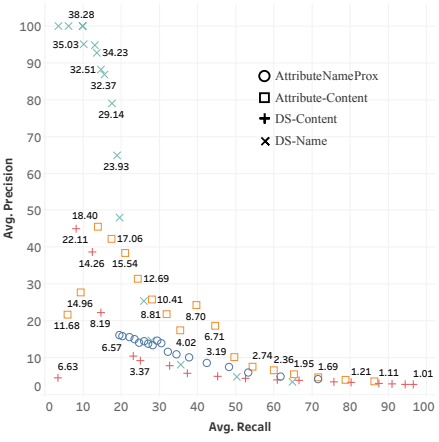

Fig. 15: Recall against precision for the different metric types.

Pre-filtering effectiveness. We compare the effectiveness of our approach against the baseline implementation of PARIS. We change the cut-off thresholds for Equation 1, and we aim to maximize the efficiency-gain while maintaining the highest recall for all candidate dataset pairs satisfying $Rel(D_y, D_z)$. The effectiveness is also evaluated by lift scores. The results from our approach and from the *'baseline'* PARIS prototype are presented in Figures 12-15. Figures 12-13 show the results for the different supervised models and PARIS, and Figures 14-15 show the results of the same evaluation metrics but for the individual meta-features in our approach, where we use the individual proximity metrics of the meta-features directly as an indicator of $Rel(D_y, D_z)$ without using any supervised learning models. We evaluate the dataset level meta-features from Section 4.1. The graphs show the average performance for all the individual metrics per specific type. We use a different minimum threshold with each proximity model or meta-feature in Figures 12-15 leading to the different plotted results per model or meta-feature. The aim of comparing both models and individual metrics is to be able to detect if the proposed supervised proximity models perform any better than simply using single independent metrics for the pre-filtering task.

For each evaluation of the models or the individual metrics, we evaluate the efficiency gain against recall first. We set a minimum target recall of 80% and a minimum target efficiency gain of 60% (i.e., filtering out at least 60% of the pairs of datasets while still proposing 80% of the true positive pairs), which are the grey shaded areas in the graphs. The minimum thresholds can be selected differently according to the requirements of the data analyst. Good performing models or proximity metrics are those that fall in this shaded area. The numbers annotated to some of the points in the graphs indicate the lift score (higher values are better). Similarly, we compare the precision against the recall in the second graphs for each evaluation (models or metrics). We also annotate some selected lift scores for some points in the graph.

When comparing our new proposed proximity models with the proximity model (DS-Prox) from our previous work [5], it can be seen that our Attribute-Prox model and the All-Prox model perform consistently better. This is expected because we are collecting finer granularity metadata to describe the datasets which makes it easier in the supervised learning task to differentiate between positive pairs and negative pairs. Although our new proposed techniques outperform our previous work in the DS-Prox model in terms of recall rates and lift scores, it comes at the price of a more computationally expensive algorithm (Algorithm 1). The complexity of the dataset level Algorithm 2 is $O([n*(n-1)]/2)$ while the complexity of the attribute level Algorithm 1 is $O([n*(n-1)*a^2]/2)$ where 'n' is the number of datasets and 'a' is the number of attributes in each dataset (we can use the average number of attributes per dataset as an approximation for 'a' when estimating the number of computations required).

If we would compare the content meta-features only model (Content-Prox) with the name meta-features only model (Name-Prox), we would see that both models perform equally the same in the pre-filtering task, although combining them in the All-prox model leads to the best results capturing the similarity of

difficult pairs that can not be retrieved by any single type individually. Therefore, it is possible to solely depend on content-based proximity models as a replacement of name-based proximity models to achieve similar results. This will be important in the cases of DLs which are not well maintained and do not have properly named datasets and attributes. We investigate in detail the performance of the All-Prox proximity model based on its true positives, false positives and false negative pairs in the Appendix[8], where we present the exact cases, we discuss the reasons of discrepancies and we give a comparative analysis of the underlying proximity metrics which led to those cases.

For each of the pruning effectiveness evaluation metrics listed above, we compute the average and standard deviation of the measure between the different folds of evaluation for our approach. The average is plotted in the graphs in Figures 12-15, and the standard deviations of each model for the threshold 0.5 (we chose the mean threshold) are given in Table 12. The standard deviation indicates the stability of our proposed metrics and models with different subsets of datasets. We aim for a low standard deviation to prove the high adaptability of our approach.

Table 12: The standard deviation of each evaluation measure for 10-fold cross-validation of each dataset pairs pre-filtering model, where $c_d = 0.5$

Proximity	**SD Recall**	**SD Efficiency Gain**	**SD Precision**	**SD Lift Score**
All-Prox	5.4	0.61	0.58	0.32
Attribute-Prox	6.5	1.0	0.7	0.24
Content-Prox	6.8	1.0	0.38	0.35
DS-Prox	7.86	0.85	0.29	0.28
Name-Prox	6.3	1.1	0.47	0.24

Dataset Pairs Pre-filtering Meta-features First, we assess if the supervised learning models perform better than a simpler approach based on the sub-components they are dependant on, which are the individual meta-features used in the models. The supervised models use multiple features in combination to score the similarity of pairs of datasets. Here, we assess the individual features as a baseline to compare against, and whether simply using a proximity metric of an individual meta-feature without any models can lead to any good result. We aggregated an average for the pruning evaluation metrics per each type of meta-feature. The results comparing recall against efficiency gain is given in Fig. 14. In our experiments, no single meta-feature was able to individually predict related pairs of datasets to achieve optimum recall and efficiency gain, as can be seen by the lack of any plotted result in the top-right box. As seen in Fig. 15, the pre-filtering task using the meta-features can not have a precision better than 10% for the higher recall rates.

[8] The appendix could be found online at https://aymanupc.github.io/all_prox_openml

We note here that the different types of meta-features are able to model different information about the datasets and their attributes as seen by the low correlations in Table 11. That is the main reason we used the combination of different types of meta-features in our proximity models which are able to combine the meta-features to give better results.

Dataset Pairs Pre-filtering Efficiency Gain Vs. Recall We also evaluated the different supervised proximity models by testing their pre-filtering performance with different proximity thresholds. As can be seen in Fig. 12, all of the proximity models were able to optimise recall and efficiency gain to achieve results in the top-right shaded area, compared to the baseline PARIS implementation that was not successful. This shows the value of approximate proximity matching and the supervised models in our approach compared to exact instance-based string matching in the baseline. The best performing models were the All-Prox and Attribute-Prox models which achieved better results than DS-Prox from our previous work [5] and better results than Name-Prox which are more common in other previous research. This means that combining both name-based meta-features and content-based meta-feature in a supervised model achieves best results in the schema matching pre-filtering task. For example, a good result can be achieved using the All-Prox model (combining all meta-feature types) with a threshold of 0.4 which achieves a recall rate of 85%, an efficiency gain of 73% and a lift score of 3.14. This means that the model is able to effectively propose most of the pairs of datasets for schema matching with the least effort possible (only proposing 27% of pairs for comparison), while achieving this with a performance that is three times better than naive random selection of dataset pairs for schema matching (as expressed by the lift score of 3.14 achieved by the All-Prox model).

Dataset Pairs Pre-filtering Precision Vs. Recall As seen in Fig. 13, the precision of the proximity models improved the performance of the schema matching pre-filtering as seen by the higher precision rates compared to the individual meta-features in Fig. 15. By combining the meta-features in a supervised model we were able to achieve higher precision rates with the same recall rates, for example, a precision of 17% with a recall rate of 75% using the All-Prox model. This is better than the best achievable precision with the individual meta-features, which can achieve a precision of 4% with the same recall rate for the attribute level meta-feature type. However, we acknowledge that the precision rates are low for all types of models and meta-features. We can therefore conclude that our proposed proximity mining approach can only be used as an initial schema matching pre-filter which is able to prune unnecessary schema matching comparisons from further steps. Our approach can not be used for the final schema matching task because it will produce false positives. Therefore, dataset pairs should be further scrutinised with more comparisons to assess their schema similarity (as seen in Fig. 1 bottom instance-based matching

layer). Such comparisons use instance-based matching similar to our previous work [4].

5.5 Experiment 3: Computational Performance Evaluation

In this experiment, we evaluate the computational performance in terms of time and storage space consumption as follows:

- **Dataset**: OML02
- **Evaluation metrics**: Computational performance
- **Relationship evaluated**: all
- **Input**: All the pairs of datasets from OML02, a model (M_{cls-ds}) and the $PARIS$ schema matching prototype.
- **Output**: attribute-level and dataset-level metadata.
- **Goal**: test the comparable computational costs of running the different components of our proximity mining approach vs. traditional instance-based schema matching techniques. We show the value of pre-filtering by means of computational costs saving.
- **Description**: we take all the annotated pairs from OML02 and we do a complete run which collects the required meta-features and metrics, and we run the algorithms to compute $Sim(D_y, D_z)$. We measure the amount of time and storage space it takes to process the pairs.

We ran the experiments for our approach using a computer running on Linux Debian, 8GB main memory, a dual-core Intel i7 processor running at 2.4GHz and 4MB cache, Java v8 for the implementation of our algorithms, and Postgres database v9.5.12 for the metadata storage and management. For the PARIS baseline implementation, we used a server with more resources as recommended by the developers. The server runs on Linux Debian, Java v8, 24GB of memory and a quad-core processor at 2.4 GhZ and 4MB cache. We present the results below.

Results We compare the computational performance by evaluating the amount of time and storage space for running our approach and the PARIS-based implementation with the DL sample OML02. The results can be seen in Table 13. We list the tasks from our approach and compare to the baseline in the last row. We compute the time for each task, the average time it takes, and the storage space used. For the attribute matching, we keep the output in memory and do not materialise it. We only materialise top-matching attribute pairs.

Based on the results in Table 13, our approach needs a total of 112 minutes and 100MB storage space for the OML02 DL sample datasets of a total size of 2.1GB (i.e., 5% metadata space overhead). This is at least 2 orders of magnitude less than the time and space consumption of the baseline PARIS implementation. The most expensive steps in our approach were those for the numeric matching tasks as they were much greater in amount than nominal attributes. Still, our approach is more efficient in terms of computational performance and pre-filtering effectiveness as shown by our results.

Table 13: The computational performance of our approach vs. the PARIS implementation in terms of time and storage space

Task	Timing	Average Time	Storage Space
Dataset Profiling	263,019ms (4:23 minutes)	1,295ms per dataset	31.25MB
Numeric Attribute Matching	1,184,000ms (19:44 minutes)	0.04ms per attribute pair	In memory
Nominal Attribute Matching	160,000ms (2:40 minutes)	0.04ms per attribute pair	In memory
Numeric Attribute Top Matching	3,250,000ms (54:10 minutes)	0.1ms per attribute pair 208ms per dataset pair (15,576 dataset pairs)	7MB
Nominal Attribute Top Matching	313,000ms (5:13 minutes)	0.08ms per attribute pair 19ms per dataset pair (16,290 dataset pairs)	2.33MB
Dataset-level All Aggregations of Attribute Similarities	500,000ms (8:20 minutes)	25ms per dataset pair (19,931 dataset pairs)	35MB
Dataset-level Name Matching	202ms (0 minutes)	0.01ms per dataset pair (20,503 pairs)	Part of Top Matching metadata
Dataset-level Content Matching	5,100ms (5.1 seconds)	0.25ms per dataset pair (20,503 pairs)	3.25MB
Attribute-level Name Matching, top pairs computation, and aggregation	1,018,663ms (16:58 minutes)	0.03ms per attribute pair (35,283,824 attribute pair) 51ms per dataset pair (19,931 dataset pair)	12.5MB
Apply the proximity models on the dataset pairs to score their similarities	1,665ms (1.66 seconds)	0.08ms per dataset pair (20,503 dataset pair)	8.5MB
PARIS Alignment Implementation	743,077,431ms (12,384:37 minutes)	36,241ms per dataset pair (0:36 minutes per dataset pair)	15,450MB (15.1GB)

5.6 Generalisability

In our experiments, we have used the OpenML DL to create a 10-fold cross-validation experimental setup. OpenML stores datasets representing heterogeneous subject-areas. Thus, we expect our proposed techniques to achieve similar results with different heterogeneous DLs. We tested our approach with different heterogeneous DL subsets covering randomly selected subject-areas in each cross-validation fold. This further improves the generalisability of our results as the results achieved proved to be stable between the different cross-validation folds. Therefore, our approach is recommended in the early-pruning and schema matching pre-filtering task in a DL environment with heterogeneous subject-areas. Under different settings, the data scientist should first test the performance of our approach on a test sample and then select the best performing cut-off thresholds accordingly. It is also crucial that the training samples selected for creating the supervised models are representative of the specific DL setting they are used for. We also note, that although our experiments were done over binary approximation for the $Rel(D_y, D_z)$ function in the ground truth due to the difficulty to find a ground truth with a similarity continuum, still our approach can be useful in dataset pairs ranking problems using the $Sim(D_y, D_z)$ continuous function.

6 Conclusion

We have presented in this paper a novel approach for pre-filtering schema matching using metadata-based proximity mining algorithms. The approach is able to

detect related dataset pairs containing similar data by analysing their metadata and using a supervised learning model to compute their proximity score. Those pairs exceeding a minimum threshold are proposed for more detailed, more expensive schema matching at the value-based granularity-level. Our approach was found to be highly effective in this early-pruning task, whereby dissimilar datasets were effectively filtered out and datasets with similar data were effectively detected in a real-life DL setting. Our approach achieves high lift scores and efficiency gain in the pre-filtering task, while maintaining a high recall rate. For future research, we will investigate the different techniques to improve the scalability of our approach by improving attribute level matching selectivity. We also want to investigate the possibility of detailed semantic schema matching at the attribute level. We will also investigate our proximity mining approach in effectively clustering the datasets into meaningful groupings of similarity.

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

References

1. Abedjan, Z., Golab, L., Naumann, F.: Profiling relational data: a survey. The VLDB Journal **24**(4), 557–581 (2015). https://doi.org/10.1007/s00778-015-0389-y,
2. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems (TOIS) **23**(1), 103–145 (2005). https://doi.org/10.1145/1055709.1055714
3. Algergawy, A., Massmann, S., Rahm, E.: A Clustering-Based Approach for Large-Scale Ontology Matching. In: East European Conference on Advances in Databases and Information Systems (ADBIS), pp. 415–428. Springer (2011).
4. Alserafi, A., Abelló, A., Romero, O., Calders, T.: Towards Information Profiling: Data Lake Content Metadata Management. In: DINA Workshop, ICDM. pp. 178–185. IEEE (2016). https://doi.org/10.1109/ICDMW.2016.0033
5. Alserafi, A., Calders, T., Abelló, A., Romero, O.: DS-prox: Dataset proximity mining for governing the data lake. In: International Conference on Similarity Search and Applications. vol. 10609 LNCS, pp. 284–299. Springer (2017). https://doi.org/10.1007/978-3-319-68474-1_20
6. Ben Ellefi, M., Bellahsene, Z., Dietze, S., Todorov, K.: Dataset Recommendation for Data Linking: An Intensional Approach. In: Proceedings of the International Semantic Web Conference: The Semantic Web. Latest Advances and New Domains. vol. 9678, pp. 36–51. Springer (2016). `http://link.springer.com/10.1007/978-3-319-34129-3`
7. Bernstein, P.A., Madhavan, J., Rahm, E.: Generic Schema Matching , Ten Years Later. Proceedings of the VLDB Endowment **4**(11), 695–701 (2011)
8. Bilke, A., Naumann, F.: Schema Matching using Duplicates. In: Proceedings of the 21st International Conference on Data Engineering. pp. 69–80. IEEE (2005)
9. Breiman, L.: Random Forests. Machine Learning **45**(1), 5–32 (2001)
10. Chen, C., Halevy, A., Tan, W.c.: BigGorilla : An Open-Source Ecosystem for Data Preparation and Integration. IEEE Data Engineering Bulletin **41**(2), 10–22 (2018)

11. Chen, Z., Jia, H., Heflin, J., Davison, B.D.: Generating Schema Labels through Dataset Content Analysis. In: Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18. pp. 1515–1522 (2018). https://doi.org/10.1145/3184558.3191601
12. Deng, D., Kim, A., Madden, S., Stonebraker, M.: SilkMoth: An Efficient Method for Finding Related Sets with Maximum Matching Constraints. Proceedings of the VLDB Endowment **10**(10), 1082–1093 (2017)
13. Furche, T., Gottlob, G., Libkin, L., Orsi, G., Paton, N.W.: Data wrangling for big data: Challenges and opportunities. In: EDBT. vol. 16, pp. 473–478 (2016)
14. Gallinucci, E., Golfarelli, M., Rizzi, S.: Schema profiling of document-oriented databases. Information Systems **75**, 13–25 (2018). https://doi.org/10.1016/j.is.2018.02.007
15. Herlocker, J., Konstan, J.A., Terveen, L.G., Riedel, J.T.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems (TOIS) **22**(1), 5–53 (2004)
16. Kandel, S., Heer, J., Plaisant, C., Kennedy, J., Van Ham, F., Riche, N.H., Weaver, C., Lee, B., Brodbeck, D., Buono, P.: Research directions in data wrangling: Visualizations and transformations for usable and credible data. Information Visualization **10**(4), 271–288 (2011)
17. Kim, J., Peng, Y., Ivezic, N., Shin, J.: An Optimization Approach for Semantic-based XML Schema Matching. International Journal of Trade, Economics and Finance **2**(1), 78 – 86 (2011)
18. Kruse, S., Papenbrock, T., Harmouch, H., Naumann, F.: Data Anamnesis : Admitting Raw Data into an Organization. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering pp. 8–20 (2016)
19. Lacoste-Julien, S., Palla, K., Davies, A., Kasneci, G., Graepel, T., Ghahramani, Z.: SiGMa: Simple Greedy Matching for Aligning Large Knowledge Bases. In: Proceedings of the 19th ACM SIGKDD international conference. pp. 572–580 (2013). https://doi.org/10.1145/2487575.2487592
20. Maccioni, A., Torlone, R.: KAYAK: A Framework for Just-in-Time Data Preparation in a Data Lake. In: International Conference on Advanced Information Systems Engineering. pp. 474–489. Springer International Publishing (2018). https://doi.org/10.1007/978-3-319-91563-0
21. Madhavan, J., Bernstein, P.a., Rahm, E.: Generic Schema Matching with Cupid. VLDB **1**, 49–58 (2001)
22. Manning, C.D., Raghavan, P., Schütze, H.: An Introduction to Information Retrieval. No. c (2009)
23. Miller, R.: Open Data Integration. PVLDB **11**(12), 2130–2139 (2018)
24. Naumann, F.: Data profiling revisited. ACM SIGMOD Record **42**(4), 40–49 (2014)
25. Oliveira, A., Tessarolli, G., Ghiotto, G., Pinto, B., Campello, F., Marques, M., Oliveira, C., Rodrigues, I., Kalinowski, M., Souza, U., Murta, L., Braganholo, V.: An efficient similarity-based approach for comparing XML documents. Information Systems **78**, 40–57 (2018). https://doi.org/10.1016/j.is.2018.07.001
26. de Oliveira, H.R., Tavares, A.T., Lóscio, B.F.: Feedback-based data set recommendation for building linked data applications. In: Proceedings of the 8th International Conference on Semantic Systems - I-SEMANTICS '12. p. 49. ACM (2012)
27. Pei, J., Hong, J., Bell, D.: A novel clustering-based approach to schema matching. In: Proceedings of the international conference on Advances in Information Systems. pp. 60–69. Springer (2006)
28. Rahm, E.: Towards large-scale schema and ontology matching. In: Schema matching and mapping, pp. 3–27. Springer Berlin Heidelberg (2011)

29. Rahm, E.: The Case for Holistic Data Integration. In: ADBIS. pp. 11–27 (2016). https://doi.org/10.1007/978-3-319-44039-2
30. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal **10**(4), 334–350 (2001). https://doi.org/10.1007/s007780100057
31. Shvaiko, P.: A Survey of Schema-based Matching Approaches. Journal on Data Semantics **3730**, 146–171 (2005).
32. Steorts, R., Ventura, S., Sadinle, M., Fienberg, S.: A Comparison of Blocking Methods for Record Linkage. In: International Conference on Privacy in Statistical Databases. pp. 253–268 (2014)
33. Suchanek, F.M., Abiteboul, S., Senellart, P.: PARIS : Probabilistic Alignment of Relations , Instances , and Schema. Proceedings of the VLDB Endowment **5**(3), 157–168 (2011). https://doi.org/10.14778/2078331.2078332
34. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to data mining. Pearson Education (2006)
35. Terrizzano, I., Schwarz, P., Roth, M., Colino, J.E.: Data Wrangling: The Challenging Journey from the Wild to the Lake. In: 7th Biennial Conference on Innovative Data Systems Research CIDR'15 (2015)
36. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. ACM SIGKDD Explorations Newsletter **15**(2), 49–60 (2014)