Effective and Scalable Data Discovery with Nextia$_{JD}$

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

ABSTRACT

We present Nextia$_{JD}$, a data discovery system with high predictive performance and computational efficiency. Nextia$_{JD}$ aids data scientists in the discovery of datasets that can be crossed. To that end, it proposes a ranking of candidate pairs according to their join quality, which is based on a novel similarity measure that considers both containment and cardinality proportions between candidate attributes. To do so, Nextia$_{JD}$ adopts a learning approach relying on profiles. These are succinct and informative representations of the schemata and data values of datasets that capture their underlying characteristics. Nextia$_{JD}$'s features are fully integrated into Apache Spark and benefits from it to parallelize the profiling and discovery processes. The on-site demonstration will showcase how Nextia$_{JD}$ can effectively support large-scale data discovery tasks with a large set of datasets the audience will be able to play with.

1 INTRODUCTION

Data-driven organizations are nowadays generating valuable insights by crossing their core data with external third party data, such as data from open data catalogs [6]. This has led to the creation of massive data repositories, or data lakes [8], of heterogeneous datasets without a proper structure or organization [7]. Yet, it is reported that data scientists spend up to 80% of their time in the process of discovering and integrating such datasets [9]. The lack of efficient strategies to automate such discovery process has a large impact on productivity. We exemplify this fact with the following scenario:

Example 1.1. *Emma is a data scientist employed by a marketing agency, hired to launch a campaign in the northern region of Spain. The objective is to find the best way to upsell a new product. To that end, Emma is provided with a reference dataset, such as that depicted in Table 1, containing the store locations and marketing channels that will be used to advertise the new product. She knows that the best strategy for this task is to use demographic data to define consumer segments, ultimately driving the kind of promotion and budget devoted to it. Thus, Emma plans to search for datasets in the agency's data lake, requiring to manually explore each dataset to find interesting ones to be crossed.*

1st Admin. Level	2nd Admin. Level	Store code	Channel
Aragon	*Zaragoza*	*ST123*	*Social networks*
Catalonia	*Lleida*	*ON456*	*Transit ads*
Catalonia	*Barcelona*	*ST093*	*Social networks*
Basque Country	*Araba*	*ON123*	*TV*
...	...	...	...

Table 1: Stores in Spain's northern region (D_{ref})

The previous example is a commonplace *data discovery* scenario. This is the process of automatically identifying and crossing relevant datasets to enable informed data analysis [1]. We put the focus on the task of discovering joinable attributes among datasets in a data lake. The problem is commonly tackled by measuring the similarities among pairs of attributes, aiming to provide the user with higher similarity pairs. We distinguish exact and approximate approaches, where there is a trade-off between their search accuracy and algorithmic complexity. Massive data lake environments, containing hundreds of datasets with thousands of attributes, require solutions with the ability to scale-up, and thus rule out exact methods. The state-of-the-art on approximate approaches to data discovery is those adopting *comparison by hash* techniques, such as MinHash [2] or LSH Ensemble [11]. These compare and predict similarities among pairs of attributes using techniques that, with high probability, hash similar elements to the same bucket (e.g., locality-sensitive hashing or locality-preserving hashing). This process is optimized by building index structures for a particular threshold, such that they allow to efficiently look up the predicted similarity. We next elaborate on applying hash-based techniques on Example 1.2.

Example 1.2. *Table 2 depicts a sample of Emma's agency data lake. She aims to automatically find datasets that will yield the best join with D_{ref}. To do so, as an indicator of a high quality join, she builds a threshold index to discern pairs of attributes with a containment similarity larger than 0.75. Then, Emma uses this index to find promising joinable pairs. Examples of the proposed pairs are D_{ref}.1st Admin Level = D_1.Area, D_{ref}.1st Admin Level = D_2.Region and D_{ref}.1st Admin Level = D_3.Product. Note, however, that the last pair is clearly a false positive, since many regions have a matching product name. Indeed, the values from D_{ref}.1stAdminLevel can also be found in datasets related to ship names, people names, places that are not from Spain, etc. This is a usual scenario in heterogeneous data lakes (i.e., files in different formats and covering different semantic topics) that tend to generate a significant amount of false positives pairs when only considering containment. The number of false positives generated by current approaches is overwhelming when working at scale.*

State of the art hash-based data discovery systems tend to optimistically propose too many candidate pairs at scale. Moreover, the arrival of new datasets requires to reconstruct the threshold indexes for efficient lookup. Such factors can overwhelm data scientists when dealing with large data lakes. Alternatively, another kind of approximate method to data discovery is the *comparison by profile* approach. These approaches extract summaries of datasets and their attributes to build a profile. Profiles are then compared to predict whether a given pair of attributes will join. Such succinct representations can be efficiently generated in a distributed fashion, and their comparison is much more efficient than comparing data values from a complexity point of view. Nevertheless, state of the art profile-based solutions, such as FlexMatcher [3] and Aurum [4], have a low quality prediction rate with respect to other approaches. This is mainly due to either the

(a) D_1 **– Spain census data**

Area	Total population	Persons under 18	Households with a computer
Aragon	3,017,804	23.2 %	84.1 %
Catalonia	973,764	20.9%	89.9%
Asturias	28,995,881	25.5%	89.2%
Galicia	6,045,680	22.1%	91.3%
...	...	...	...

(b) D_2 **– Average life expectancy per region**

Region	Life expectancy (Women)	Life expectancy (Men)
Catalonia	77.9	71.9
Galicia	82.6	77.5
Cantabria	78.8	73.3
Andalusia	81.4	76.8
...	...	...

(c) D_3 **– One million products reviews**

Product	Brand	Kind	Rating
Asturias	Sidra de Asturias	Cider	7.22
Catalonia	K. McRoberts	Book	8.83
Echo dot	Amazon	Smart speakers	8.3
Aragon	Ambar	Beer	8.22
Georgia	Fossil	Watch	5.4
...		...	...

Table 2: Three datasets proposed in a data discovery (D_1, D_2 and D_3)

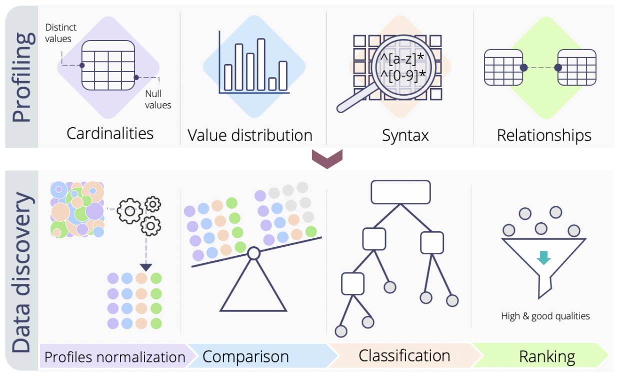


Figure 1: Profiling and data discovery stages implemented by Nextia$_{JD}$

usage of a predictive binary class (i.e., joinable or non-joinable) that generate too many false positives in practice; or to the adoption of rather basic profiles that do not accurately describe the underlying data.

In order to overcome the previous issues, we present Nextia$_{JD}$[1], a novel data discovery system with high predictive performance and computational efficiency. Nextia$_{JD}$ aims to fill the gap generated by the low predictive performance of current profile-based methods, as well as the limited precision and scalability of hash-based ones on data lakes. Nextia$_{JD}$ adopts a novel learning-based method based on data profiles. Importantly, the steps related to profiling and classification can be efficiently run in parallel. Our experiments (see [5] for more details) have shown that the predictive performance of Nextia$_{JD}$ is comparatively better than that obtained using state-of-the-art profile-based solutions, and the rate of false positives (i.e., precision) is improved w.r.t. hash-based ones. Additionally, Nextia$_{JD}$ also outperforms these systems in terms of scalability. This is achieved by integrating Nextia$_{JD}$ into the Apache Spark[2] ecosystem for distributed data processing, providing a competitive advantage with respect to the state of the art on scalability. Nextia$_{JD}$'s predictive model is based on random forest classifiers, a highly expressive model robust to outliers and noise. These models, unlike current approaches, predict a categorical quality for a candidate join based on both the containment and cardinality proportion of the involved attributes, and provide a join quality ranking that facilitates to disregard false positives.

Our demonstration will let EDBT participants impersonate Emma on her data discovery tasks. These involve exploring a data lake, generating a ranking based on the join quality of attribute pairs, as well as generating data processing pipelines from them. Similarly to other contemporary data discovery solutions (e.g., [10]), Nextia$_{JD}$ is accessible via a friendly notebook interface, nowadays the customary tool to develop and visualize data science tasks. Nevertheless, NextiaJD is able to scale-up and manage more and larger datasets. Additionally, the audience will be encouraged to try Nextia$_{JD}$ on datasets of their interest.

This will demonstrate how Nextia$_{JD}$ facilitates data discovery by reducing the time on high quality data exploration and discovery, and thus increasing the productivity of data scientists.

Outline. We next introduce Nextia$_{JD}$'s demonstrable features to resolve the motivational example and other data discovery scenarios. We first provide an overview of Nextia$_{JD}$, followed by a presentation of its core features. Lastly, we outline our on-site demonstration, involving the motivational scenario as well as other more complex real-world use cases.

2 Nextia$_{JD}$ IN A NUTSHELL

Apache Spark has emerged as the leading framework for Big Data processing due to its scalability and performance. It has been extended with modules to enable structured data processing and machine learning, namely SparkSQL and MLlib. Nextia$_{JD}$ extends Spark's source code with new operators to discover joinable datasets: `attributeProfile` and `discovery`. Figure 1, depicts a high-level overview of the stages involved in these operators.

2.1 Attribute profiling

The profiling operator implements the computation of a dataset's profile, which is composed of attribute meta-features. These represent the underlying distribution and characteristics of attributes. Hence, the method `attributeProfile` lazily computes the attribute profiles from a `DataFrame` object once and stores them for later reuse. This process can be triggered at ingestion time, or later in the discovery phase. Nextia$_{JD}$ takes full advantage of the Spark's Catalyst Optimizer to efficiently distribute the workload on very large datasets.

Kinds of profiles. Nextia$_{JD}$ collects extensive meta-features about the structure and content of String attributes in a `DataFrame`. We consider three kinds of meta-features: cardinalities, value distribution, and syntax. Cardinalities provide a broad view of an attribute via meta-features like the number of distinct values, uniqueness, or incompleteness. The value distribution builds a histogram by collecting the number of occurrences and aggregating it to compute meta-features such as the mean, standard deviation, or quantiles. Finally, syntax meta-features aim to describe the shape of data and their patterns. Nextia$_{JD}$ collects meta-features such as the length of values, numbers, or alphabetic values. Here, Nextia$_{JD}$ also exploits several regular expressions to identify specific data types such as telephones, IPs, or emails.

Overall, Nextia$_{JD}$ computes 48 meta-features that compose an attribute's profile. Figure 2, depicts the Scala code used to compute attribute profiles for D_{ref}, as well as an excerpt of its

[1]More info and resources are available at https://www.essi.upc.edu/dtim/nextiajd/
[2]https://spark.apache.org/

```
Spark.read.csv("Dref.csv").attributeProfile()
```

Attribute	Cardinality	NULLs	Entropy	% of Spaces	⋰
1st Admin. Level	17	0	12.59	5%	…
2nd Admin. Level	50	130	0.73	12%	…
Store code	8	5	8.22	1%	…
…					

Figure 2: Code to compute profiles and output's excerpt

output. Additionally, prior to the discovery process, Nextia$_{JD}$ also computes binary meta-features, which denote the characteristics of the relationship between pairs of attributes. Precisely, we measure the degree of similarity and dissimilarity between attribute names by computing the Levenshtein distance. Nextia$_{JD}$ also estimates a best-case containment scenario, assuming all unique values are covered in both attributes. The current set of meta-features used result from a principal component analysis and therefore, all of them are guaranteed to contribute with relevant information to make the decision. Indeed, Nextia$_{JD}$ computes richer profiles compared to other profile-based approaches.

2.2 Data discovery

The data discovery operator exposes the functionality to discover joinable attributes by using the profiles. We distinguish two scenarios: discovery-by-attribute and discovery-by-dataset. The former focuses on the discovery from a reference attribute, while the latter exhaustively searches all attributes in a reference dataset. Both settings require a reference dataset and a list of candidate datasets. Additionally, discovery-by-attribute requires a reference attribute name. This operator encapsulates and hides from the analyst the complexity required to implement the different stages in the discovery pipeline: profile normalization, comparison, classification, and ranking. Thus, Nextia$_{JD}$ does not require to parameterize or process the input data.

Normalization. Meta-features in a profile are represented in different magnitudes, therefore normalization plays an important role to guarantee a meaningful comparison between profiles. Nextia$_{JD}$ adopts the Z-score normalization method for all meta-features in a profile. To do so, a UDF function computes the mean and standard deviation for a given meta-feature using the respective SparkSQL aggregation functions.

Comparison. Comparing profiles requires computing distances among meta-features corresponding to a pair of attributes. Once pairs are created, we merge the profiles subtracting the normalized meta-features from the reference attribute and the to-be-compared attribute by using Spark SQL.

Classification. Nextia$_{JD}$ adopts a learning approach that allows us to classify pairs of attributes producing high quality joins. Precisely, Nextia$_{JD}$ aims to predict the join quality, which is an asymmetric rule-based measure combining both containment similarity and cardinality proportion. In [5] we introduced the concept and role of the cardinality proportion, which complements the containment metric to remove a substantial amount of the false positives generated by it. In short, the cardinality proportion contextualizes containment, and as such, different cardinalities tend to identify different semantics or granularity levels for heterogeneous data lakes. Such metric yields a quality class from a totally-ordered set $S = \{\texttt{None}, \texttt{Poor}, \texttt{Moderate}, \texttt{Good}, \texttt{High}\}$. Hence, the definition of join quality is as follows:

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

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

Nextia$_{JD}$ embeds a set of general purpose models, one per quality label in the previous definition, trained with Random Forest classifiers from Spark MLlib. These models were trained by transforming a multi-class classification problem into a binary one per class. Each of these models takes as input the normalized unary and binary meta-features of the pair of attributes for which we aim to predict their join quality. These models were trained following good practices in model learning and the ground truth (including labeling), data preparation, and validation processes are thoroughly presented in a reproducible manner at: https://www.essi.upc.edu/dtim/nextiajd/. As part of this process, we empirically determined 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 on our training dataset composed of 138 real datasets. The models validation was conducted with 139 real datasets from different topics and file sizes and yielding a high predictive performance [5]. As a result, for each candidate join pair the discovery operator associates a join quality label (i.e., from None to High) and five probability scores, one per model.

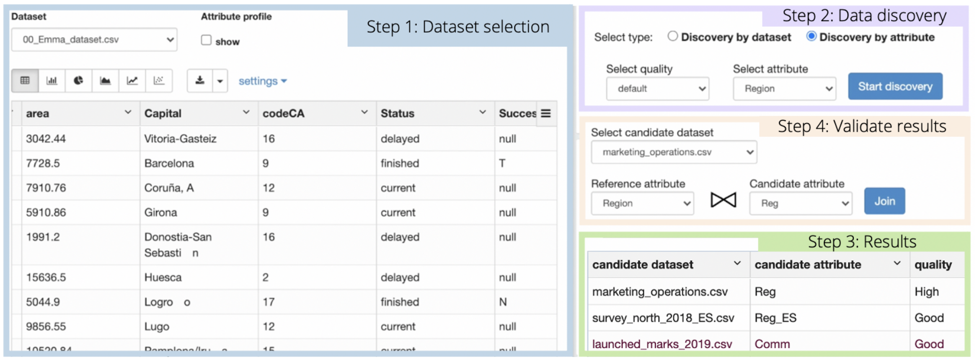
A key distinguishing factor of Nextia$_{JD}$ with regard to other profile-based approaches (e.g., FlexMatcher [3]), is that it relies on general purpose models that can be used for any data discovery process with heterogeneous datasets.

```
NextiaJD.discovery(Dref, Seq[D1,D2,D3],
"1st␣Admin.␣Level")
```

Dataset	Attribute	Quality	Probability
D_2	Region	High	0.95
D_1	Area	High	0.93

Figure 3: Code to trigger a discovery process, and the two first elements of the ranking it generates

Ranking. Finally, an evaluation is performed in the probabilities of a candidate join pair to assign a single probability. In short, the highest probability wins, except for cases where several probabilities are close to each other. In those cases, we follow a rule-based strategy to avoid misclassifications due to the fact that the probability for the None class is the only one predicting no join. We identified two main cases generating the most misclassifications: (i) when the no join probability (i.e., the probability for None) is above 50% and (ii) when the join-related labels (i.e., from Poor to High) are all below 50% and the None probability is close to them (measured by an empirical threshold). In these cases, the final decision is modified to the second highest probability (which in practice, given these rules, mostly means to that of None). Then, Nextia$_{JD}$ generates a partial order by considering, first,


Step 1: Dataset selection

Dataset: 00_Emma_dataset.csv

Attribute profile: ☐ show

settings

area	Capital	codeCA	Status	Succes
3042.44	Vitoria-Gasteiz	16	delayed	null
7728.5	Barcelona	9	finished	T
7910.76	Coruña, A	12	current	null
5910.86	Girona	9	current	null
1991.2	Donostia-San Sebasti n	16	delayed	null
15636.5	Huesca	2	delayed	null
5044.9	Logro o	17	finished	N
9856.55	Lugo	12	current	null

Step 2: Data discovery

Select type: ○ Discovery by dataset ● Discovery by attribute

Select quality: default

Select attribute: Region

Start discovery

Step 4: Validate results

Select candidate dataset: marketing_operations.csv

Reference attribute: Region ⋈ Candidate attribute: Reg

Join

Step 3: Results

candidate dataset	candidate attribute	quality
marketing_operations.csv	Reg	High
survey_north_2018_ES.csv	Reg_ES	Good
launched_marks_2019.csv	Comm	Good

Figure 4: Nextia$_{JD}$ GUI in a Zeppelin notebook

the predicted quality label (and the totally-ordered set related to them). For pairs yielding the same quality label, we rank them considering the probability yielded by the model (i.e., that of the model of the quality labeled finally assigned). An example of the ranking produced for Emma's use case is presented in Fig. 3. By default, Nextia$_{JD}$ only shows High and Good qualities. However, other qualities can be requested on-demand.

3 DEMONSTRATION OVERVIEW

Computational notebooks have become the de-facto tool for data science projects. In our experience in several Data Science projects, data scientists like to explore datasets with their usual analytical tools. Using a third party tool to perform data discovery is disruptive for their day-by-day tasks, and they tend to avoid it. For this reason, we created Nextia$_{JD}$, which fills a current gap to bring data discovery closer to data scientists. We argue that embedding data discovery into notebooks and taking advantage of their interactive capabilities will improve data scientists' productivity. Therefore, for our demonstration, we will use a Zeppelin notebook to present the main functionalities: profiling and data discovery, and how they can be used in the day-by-day of data scientists. We have also created an informative companion website[3] where the notebooks[4], source code and experiments are publicly available. It is worthy to say that Nextia$_{JD}$ is not tied to this demonstration platform and can be integrated into any technology that supports Spark in Java or Scala clients.

We encourage attendees to impersonate Emma and follow the workflow she would have to execute using Nextia$_{JD}$. Note that we assume for this demonstration that datasets were profiled when ingested into a repository to be ready for use in further discoveries. However, new datasets can be processed and profiled on demand if required during the demo. Figure 4 shows Nextia$_{JD}$'s GUI in a Zeppelin notebook:

(1) **Dataset selection.** First, attendees can select and preview datasets available in our heterogeneous data lake containing Emma's dataset and datasets from different topics such as movies, territories, finance, etc. Through this step, attendees can also preview the profiling computed to have a better perspective of what kind of meta-features Nextia$_{JD}$ collects.
(2) **Data Discovery.** Once a dataset is selected, we proceed to the data discovery task. Users can perform two types of setting: discovery-by-dataset or discovery-by-attribute. Through this step, it is possible to select the desired quality. Nextia$_{JD}$ will execute the data discovery operator and will handle all steps: normalization, comparison, classification, and ranking.
(3) **Explore results.** After data discovery, results are visualized in a table where we show the attributes pairs found, the source of the datasets, the quality predicted, and the probability.
(4) **Validate results.** Once the attendees find an interesting pair proposed by Nextia$_{JD}$, they can validate the result by executing the join operation. This operation will update the dataset preview with the result of the join. Additionally, the similarity and cardinality proportion obtained by the join operation is also computed and shown.

Last, but not least, we are aware that some data scientists are advanced users. In these cases, they do not need to use Nextia$_{JD}$'s GUI but directly use the new it offers as an extension of Spark. These are compiled and ready in the Spark fork available from our website. This is shown in the live demo on our website. Overall, this demonstration will offer a comprehensive dive into Nextia$_{JD}$.

[3]https://www.essi.upc.edu/dtim/nextiajd/
[4]Nextia$_{JD}$ online notebooks have deactivated parallelism to keep them 24/7 in a budget machine. Find instructions to install Spark and Nextia$_{JD}$ in https://www.essi.upc.edu/dtim/nextiajd/#resources

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] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *ICDE*. IEEE, 709–720.
[2] Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *SEQUENCES*. IEEE, 21–29.
[3] 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.
[4] 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.
[5] Javier Flores, Sergi Nadal, and Oscar Romero. 2021. Scalable Data Discovery Using Profiles. In *EDBT*. To be published as short paper.
[6] 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.
[7] Fatemeh Nargesian, Ken Q. Pu, Erkang Zhu, Bahar Ghadiri Bashardoost, and Renée J. Miller. 2020. Organizing Data Lakes for Navigation. In *SIGMOD Conference*. ACM, 1939–1950.
[8] 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.
[9] 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.
[10] Yi Zhang and Zachary G. Ives. 2020. Finding Related Tables in Data Lakes for Interactive Data Science. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. ACM, 1951–1966.
[11] 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.