Supporting Data Discovery Tasks at Scale with Freyja

Marc Maynou
Universitat Politècnica de Catalunya
Barcelona, Spain
marc.maynou@upc.edu

Sergi Nadal
Universitat Politècnica de Catalunya
Barcelona, Spain
sergi.nadal@upc.edu

ABSTRACT

Data lakes, massive repositories of heterogeneous data, were popularized by implementing a *load-first model-later* approach to data integration in contrast to the traditional *model-first load-later* implemented in data warehouses. Yet, such settings create new challenges related to *a)* how to effectively discover relevant data and automatically find meaningful relationships; and *b)* how to do so efficiently at scale. In order to address them, we present Freyja, a data discovery system designed to navigate through vasts amounts of data and support users on finding joinable pairs of attributes. Freyja is based on the definition of a novel similarity measure, designed to automatically discovery semantically-relevant relationships. The adopted metric is tailored to data lake contexts, where datasets are denormalized and data from several domains are stored. In order to optimize and scale the computation of such metric, Freyja adopts a learning approach relying on profiles, succinct and lightweight representations of the underlying characteristics of the data attributes. Freyja leverages analytical databases to efficiently compute column profiles. In this demo, we demonstrate how Freyja, which is offered as a Python library, supports data augmentation tasks (i.e., augmenting training datasets with new features to improve the performance of a machine learning model).

1 INTRODUCTION

Modern data-driven systems require vast amounts of high quality data to perform analyses and develop sophisticated predictive models. Data lakes [8], repositories where numerous agents deposit their data via a flexible schema-on-read approach, offer the opportunity to find such relevant assets, as they hold a plethora of potentially useful information provided by different stakeholders and organizations. Adequately navigating through these vast data repositories and identifying suitable relationships, allows data scientists to execute downstream tasks with extended sets of data, improving the capabilities of the defined processes [11]. We exemplify one of such cases with the following scenario:

Example 1.1. *Anna is a data scientist hired by the municipality of Barcelona to analyze the skyrocketing rental prices in the city. Tasked with uncovering the underlying factors contributing to these increases, she decides to develop a robust predictive model to accurately forecast rental prices, segmented by neighborhood. Her goal is to obtain a model that can accurately predict future values based on the characteristics of the population. She is provided with a reference dataset (depicted in Table 1), which contains the average rental price for each neighborhood, along with other characteristics. After deploying an initial model, she considers that by increasing the number of features, the problem could be better characterized. Hence, she now turns to the government's data lake to extend the available information and perform more nuanced analyses. She plans to do so by finding datasets that she can join with through the neighborhood column, employing the newly added columns as model features.*

Neighborhood	Avg. Rental (€)	Avg. Age	Avg. Income (€)
El Raval	834	38.4	11,304
Sants	874	42.4	20,501
Les Corts	1065	46.3	31.123
La Salut	970	44.5	27,477

Table 1: Rental prices of Barcelona's neighborhoods (D_{ref})

The presented scenario describes a *data discovery* task [7]. Data discovery can be defined as the process of automatically identifying and combining relevant datasets from various sources to extend the amount of available information for downstream tasks. An instance of one such subsequent task is also presented in the example: *data augmentation* [3], which is described as incorporating new data in the training of a predictive model to improve its efficacy when addressing the target function.

In order to aid in such context we present Freyja[1], a data discovery system that focuses on the task of *join discovery* [2], which specifically explores the detection of joinable datasets to increase the number of data attributes available. Our goal is to present a ranked list of potentially joinable pairs between an attribute of a reference dataset (which we label as the query attribute) and attributes from other datasets, sorting the pairs by their degree of joinability. Hence, we particularly want to increase the number of features of a designated dataset, doing so by performing join operations through the query attribute. A simplified approach to perform this task is described next:

Example 1.2. *Figure 1 showcases a subset of the tables found in the data lake explored by Anna. Her goal is to find attributes that represent good candidates to join with column D_{ref}.Neighborhood to extend the available data. To do so, she opts for a straightforward approach: rank the quality of a join based on the degree of overlapping (i.e. containment) between the columns. In doing so, the pairs (D_{ref}.Neighborhood, D_1.Neighborhood), (D_{ref}.Neighborhood, D_2.Neigh.) and (D_{ref}.Neighborhood, D_3.Product) are considered as relevant joins. Note, however, that the latter is a **false positive**.*

The process described above is overly inefficient, as data lakes differ from conventional structured data repositories in two major facets. Firstly *(i)* they contain **massive volumes of data**, much higher than traditional databases and with datasets that can reach millions of rows. Computing overlaps is unfeasible in such scenario. Secondly *(ii)*, as different agents provide the data assets, the **stored datasets are highly heterogeneous**. This refers to both semantic diversity (i.e. large variety of disparate topics or domains) as well as syntactic variability (e.g. large differences on the number of attributes, their values or their cardinalities). Hence, false positives joins, such as the one depicted in Example 1.2, are common in data lakes, as the heterogeneity of the data makes it difficult to discern whether two sets of similar values represent the same entities.

[1]Link to the paper's companion website: https://freyja-data-discovery.github.io/

D_1 – **Inequality metrics**

Neighborhood	80/20 rate	Gini index
La Salut	3.85	27.8
Tres Torres	3.04	31.2
Les Corts	3.05	33.3
El Raval	3.12	34.2

D_2 – **Population statistics**

Neigh.	Population	% Male
El Raval	46,934	47.48
Sants	47,063	51.47
Les Corts	46,740	53.57
Guinardó	38,153	53.20

D_3 – **List of products**

Product	Author	Type
El Raval	Zoo	Music CD
Sants	Tania Juste	Book
Les Corts	Josep Maria Casasus	Book
La Salut	Michel Sardou	Music CD

Figure 1: Subset of datasets in a data lake

Modern data discovery systems address data heterogeneity by proposing approaches that, rather than computing set-overlapping metrics, consider the semantics of the data beyond value intersection (e.g., embedding representations [2, 4], knowledge bases [9] or deep learning architectures [5, 6]). Nonetheless, these methodologies are complex to set up and expensive to execute, often demanding large pools of resource and incurring high execution costs. In contrast, Freyja aims to maintain the effectiveness of modern approaches (i.e. to consider the semantics of the data attributes) while offering a scalable and efficient solution that can be used in most environments with minimal set up. To do so, Freyja adopts a **new joinability metric** that, unlike traditional options such as containment or Jaccard, incorporates an assessment of the semantics of the data attributes (we refer the reader to [10] for further details). However, this metric, which measures the quality of a join, still relies on costly set-overlap computations. Hence, Freyja prevents the need of computing such pairwise comparisons by employing a **predictive model that approximates** the quality of a join in a fraction of the time needed to compute it. To do so, Freyja computes and stores extensive data profiles that capture the underlying characteristics of the data attributes (e.g. incompleteness, entropy) [1].

Freyja uses a lightweight and general-purpose predictive model, thus no retraining is needed when facing new data lakes. Profiles are obtained by performing column-wise operations over the data, for which we employ analytical databases (i.e. DuckDB) to optimize the process. Computing the profiles is easily parallelizable, as profiles are completely independent from each other. This also applies to comparing them (i.e. calculating the differences), as each procedure is isolated. Moreover, the entire workflow is highly scalable, as profiles always have the same length (i.e. number of metrics) regardless of the size of the original data. Hence, the overall cost scales linearly with regards to the number of datasets in the repository. Finally, each profile occupies a few KBs, so the entire pipeline can be executed in-memory, even for large data lakes. Figure 2 presents Freyja's high-level pipeline.

Demonstration. EDBT attendees will be able to act as Anna in her data discovery task, doing so from the comfort of a Python notebook. Notebooks are nowadays the customary tool to explore and analyze data, as they provide an expressive environment to run experiments and present results. However, most data discovery systems are cumbersome to operate due to requiring complex set ups (e.g. accommodating large knowledge bases) or high-end machines to run expensive computations (e.g. deep learning tasks). This limits the usage of such approaches, specially because there are no libraries to easily invoke their defined tasks. Opposedly, Freyja requires minimum setup and is highly portable. Moreover, it can run on the vast majority of PCs, as it does not have elevated memory or GPU/CPU requirements. Our aim is to prevent data discovery from being restricted only to high-performance environments.

2 FREYJA

Freyja provides three processes: *(i)* compute profiles, both for single files as for entire data lakes, *(ii)* obtain, for a given query attribute and a data lake, the joinability ranking (sort all potential pairs based on their degree of joinability), and *(iii)* execute a data augmentation pipeline, which we provide as an instance of a downstream task to showcase the practical utility of Freyja.

2.1 Attribute profiling

Profiles are summarized representations of the underlying characteristics of sets of values. A profile is composed of several features, each capturing different properties of the data attributes (e.g. incompleteness, entropy, median value, etc.). By working with these features we can have a high-level understanding of the original data while preventing the need to operate with extensive lists of values, so their utility goes beyond our particular application for data discovery. Freyja leverages analytical databases (e.g. DuckDB) to execute optimized column-wise operations, allowing for the efficient computation of the aggregations needed to obtain the profile features. Moreover, profiles need to be computed only once per dataset and can be retained for future utilization in subsequent data discovery processes. This can be done in an offline stage previous to the data discovery task, and be reused every time a new analysis needs to be conducted. Figure 3 showcases how to employ Freyja to generate data profiles. Note that our goal is to use profiles to define joins, so only profiles of non-numerical columns are obtained, as computing joins with numerical data generally leads to non-significant results.

Types of features. Profiles are composed of an extensive selection of features (extracted from state-of-the-art on data profiling [1]), aimed at representing all the relevant properties of data that the model is going to use to predict the joinability metric. That is, our goal is to offer a multi-faceted, high-level characterization of the data. Features can be divided into three main categories: *general properties*, *value distribution* and *syntactic*. The first category measures well established properties to assess the characteristics of the underlying data, such as entropy, uniqueness or incompleteness. *Value distribution* features capture insights regarding the distribution of the values, such as octiles and average frequency. *Syntactic* features analyze the structure of the data values themselves: longest/shortest strings, number of strings or data types. In total, Freyja collects 62 individual features, offering a wide spectrum of properties to represent the data.

```
dref = read_csv(dref_path)
dref_profile = freyja.compute_profile(dref)
```

Feature	Entropy	Min. Freq	Avg. length	...
Neighborhood	2.4	1	11,304	...

Figure 3: Code snippet to compute profiles

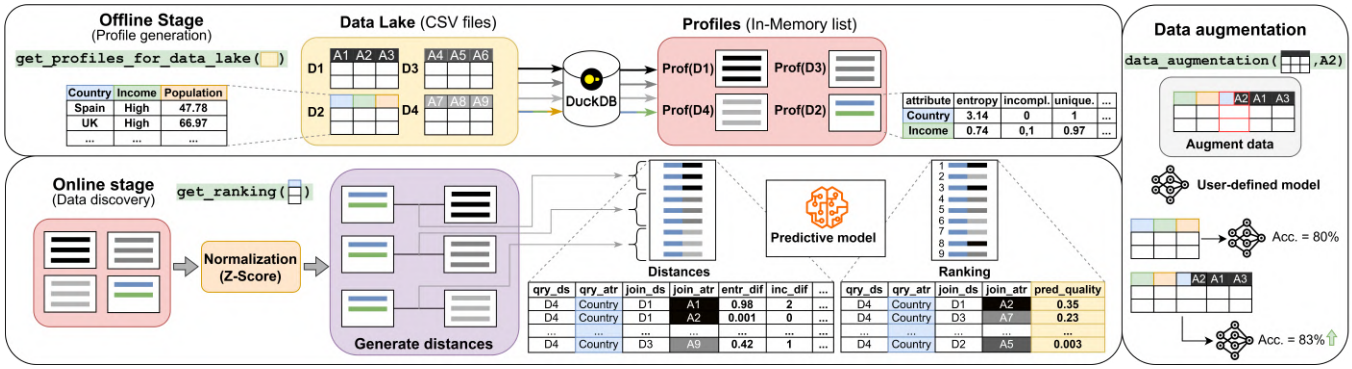

Figure 2: Freyja system architecture

2.2 Joinability rankings

The data discovery process is as follows: given the profile of a query attribute, $P(A_q)$, and the profiles for all columns (candidate attributes) of a given data lake D_l, $\{P(A_c) \mid A_c \in D_l\}$, we obtain a ranking of all the possible joins (A_q, A_c), sorted by their degree of joinability. This method has two main steps. Firstly *(i)*, we calculate the *distances* between the query attribute and all the candidate attributes, which represent a high level comparison between the properties of the two underlying columns. Secondly *(ii)*, we execute our predictive model to map every set of *differences* between the query attribute and each candidate attribute to a joinability value.

Calculating distances. This process amounts to, for every candidate pair to join, computing the difference between each of the individual metrics of both profiles (e.g. $D_a.Att1$.entropy - $D_b.Att_3$. entropy). However, some features, such as frequency distributions, are represented in different magnitudes. Hence, to define meaningful comparisons we first normalize the features. More precisely, for a given dataset, we normalize conjointly the profiles of all its attributes by applying the Z-score normalization, obtaining the mean and standard deviation for all instances of a feature in all profiles of the dataset. For instance, for a dataset D the feature *minimum frequency* is normalized by computing the mean and standard deviation of all *minimum frequency* values in D, one per profile.

Predictive model. The novel joinability metric is still costly to compute, thus Freyja employs a learning approach based on a pre-trained model that can be used to predict the joinability metric for two sets of values based on their profiles. Therefore, Freyja evaluates the the model for each pair (A_q, A_c) we want to assess the joinability of. Our instance of the model has been trained with an extensive corpus containing pairs of attributes with disparate joinability values. This has allowed the model to understand how the distinctions in the values of the individual profile features correlate with the novel joinability metric. Hence, it does not need to be re-trained for every data lake, as it is general enough to adapt to most environments.

Our novel joinability metric generates a continuous value that quantifies the quality of the join between two attributes, providing a precise measurement of how well these attributes can be combined. This continuous value enables us to create an efficient ranking system that orders candidate pairs based on their ability to produce highly relevant and meaningful joins. This ranking is essential, as, without it, the output could consist of an overwhelming number of potential joins, each of which would require extensive manual validation, a time-consuming and resource-intensive task. Our model effectively replicates this behavior, but at a significantly lower cost, replacing expensive set-overlap computations with model predictions. Experimental results show that the prediction error relative to the original metric is negligible [10]. Figure 4 exemplifies how to get a joinability ranking.

```
dl_profiles = freyja.profiles_for_dl(dl_path)
ranking = freyja.ranking(dref_profile, "neighborhood",
      dl_profiles, 3)
# 4th parameter indicates the top-k positions to show
```

Candidate dataset	Candidate attribute	Predicted metric
D_2	Neigh.	0.455
D_1	Neighborhood	0.455
D_3	Product	0.131

Figure 4: Code snippet to obtain the top-3 joins

2.3 Data augmentation

The goal of data discovery is to increase the amount of data available in downstream tasks, ultimately improving the effectiveness of data-driven models. By identifying relevant datasets and integrating useful attributes, data discovery enables more comprehensive analyses and enhances model performance. To further extend the capabilities of Freyja, we provide an implementation of one such tasks: data augmentation. This implementation is seamlessly integrated with the rest of the pipeline, ensuring smooth interaction with other components. Given a joinability ranking and a designated candidate attribute (selected from those appearing in the ranking and identified by an index), we execute the join between the query attribute and the candidate attribute. This process merges the relevant data from different datasets into a single, cohesive collection, thereby enriching the available information and improving the capabilities of a model. The augmentation method is designed to be flexible, allowing users to experiment with different ranking positions to assess their impact on the downstream task.

Figure 5 showcases the full data augmentation pipeline, illustrating its role in refining model performance. It presents a previously defined model undergoing three evaluation scenarios: using only the base data, augmenting it with the highest-ranked candidate attribute, and repeating the process with the second-highest-ranked candidate. These evaluations demonstrate how strategic data augmentation can enhance a model's learning process, ultimately leading to better performance in downstream tasks.

```
dref = read_csv(dref_path)
run_model(d_ref, "d_ref")

dref_profile = freyja.compute_profile(dref)
dl_profiles = freyja.profiles_for_dl(dl_path)

ranking = freyja.ranking(dref_profile, "neighborhood",
      dl_profiles, 20)

dref_d2 = freyja.da(dref, "neighborhood", ranking, 1)
run_model(dref_d2, "d_ref joined with d_2")

dref_d1 = freyja.da(dref, "neighborhood", ranking, 2)
run_model(d_ref_d_1, "d_ref joined with d_1")
```

```
Data used to generate the model: d_ref
- Root Mean Squared Error: 76.44
- R-squared: 0.7188
------------------------
Data used to generate the model: d_ref joined with d_2
- Root Mean Squared Error: 39.1857
- R-squared: 0.9029
------------------------
Data used to generate the model: d_ref joined with d_1
- Root Mean Squared Error: 47.59
- R-squared: 0.8568
```

Figure 5: Data augmentation with Freyja

3 DEMONSTRATION OVERVIEW

The main driver underneath the development of Freyja was to simplify the access and usage to data discovery tools, preventing the need for elaborate setups and costly infrastructure. This goal led to the creation of a solution that enables users to work with their data without the technical barriers that often accompany complex platforms. To that end, we offer Freyja as a Python library, which streamlines its integration into existing workflows. Its functionalities are showcased via notebooks, expressive environments that are ideal for trial-and-error approaches and for presenting results due to their structure, which divides the process into independent code blocks. This modular approach is particularly well-suited for experimentation and analysis and, as a result, have become the de facto tool for data scientists to explore and analyze their data, as they allow for iterative experimentation and seamless integration of both visualizations and explanatory text in a single environment. By embedding Freyja within this environment, we significantly lower the barrier for adoption, eliminating the steep learning curves that are characteristic of more complex, extrinsic tools. The simplicity and efficiency of Freyja's design ensure that users can focus on what truly matters: discovering insights from their data. Furthermore, as illustrated in the code snippets above, interacting with Freyja is straightforward and intuitive. The functions primarily operate over in-memory objects, which reduces the need for additional setup or infrastructure. This design philosophy contributes to the library's accessibility and ease of use, making it a valuable tool for both beginners and experienced data scientists alike.

Attendees will be able to execute a full data discovery pipeline, mimicking the processes illustrated in the above examples. Specifically, the available tasks are those defined in previous section, which map to the main stages implemented by Freyja's pipeline, allowing us to showcase the full potential of the tool on a practical setting. This hands-on experience will provide valuable insights into how Freyja can simplify, accelerate and enhance the data discovery process.

(1) **Create profiles**. Participants can initiate the process of computing profiles both for single datasets as well as collections of datasets. In the demo we include a small data lake so the users can get a sense of the efficiency of Freyja. The profiles and set of metrics they contain can be explored, which allows the attendees to get a better understanding at the type of information collected.
(2) **Obtain joinability ranking**. Once a set of profiles has been generated, users can execute the discovery process, which will yield a joinability ranking, with the more relevant joins placed in the top positions. This ranking is accompanied by a numerical evaluation, which highlights the differences in the quality of the joins and provides a clear indication of their significance. The evaluation makes it easier for users to assess and select the most meaningful joins, streamlining the decision-making process and ensuring the most impactful relationships are identified.
(3) **Data augmentation**. Attendees will be able to employ the obtained ranking in a data augmentation task. By selecting one of the candidate columns presented in the ranking, an automatic augmentation process can be executed, which generates a joined dataset that can be immediately applied to test and evaluate a model's performance. This seamless integration ensures that the augmented data can be put to use quickly, facilitating more efficient experimentation and model improvement.

In the on-site demonstration we will encourage attendees to propose new use cases and datasets, in order to showcase the flexibility and efficiency of Freyja.

ACKNOWLEDGMENTS

This work has been partly supported by the Horizon Europe Programme under GA.101135513 (CyclOps) and the Spanish Ministerio de Ciencia e Innovación under project PID2023-152841OA-I00 / AEI/10.13039/501100011033 (TALC).

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] Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David Karger. 2020. ARDA: automatic relational data augmentation for machine learning. *arXiv preprint arXiv:2003.09758* (2020).
[4] Yuyang Dong, Kunihiro Takeoka, Chuan Xiao, and Masafumi Oyamada. 2021. Efficient Joinable Table Discovery in Data Lakes: A High-Dimensional Similarity-Based Approach. In *ICDE*. IEEE, 456–467.
[5] Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. 2022. DeepJoin: Joinable Table Discovery with Pre-trained Language Models. *arXiv preprint arXiv:2212.07588* (2022).
[6] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée Miller. 2022. Semantics-aware dataset discovery from data lakes with contextualized column-based representation learning. *arXiv preprint arXiv:2210.01922* (2022).
[7] 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.
[8] Rihan Hai, Christos Koutras, Christoph Quix, and Matthias Jarke. 2023. Data lakes: A survey of functions and systems. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (2023), 12571–12590.
[9] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J Miller, and Mirek Riedewald. 2023. Santos: Relationship-based semantic table union search. *Proceedings of the ACM on Management of Data* 1, 1 (2023).
[10] Marc Maynou, Sergi Nadal, Raquel Panadero, Javier Flores, Oscar Romero, and Anna Queralt. 2024. FREYJA: Efficient Join Discovery in Data Lakes. arXiv:cs.DB/2412.06637 https://arxiv.org/abs/2412.06637
[11] 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.