Short Paper

Towards Scalable Data Discovery

Javier Flores, Sergi Nadal, Oscar Romero
Universitat Politècnica de Catalunya
Barcelona, Spain
jflores|snadal|oromero@essi.upc.edu

ABSTRACT

We study the problem of discovering joinable datasets at scale. We approach the problem from a learning perspective relying on profiles. These are succinct representations that capture the underlying characteristics of the schemata and data values of datasets, which can be efficiently extracted in a distributed and parallel fashion. Profiles are then compared, to predict the quality of a join operation among a pair of attributes from different datasets. In contrast to the state-of-the-art, we define a novel notion of join quality that relies on a metric considering both the containment and cardinality proportion between join candidate attributes. We implement our approach in a system called Nextia$_{JD}$, and present experiments to show the predictive performance and computational efficiency of our method. Our experiments show that Nextia$_{JD}$ obtains similar predictive performance to that of hash-based methods, yet we are able to scale-up to larger volumes of data. Also, Nextia$_{JD}$ generates a considerably less amount of false positives, which is a desirable feature at scale.

1 INTRODUCTION

Data discovery requires to identify interesting or relevant datasets that enable informed data analysis [2, 9]. Discovery and integration of datasets is nowadays a largely manual and arduous task that consumes up to 80% of a data scientists' time [19]. This only gets aggravated by the proliferation of large repositories of heterogeneous data, such as *data lakes* [15] or open data-related initiatives [14]. Due to the unprecedented web-scale volumes of heterogeneous data sources, manual data discovery becomes an unfeasible task that calls for automation [11]. Hence, we focus on the very first task of data discovery: the problem of discovering joinable attributes among structured datasets in a data lake. We distinguish three approaches: *comparison by value*, *comparison by hash* and *comparison by profile*. Table 1, overviews recent contributions. Comparison by value relies on auxiliary data structures such as inverted indices or dictionaries to minimize the lookup cost. Alternatively, the comparison by hash approach expects that similar values will collision in the same bucket, also employing index structures for efficient threshold index. Comparison by profile methods leverage on profiles extracted from datasets and their attributes. These are compared to predict whether a given pair of attributes will join.

Search accuracy		
Exact ..... Approximate		
Comp. by value [6, 20, 22]	Comp. by hash [4, 10, 21, 23]	Comp. by profile [5, 7, 8, 12]
Expensive ..... Efficient		
Algorithmic complexity		

Table 1: Overview of approaches by technique, arranged according to accuracy and algorithmic complexity

1.1 Data discovery at scale

Unfortunately, as we experimentally show in Section 3, the state-of-the-art in data discovery does not meet the expectations for web-scale scenarios. Unlike traditional relational databases, these are characterized by *a)* a wide heterogeneity among datasets (e.g., large differences on the number of attributes and / or their cardinalities); *b)* massive volumes of data; and *c)* the presence of a variety of topics, or domains. Overall, these distinguishing features deem current solutions ineffective due to their inability to scale-up as well as the low precision of the results obtained.

Inability to scale-up. Solutions that yield exact results or with a bounded error (i.e., comparison by value and hash) require the construction and maintenance of index structures for efficient lookup. This is a task that becomes highly demanding in terms of computing resources on large-scale datasets. In fact, as we have empirically observed, the available implementations fail to handle datasets of few GBs. Furthermore, most available approaches do not allow incremental maintenance.

Low precision. Comparison by hash solutions employ either containment or Jaccard distance as similarity measures to decide joinability among pairs of attributes. It has been reported, however, that the estimation of such measures is highly imprecise when the cardinality (i.e., the number of distinct values) of an attribute is comparatively larger than the other's [16], which is a common characteristic in real-world web-scale applications. As result, the precision of current approaches is highly affected due to the large number of false positives. To showcase this fact, we designed an experiment collecting 138 datasets from open repositories such as Kaggle and OpenML[1]. Precisely, we devised an heterogeneous collection of datasets ranging different topics, which yielded a total of 110, 378 candidate pairs of string attributes, where 4, 404 of those have a containment higher or

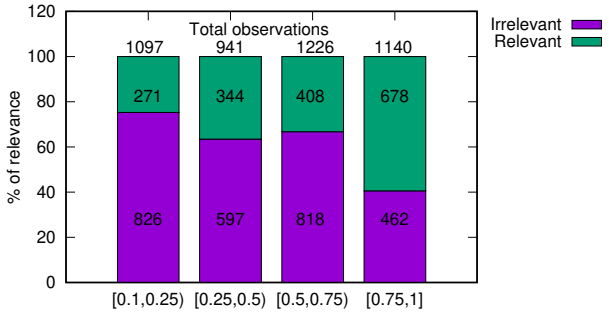

Figure 1: Distribution of relevant and irrelevant results for different containment values on a web-scale repository

[1]Repository available at https://mydisk.cs.upc.edu/s/GeYwdYH7xsGqbaX

equal than 0.1. Indeed, as shown in Figure 1, even for very high containment values (i.e., above 0.75), the number of irrelevant results (i.e., false positives) represents 40% of the total. Such results were obtained by manually analyzing the proposed candidate pairs. However, this approach is unfeasible for large scenarios and better join metrics are needed.

1.2 Profile-based methods to the rescue

The above discussion highlights the limitations of value and hash-based data discovery over web-scale scenarios. Alternatively, the comparison by profile approach suits betters for large scale scenarios as they rely on the detection of similarities or discrepancies between profiles. Working with summaries instead of data values is much more efficient from a complexity point of view. Yet, despite the clear performance benefits of profile-based approaches, there is nowadays a large gap in the trade-off regarding the quality of their results mainly due to the adoption of rather basic profiles (e.g. [8]) that do not accurately describe the underlying data or representative profiles (e.g. [5]) that are used to discover a binary class (e.g. joinable or non-joinable). To that end, we propose a novel approach to data discovery which aims to cover the gap generated by the low predictive performance of profile-based methods, as well as the limited precision and scalability of hash-based systems on large data lakes.

We, first, propose a novel metric to denote the quality of a join. Opposite to the related work, mostly focused on containment or Jaccard distance, we also consider the cardinality proportion between attributes as an indicator of a higher join quality. This allows us to get rid of a substantial amount of false positives, reducing the number of pairs to analyze. This is specially relevant in large-scale settings, where as shown in Figure 1, the number of candidate pairs is too large to manually disregard false positives. Second, we propose a novel learning-based method based on profiles to discover joinable attributes for large-scale data lakes. Our assumptions apply to scenarios where data is typically denormalized and file formats embed tabular data (i.e., not nested). We rely on state-of-the-art relational data profiling techniques [1] to compute informative profiles for datasets. This task, which can be done offline and parallelized over distributed computing frameworks (e.g., Apache Spark), allows us to extract and model the underlying characteristics of attributes. Next, profiles are compared in order to predict their expected join quality. The predictive model is based on random forest classifiers, which are highly expressive and robust to outliers and noise [3]. Additionally, such models can be trained and evaluated in a distributed fashion [17], thus yielding a fully distributed end-to-end framework for data discovery. We show that our method is generalizable and that proposes a meaningful ranking of pairs of attributes based on the predicted join quality.

Contributions. We summarize our contributions as follows:

- We introduce a qualitative metric for join quality, which considers containment and cardinality proportion between attributes.
- We learn a model based on random forest classifiers to efficiently rank candidate pairs of joinable attributes.
- We show that our approach is scalable and outperforms the current state of the art, yielding higher predictive performance results than profile-based solutions and similar quality (F_1-score) to hash-based ones. Yet, our approach yields better precision than hash-based approaches and produce less false positives.

2 MEASURING THE QUALITY OF A JOIN

Unlike the state-of-the-art, which mainly uses containment and Jaccard similarities to decide the degree of joinability among pairs of attributes, we define a qualitative metric to measure the expected join quality. We consider containment as a desirable metric to maximize. Yet, we make the observation that datasets on a data lake do not relate to each other as in a relational database. In such scenarios, it is common to find datasets with few data values in common that, in turn, may represent different semantic concepts. In order to exemplify this idea, let us consider the datasets depicted in Table 2. In this example, the reference dataset D_{ref} might be joined with any of the two candidate datasets D_1 (at the EU level) and D_2 (worldwide). Current approaches would propose both as joinable pairs, since they yield the same containment. However, we aim at distinguishing the join quality between them and use their *cardinality proportion* for that purpose. Let us consider the following cardinalities corresponding to the city attributes: $|D_{ref}|$ = 8124, $|D_1|$ = 54500 and $|D_2|$ = 982921. We use the cardinality proportion as a measure to infer whether their data granularities are similar. In this sense, the third dataset is much larger than $|D_{ref}|$ and yield a worse proportion and therefore we rank it worse. Importantly, we assume these datasets store independently generated events and such big difference in their cardinality most probably mean they embed different semantics or sit at different granularity levels. In general, such situations are a source of false positives for current solutions, specially, when considering small tables.

2.1 Join quality

We now formalize the metric for join quality as a rule-based measure combining both containment and cardinality proportion. We define a totally-ordered set of quality classes S = {`None`, `Poor`, `Moderate`, `Good`, `High`} as indicator of the quality of the resulting join. Indeed, we advocate not to define a binary class (i.e., either joinable or not), since we would not be able to rank the positive ones. As experienced with the state-of-the-art, in realistic large scenarios, a binary class yields a long list of results, which

(a) D_{ref} – Tourism income in Spain

City	Seaside	Amount
Barcelona	Y	350M
Girona	Y	110M
Lleida	N	75M
Tarragona	Y	83M
…	…	…

(b) D_1 – EU demographic data

Unit	Population	Avg. salary	Cost of living
Antwerp	1,120,000	44,000€	2,896€
Barcelona	1,620,343	31,000€	2,422€
Berlin	4,725,000	49,000€	2,737€
Bristol	1,157,937	30,000£	2,397£
…	…	…	…

(c) D_2 – Worldwide demographic data

Name	Country	Population
Barcelona	Spain	1,620,343
Canberra	Australia	426,704
Chicago	United States	2,695,598
Curitiba	Brasil	1,908,359
…	…	…

Table 2: A reference dataset (D_{ref}) and two candidate datasets to be joined. D_1 is curated with extensive data at european level, while D_2 is curated at the worldwide level with less details

are difficult to explore and compare. Therefore, we propose the following multi-class join quality metric.

Definition 2.1. Let A, B be sets of values, respectively the *reference* and *candidate* attributes. The join quality among A and B is defined by the expression

$$Quality(A,B) = \begin{cases} (4)\ \texttt{High}, & C(A,B) \geq C_H \wedge \frac{|A|}{|B|} \geq K_H \\ (3)\ \texttt{Good}, & C(A,B) \geq C_G \wedge \frac{|A|}{|B|} \geq K_G \\ (2)\ \texttt{Moderate}, & C(A,B) \geq C_M \wedge \frac{|A|}{|B|} \geq K_M \\ (1)\ \texttt{Poor}, & C(A,B) \geq C_P \\ (0)\ \texttt{None}, & \text{otherwise} \end{cases}$$

The rationale behind the quality metric is to constrain the candidate pairs of attributes to two thresholds per class: containment (C_i) and cardinality proportion (K_i). Precisely, we fix that for any pair of classes $S_i, S_j \in S$ where $S_i > S_j$, the containment and cardinality proportion must be higher (i.e., $C_H > C_G > C_M > C_P$ and $K_H > K_G > K_M$). Intuitively, a larger containment and a similar cardinality proportion guarantees that the two attributes share common values and their cardinalities are alike. Consequently, most probably, they have a semantic relationship. We consider the values $C_H = 3/4 = 0.75, C_G = 2/4 = 0.5, C_M = 1/4 = 0.25, C_P = 0.1$ for containment, and $K_H = 1/4 = 0.25, K_G = 1/8 = 0.125, K_M = 1/12 = 0.083$ for cardinality proportion. These have been empirically defined from our training set, yet, as we show in Section 3 they are generalizable to other datasets.

To demonstrate the benefits of the proposed metric, we ran an experiment following the same methodology as that depicted in Section 1.1. Using the same collection of 110,378 candidate pairs, we evaluated their join quality. As a result, in Figure 2 we depict the distribution of the join quality distinguishing relevant and irrelevant results. There are two key observations to be made. On the one hand, the number of results labeled with higher quality classes is considerably smaller than those for high containment values. Thus, the largest number of observations are labeled as of *Poor* quality or *None*. On the other hand, the proportion of relevant cases is, in general, much larger than irrelevant ones but specially significative for higher quality classes. As expected, the lower the quality class, the more irrelevant cases will be found. Drilling down in the obtained results, we have manually studied these irrelevant cases where the quality class is High. We have observed that these situations occur when the proposed pairs do not have a semantic relationship but they share a syntactic one (e.g., some artists use a country name). Thus, it would only be possible to disregard them considering semantics.

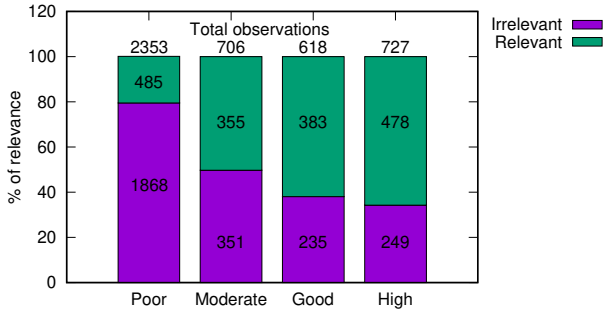

Figure 2: Distribution of relevant and irrelevant results for different quality classes on a web-scale repository

2.2 A learning approach to join discovery

Here, we describe our approach and present the process of building profiles and the predictive model.

Attribute profiling. Profiles are composed of meta-features that represent the underlying characteristics of attributes. Such profiles are the key ingredient for high accuracy predictions, thus we require an exhaustive summary of attributes. To this end, we base our profiling on state-of-the-art relational data profiling techniques [1]. We distinguish meta-features corresponding to unary and binary profiles. We further distinguish the former into meta-features modeling cardinalities, value distribution and syntax. Before comparing profiles and due to the fact attribute meta-features are represented in different magnitudes, we normalize them to guarantee a meaningful comparison using the Z-score. Finally, once meta-features have been normalized we compute the distances among pairs of attributes. Here, we also compute binary meta-features. The result of this stage is a set of distance vectors D where, for each D_i, values closer to 0 denote high similarities.

Predictive model. Once distance vectors are computed, we can train the predictive model. Precisely, the goal is to train a model that, for a pair of attributes A, B, its prediction is highly correlated to the true class (i.e., $Quality(A,B)$). The training process was performed using the collection of datasets discussed in Section 1.1. The ground truth was labeled using the newly proposed quality metric (Definition 2.1), which served as training dataset for the random forest classifiers. In order to reduce the false positive rate, the different classifiers are connected in a classifier chain architecture. This is an effective approach for multi-label classification [18]. Each classifier predicting the probability of class i is trained with the set of distance vectors D and the probabilities of classes $0, \ldots, i-1$, improving the predictive accuracy of the classifier. Figure 3 depicts a high-level overview of the architecture used for training. Then, the prediction returned by the classifier is the one with highest probability from each RF_i. To assign the predicted quality class to a candidate attribute, we assign the label considering the highest probabilities from all classifiers.

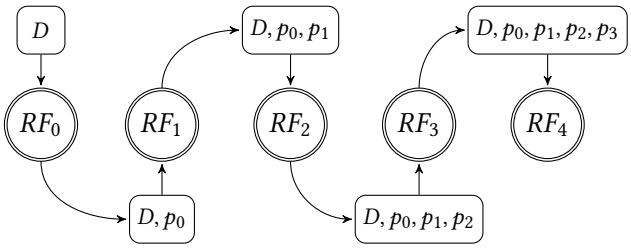

Figure 3: Chain of 5 random forest classifiers, each RF_i is fed with the distances vectors D and the probabilities from the previous classifiers $p_0, \ldots, p_{i-1}$

3 EVALUATION

In this section, we present the evaluation of our approach. On the one hand, we evaluate the ability of the model to discover quality joins through several experiments as well as its generalizability. On the other hand, we compare its performance with representative state-of-the-art solutions. In order to present transparent experiments and guarantee the reproducibility of results, we created an informative companion website[2].

[2]https://www.essi.upc.edu/dtim/nextiajd/

Implementation. Nextia$_{JD}$ is implemented as an extension of Apache Spark. The classification model was trained using its distributed machine learning library MLlib. The runtime methods (i.e., profiling and ranking) are implemented as new operators over the structured data processing library SparkSQL. We leverage on the Catalyst optimizer to efficiently compute the profiles and compare them. Our implementation supports two modes of operation *discovery-by-attribute* and *discovery-by-dataset*. The former receives as input a reference Spark dataframe (i.e., a dataset) and one of its attributes, and generates a ranking against a collection of dataframes (i.e., other datasets). The *discovery-by-dataset* mode does not receive as input a reference attribute, so it runs the discovery process for all attributes from the reference dataframe. Notably, implementing Nextia$_{JD}$ on top of Spark brings many other benefits. Firstly, we can benefit from many source connectors and we can easily ingest the most common data formats (e.g., CSV, JSON, XML, Parquet, Avro, etc.). Secondly, our extension benefits from the inherent capacity of Spark to parallelize tasks on top of distributed data.

Test set. For evaluation purposes, we collected 139 independent datasets from those used for the ground truth. We further divided such datasets into 4 testbeds (extra-small, small, medium and large) according to their file size. Table 3 shows the characteristics of each testbed.

Testbed	***XS***	***S***	***M***	***L***
File size	$0-1$ MB	$1-100$ MB	100 MB -1 GB	>1 GB
Datasets	28	46	46	19
Attributes	159	590	600	331

Table 3: Characteristics per testbed

Alternatives. We compare our approach with the following state-of-the-art data discovery solutions representatives of, respectively, hash-based and profile-based methods whose source code is openly available: LSH Ensemble [23] and FlexMatcher [5]. No fine tuning was performed in such systems, running the code as provided out-of-the-box. These systems differ in their mode of operation, the former being an approach based on comparison by hash, while the latter on comparison by profile. Note that, for computational performance reasons, we rule out approaches based on comparison by value. Such solutions are not comparable to ours, since their kind of search accuracy is exact and by nature they are not suitable for large-scale scenarios.

3.1 Predictive performance

Here we assess the classifier's predictive performance evaluating the ranking of candidate equi-join predicates for each testbed.

Methodology. We depict a confusion matrix to capture the relationship between the true and predicted classes. We also provide performance metrics for the classifier such as precision, recall and F_1 score. We first discuss the experiment for all testbeds together, and later do a fine-grained discussion for each testbed.

Results. Figure 4 and Table 4, show, respectively, the confusion matrix and performance metrics for all testbeds. Overall, we evaluated 467,965 attributes pairs. We can validate the good performance of the proposed approach by the fact that class 4, denoting the highest quality joins, has the best precision. Our method aims at proposing a ranking according to the predicted join quality, which for the highest value it has almost no false positives. It is also relevant to note that the prediction for class 0, denoting the no join quality, also has both high precision and recall. This is particularly relevant to filter out irrelevant results, and thus reduce the search space when presenting a ranking to the user. We additionally note that, as depicted by the precision and recall measures, predictions corresponding to classes 1 and 2 are highly inaccurate. Nevertheless, Nextia$_{JD}$ is prepared to be used as an interactive tool. Thus, if we analyze these results from the point of view of a user, most misclassifications are between similar classes and thus irrelevant. Nextia$_{JD}$ shows the results in strict order. First, classes 4 and 3, and then classes 2 and 1 on demand. In this sense, a binary classification meaning *likely relevant* or *likely irrelevant* would be a fairer way to evaluate Nextia$_{JD}$. When considering these results as a binary problem (relevant: classes 3-4; irrelevant: classes 0-2), the evaluated metrics improve considerably, as shown in Figure 9 and Table 5. Relevantly, we note that Nextia$_{JD}$ generates very few false positives; a desirable property for large-scale data discovery problems. We nevertheless highlight the relevance of distinguishing classes 1 and 2 from 0, either for advanced users or because Nextia$_{JD}$ could be used for other automatic data discovery problems where such distinction would be relevant.

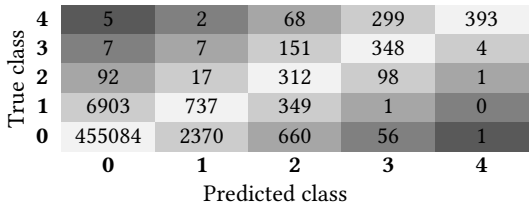

Figure 4: Confusion matrix for all testbeds (clearer cells denote a closer proximity w.r.t. the true class)

	Precision	**Recall**	**F_1 score**
(0) None	0.9848	0.9933	0.9890
(1) Poor	0.2352	0.0922	0.1325
(2) Moderate	0.2025	0.6000	0.3029
(3) Good	0.4339	0.6731	0.5276
(4) High	0.9849	0.5123	0.6740

Table 4: Performance metrics per class for all testbeds

3.2 Comparison with the state-of-the-art

In this experiment we aim at comparing our approach to other data discovery approaches. We perform such evaluation by measuring and comparing their computational complexity and predictive performance.

Methodology. All systems under evaluation, including ours, implement data discovery in two steps. The first step, which we denote *pre*, builds the core data structures from the datasets. For hash-based methods, such as LSH Ensemble, building the index, while profile-based methods, such as FlexMatcher and ours, create the profiles. Additionally, FlexMatcher will create the predictive models for each new data discovery task. Then, the second step, which we denote as *query*, consists of computing the prediction leveraging on the previously built data structures. Whenever possible, we decouple both steps and thus read the data structures from disk. This is, however, not the case for LSH

Ensemble, as it does not offer any resources to store the index on disk, forcing us to maintain it in memory.

The three systems analyzed have slightly different objectives. In order to perform a fair comparison, we analyze the results from the user perspective. That is, the number of results provided and its degree of relevantness. For that, we use a binary scale, which maps to the output obtained in LSH Ensemble and FlexMatcher. For Nextia$_{JD}$, we will reuse the relevantness mapping discussed in the previous experiment: we map classes $\{0, 1, 2\}$ to the irrelevant class, and classes $\{3, 4\}$ to the relevant one. Applying the same rationale, in LSH Ensemble we consider relevant those pairs with a containment above 50% (i.e., the threshold we considered for our class 3). Finally, FlexMatcher is not parameterizable with a quality threshold and already provides a binary output (i.e., non-joinable/joinable). In this case, non-joinable maps to irrelevant and joinable to relevant.

Results. We evaluated the performance of Nextia$_{JD}$, LSH Ensemble and FlexMatcher on each testbed. Both LSH Ensemble and FlexMatcher suffered from scalability issues and were not able to execute testbed *L*. Figure 6, depicts the runtime of the *pre* phase for each testbed. We can observe that the runtime of all systems is in the same orders of magnitude, however our *pre* is larger than the rest. This is mainly due to the fact that, to ensure a fair comparison, we did not set Spark on cluster mode. It is well-known that using Spark on centralized mode adds extra overhead of tasks when generating the required data structures, managing partitions, etc. This is not the case for the other systems, which are provided as standalone programs. Nevertheless, Nextia$_{JD}$ benefits from Spark's robustness and it is the only approach, even in centralized mode, capable of dealing with a large-scale testbed. Furthermore, we note that Nextia$_{JD}$ is the only solution able to precompute its *pre* step (except for binary meta-features).

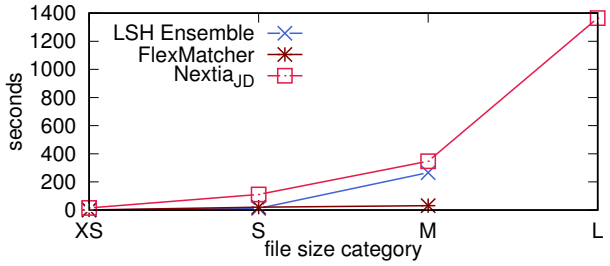

Figure 6: Pre runtime

Regarding the computational performance of the *query* phase, we distinguish the *discovery-by-attribute* and *discovery-by-dataset* scenarios, respectively in Figures 7 and 8. Note that *discovery-by-attribute* is not available in FlexMatcher. LSH Ensemble excels in both *query* tasks. This is due to the fact there is no mechanism to persist the index and this task is reduced to an in-memory lookup. FlexMatcher also benefits from fully running in memory but, in this case, the *query* step suffers from the need to compute some on-the-fly learning models. As general observation, both approaches are thought to compute their core structures and run in memory, which is the main reason hindering their ability to scale-up. Overall, Nextia$_{JD}$ shows a good behaviour in the *query* step. Importantly, Nextia$_{JD}$ is not affected by the dataset cardinality at *query* time. Indeed, the runtime is directly proportional to the number of attributes, or profiles, to compare.

We now put the focus on comparing the predictive performance of the three approaches. Figure 9 and Table 5, depict,

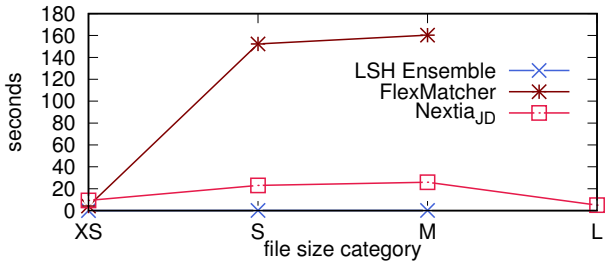

Figure 7: Query runtime (*discovery-by-dataset*)

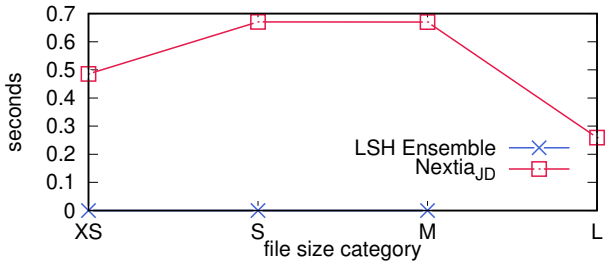

Figure 8: Query runtime (*discovery-by-attribute*)

respectively, the confusion matrices and performance metrics using the binary class mapping. Relevantly, the predictive quality of Nextia$_{JD}$ and LSH Ensemble are comparable. While LSH Ensemble finds more true positives, it generates much more false positives. As result, Nextia$_{JD}$ precision is better. Finally, and aligned with our claim that contemporary profile-based data discovery methods fall short in terms of quality, FlexMatcher generates an extremely large number of false positives reducing its overall quality and making it unfeasible for large-scale scenarios.

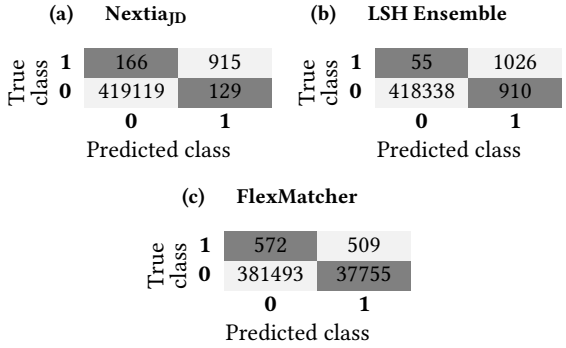

Figure 9: Combined confusion matrices for each system on testbeds *XS*, *S*, *M*

	Precision	Recall	F_1 score
Nextia$_{JD}$	0.8764	0.8464	0.8611
LSH Ensemble	0.5299	0.9491	0.6800
FlexMatcher	0.0133	0.4708	0.0258

Table 5: Performance metrics using binary classes for each system under evaluation on testbeds *XS*, *S*, *M*

Then, Figure 10 drills deeper into the comparison between Nextia$_{JD}$ and LSH Ensemble. We assigned a quality class to LSH Ensemble by running it several times and using a different containment threshold each time, as defined in our quality classes

(i.e., 0.75, 0.5, 0.25 and 0.1). There, we observe relevant differences on the predictions computed. In general, our approach is more conservative, in the sense that we produce less false positives at expenses of sacrificing some true positives. Overall, this improves the precision of our approach by reducing the number of false positives shown to the user. As general observation, both approaches follow slightly different objectives and Nextia$_{JD}$ is more suitable for large-scale scenarios, both for its scale-up capacity and high precision, which guarantees the user will not be overwhelmed with large rankings including false positives.

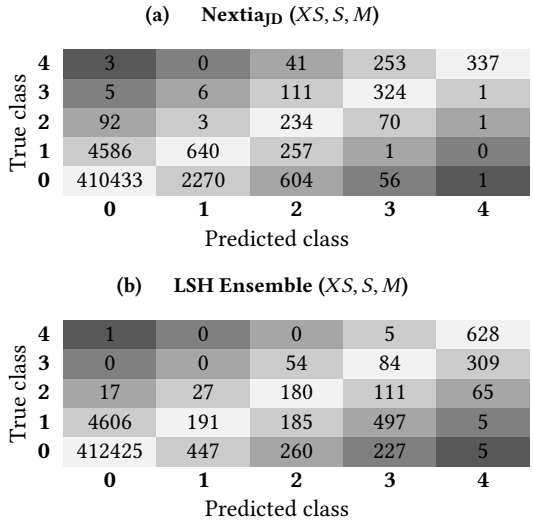

Figure 10: Confusion matrices using 5 quality classes for Nextia$_{JD}$ and LSH Ensemble

4 CONCLUSIONS AND FUTURE WORK

We have presented a novel learning-based approach for data discovery on large-scale repositories of heterogeneous, independently created datasets. Our work is motivated by (i) the poor predictive performance of current profile-based solutions, and (ii) the inability to scale-up and low precision of hash-based ones, which is undesirable for large-scale scenarios. In order to overcome these limitations, we propose a scalable method yielding good precision, and grounded on a novel qualitative definition of join quality. We implemented our approach in a tool called Nextia$_{JD}$. We have experimentally shown that despite being a profile-based approach, Nextia$_{JD}$ presents a similar predictive performance to that of hash-based solutions, yet better adapted for large-scale scenarios, while benefiting from linear scalability.

We do believe profile-based solutions are the right way to go at scale. However, there are some open problems that should be addressed in the future. First, a better join definition metric able to discriminate semantic joins. Current metrics are based on containment / Jaccard similarity and, as previously discussed, these metrics have a very good recall but very low precision. Annotating the required ground truth based on these metrics bias the learning due to the amount of false positives. Current annotated semantic ground truths are unfortunately too small (e.g., Valentine [13]), or generated from approximate metrics like ours, which smoothes the problem but still suffers from it. Last, but not least, profile-based approaches are promising to detect non-syntactic (i.e., with a different encoding per value) semantic join relationships. These are pairs of attributes that maintain the same underlying data distribution but require some transformation in order to join. Based on such predictions, it should be possible to propose such required transformations to join.

ACKNOWLEDGMENTS

This work is partly supported by Barcelona's City Council under grant agreement 20S08704. Javier Flores is supported by contract 2020-DI-027 of the Industrial Doctorate Program of the Government of Catalonia and Consejo Nacional de Ciencia y Tecnología (CONACYT, Mexico).

REFERENCES

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. 2015. Profiling relational data: a survey. *VLDB J.* 24, 4 (2015), 557–581.
[2] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *ICDE*. IEEE, 709–720.
[3] Leo Breiman. 2001. Random Forests. *Mach. Learn.* 45, 1 (2001), 5–32.
[4] Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *SEQUENCES*. IEEE, 21–29.
[5] Chen Chen, Behzad Golshan, Alon Y. Halevy, Wang-Chiew Tan, and AnHai Doan. 2018. BigGorilla: An Open-Source Ecosystem for Data Preparation and Integration. *IEEE Data Eng. Bull.* 41, 2 (2018), 10–22.
[6] Dong Deng, Albert Kim, Samuel Madden, and Michael Stonebraker. 2017. SilkMoth: An Efficient Method for Finding Related Sets with Maximum Matching Constraints. *Proc. VLDB Endow.* 10, 10 (2017), 1082–1093.
[7] AnHai Doan, Pedro M. Domingos, and Alon Y. Halevy. 2003. Learning to Match the Schemas of Data Sources: A Multistrategy Approach. *Mach. Learn.* 50, 3 (2003), 279–301.
[8] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A Data Discovery System. In *ICDE*. IEEE Computer Society, 1001–1012.
[9] Raul Castro Fernandez, Essam Mansour, Abdulhakim Ali Qahtan, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2018. Seeping Semantics: Linking Datasets Using Word Embeddings for Data Discovery. In *ICDE*. IEEE Computer Society, 989–1000.
[10] Raul Castro Fernandez, Jisoo Min, Demitri Nava, and Samuel Madden. 2019. Lazo: A Cardinality-Based Method for Coupled Estimation of Jaccard Similarity and Containment. In *ICDE*. IEEE, 1190–1201.
[11] Behzad Golshan, Alon Y. Halevy, George A. Mihaila, and Wang-Chiew Tan. 2017. Data Integration: After the Teenage Years. In *PODS*. ACM, 101–106.
[12] Mayank Kejriwal and Daniel P. Miranker. 2015. Semi-supervised Instance Matching Using Boosted Classifiers. In *ESWC (Lecture Notes in Computer Science, Vol. 9088)*. Springer, 388–402.
[13] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2020. Valentine: Evaluating Matching Techniques for Dataset Discovery. *CoRR* abs/2010.07386 (2020). arXiv:2010.07386 https://arxiv.org/abs/2010.07386
[14] Renée J. Miller, Fatemeh Nargesian, Erkang Zhu, Christina Christodoulakis, Ken Q. Pu, and Periklis Andritsos. 2018. Making Open Data Transparent: Data Discovery on Open Data. *IEEE Data Eng. Bull.* 41, 2 (2018), 59–70.
[15] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. 2019. Data Lake Management: Challenges and Opportunities. *Proc. VLDB Endow.* 12, 12 (2019), 1986–1989.
[16] Azade Nazi, Bolin Ding, Vivek R. Narasayya, and Surajit Chaudhuri. 2018. Efficient Estimation of Inclusion Coefficient using HyperLogLog Sketches. *Proc. VLDB Endow.* 11, 10 (2018), 1097–1109. https://doi.org/10.14778/3231751.3231759
[17] Biswanath Panda, Joshua Herbach, Sugato Basu, and Roberto J. Bayardo. 2009. PLANET: Massively Parallel Learning of Tree Ensembles with MapReduce. *Proc. VLDB Endow.* 2, 2 (2009), 1426–1437.
[18] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Mach. Learn.* 85, 3 (2011), 333–359.
[19] Michael Stonebraker and Ihab F. Ilyas. 2018. Data Integration: The Current Status and the Way Forward. *IEEE Data Eng. Bull.* 41, 2 (2018), 3–9.
[20] Chuan Xiao, Wei Wang, Xuemin Lin, Jeffrey Xu Yu, and Guoren Wang. 2011. Efficient similarity joins for near-duplicate detection. *ACM Trans. Database Syst.* 36, 3 (2011), 15:1–15:41.
[21] Yang Yang, Ying Zhang, Wenjie Zhang, and Zengfeng Huang. 2019. GB-KMV: An Augmented KMV Sketch for Approximate Containment Similarity Search. In *ICDE*. IEEE, 458–469.
[22] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. 2019. JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. In *SIGMOD Conference*. ACM, 847–864.
[23] Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. 2016. LSH Ensemble: Internet-Scale Domain Search. *Proc. VLDB Endow.* 9, 12 (2016), 1185–1196.