Finding What You're Looking For: A Distribution-Aware Dataset Search Engine in Action

Lennart Behme
BIFOLD & Technische Universität Berlin
Germany
l.behme@tu-berlin.de

Leonard Geißler
Technische Universität Berlin
Germany
l.geissler@tu-berlin.de

Pratham Agrawal
Indian Institute of Technology Delhi
India
cs1210891@iitd.ac.in

Emil Badura
Technische Universität Berlin
Germany
badura@tu-berlin.de

Benjamin Ueber
Technische Universität Berlin
Germany
b.ueber@tu-berlin.de

Kaustubh Beedkar
Indian Institute of Technology Delhi
India
kbeedkar@cse.iitd.ac.in

Volker Markl
BIFOLD, Technische Universität Berlin & DFKI
Germany
volker.markl@tu-berlin.de

Abstract

The growing volume of academic, commercial, and governmental datasets distributed across countless independent repositories calls for dataset search engines that can answer queries only using publicly shared metadata instead of relying on raw data access. However, the keyword search interfaces of existing metadata-based search engines fail to capture complex user needs, such as distributional requirements, thereby limiting their effectiveness. In this demonstration, we present the first end-to-end system for distribution-aware dataset search over decentralized data repositories. Our prototype combines existing search techniques with recently proposed percentile predicates to provide more powerful query capabilities. Based on our novel Dataset Query Language and a distribution-aware index, the system enables efficient, flexible search without access to raw data. To demonstrate its utility, we curated over 150 000 profiles of tabular datasets from Kaggle and enriched them with statistical information, enabling attendees to explore distribution-aware search and the trade-offs involved in system configuration.

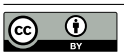
CCS Concepts

• **Information systems** → **Information retrieval**; **Data management systems**.

Keywords

data discovery; dataset search; distribution-aware search

ACM Reference Format:
Lennart Behme, Leonard Geißler, Pratham Agrawal, Emil Badura, Benjamin Ueber, Kaustubh Beedkar, and Volker Markl. 2025. Finding What You're Looking For: A Distribution-Aware Dataset Search Engine in Action. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3722212.3725104



SIGMOD-Companion '25, Berlin, Germany

ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3725104

1 Introduction

In a rapidly evolving data landscape, organizations generate and store vast volumes of information across diverse repositories, making effective and efficient data discovery a cornerstone of any data-driven task. The demand for advanced data discovery systems in academia and industry has grown alongside the widespread adoption of machine learning, which thrives on large, high-quality datasets. As data sharing and trading gain importance both within organizations and across broader ecosystems [3, 10], dataset search engines are rising as a critical component of data-driven processes.

The Landscape of Dataset Search Engines. Despite their importance, current dataset search engines often fall short due to two fundamental limitations in their design. First, they frequently assume unrestricted access to all datasets for indexing and processing [6, 8]. This assumption is impractical in distributed environments, where datasets are spread across several data repositories. Consolidating them into a central server is not only costly but also conflicts with data owners' desire to maintain control over their proprietary information. Instead, large-scale dataset search engines like Google Dataset Search [12] and Kaggle or B2B platforms like Dawex [7] rely on a federated data architecture, where metadata are shared with a central entity while the underlying raw data remain decentralized. Second, existing systems predominantly rely on keyword search interfaces or example-based queries, often augmented by filters [5, 12]. While these methods are straightforward, they fail to address more complex use cases where data consumers have multiple requirements about a dataset, such as requiring specific data distributions for individual dataset attributes. For instance, machine learning practitioners seeking balanced datasets with equal representation across target groups cannot specify such criteria using keywords.

In their survey among data professionals from academia and industry, Hulsebos et al. [9] find that many data consumers ask for more powerful query interfaces and would like to search for statistical properties of datasets, underscoring the need for dataset search engines that offer advanced search operators, specifically for distributional requirements.

Distribution-Aware Dataset Search. Asudeh and Nargesian [2] recently advocated incorporating distributional search predicates

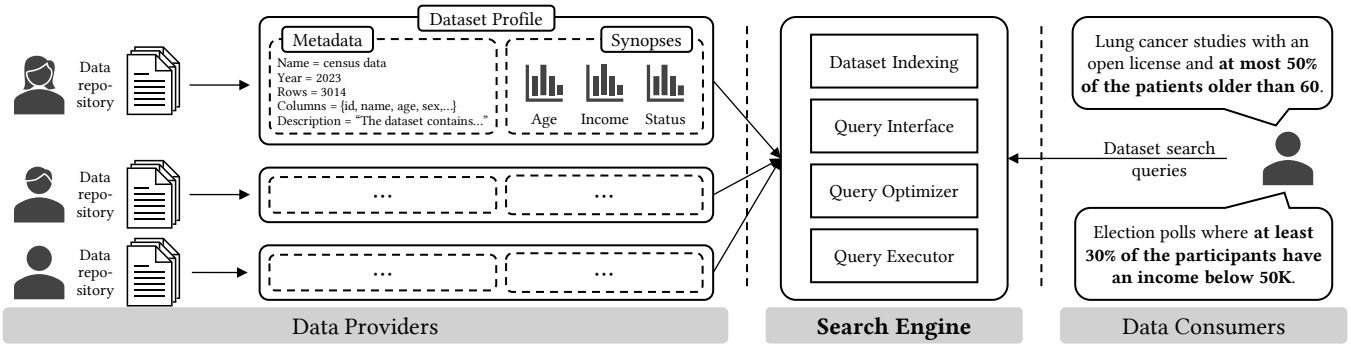


Figure 1: Overview of the problem setting (adapted from [4]). Data providers publish dataset profiles containing standard metadata and dataset statistics. Data consumers have complex search intents, including conventional keyword predicates and distributional requirements. The search engine (this demonstration) has to index datasets and efficiently answer search queries.

into the design of data markets and dataset search engines. Distribution-aware dataset search extends beyond conventional metadata-based search by identifying datasets based on statistical properties, such as value distributions, to support tasks like enhancing fairness or robustness in machine learning. In our previous work [4], we introduced percentile predicates as a flexible means to express distributional requirements and proposed Fainder, an efficient search index for evaluating them. To realize the full potential of our contributions, it is essential to integrate them into a comprehensive dataset search engine with a versatile query interface that combines existing keyword search techniques and percentile predicates.

Our Demonstration. We present the first end-to-end system for distribution-aware dataset search over decentralized data repositories. Figure 1 outlines our problem setting, illustrating the three roles that participants can take in our demonstration. Data providers who own datasets they want to make discoverable upload dataset profiles enriched with statistical information using the industry-wide Croissant standard [1]. The search engine ingests these profiles and constructs search indices, such as Fainder, to enable efficient query execution. As system operators, participants can experiment with different index configurations to understand their impact on runtime and result quality. Most importantly, as data consumers, they can experience the benefits of distribution-aware search when querying a large dataset collection with a specific task in mind.

Building on the conceptual query model from our prior work [4], we developed the declarative Dataset Query Language (DQL) as our system's query interface. DQL is based on Boolean algebra and seamlessly integrates existing keyword search techniques with percentile predicates while offering extensibility for future advanced search operators. Leveraging DQL's declarative nature, our prototype explores query optimization strategies, such as pruning sub-plans based on intermediate results, to reduce query execution time. Data consumers can use DQL directly or utilize our graphical query builder for quick and intuitive query construction.

A Dataset Collection for Distribution-Aware Search. To provide participants with a realistic and engaging search experience, we curated over 150 000 dataset profiles of tabular datasets from Kaggle and enriched them with statistical information. This collection, to the best of our knowledge, is the largest of its kind for distribution-aware dataset search and serves as a valuable resource for advancing research in the field beyond our demonstration.

2 System Overview

We describe the functionalities and architecture of our demonstration from the perspectives of the three types of users who interact with it: data providers, system operators, and data consumers.

2.1 Data Provider Perspective

Data providers can make their datasets discoverable by uploading metadata profiles to our system. We adopted the Croissant dataset specification [1] as a schema for dataset profiles, an industry standard used across platforms such as Google Dataset Search, Kaggle, Hugging Face, and OpenML. To enable distribution-aware search, we extended the specification[1] with additional fields containing dataset and column statistics (e.g., histograms) within the "recordSet" key. Data providers can use tools such as the profile editor from the MLCommons Croissant working group[2] along with standard data profiling libraries to automate the process of creating dataset profiles. Our flexible profile schema allows data providers to omit optional fields while supporting them in adding additional fields to dataset profiles, assuming data consumers search for them.

2.2 System Operator Perspective

The system architecture of our demonstration consists of two components: a web-based user interface (UI) and a backend responsible for query parsing, optimization, and execution. We show an overview of the architecture and component interaction in Figure 2. While data providers and consumers primarily interact with the UI, system operators can monitor and configure the system via dedicated API endpoints to understand and optimize query execution.

When a data consumer issues a dataset search query, the backend first parses the query and constructs a query plan. Next, the query optimizer annotates and rewrites the plan to reduce execution time. While a principled approach to query optimization in dataset search engines remains an open research problem, our system explores different optimization techniques. For example, our optimizer uses heuristics to decide whether sibling operators should be executed in parallel or whether the output of one operator can be used as a filter for subsequent operators (e.g., because of a logical conjunction in the parent operator). Finally, the query executor receives the

[1] https://docs.mlcommons.org/croissant/docs/croissant-spec.html
[2] https://github.com/mlcommons/croissant/tree/main/editor

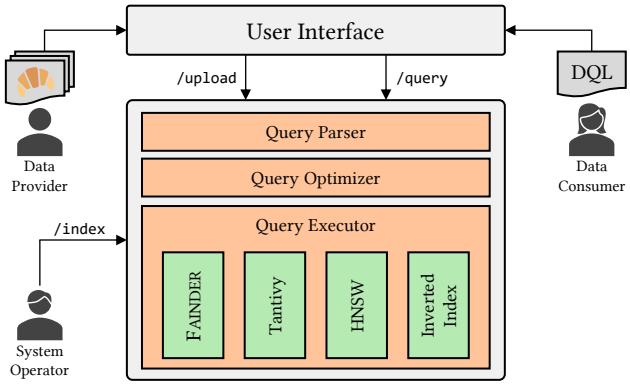


Figure 2: System architecture overview. Data providers upload Croissant files with dataset profiles, consumers run DQL search queries, and system operators configure the underlying indices to optimize performance and result accuracy.

annotated query plan and returns the collection of dataset profiles that match the query. Our prototype uses several index structures, such as HNSW [11] for semantic similarity search or FAINDER [4] for percentile predicates, to accelerate query execution.

Conceptually, FAINDER organizes a heterogeneous histogram collection by clustering similar histograms and transforming their bins to a unified cluster bin distribution. At query time, it employs a two-stage binary search: first over the cluster bins and then within the cumulative density array precomputed for each cluster during index construction. FAINDER supports multiple execution modes to balance efficiency and accuracy: FAINDER APPROX prioritizes runtime efficiency, delivering fast results with trade-offs in precision and recall, while FAINDER EXACT combines two approximate pruning stages with a final exact scan. For a more detailed explanation of FAINDER, refer to our previous publication [4].

In addition to the indices for percentile predicates and semantic similarity search, we use the full-text search library Tantivy to leverage decades of expertise in efficient keyword search instead of reinventing the wheel. Our query executor extracts keyword predicates from query plans, passes them to Tantivy, and incorporates the results, as well as Tantivy's result ranking, into the remaining query execution. System operators can configure their search engine instance by choosing appropriate parameters for each index structure. The parameters, such as the bin budget or the number of clusters for a FAINDER index, impact query execution time, hardware utilization, and result accuracy in the case of approximate indices. When new datasets are uploaded or data providers request their removal, system operators can periodically (e.g., daily) trigger an offline reindexing process.

2.3 Data Consumer Perspective

Data consumers primarily interact with our demonstration by running dataset search queries and reviewing search results. To facilitate these interactions, we developed the domain-specific language DQL, which powers our search engine's query interface. Our demonstration offers two methods for writing queries: a graphical query builder for intuitive and iterative search refinement and a DQL editor for advanced users. The graphical query builder simplifies the search process for data consumers by providing an interactive

```
KEYWORD('lung cancer') # 'lung cancer' included in dataset metadata
AND COLUMN( # has a column with the following properties
    NAME('age';3) # col name equals 'age' or its three closest semantic neighbors
    AND PERCENTILE(0.5;ge;60) # at least 50% of values are >= 60
) AND COLUMN(
    NAME('Gender';0) # col name exactly equals 'Gender'
    OR NAME('sex';0)) # or col name exactly equals 'sex'
```

Listing 1: Exemplary DQL query showing language features.

interface on top of DQL. Users can begin their search with generic keyword queries and iteratively narrow down the results by adding filters, such as percentile predicates. This allows them to explore intermediate results and refine their query accordingly.

For advanced users, directly writing DQL queries unlocks the full potential of the language's expressive capabilities. Listing 1 shows an exemplary dataset query in DQL. The design of DQL bridges existing keyword search approaches with advanced search operators, such as distribution-aware search, while maintaining extensibility for future innovations in dataset search. DQL offers several key advantages over existing search interfaces, including versatility, as it can be effortlessly wrapped with graphical UIs to cater to varying levels of user proficiency; extensibility, allowing new dataset search operators to be added seamlessly; declarativity, enabling execution plan optimization; and independence, as it is not tied to any specific environment, ensuring broad applicability.

Before executing a query, data consumers can select the FAINDER index mode (e.g., full precision or full recall) to suit their result quality and runtime requirements. Furthermore, DQL allows them to specify whether they prefer exact matching on column names or want to leverage our system's language embedding model to extend their search over columns similar to their column name predicates.

3 Demonstration Scenarios

Our demonstration[3] offers a range of scenarios, enabling the audience to explore our system from the perspectives of all three user roles. As data consumers, participants can experience the benefits of distribution-aware dataset search when querying an extensive dataset collection with a specific use case. As system operators, they can examine how FAINDER's construction and runtime parameters influence search results, execution time, and resource utilization. Finally, as data providers, they can create and upload new dataset profiles and observe their discoverability after reindexing.

To equip data consumers and system operators with a large and realistic dataset profile collection for distribution-aware search, we curated over 150 000 tabular datasets from Kaggle and augmented their existing Croissant files with additional dataset- and column-level statistics, including value counts and histograms[4].

Experiencing Distribution-Aware Dataset Search. Our demonstration presents the first end-to-end system for distribution-aware dataset search. Figure 3 collates a series of screenshots that illustrate the user experience within this scenario. Participants begin by exploring a dataset collection using high-level keyword queries. They then iteratively refine their search using our query builder, experiencing first-hand how distribution-aware search enhances their ability to discover relevant datasets. For each iteration, the UI

[3] https://github.com/lbhm/fainder-demo
[4] https://github.com/lbhm/dataset-scrapers

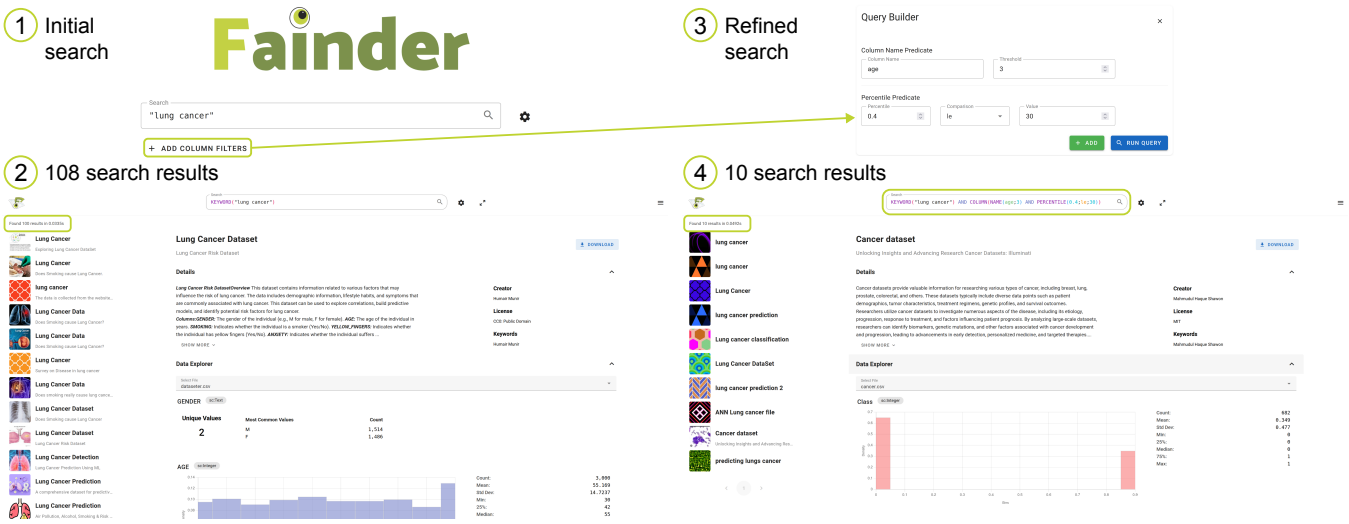


Figure 3: Overview of the dataset search experience. Data consumers start by writing high-level keyword queries (1) to get an overview of the dataset collection (2). Then, they iteratively refine their query using DQL or our graphical query builder (3), giving them a more manageable result set (4). In this example, the user searches for columns named similar to the term "age" where at least 40% of the values are smaller than 30. Data consumers can also specify advanced parameters for semantic search and percentile predicates, such as the number of neighbors to consider or the Fainder mode to use.

displays matching datasets and query statistics, helping participants become familiar with the dataset collection as they concretize their queries. For instance, they can add column name and percentile predicates to precisely search for intended datasets. Once comfortable with the system, participants can also write complex queries directly in DQL to experience the perspective of an advanced user.

Examining Different Fainder Configurations. As we describe in our research paper on Fainder [4], the index is highly flexible, offering a range of configuration parameters at both construction and execution time to address user needs. Our demonstration allows participants to interact directly with Fainder's configuration, providing an interactive experience to explore its customization options. To facilitate this exploration, we present a variety of index configurations to the audience, highlighting their trade-offs. Some configurations deliberately include suboptimal parameter choices to illustrate potential pitfalls in index tuning. Participants are encouraged to propose their own parameter settings and construct an index accordingly, fostering a hands-on understanding of the system's flexibility. Following index construction, participants can write their own benchmark queries and run them using different index configurations and execution modes. Through the UI, they can examine the impact on result quality, while system operator logs provide insights into execution statistics and hardware resource consumption. This comprehensive approach gives participants a holistic view of the trade-offs and considerations involved in configuring Fainder.

Providing New Datasets and Reindexing. To address all user perspectives with our demonstration, we offer pre-filled Croissant files containing additional dataset profiles to provide participants with a hands-on experience of the data provider perspective. Participants can modify these templates as desired and upload them to the search engine. Afterward, they can trigger the recreation of search indices, observing the index creation process behind the scenes as their newly submitted datasets are incorporated into the search results. Finally, participants can run new search queries to examine when and how their uploaded datasets appear in the results.

Acknowledgments

We gratefully acknowledge funding from the German Federal Ministry of Education and Research under the grants BIFOLD24B and 01IS17052 (for the Software Campus project FDaaS).

References

[1] M. Akhtar, O. Benjelloun, C. Conforti, et al. 2024. Croissant: A Metadata Format for ML-Ready Datasets. DEEM. DOI: 10.1145/3650203.3663326.
[2] A. Asudeh and F. Nargesian. 2022. Towards Distribution-aware Query Answering in Data Markets. *Proc. VLDB Endow.*, 15, 11, 3137–3144. DOI: 10.14778/3551793.3551858.
[3] S. A. Azcoitia and N. Laoutaris. 2022. A Survey of Data Marketplaces and Their Business Models. *SIGMOD Rec.*, 51, 3, 18–29. DOI: 10.1145/3572751.3572755.
[4] L. Behme, S. Galhotra, K. Beedkar, and V. Markl. 2024. Fainder: A Fast and Accurate Index for Distribution-Aware Dataset Search. *Proc. VLDB Endow.*, 17, 11, 3269–3282. DOI: 10.14778/3681954.3681999.
[5] S. Castelo, R. Rampin, A. Santos, et al. 2021. Auctus: A Dataset Search Engine for Data Discovery and Augmentation. *Proc. VLDB Endow.*, 14, 12, 2791–2794. DOI: 10.14778/3476311.3476346.
[6] R. Castro Fernandez, Z. Abedjan, F. Koko, et al. 2018. Aurum: A Data Discovery System. ICDE. DOI: 10.1109/ICDE.2018.00094.
[7] Dawex. 2024. Data Marketplaces, Data Hubs & Data Spaces. Retrieved Feb. 6, 2024 from https://www.dawex.com/en/.
[8] M. Esmailoghli, C. Schnell, R. J. Miller, and Z. Abedjan. 2025. Blend: A Unified Data Discovery System. To appear in ICDE. DOI: 10.48550/arXiv.2310.02656.
[9] M. Hulsebos, W. Lin, S. Shankar, and A. Parameswaran. 2024. It Took Longer than I was Expecting: Why is Dataset Search Still so Hard? HILDA. DOI: 10.1145/3665939.3665959.
[10] J. Kennedy, P. Subramaniam, S. Galhotra, and R. C. Fernandez. 2022. Revisiting Online Data Markets in 2022. *SIGMOD Rec.*, 51, 3, 30–37.
[11] Y. A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42, 4, 824–836. DOI: 10.1109/TPAMI.2018.2889473.
[12] N. Noy, M. Burgess, and D. Brickley. 2019. Google Dataset Search: Building a Search Engine for Datasets in an Open Web Ecosystem. WWW. DOI: 10.1145/3308558.3313685.