A Graph RAG Approach to Enhance Explainability in Dataset Discovery

Data Science and Engineering (2026) 11:30–52
https://doi.org/10.1007/s41019-025-00313-x

RESEARCH PAPERS

A Graph RAG Approach to Enhance Explainability in Dataset Discovery

Claudia Diamantini[1] · Alessandro Mele[1] · Alex Mircoli[1] · Domenico Potena[1] · Cristina Rossetti[1,2] · Emanuele Storti[1]

Received: 16 February 2025 / Revised: 10 July 2025 / Accepted: 12 August 2025 / Published online: 13 November 2025

Abstract
Discovering relevant datasets in large, heterogeneous data ecosystems, such as Data Lakes or Data spaces, is a complex task, often hindered by a lack of transparency and user-centric explanations in the discovery process. Explainability is critical for enabling users to understand why specific datasets are recommended, what information they contain, and how they align with user-defined criteria and preferences. To address these challenges, this work proposes a novel Graph Retrieval-Augmented Generation (Graph RAG) framework to enhance explainability in a platform for discovery of summary data sources. The proposed approach leverages a Knowledge Graph (KG) to interpret user requests, extracting relevant contextual information. These enriched requests are then transformed by a Large Language Model (LLM) into actionable dataset queries for a dataset discovery platform. Candidate solutions are evaluated and enriched with statistical insights on value distributions and contextual knowledge from the KG. Finally, the LLM ranks these solutions based on user preferences, producing a final report. This dual strategy of query enrichment and contextual explanation fosters transparency and enhances user understanding of the discovery process. We demonstrate the effectiveness of the approach through an experimental validation, highlighting its potential to improve both the accuracy and interpretability of dataset discovery.

Keywords Dataset discovery · Knowledge graph · Retrieval augmentation generation · LLM · Multidimensional model

1 Introduction

In today's data-driven landscape, organizations increasingly rely on vast and heterogeneous collections of datasets to support analytical and decision-making processes. Enabling technologies feature both centralized Data Lakes and decentralized Data Spaces, which have a great potential to facilitate data sharing and integration across organizations, and offer flexible and scalable storage for diverse data sources. However, their flexibility introduces significant challenges in data management [49], including achieving full data interoperability and making sense of disparate data through valuable and interpretable insights. Key factors in addressing these challenges include the ability to represent heterogeneous datasets and their metadata as a support for data governance, as well as to provide mechanisms for discovering, accessing, and analyzing relevant datasets. Overcoming these challenges requires not only effective dataset discovery approaches but also transparent and explainable systems that help users understand why specific datasets are relevant to their queries and how they align with analytical goals. As a matter of fact, providing clear, user-centric

✉ Emanuele Storti
e.storti@univpm.it

Claudia Diamantini
c.diamantini@univpm.it

Alessandro Mele
a.mele@pm.univpm.it

Alex Mircoli
a.mircoli@univpm.it

Domenico Potena
d.potena@univpm.it

Cristina Rossetti
cristina.rossetti@polito.it

[1] Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, via Brecce Bianche, Ancona 60131, Italy

[2] Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino 10129, Italy

explanations is indeed crucial for improving transparency, facilitating informed decision-making, and enabling effective utilization of data resources. While explainability is a key aspect of dataset discovery, it remains underexplored in most existing systems. Current solutions do not often adequately contextualize results and, when multiple datasets meet query criteria, users are left with little support in comparing alternatives or understanding the content and structure of the results and the relevance to their requests. This is particularly critical for huge repositories of summary data sources, namely datasets including statistical content. Such peculiar data sources are characterized by a set of *measures* (also called indicators in the following), that is data of interest for the analysis that are aggregated along a number of *dimensions*, typically at different granularities called *levels*. Such data sources are also termed *data cubes*. The integration of different data cubes may be done by joining data items on the (sub-) set of dimensions, in order to compare and analyse different measures. Inconsistent, sparse, or noisy data cubes are more the rule than an exception, and can make data discovery and integration a time-consuming and frustrating endeavor.

Let us consider for example a set of data sources related to global pollution, and let us suppose a user wants to compare particulate matter emissions, namely $PM_{2.5}$ and PM_{10}, on different countries and years, focusing in particular on European countries and the last 3 years. Besides traditional issues like schema heterogeneity (e.g. $PM_{2.5}$ column can be named $particulate\_matter\_2.5$ in a source, and $Fine\_particulate\_matter$ on another) the production of data at different pace, or at different aggregation levels can make the integration of apparently joinable sources a useless effort, producing a table formed of very few results, and/or concentrated on ranges of level values of little interest. Hence, in the presence of different alternatives, discovery should be able to return the user a list of possible alternative datasets, ranked by relevance with respect to user preference (such as prioritizing datasets covering countries in Europe and years 2023-2025, and be able to explain the adopted criteria in a synthetic, user-comprehensible form.

To address these issues, we propose a novel approach to enhance explainability in a platform for discovering summary data sources, that combines the expressive capabilities of Knowledge Graphs (KGs) and advanced Natural Language Processing through a Graph Retrieval-Augmented Generation (RAG) approach. A KG is leveraged to interpret and contextualize user requests expressed in natural language, extracting dimensions, levels, and indicators relevant to the request. This enriched query is then translated by a Large Language Model (LLM) into actionable dataset queries targeting the dataset discovery platform. The query results, in terms of alternative combinations of data sources capable of addressing the user request, are further enriched with detailed metadata, including value distributions (e.g., the percentage of records associated with a specific value) and semantic relationships (e.g., hierarchical groupings of entities, such as Italy being part of Europe). These enriched results are then processed by the LLM for ranking them based on explicit user preferences, ensuring alignment with analytical priorities (e.g., datasets focused on European countries). By integrating this dual-layered approach of query contextualization and result enrichment, the proposed framework is capable of both interpreting user requests related to dataset discovery and addressing the critical need for result explainability. This also aligns with the principles of Data Spaces by promoting semantic interoperability, contextual enrichment, and transparency in data governance processes [33].

The contributions of this paper are as follows:

- we define a Knowledge Graph representation model. Part of the graph captures static (1) background knowledge, including definitions of dimensions and indicators. Another part describes dynamic (2) source-related knowledge including metadata for the characterization of data sources, including source profiles, i.e., summary representations of their content aligned with the KG.
- We introduce a novel Graph RAG approach that integrates Knowledge Graphs and LLMs to contextualize and enrich the user request. The LLM is exploited for multiple objectives: (1) to translate the user request, expressed in natural language, in a properly formatted query delivered to the dataset discovery platform, (2) to filter and rank the results according to the user preferences and (3) to enhance the explainability of dataset discovery results by providing statistical and semantic insights to users.
- We perform an experimental evaluation of the proposed approach, comparing multiple LLMs and different scenarios to assess its effectiveness across key dimensions, including the correctness of the provided ranking and its explainability, in terms of coherence, general quality and compactness of the responses.

The manuscript extends previous work on dataset discovery [13] by extending the representation capabilities of the Knowledge Graph and by building a Graph RAG solution upon it , leveraging the Knowledge Graph to extract relevant contextual information. This helps to improve the usability, precision and interpretability of dataset recommendations. A further extension involves integrating user preferences in the request, ensuring the results align with specific analytical goals.

The rest of this paper is organized as follows. Section 2 reviews related work in dataset discovery, explainability of human-agent systems, Knowledge Graphs, and the use of Graph RAG. Section 3 describes the data model of the platform, while Sect. 4 discusses the proposed methodology, detailing the Graph RAG framework and its components. Section 5 evaluates its performance through experiments. Finally, Sect. 6 concludes the paper and outlines future research directions.

2 Related Work

Modern data ecosystems demand robust mechanisms enabling end-users to collect, explore, and analyze data sources suited to their specific needs. The effectiveness of these solutions is deeply influenced by how metadata associated with data sources is managed. Metadata models play a pivotal role in representing essential information such as data provenance, structure, semantics, and relationships, thereby facilitating advanced dataset discovery functionalities and enhancing result explainability. Metadata, in the form of a graph, can then be leveraged to enrich Retrieval-Augmented Generation for LLMs, ultimately aimed to deliver more relevant results and foster a deeper understanding of the relationships and patterns within metadata. In the following, we survey relevant related literature on these topics.

2.1 Dataset Discovery and Exploration

Dataset discovery is the process which allows users to find data sources that are relevant to a user query. This process can be challenging for several reasons, such as the heterogeneities of data sources or the huge amount of stored data. Additionally, the user might not know in advance what to specifically search, therefore tools to effectively support the end users in discovering new data sources are needed.

The simplest approach for dataset discovery, widely employed by data repositories and for web tables, relies on keyword search. This method can be applied to metadata or, when accessible, to the dataset's source content (see survey by Chapman et al. [10]). In a second group of approaches, the input for the search is represented by a dataset, which can be provided by the user or produced as a result of a previous analysis task. In this way, source integration and source discovery are addressed in a combined way, following the idea of finding datasets that are similar to a query dataset and that can be integrated in some way (either by joins, unions or aggregates). For instance, in the context of open datasets, Table Union Search [34] aims at identifying the k tables that have the highest likelihood of being *unionable* with a search table T on some subset of attributes, by considering the similarity between domains and values of the attributes. The approach relies on Locality-Sensitive Hashing (LSH, Zhu et al. [57]) to obtain an approximate yet efficient solution. Other work focuses on joinable dataset search, e.g., D3L [6] in which both schema– and instance–based features are used [44], or Auctus [8] where data summaries are in part exploited. Finally, some papers propose a *query-based search*, which relies on a specific query language and a model, that in most cases is a graph [9, 17, 19]. As an example, a Knowledge Graph is exploited in Aurum [9] to profile the data sources in a repository and find similarity-based relationships and primary-foreign key candidates. For each column, a profile is built by generating signatures from extracted information (e.g., cardinality, data distribution, MinHash [7]). These signatures are then indexed using Locality-Sensitive Hashing: if two columns share a bucket, an edge is created between the corresponding nodes, with the similarity value as the edge weight. Users can then search datasets using a Source Retrieval Query Language. The KGLac architecture [19] enhances data discovery operations, such as schema similarity search or joinable tables discovery, through a Knowledge Graph where stored relationships are computed exploiting metadata by a data profiler which calculates content-based and semantic-based embeddings of datasets.

A related problem is the *correlated dataset search*, where an additional challenge arises: besides identifying possible joins, it is also necessary to compute, or estimate, the joinability (or correlation). As a measure of joinability, most approaches propose the Jaccard Index and the Jaccard Containment similarity. While some algorithms provide an exact solution to the problem, e.g., JOSIE [58], scalability issues can be addressed by providing approximate answers, e.g., Lazo [16], Asymmetric MinWise Hashing (ALSH, Shrivastava and Li [48]) and LSH Ensemble [57], balancing precision and recall. Such techniques apply indexing structures and data sketches (typically through hashing) to reduce the dimensionality of the datasets and perform time-effective estimation of the Jaccard index or the containment set. In a previous work of ours [13], we presented a framework for analytic query-driven dataset discovery in Data Lakes, focusing on LSH Ensemble-based efficient integration and mapping of summary data sources to a Knowledge Graph, which serves as a global model following a Local-as-View approach. Analytic queries are expressed in terms of target indicators and relevant dimensions, and the system identifies combination of suitable data sources. Compared to the mentioned approaches, the framework is capable of preselecting only relevant sources by exploiting mappings, avoiding pair-wise comparisons among a large number of sources.

2.2 Explainability in Human-Agent Systems

In the domain of human-agent interaction, explainability has emerged as a critical non-functional requirement, essential for promoting transparency, accountability, and trustworthiness. It also contributes to the overall usability and auditability of software systems [42]. Despite its recognized importance, a precise and universally accepted definition of explainability from a software engineering perspective is currently missing. Likewise, systematic methods for its elicitation and validation are still under development, although recent efforts have attempted to conceptualize and model its various dimensions [5].

Research on explainability has predominantly gained momentum within the context of machine learning and artificial intelligence (e.g., XAI), where systems often produce non-deterministic outputs subject to generalization, abstraction, inference, and ambiguity. Nevertheless, explainability is equally pertinent to other classes of systems, such as information retrieval and recommendation engines, especially if they are supported by AI models, such as LLMs. In these contexts, providing clear justifications for system outputs, especially in relation to user queries or preferences, is essential to ensuring the perceived appropriateness and relevance of results. As several work report, e.g. Chapman et al. [10], even when users find a solution to a query, they might struggle with issues like trustworthiness (e.g., concerns about data quality, accuracy, or reliability) or may encounter practical difficulties in using these datasets, e.g. for unclear documentation. In these cases, much time risks to be devoted to examining and comparing results [31]. To this aim, text summaries are often used, along with summary statistics or visualisations, as a way to explain a complex result to users. In particular, conversational search and LLMs have proven particularly useful in cases where users have insufficient background knowledge [46]. In more complex cases, the information retrieval task at hand may require an entire set of results from multiple different sources with interdependencies, as in the case of a dataset discovery platform. To the best of our knowledge, however, no prior work has explicitly addressed explainability as a first-class objective in dataset discovery. Existing methods often focus on usability or metadata-driven retrieval, but fall short of providing user-centered explanations justifying dataset recommendations. Indeed, existing dataset discovery platforms either provide no explanation of the results to the user or offer only very limited support, e.g., visual exploration of datasets is possible in Auctus [8], including previews, summaries, and spatial visualizations but only for single sources, while KGLac [19] offers entity-level tracing through KG connections, and Aurum [9] supports provenance and lineage tracing for each match.

2.3 Data Models and Vocabularies for Profile Representation

Leveraging well-designed metadata models ensures that data from diverse sources can be integrated, making it easier for users to navigate large and complex datasets within data ecosystems. Therefore, the role of metadata management extends beyond simple documentation, functioning as a cornerstone for the usability, interpretation, and interoperability of data systems. Effective data governance, therefore, requires a well-defined metadata model that aligns with the FAIR principles (Findable, Accessible, Interoperable, and Reusable) [53]. In the context of Data Lake literature [45], functional metadata are classified by Oram et al. [37] as technical (i.e., data format, structure or schema), operational (i.e., data location, file size, number of records) and business-related (i.e., field names and explanations), although such classes are not entirely distinct and can overlap [14]. On the other hand, structural metadata are associated with the concept of object, a generalization of a data source that can be structured (e.g., SQL tables), semi-structured (e.g., XML, JSON), or unstructured (e.g., images, audio or video).

Metadata are also core components of other data-sharing ecosystems like Data Space architectures. In fact, Data Spaces add decentralized governance to Data Lakes, facilitating the sharing and exchange of data across different organizations without the need of a single, centralized repository. This makes FAIR principles even more important and challenging.

Several vocabularies and ontologies have been proposed in the literature to describe (statistical) data sources [21]. Among them, "Vocabulary of Interlinked Datasets" (VoID)[1] is an RDF Schema vocabulary for expressing high-level metadata about RDF datasets, such as the vocabularies used in the dataset and the size of the dataset. Similarly, Dublin Core Metadata Initiative (DCMI) Terms[2] aims at interoperable metadata descriptions for resources in the Web, also including basics data provenance metadata. RDF Data Cube[3] is an RDF Schema vocabulary based on the SDMX standard designed for representing statistics in a multidimensional attribute space, while DCAT[4] is an RDF vocabulary designed to facilitate interoperability between data catalogues published on the Web. For each source within the catalogue, some basic metadata are provided, i.e., resource URL, title, file type and dimension. For what concerns tools for statistics generation, LODstat [2] provides schema and data statistics for RDF data sources, i.e., number of triples,

[1] https://www.w3.org/TR/void/.

[2] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#.

[3] https://www.w3.org/TR/vocab-data-cube/.

[4] https://www.w3.org/TR/vocab-dcat-3/.

triples containing blank nodes, or literal values with the respective data types. PROLOD++ [1] is a tool for profiling and mining RDF data sources. Profiles provide statistics such as frequency counts and distributions related to subjects, predicates and objects. Similarly, RDFStats [28] is a generator for statistics of RDF data sources, which provides histograms of classes, properties and value types. In Zhao et al. [56], a set of metadata to support data discovery and analysis in a Data Lake context are proposed. Metadata like data types, distribution of values, relationships between data sources such as similarities, are computed to provide users with a better understanding of data in order to select the appropriate datasets for their purposes. A measure of similarity/dissimilarity among datasets is computed and stored as a metadata to support dataset search. We emphasize that the mentioned vocabularies and tools do not fully provide support for the type of metadata we are interested in, since they are not tailored to the description of the peculiar characteristics of multidimensional data sources, namely facts, indicators and dimensions.

2.4 Graph-Based Retrieval-Augmented Generation

Large Language Models (LLMs) marked a revolution in the field of Natural Language Processing (NLP), due to their capabilities in language processing and text generation. The training of an LLM consists of providing the model with large amounts of textual data (*corpora*) as input. However, this paradigm suffers from certain issues: (i) the LLM will incorporate the knowledge acquired up to that point, and (ii) since *corpora* are general-purpose texts, the model could easily fail into hallucinations [24] with domain-specific queries. Since training LLMs is extremely expensive, incorporating custom knowledge into them is a nontrivial challenge. Several solutions have been proposed in the literature to address this problem, including fine-tuning, i.e., training the LLM in a task-specific dataset [20, 22, 38, 41], Retrieval-Augmented Generation (RAG), i.e., enriching the prompt with information extracted from external knowledge bases [30], and prompt engineering, i.e., supporting the model by providing detailed instructions for the task [25, 52, 55]. RAG emerges as dominant solution, allowing updates to the LLM knowledge base with no necessity of retraining the model. Specifically, RAG dynamically queries external knowledge bases, extracts domain-specific knowledge, and incorporates it into the LLM output. Although RAG achieves excellent results and is widely used, it is highly sensitive to noise in documents [40] and suffers from some limitations in scenarios where documents are, in some way, interconnected each others, e.g., social media postings, research papers [23, 39]. Graphs represent the candidate data structures for modeling knowledge bases: they reduce verbosity and naturally represent connections between entities (e.g., documents or complex objects). To combine the potential of graphs with traditional RAG, Graph Retrieval-Augmented Generation (Graph RAG) techniques have been proposed in the literature [15, 23, 32]. According to Procko and Ochoa [40], Graph RAG techniques can be categorized into three main types: Graph-Based Indexing, Graph-Guided Retrieval, and Graph-Enhanced Generation. Graph-Based Indexing techniques aim at constructing the initial knowledge base, which can be achieved either by building the graph database from scratch for specific tasks or leveraging specific Web resources such as Open Knowledge Graphs. A crucial aspect is the definition of indices between the entities, which helps improve the efficiency and effectiveness of the search stage. Graph-Guided Retrieval, on the other hand, focuses on selecting the candidate graphs from external graph databases. Finally, Graph-Enhanced Generation aims to combine the retrieved graph data with the query to enhance the quality of the result. Graph RAG techniques are widely adopted in various NLP tasks, e.g., Question Answering (QA), Information Extraction, and Dialogue systems [39]. Most of the literature in the Graph RAG scenario is concerned with QA [40], traditionally addressed by Knowledge Graphs. The combination of LLMs with Knowledge Graph Question Answering (KGQA) frameworks is a commonly researched application of RAG [3, 15, 18, 47, 54]. Among them, G-Retriever [18] offers a flexible QA framework to interact with a graph through a conversational interface and addresses the issue of hallucination with a Prize-Collecting Steiner Tree optimization problem. Bahr et al [3] developed a framework to leverage analytical and semantic QA capabilities on Failure Mode and Effects Analysis (FMEA) data, by defining an ontology for FMEA observations, an algorithm for creating vector embeddings from the FMEA KG, and a KG enhanced RAG framework. KG-RAG [43] is a framework for KGQA which derives a Knowledge Graph from unstructured text and then performs information retrieval over it. The retrieval methodology leverages the Chain of Explorations (CoE) algorithm which benefits from LLMs reasoning to explore nodes and relationships within the Knowledge Graph sequentially.

3 Data Model

In this work, we consider a data ecosystem comprising multiple data sources. Since the focus is on summary/statistical datasets, we assume a generic structure based on the multidimensional model [26], where indicators are described along dimensional hierarchies. The approach relies on a Knowledge Graph based on the RDF representation model. The Knowledge Graph is structured in two distinct layers,

called *Background Knowledge Graph* (BKG) and *Source-related Knowledge Graph* (SKG). The former provides definitions of indicators and dimensions, that constitute stable information within the data ecosystem. The latter provides a description of source metadata, mappings to elements of BKG, as well as source profiling. As such, it is specific to each dataset and more subject to change.

3.1 Background Knowledge Graph

Hereby, we provide the notions of dimension, related levels and members, and indicators.

Definition 1 (Dimension) A dimension $\mathcal{D}_i \in \mathbb{D}$ is a perspective of analysis and is defined as the set $\mathcal{D}_i = \{L_1^{\mathcal{D}_i}, \ldots, L_n^{\mathcal{D}_i}\}$, where $L_j^{\mathcal{D}_i}$ is a (dimensional) level of $\mathcal{D}_i$.

For the sake of simplicity, in the following we refer to a dimensional level as L_j, when the dimension is clear from the context. Levels of $\mathcal{D}_i$ are organized in a hierarchy through a *partial order* $\leq_L$, a reflexive, antisymmetric, and transitive relation defined over $\mathcal{D}_i$. To make an example, given levels *region* and *country* of the geographic dimension (*GEO*), the relation *region* $\leq_L$ *country* holds.

Given a level L_j, we denote by $\delta(L_j) = \{m_1^{L_j}, \ldots, m_q^{L_j}\}$ the set of its *members*, i.e., elements of a level. Likewise, a member will be denoted as m_p, whenever the dimensional level it belongs to is evident from the context. For example, $Venice \in \delta(city)$ states that *Venice* is a member of the level *city*.

Given a dimension $\mathcal{D}$, levels $L_i, L_j \in \mathcal{D}$ and a partial order relation $\leq_L$ on $\mathcal{D}$, a relation $partOf_{L_i,L_j}$ can be defined among the set of corresponding members $\delta(L_i)$ and $\delta(L_j)$. This relation enables the roll-up operator along $\mathcal{D}$.

Definition 2 (Indicator) An indicator $ind_i \in \mathcal{I}$ is a quantitative measure that can be related to one or more dimensions $\mathcal{D}_j \in \mathbb{D}$.

In our example case about global pollution, background knowledge consists of the geographical dimension (*GEO*) and the temporal dimension (*TIME*). *GEO* is articulated in a hierarchy composed of four dimensional levels (*city* $\leq_L$ *region* $\leq_L$ *country* $\leq_L$ *continent*), while *TIME* is composed of three dimensional levels (*day* $\leq_L$ *month* $\leq_L$ *year*). Furthermore, some indicators are defined, namely CO_2, $PM_{2.5}$ and PM_{10}.

Background knowledge is encoded in the BKG and is based on KPIOnto[5], an OWL2-RL ontology that provides the terminology for defining indicators and some additional properties (e.g., description, unit of measure, objective), along with dimensional hierarchies and members. In particular, class `kpi:Level` defines a level which includes a number of `kpi:Members` and belongs to a `kpi:Dimension`. The Object Property `kpi:rollUp` encodes the partial order relation among levels. In the bottom of Fig. 2, an example is reported on the representation of the dimensional hierarchy for the dimension GEO, where namespace "kpi" refers to KPIOnto and "bkg" to the BKG.

3.2 Source-related Knowledge Graph

We provide here the definition of data source and the notion of profile, along with the description of the graph schema adopted for their representation.

Definition 3 (Data Source) A data source is defined as $S_k = \{a_1, \ldots, a_n\}$ where $a_i \in S_k$ is an attribute and $\nu(a_i) = \{v_1, \ldots, v_m\}$ is the set of values of a_i.

The characterization of a source includes also the indicators and levels of the Knowledge Graph which the attributes of the source can be mapped to. With reference to the example case, let us assume a data source includes attributes *nation* and *day* that can be mapped respectively to the levels *GEO.country* and *TIME.day*, and an attribute *particulate_matter_2.5* that can be mapped to the indicator $PM_{2.5}$. As discussed in [13], mappings for dimensional attributes can be discovered in a semi-automatic fashion, leveraging approximate solutions based on hashing schemes. This approach identifies, for a given attribute a_i, the dimensional level L_j whose members $\delta(L_j)$ most closely matches its values $\nu(a_i)$. We denote by $ind(S_k)$ and $dim(S_k)$ the set of indicators and dimensions of the Knowledge Graph, respectively, that are mapped to attributes in S_k.

A further relevant source-related information regards a summary representation of the content of a data source, namely its profile.

Definition 4 (Profile item) Given a dimension $\mathcal{D} \in \mathbb{D}$, a level $L_j \in \mathcal{D}$, a source S_k, an attribute $a_i \in S_k$ mapped to L_j and a member $m_p \in \delta(L_j)$, the profile item of a_i for the member m_p is defined as $\langle m_p, y_p \rangle$, where $y_p = \mid \{v \in \nu(a_i) : v = m_p\} \mid$.

We refer to $\Psi(a_i) = \{\langle m_1, y_1 \rangle, \ldots, \langle m_n, y_n \rangle\}$ as the profile of the attribute a_i, i.e., the set of its profile items. The

[5] https://w3id.org/kpionto/.

profile represents the frequency with which each value of a_i occurs. It is defined only for attributes that can correctly be mapped to dimensional levels.

Source-related knowledge is encoded in the SKG based on an ontological vocabulary. The vocabulary defines classes and properties for the definition of a source, to represent its structure in terms of attributes, and to map an attribute to the corresponding indicator or level in the BKG. The vocabulary allows for a uniform representation of heterogeneous sources. This accommodates both structured data sources (such as SQL databases) and semi-structured data sources (such as JSON or XML files) in a consistent manner. This uniformity simplifies the integration and querying of diverse datasets, thus enhancing overall interoperability. An example is shown in Fig. 1, where the prefix "dl" denotes the namespace for the vocabulary, while "skg" refers to the namespace of the SKG. Each data source is represented as an instance of the `Source` class, which extends the class `void:Dataset` from the VoID vocabulary to provide enhanced integration capabilities. Descriptive metadata are reused from the DCMI standard and include the `title` of the source, its `description`, the file `format` and a set of `subjects` (typically linking to DBpedia resources). Additional metadata include `creator`, `publisher` and any `contributors`, as well as the creation `date` and the `license`. Basic statistics such as the number of dimensions are represented through the property `dl:domains`, while the cardinality through the property `dl:items`. Further metadata include the file path of the data source (`dl:location`), provenance and lineage information, and structural metadata. Structural metadata define each attribute as an instance of the `dl:Domain` class, with a connection with its source through the property `dl:contains`.

Attributes related to an indicator or a dimension, on the other hand, are instances of the `Domain` class. A property `mapTo` allows to map the attribute to a corresponding indicator or level in the BKG. This is exemplified in Fig. 2 for an attribute mapped to level *GEO.country*. A profile for a dimensional attribute is defined as an instance of the `DProfile` class and includes summary information on its value distribution as a set of `DProfileElement` (corresponding to profile items). In turn, this is linked to a member (property `toMember`) and a number of occurrences (property `dl:frequency`). A special profile element is defined for all values that cannot be mapped to any member. In this case, a special value `dl:others` is adopted. In the example in Fig. 2, the "nation" attribute includes 282501 instances of member "Germany" of the GEO.country level, 283549 instances of member "Italy", and 10345 other values which cannot be aligned to any member.

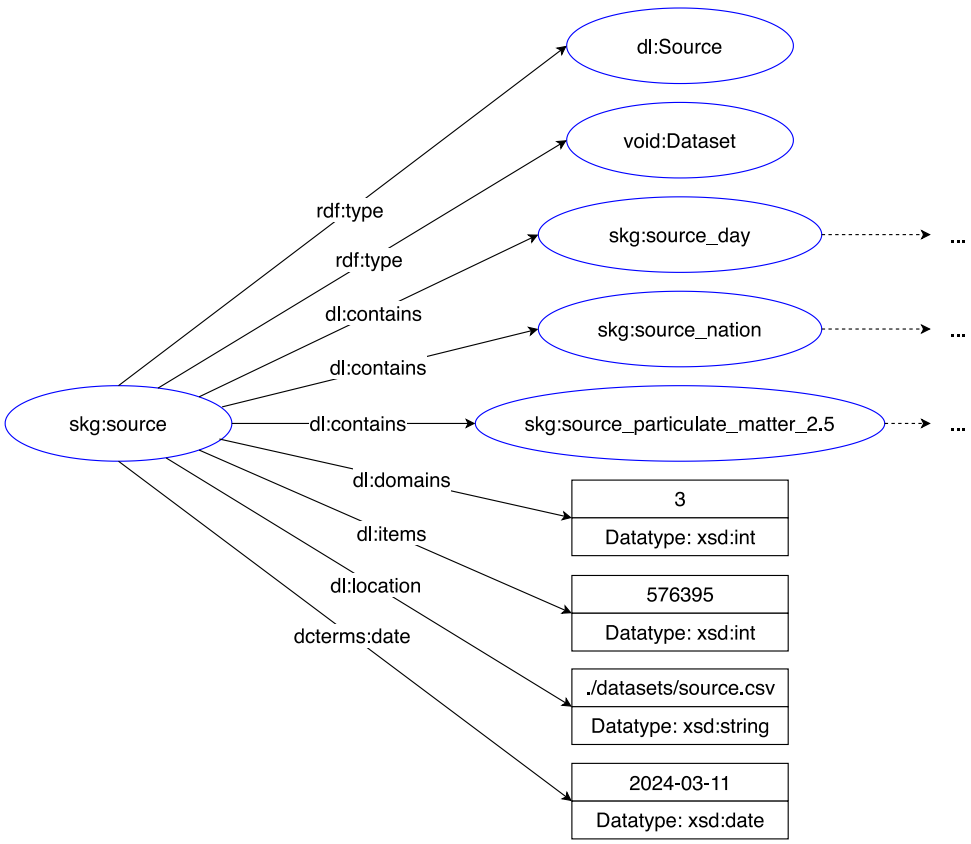

Fig. 1 Example of source-related knowledge representation (source metadata) in the SKG. Dashed lines link to attribute metadata

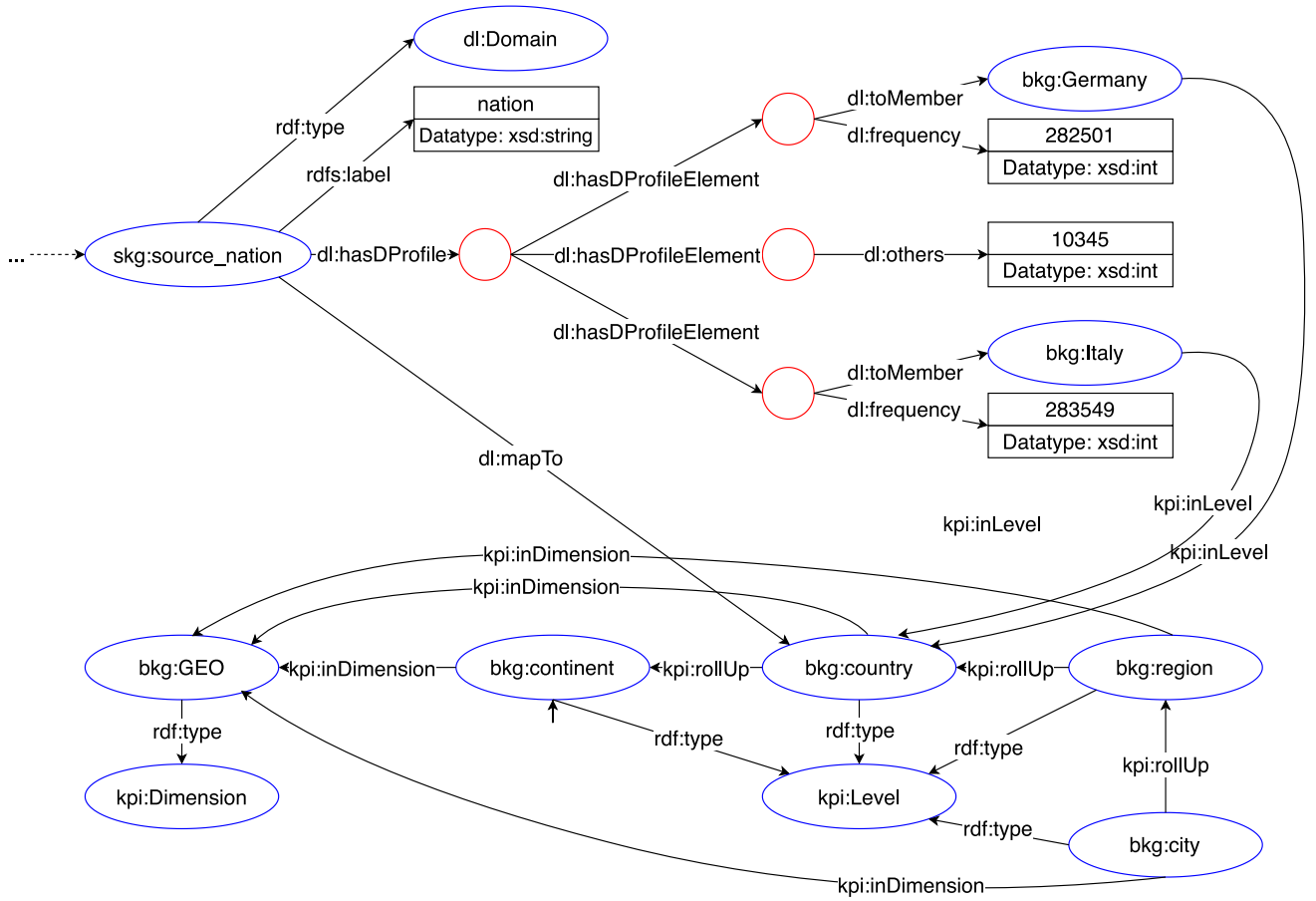

Fig. 2 Example of source-related knowledge representation in the SKG (attribute metadata)

4 Methodology

The proposed approach for enhancing explainability in dataset discovery is structured around a Graph Retrieval-Augmented Generation (Graph RAG) framework. As shown in Fig. 3, the architecture integrates multiple components to transform user requests in natural language into actionable and explainable dataset discovery results. The contextualization of the user request ensures the system fully understands the user's intent and prepares the request for downstream processing.

The LLM plays a dual role in the approach. First, it converts enriched user requests into executable queries. Second, it synthesizes the final results, incorporating statistical insights, semantic explanations, and user preferences into a coherent response. Its natural language capabilities are key to maintaining a user-centric, explainable interface.

The workflow is managed by a module for coordination, which is not depicted in Fig. 3. It is in charge of providing input to the next stage and collecting the relative output, thus serving as an orchestrator of the workflow.

4.1 Request Formulation and Enrichment

The initial step of the workflow involves the definition of the user request and its enrichment through background knowledge from the Knowledge Graph, with the aim to produce the starting prompt to the LLM.

The request, representing the objective of the analysis, is expressed by the user in natural language. We do not assume that the user has full knowledge of the indicators available in the data sources or the analysis dimensions these sources include, nor their hierarchical structure across levels. Similarly, while the users may receive guidance through a GUI, typically in the form of examples, they are free to formulate their request using their preferred linguistic structure and syntax, without any predefined constraints. An example of an analytical request is *"I want to analyze the trends of PM2.5 air pollution by country and on a daily basis"*, where the subject of measurement can be expressed in various alternative wordings, such as *"particulate matter below 2.5 micron"* or *"fine particulate"*. Similarly, the dimensions may be phrased differently, e.g., *"nation and day"* or *"daily at country-level"*.

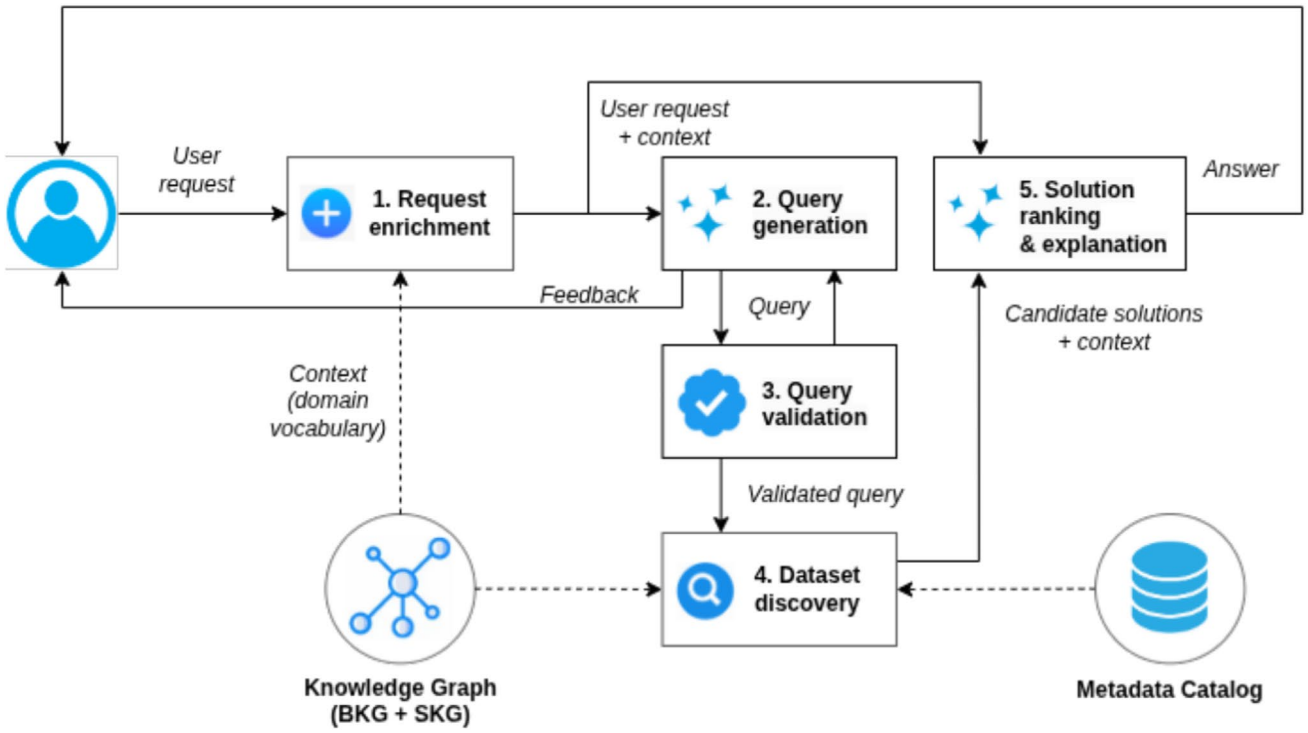

Fig. 3 Architecture of the Graph RAG approach

The request can be supplemented by one or more preferences about the content of the data sources, such as *"I am particularly interested in results for Europe in the first half of 2024"*. Preferences are used at a later stage of the workflow to drive the ranking of the alternative results.

The approach is summarized in Algorithm 1. The enrichment of the user request involves the extraction of contextual knowledge from the Knowledge Graph about the terminology used in the organization for indicators (line 1), dimensions and corresponding levels (line 2). Extracted information for indicators includes their full name along with properties such as the textual description, the unit of measure and the measurement objective. To make an example, the following information are added to the prompt about the indicator CO2: "CO2 (Carbon Dioxide, greenhouse gas), measured in ppm (parts per million)". As for dimensions, their names and the set of their levels are extracted and used for the enrichment, e.g. "Dimension 'GEO' has levels: continent, country, region and city." Since the size of the dimensional hierarchies is not critical in most real-world scenarios, a *static* enrichment with all indicators in $\mathcal{I}$, dimensions in $\mathbb{D}$ and corresponding levels in the Knowledge Graph is suitable for most practical cases. A more *dynamic* approach to enrichment can be employed when the size of the dimensional hierarchy exceeds the maximum length allowed for the prompt of the LLM or to limit its number of tokens. In these situations, the request can be preliminary interpreted and categorized according to some criteria. For instance, in multi-purpose data catalogs organized on a domain basis, the indicators and dimensions most relevant to the identified topic of interest can be prioritized and included as context, e.g., if the request deals with environmental pollution, indicators on gender balance or employment rate are unlikely to be useful for the analysis, and can be left out of the contextual part of the prompt. Further practical optimizations can be devised, such as employing a stateful or hybrid interaction mode with the LLM, reusing (a part of) the context.

4.2 Query generation and validation

The following step consists in translating the user request, complemented with the encoded contextual knowledge, into a valid query for the dataset discovery platform. In the following, we introduce the notion of well-defined query.

Definition 5 (Well-defined query) A well-defined query is a tuple $Q = \langle \mathcal{I}_Q, \mathcal{L}_Q \rangle$, where $\mathcal{I}_Q = \{ind_1, \ldots, ind_{n_Q}\} \subseteq \mathcal{I}$ is a set of indicators, $\mathcal{L}_Q = \{\mathcal{L}_1, \ldots, \mathcal{L}_{m_Q}\}$ such that $\forall \mathcal{L}_j, \mathcal{L}_z \in \mathcal{L}_Q, \mathcal{L}_j \in D_h, \mathcal{L}_z \in D_g : D_h \neq D_g$, is a set of

levels representing the dimensional pattern at which the result is requested.

The formal specification of the syntax for a query can be expressed in Extended Backus-Naur Form (EBNF):

$$
\begin{aligned}
&\langle QUERY\rangle ::= \texttt{<}\ \langle INDICATOR\_SET\rangle\ \texttt{,}\ \langle LEVEL\_SET\rangle\ \texttt{>}\\
&\quad |\quad \langle INDICATOR\_SET\rangle ::= \texttt{\{}\ \langle INDICATOR\rangle\ (\texttt{,}\ \langle INDICATOR\rangle)^*\ \texttt{\}}\\
&\quad |\quad \langle INDICATOR\rangle ::= \langle IDENTIFIER\rangle\\
&\quad |\quad \langle LEVEL\_SET\rangle ::= \texttt{\{}\ \langle LEVEL\rangle\ (\texttt{,}\ \langle LEVEL\rangle)^*\ \texttt{\}}\\
&\quad |\quad \langle LEVEL\rangle ::= \langle IDENTIFIER\rangle\\
&\quad |\quad \langle IDENTIFIER\rangle ::= \text{URI}
\end{aligned}
$$

In our example case, the user is interested in analyzing $PM_{2.5}$ and PM_{10} indicators aggregated by country and year levels. Let the input textual request be: *"I want to analyze particulate matter emissions by country and year, with a focus on Europe and the last 3 years"*. The user's request is translated in the following well-formed query: $\langle\{Pollution\_PM2.5, Pollution\_PM10\}, \{GEO.country, TIME.year\}\rangle$, where *GEO.country* and *TIME.year* respectively refer to the levels *country* of dimension *GEO* and the level *year* of dimension *TIME*.

It is worth noting that, despite the apparent simplicity of the query language, the kind of queries we can manage fall into the classical *Selection-Projection-Join* (SPJ) class. Projection is clearly defined by the specification of the set of indicators and levels, while joins are automatically discovered by one of the most prominent features of the dataset discovery system proposed, thus relieving the user from explicit definition of join conditions. Selection is managed as well. In practice, desired conditions on data values are specified as soft requirements in the form of user preferences in the textual request (e.g. "with a focus on Europe and the last 3 years" in the previous example), and managed at a later stage, when ranking is performed.

We also like to note that, for the sake of simplicity, we do not introduce aggregation operators here, in the view of the fact that aggregation is performed on the result of the source discovery step, although in principle the underlying system would be able to manage it.

The actual generation of the query is performed by the LLM through a prompt which includes a set of instructions along with the user request and the contextual knowledge about available indicators and dimensional hierarchies. This approach is named *Semantic Parsing*, and refers to the process of translating natural language into a structured formal representation, such as a query in a specific database or knowledge graph query language. The use of LLM for this task has been experimented in the Literature with various approaches, both with pre-trained Language Models [e.g., 4] and general-purpose ones [e.g., 36].

Instruction-tuning [51] is meant to drive the LLM in the generation of the correct output, in this case a query aligning with the right syntax and semantics. Along with the syntax for a well-formed query, the prompt also includes examples of queries, following the *few-shot learning* prompting approach [50]. This technique involves providing a small set of well-structured examples of well-formed queries. These examples showcase the correct format, structure, and use of indicators and dimensions ensuring that the query aligns with the syntax and semantics of the dataset discovery platform. With respect to zero-shot prompting, which does not include any examples and only relies on the knowledge of the model, few-shot learning is capable of steering the model towards the right goal, obtaining more accurate responses with fewer hallucinations or ambiguities. A validator is then used to verify the query's syntactic correctness and possibly drive its re-generation. Finally, in order to deal with incomplete or ambiguous user requests that can be misinterpreted, the LLM is required to abstain from generating low-confidence outputs. To this aim, we employ a prompting-based approach in which the prompt includes specific instructions to reject the task if this is not clear enough. An example of an ambiguous request is *"I want to analyse pollution"*, which does not clarify neither which specific pollution indicator to select among the available ones, nor the analysis dimensions. In such cases, the LLM will reply a feedback to the user to disambiguate the request.

The query generation procedure is summarized in Algorithm 1. Once dimensional levels and indicators have been extracted, as seen in the previous subsection, a prompt is produced (function *write_query_prompt*, line 6) and submitted to the LLM for the query generation (line 7). The obtained query is then validated in order to identify possible syntactic errors (line 8). In case the query is valid, it is returned along with the successful status (lines 9-10). Conversely, the *patch_prompt* function (line 12) is called to adjust the prompt taking into account the identified syntax error, and the loop continues with the updated version. The process goes on a number of times up to MAX_ATTEMPTS. In case no valid query can be generated, the function returns a tuple

including an empty query and the unsuccessful status. As such, the procedure implements an iterative and feedback-driven approach to translating natural language requests into structured queries, leveraging a prompt patching mechanism to improve generation results when needed.

Input: user *request*
Output: tuple $\langle Q, status \rangle$

```
inds ← KG.get_Indicators()
levs ← KG.get_Dimensional_Levels()
patch ← ∅
i ← 0
while i ≤ MAX_ATTEMPTS do
    prompt ← write_query_prompt(request,inds,levs,patch)
    Q ← LLM.execute(prompt)
    status, error ← validate(Q)
    if status=="valid" then
        return ⟨Q, status⟩
    else
        patch ← patch_prompt(error)
        i ← i + 1
return ⟨∅, status⟩
```

Algorithm 1 Request enrichment and query generation

4.3 Dataset Discovery

The following step involves the submission of the query to the dataset discovery platform to identify a set of solutions.

To determine whether a data source can be used to compute an answer to a query, a requirement is that it must have a *compatible dimensional schema*, i.e., the source must include the same levels as the query. In many practical scenarios, this constraint can be relaxed by considering a source compatible if it includes a super-set of the query's levels. In such cases, roll-up relations can be leveraged to provide data at the appropriate aggregation level, as outlined in [12].

4.3.1 Solutions Discovery

A solution for a query is a set of data sources that (i) have a dimensional schema compatible with the query and, when joined together, (ii) provide the requested set of indicators. In order to be actionable, the sources in a solution need to be joined using the levels in the dimensional schema as join attributes.

Definition 6 (Solution to a query) Given a query $Q = \langle \mathcal{I}_Q, \mathcal{L}_Q \rangle$, a solution $\mathbb{S}$ is a combination of data sources $\{S_1, \ldots, S_n\}$, such that (i) $\mathcal{L}_Q = dim(S_i), \forall i = 1, \ldots, n$ and (ii) $\mathcal{I}_Q \subseteq ind(\bowtie_{i=1}^{n} S_i)$.

A solution is generated by the platform by exploiting the mappings between the Knowledge Graph and the metadata of the sources discussed in Sect. 3. This is a peculiar feature of our approach, whereas many works in the Literature require the exploration of the search space during query execution.

Given the user's request $\langle \{Pollution\_PM2.5, Pollution\_PM10\}, \{GEO.country, TIME.year\} \rangle$, a possible solution can be a set of two sources $\{S_1, S_2\}$ where S_1 has an attribute mapping to $Pollution\_PM2.5$ indicator and S_2 has an attribute mapping to $Pollution\_PM10$ indicator. Both sources have the dimensional schema compatible with the query, i.e., they contain columns which map to dimensional levels *GEO.country* and *TIME.year*. The user obtains the requested information when joining together S_1 and S_2 using the attributes mapped to *GEO.country* and *TIME.year* as join attributes.

It is worth noting that multiple solutions, i.e., alternative sets of sources satisfying the conditions, can be retrieved by the discovery platform. Such solutions will be provided to the user to let them select the one most suitable to their needs. In order to reduce the number of candidate solutions, pre-filtering is done by removing those solutions in which the cardinality of the join is very small. This can happen when joining sources with non-overlapping content, e.g., joining a source about pollution in Europe and a source about pollution in the Far East would result in an empty result set, although the dimensional schema is the same. However, an exact calculation of the joins would be highly inefficient in case of a very large dataset repository such as a Data Lake or a Data Space. For such a reason, the platform employs an algorithm to compute an approximate evaluation of the cardinality, through hashing schemes such as MinHash [7] and LSHEnsemble for approximate evaluation

of set containment [57]. The estimated cardinality is also used to provide a basic ranking of the alternative solutions.

4.3.2 Estimation of the Profile for a Solution

In order to support the users in the choice among multiple solutions, summary information of the content of each solution is provided in terms of data distribution of the result of the join. Again, exact profiling would be infeasible in Big Data scenarios. For such a reason, we complement each solution $\mathbb{S}_i$ with its *estimated profile* E_i, which represents the estimated distribution of values for each level of the dimensional schema. As such, the estimated profile provides an explanation of $\mathbb{S}_i$ in terms of the estimated distribution of the data resulting from joining the data sources in $\mathbb{S}_i$.

Definition 7 (Estimated profile) Given a query $Q = \langle \mathcal{I}_Q, \mathcal{L}_Q \rangle$ with $\mathcal{L}_Q = \{L_1, \ldots, L_m\}$ and a solution $\mathbb{S} = \{S_1, \ldots, S_n\}$, we denote the estimated profile of $\mathbb{S}$ by a set $E = \{\Psi_e(L_1), \ldots, \Psi_e(L_m)\}$. We denote by $\Psi_e(L_j) = \{\langle m, y_e \rangle | m \in L_j, y_e \geq 0\}$ the estimated profile of a level L_j, where the value y_e represents the estimated number of values aligned with member $m \in \delta(L_j)$ in the result of the join $\bowtie_{i=1}^{n} S_i$.

Operationally, given a level L_j, the profile $\Psi_e(L_j)$ is calculated as the intersection of profiles of attributes of all sources in $\mathbb{S}_i$ which map to the same level L_j. On the other hand, the value y_e can be estimated in several ways. In this work, we rely on a histogram-based approach assuming independence of the dimensional attributes, which is a simple yet efficient approach and one of the most adopted solutions for industrial DBMSs [29]. Denoting by $o_1, \ldots, o_n$ the number of occurrences of a member m in $S_1, \ldots, S_n$, the number of occurrences of m in the join of $S_1, \ldots, S_n$ is $\leq min(o_1, \ldots, o_n)$. Therefore, we take this value as an upper bound for y_e. This is motivated by the fact that each record in a source has a unique combination of values for the dimensional schema (we rely on the classical definition of relation as a set).

To give an example, let us consider the profiles of two attributes from two sources $a_x \in S_1$ and $a_y \in S_2$, both mapped to level *GEO.country*. The profile of the former is {<Italy, 20>, <France, 70>, <Germany, 10>}, while the one for the latter is {<Italy, 200>, <Spain, 400>, <Portugal, 350>, <France, 50>}. The cardinalities of the sources are 100 for S_1 and 1000 for S_2. Since Italy occurs 20 times in a_1 and 200 times in a_2, the upper bound for the number of occurrences in the join is $min(20, 100) = 20$. As a result, $\Psi_e(GEO.country)$={<Italy, 20>, <France, 50>}. Please note that, in general, the intersection can overestimate the actual distribution, as it is computed on each attribute of the sources separately.

Finally, the result set of the dataset discovery step is defined as follows.

Definition 8 (Result set) Given a query Q, its result is a set $R_Q = \{Z_1, ..., Z_r\}$, where $Z_i = \langle \mathbb{S}_i, E_i \rangle$ is a tuple including a solution $\mathbb{S}_i$ and its estimated profile E_i.

As shown in Algorithm 2, the procedure starts with the execution of the query on the dataset discovery platform (line 2). Then, for each retrieved solution $\mathbb{S}_i$, the estimated profile is computed (lines 4-8): in particular, for each data source S_j in $\mathbb{S}_i$ its profile π_j is extracted from the SKG (line 6) and appended to the list of profiles P_i for the solution (line 7). Then, its estimated profile E_i is computed (line 8) and added to the result set R_Q (line 9).

4.4 Preference-Oriented Solution Ranking and Explanation

The last step of the workflow consists in ranking the solutions produced by the dataset discovery platform according to the user preferences and providing the user with a textual report including the rank and a comparison of the solutions.

As summarized in Algorithm 2, this task is performed by the LLM. At first, a prompt is built including (1) a set of instructions on the task to perform (reported in Sect. 5, (2) the user preference expressed in the original request, (3) the list of solutions $\{\mathbb{S}_1, \ldots, \mathbb{S}_m\}$ and their estimated profiles $\{E_1, \ldots, E_m\}$ and (4) background knowledge on the dimensional levels of the query (line 10). The instructions are aimed to make the task of ranking and comparison explicit. The report will focus on explaining the differences among the solutions, especially with respect to user preferences. In case a ranking is not computable due to ambiguous user preferences, the original ranking of the dataset discovery is preserved (only based on estimated cardinality). The report will only focus on describing the different solutions. Finally, the prompt is submitted to the LLM (line 11) and the response is returned to the user, in the form of a report including the rank and an explanation comparing the solutions. To this aim, we refer to the following definition of explainability, adapted from the general definition proposed in Chazette et al. [11].

Definition 9 (Explainable ranking for dataset discovery)

A ranked set R_{rank} of solutions for a query Q is explainable with respect to a user A and preference P, if and only if there exists an explanation E such that:

$$E \models \text{Understanding}(A, R_{\text{rank}}, P)$$

where E is a structured natural language report satisfying the following conditions:

(i) Preference interpretation: the system identifies, extracts, and represents preferences P in a form usable for ranking decisions;
(ii) Justification of Ranking: E justifies the ordering of items in R_{rank} by showing how dataset profiles and domain knowledge align to P;
(iii) Intent Alignment Evaluation: A can assess whether the system's interpretation of P aligns with the intended prioritization criteria.

According to the definition, the system (i) must be able to accurately identify and interpret the implicit and explicit preferences expressed. For instance, the user might say "I want recent datasets relevant to particulate matter in America" and the system must understand not just keywords, but also concepts like recency, geography, or domain-specific relevance. Then, (ii) the explanation should explicitly link features of the solutions (in the form of profiles) to user preferences, so it is not enough to justify that a solution is better than the other, as users need to know why, e.g., "Solution A includes American countries and the last five years", which matches the preferences on the geographic and temporal dimension. Furthermore, (iii) the user must have the possibility to evaluate whether the system understood them correctly.

According to the categorization proposed in Bernard and Balog [5], our approach to explainability operates at the *local explanation level* describing the relationship between a specific input (i.e., query result) and output (i.e., the ranking). The explanations are presented in a *textual format* and are implemented as a *post-processing* step, modifying the output of the dataset discovery platform to incorporate explainability. This approach is *model-agnostic*, as it does not rely on the internal mechanisms of the underlying retrieval model.

Considering our example case and the query $\langle\{Pollution\_PM2.5, Pollution\_PM10\}, \{GEO.country, TIME.year\}\rangle$, let us assume the dataset discovery platform responds providing a result set R_Q formed by four distinct solutions. Each solution ($\mathbb{S}_i = \{S_{i1}, \ldots, S_{in}\}$) includes a set of data sources that, once joined, are capable of responding to the user request. In Fig. 4, the estimated profiles (E_i) for each solution are illustrated as a pair of histograms for the two required dimensions, showing the estimated distribution of, respectively, the *GEO.country* and the *TIME.year* dimensions. The occurrences (y-axis) are normalized with respect to the total number of rows provided by the joins. In the absence of further support, the user should manually analyze the resulting profiles and figure out which solution is best aligned with their needs. This process can be time-consuming and may become unfeasible in real-world scenarios where both the number of solutions and the number of distinct members in sources' profiles can be huge. In order to address this issue, the system returns a ranking of the obtained solutions based on the user's preferences, along with natural language explanations of the reasons behind the ranking. In this example case, the user expresses preferences for "European countries" and "the last 3 years". Thus, the system should identify $\mathbb{S}_2$ as the best solution since it contains data regarding years 2023-2025 and European countries. The solution $\mathbb{S}_1$ does not contain occurrences on $2023 - 2025$ but could still be interesting since it has only European data. Solutions $\mathbb{S}_3$ and $\mathbb{S}_4$ represent those of least interest as they have neither data on 2023-2025 nor a majority of European countries. All this information is returned by the system in natural language, making the explanations easy to understand by even less-experienced users.

Input: user $request$ and $preference$, indicators $inds$, dimensional levels $levs$, query Q
Output: ranked solution and explanation R_{rank}

$R_Q \leftarrow []$
$\rho \leftarrow \text{DD.run}(Q)$
for each $\mathbb{S}_i = \{S_{i_1}, \ldots, S_{i_n}\} \in \rho$ **do**
    $P_i \leftarrow []$
    for each $S_j \in \mathbb{S}_i$ **do**
        $\pi_j \leftarrow \text{KG.get\_profile}(S_j)$
        $P_i.\text{append}(\pi_j)$
    $E_i \leftarrow \text{compute\_estimated\_profile}(P_i)$
    $R_Q.\text{append}(\langle \mathbb{S}_i, E_i \rangle)$
$prompt \leftarrow \text{write\_rank\_prompt}(request, preference, inds, levs, R_Q)$
$R_{rank} \leftarrow \text{LLM.execute}(prompt)$
return R_{rank}

Algorithm 2 Discovery and ranking

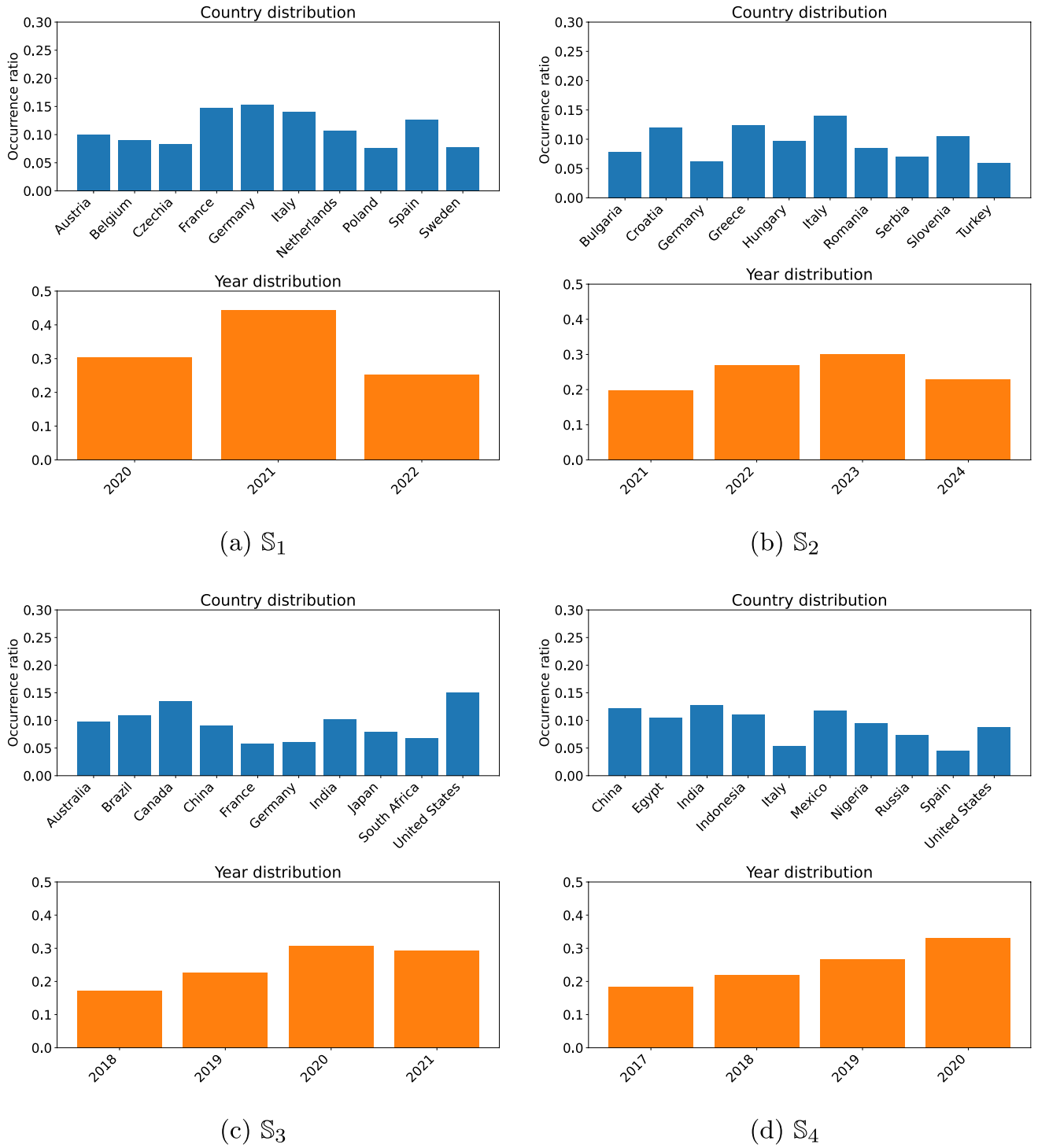

Fig. 4 Example case: estimated profiles for the four solutions of the dataset discovery platform

5 Evaluation

This section is devoted to describing experiments aimed at evaluating the performance of a prototypical implementation of the proposed Graph RAG-based approach for dataset discovery. An evaluation of the dataset discovery platform (Fig. 3, step 4) is available in a previous work [13].

5.1 Experimental Setup

In order to evaluate the proposed dataset discovery platform, we developed a prototype following a Service-Oriented Architecture, where each functionality (both for graph management and dataset search) is provided as a service. The prototype was implemented in Python 3.7, using Flask as a micro-framework to deploy RESTful services. Experiments were carried out on datasets monitoring air pollutants, such as CO_2 and $PM_{2.5}$, aggregated by geographical, temporal, and sectoral dimensions. The BKG represents pollutant indicators and the following dimensions: GEO, TIME, and SECTOR. The dimension levels are:

- *GEO dimension*: region, country, and continent;
- *TIME dimension*: month and year;
- *SECTOR dimension*: macro-sector and subsector.

To structure the schema, extract air pollutants and obtain value distributions in the different dimensional levels, we referred to real-world datasets from the Climatetrace[6], Ourworldindata[7] and European Data[8] repositories. We considered a total of 24 air pollutants gathered from all the repositories (CO2, CO2e-100, CO2e-20, CH4, N2O, PM2.5, PM10, NMVOCs, CO, NH3, NOx, SO2, BC, OC, AS, CD, CR, CU, HG, NI, PB, SE, SOx, ZN) and 10 macro-sectors (Agriculture, Buildings, Fluorinated Gases, Fossil Fuel Operations, Forestry and Land Use, Manufacturing, Mineral Extraction, Power, Transportation, Waste) each divided into its subsectors from the Climatetrace repository.

We performed three different typologies of experiments on the datasets:

- *query generation*: we tested the system's ability to generate well-formed queries based on user requests or, if this is not possible, to fall back to requesting clarification from the user;
- *solutions ranking*: we evaluated solutions rankings proposed by the system based on user preferences.
- *explainability*: we evaluate the LLMs' ability to provide sound explanations to the provided solutions rankings.

For what concerns the considered LLMs, we used the *GPT-4o* model available through the OpenAI APIs[9] for the experiment on query generation and we compared three different LLMs, namely *GPT-4o*, *DeepSeek-R1* and *Llama-70b* in the other experiments. The LLMs were chosen to allow for a comparative analysis both between different proprietary models and between proprietary and open source models. The details of the experiments and the results are presented in the following subsections. For a complete list of tested prompts and corresponding outputs, please refer to the project repository at https://github.com/KDMG/graph_RAG_dataset_discovery.

5.2 Query Generation

The system's ability to generate queries was evaluated using 22 user requests, 6 of which contained vague/imprecise instructions and/or typos, and 2 were malformed, including ambiguity or mistakes, to evaluate the system's response in more complex scenarios. The prompt structure was optimized through a prompt engineering phase in order to get better answers from the LLM. Prompts contained the following information:

- a brief specification of the task to be performed;
- a description of the indicators and the dimensions contained in the datasets;
- the expected syntax of the query;
- some examples of well-defined queries;
- instructions on how to deal with vague or imprecise requests;
- the user request (e.g., *"I want to analyse pollution (PM2.5 and CO2) by country and year"*).

The option to answer "Not sure" or to ask the user for clarification was given if the system was not able to generate a query with certainty based on the information provided. As reported in Table 1, the system was capable of correctly manage all cases. In particular, it provided a well-formed query in 16 out of 22 cases. These queries were syntactically correct as they respected the format $\langle\{set\_of\_indicators\}, \{set\_of\_dimension\_levels\}\rangle$ described in Sect. 4.2, while the specified indicators and dimensions were coherent with the provided lists of existing indicators and dimensions. In 4 cases, which correspond to ambiguous requests the answer "Not sure" was generated with the request of clarification. Finally, in 2 cases, corresponding to ambiguous and malformed requests, the system produces a query that was correctly assessed as invalid by the Query Validation, and therefore returned to the user for clarification. In general, the system was able to generate correct queries in all cases where the user request contained sufficient information on indicators and aggregation levels, i.e., from case 1 to case 14. It can be noticed that in case 12 (*"I want to analyse air pollution aggregated by region, month and sector"*), the system correctly identifies the indicators related to air emissions and, in absence of further

[6] https://climatetrace.org/data.

[7] https://ourworldindata.org/.

[8] https://data.europa.eu/.

[9] https://openai.com/index/openai-api/. API calls have been submitted on May 2025.

Table 1 User requests and results for the query generation experiment

Case	Prompt	Response
1	I want to analyse pollution (PM2.5 and CO2) by country and year	<{pollution_PM2_5, pollution_CO2}, {GEO.country, TIME.year}>
2	I want data sources which measure CH4 emissions by country, year and macro sectors	<{pollution_CH4}, {GEO.country, TIME.year, SECTOR.macrosector}>
3	Gather data about PM2.5 aggregated by region and subsector for all the years	<{pollution_PM2_5}, {GEO.region, TIME.year, SECTOR.subsector}>
4	I would like to join data sources containing NH3 and CO2 pollutants where indicators are aggregated by country, year and subsectors	<{pollution_NH3, pollution_CO2}, {GEO.country, TIME.year, SECTOR.subsector}>
5	I wish to analyse CH4, CO and PM10 air pollution indicators measured by country and months, also with information about subsectors	<{pollution_CH4, pollution_CO, pollution_PM10}, {GEO.country, TIME.month, SECTOR.subsector}>
6	Collect data sources containing PM10 values for all the countries and years	<{pollution_PM10}, {GEO.country, TIME.year}>
7	Find all the datasets with regions, months and subsectors, and NOx values	<{pollution_NOx}, {GEO.region, TIME.month, SECTOR.subsector}>
8	I want to analyse country data and the emissions of CO2	<{pollution_CO2}, {GEO.country}>
9	Gimme data aggregated by region and subsector for all the years	<{pollution_CO2, pollution_CO2e100, pollution_CO2e20, pollution_CH4, pollution_N2O, pollution_PM2_5, pollution_PM10, pollution_NH3, pollution_SOx, pollution_SO2, pollution_CO, pollution_NOx, pollution_NMVOCs, pollution_BC, pollution_OC, pollution_AS, pollution_CD, pollution_CR, pollution_CU, pollution_HG, pollution_NI, pollution_PB, pollution_SE, pollution_ZN}, {GEO.region, TIME.year, SECTOR.subsector}>
10	Collect greenhouse gas emissions for country, year and subsector	<{pollution_CO2, pollution_CO2e100, pollution_CO2e20, pollution_CH4, pollution_N2O}, {GEO.country, TIME.year, SECTOR.subsector}>
11	Find datasets containing CO2 measured by geography and temporal dimensions	<{pollution_CO2}, {GEO.country, TIME.year}>
12	I want to aanlyse air pollution aggregated by region, month and sector	<{pollution_CO2, pollution_CO2e100, pollution_CO2e20, pollution_CH4, pollution_N2O, pollution_PM2_5, pollution_PM10, pollution_NH3, pollution_SOx, pollution_SO2, pollution_CO, pollution_NOx, pollution_NMVOCs, pollution_BC, pollution_OC, pollution_AS, pollution_CD, pollution_CR, pollution_CU, pollution_HG, pollution_NI, pollution_PB, pollution_SE, pollution_ZN}, {GEO.region, TIME.month, SECTOR.macrosector}>
13	Find data about PM10 and PM2.5 for sectors	<{pollution_PM10, pollution_PM2_5}, {SECTOR.macrosector}>
14	Find particulate matter emissions in datasets with continents and years	<{pollution_PM2_5, pollution_PM10}, {GEO.continent, TIME.year}>
15	I would like to obtain data about CO2, NOx and NO2 for each region, month and subsector	<{pollution_CO2, pollution_NOx}, {GEO.region, TIME.month, SECTOR.subsector}>
16	I want to analyse pollution	**Not sure**
17	Give me data about air emissions aggregated by industry	<{pollution_CO2, pollution_CH4, pollution_N2O, pollution_PM2_5, pollution_PM10, pollution_NH3, pollution_SOx, pollution_SO2, pollution_CO, pollution_NOx, pollution_NMVOCs, pollution_BC, pollution_OC}, {SECTOR.macrosector}>
18	Gather data suorces contianing SO2, C4H and AS for cities and centuries	**Not sure**
19	Give me results for SO2, Nox and N2O	**Not sure**
20	I want pollutant indicators for cities	**Not sure**
21	Sector by year and country	**< { },{GEO.country, TIME.year, SECTOR.macrosector} >**
22	Year by year and sector	**< { },{TIME.year, SECTOR.macrosector} >**

Responses in bold are correctly identified as ambiguous/invalid and returned for clarification

specifications, aggregates at the highest level of the *sector* dimension (SECTOR.macrosector). Similar considerations can be made for case 17 (*"Give me data about air emissions aggregated by industry"*), where the LLM also associates the term "industry" to the *sector* dimension. User requests containing slang words (e.g., *"Gimme [...]"* in case 9) and/or typos (e.g., *"I want to aanlyse [...]"* in case 12) are also correctly interpreted, making the system robust to stylistic differences and typing errors in prompt writing. In case 15 (*"I would like to obtain data about CO2, NOx and NO2 for each region, month and subsector"*), where the prompt asks for the NO2 indicator, which does not exist in the datasets, the system ignores the request for such an indicator and provides a query that only extracts the other two existing indicators (e.g., CO2, NOx). In case of very vague queries (e.g., *"I want to analyse pollution"* in query 16) the system correctly responds "Not sure" to point out the lack of sufficient information for a correct generation of the query. For cases 21 and 22, the request itself was malformed, despite using valid terminology. Although the Query Generation module produced a query, it was correctly flagged as invalid, and returned back. As a result, the system asked the user for clarification at the second iteration.

5.3 Solutions Ranking

In the second experiment, the goal was to evaluate the preference-oriented solution rankings generated by the dataset discovery platform and the corresponding report. To this purpose, we considered 40 user requests with specific preferences (e.g., *"I prefer data on winter months about Power subsectors"*). These last range from clear and precise to more ambiguous formulations. For each request, we obtained three solutions from the dataset discovery platform and asked the three LLMs to rank them, based on the user preferences. The prompts were structured as follows:

- a role assignment to the LLM;
- a description of the members of each dimension in the datasets, i.e., background knowledge from the Knowledge Graph;
- the preferences of the user query, expressed in natural language;
- information about the task to be performed, i.e., to rank the proposed solutions according to user preferences and explain the criteria used to accomplish this task;
- for each solution, its estimated profile *E* (see Definition 7). To normalize the results, we reported the estimated number of values for each member of the estimated profile as a percentage rather than an absolute value.

A preliminary evaluation was manually performed by three human evaluators, who were not among the authors and had no familiarity with the work. They were asked to rank the solutions based on the information provided in the prompt. Their rankings were considered as the ground truth and used for the evaluation of the results of the LLMs. In cases where there was no complete agreement among the annotators, the best ranking was chosen using a majority voting criterion. The agreement among rankings proposed by the human evaluators was evaluated through the Krippendorff's alpha coefficient [27], which is defined as:

$$\alpha = 1 - \frac{D_o}{D_e}$$

where:

- D_o: Observed disagreement, calculated as the sum of differences between observed pairs of evaluations.
- D_e: Expected disagreement, calculated as the disagreement expected by chance.

The observed disagreement D_o is calculated as:

$$D_o = \sum_{c \in C} \sum_{k<l} \delta^2(v_{c,k}, v_{c,l}) \cdot o_{c,k} \cdot o_{c,l}$$

The expected disagreement D_e is calculated as:

$$D_e = \sum_{k<l} \delta^2(v_k, v_l) \cdot p_k \cdot p_l$$

The definitions for the symbols are the following:

- C: Set of all items or cases.
- $\delta(v_{c,k}, v_{c,l})$: Distance metric between two values $v_{c,k}$ and $v_{c,l}$, often squared difference for numerical data.
- $o_{c,k}$: Observed proportion of evaluators assigning value k to item c.
- $o_{c,l}$: Observed proportion of evaluators assigning value l to item c.
- p_k: Proportion of all evaluations assigned to value k.
- p_l: Proportion of all evaluations assigned to value l.

Krippendorff's alpha coefficient among evaluators was $\alpha = 0.936$, which means that evaluators proposed consistent rankings in most cases.

The rankings proposed by the human evaluators, the majority ranking as well as those generated by the three considered LLMs, are shown in Table 2. The rankings reported in the columns Human1, Human2 and Human3 refer to the solutions, proposed by the respective annotators, which

Table 2 Majority ranking of the human evaluators and comparison among the rankings proposed by the three LLMs

Prompt	Human1	Human2	Human3	Majority Ranking	GPT-4o	DeepSeek-R1	Llama-70b
1				CAB	CAB	**ACB**	**ACB**
2				CAB	CAB	**ACB**	CAB
3				ABC	ABC	ABC	**BAC**
4				BCA	BCA	BCA	**BAC**
5				CBA	CBA	CBA	**BAC**
6				CBA	CBA	CBA	**CAB**
7				ABC	ABC	**ACB**	**CAB**
8				CBA	CBA	CBA	CBA
9			BCA	CBA	CBA	CBA	CBA
10				BAC	BAC	**CBA**	**CBA**
11	BCA	ACB	BAC	-	BCA	ABC	CAB
12				CAB	CAB	CAB	CAB
13				CBA	CBA	CBA	CBA
14		BAC		BCA	**ABC**	BCA	**ACB**
15				ABC	ABC	ABC	**ACB**
16				ABC	ABC	ABC	ABC
17		ACB		CAB	CAB	**ACB**	**ACB**
18			ACB	ABC	ABC	**BCA**	**BAC**
19				CAB	CAB	**CBA**	**BCA**
20				ABC	ABC	ABC	ABC
21				BCA	**BAC**	**BAC**	**BAC**
22				CBA	**BCA**	CBA	CBA
23				CAB	CAB	CAB	**BCA**
24				BAC	BAC	BAC	BAC
25		ACB		CAB	**ACB**	**ACB**	CAB
26				ABC	ABC	ABC	**CAB**
27				CAB	CAB	CAB	CAB
28				BAC	BAC	BAC	BAC
29				ACB	ACB	**CAB**	**CBA**
30	CAB			CBA	CBA	CBA	CBA
31				CBA	CBA	CBA	CBA
32			CBA	CAB	CAB	CAB	CAB
33				CBA	CBA	CBA	**CAB**
34				CBA	CBA	CBA	**CAB**
35		ACB		CAB	CAB	CAB	CAB
36			CAB	BAC	BAC	BAC	**B (incomplete)**
37			CAB	ACB	ACB	ACB	**CAB**
38			CAB	ACB	**BAC**	**CAB**	**CAB**
39		ACB		CAB	CAB	CAB	CAB
40	BCA			CAB	**BCA**	**BAC**	**BAC**
Errors					**6**	**12**	**22**
Score					**203**	**179**	**133**

Wrong rankings are in bold

differ from the majority ranking. It can be noted that in one case (i.e., prompt #11) no agreement was reached among human annotators due to the complexity of the request, which lent itself to various interpretations, and hence we did not consider that prompt in the evaluation of results. The generated rankings were evaluated by means of two metrics:

- the *number of errors*, i.e. the number of rankings different from the majority rankings proposed by human annotators, independently from the number of solutions put in the wrong position;
- an overall *score*, calculated by assigning 3 points for each first position correctly identified, 2 for each second position and 1 for each third position.

In Table 2, it may be observed that GPT-4o outperforms the other models, as it makes considerably fewer errors than other models. The rankings proposed by DeepSeek-R1 are

Table 3 Comparison between the rankings proposed by GPT-4o with and without information from the Knowledge Graph

Prompt	Majority ranking	GPT-4o	GPT-4o w/o KG
1	CAB	CAB	**ACB**
2	CAB	CAB	CAB
3	ABC	ABC	ABC
4	BCA	BCA	BCA
5	CBA	CBA	**BAC**
6	CBA	CBA	CBA
7	ABC	ABC	**ACB**
8	CBA	CBA	CBA
9	CBA	CBA	CBA
10	BAC	BAC	**CAB**
11	-	BCA	BAC
12	CAB	CAB	CAB
13	CBA	CBA	CBA
14	BCA	**ABC**	BCA
15	ABC	ABC	**A (incomplete)**
16	ABC	ABC	**BAC**
17	CAB	CAB	**ACB**
18	ABC	ABC	**BAC**
19	CAB	CAB	CAB
20	ABC	ABC	ABC
21	BCA	**BAC**	**BAC**
22	CBA	**BCA**	CBA
23	CAB	CAB	CAB
24	BAC	BAC	BAC
25	CAB	**ACB**	**ACB**
26	ABC	ABC	ABC
27	CAB	CAB	CAB
28	BAC	BAC	BAC
29	ACB	ACB	**BAC**
30	CBA	CBA	CBA
31	CBA	CBA	CBA
32	CAB	CAB	CAB
33	CBA	CBA	CBA
34	CBA	CBA	**CAB**
35	CAB	CAB	CAB
36	BAC	BAC	BAC
37	ACB	ACB	**CAB**
38	ACB	**BAC**	**BAC**
39	CAB	CAB	CAB
40	CAB	**BCA**	**BAC**
Errors		**6**	**15**
Score		**203**	**166**

Wrong rankings are in bold

often only partially wrong, since the final score (179) is quite high if compared to the number of errors (12). For instance, in all prompts except case #10, the ranking generated by DeepSeek-R1 differs from the optimal one for a single swap of solutions. Moreover, the rankings generated by proprietary models show significantly superior performance than the considered open source model, namely Llama-70b, partly due to differences in parameter size.

For what concerns the best model, i.e. GPT-4o, it can be observed that the system proposed the best ranking in 33 out of 39 (84.6%) considered cases, while in an additional case (2.5%) the evaluators could not reach a consensus on a definitive best ranking. This demonstrated the ability of GPT-4o to interpret user preferences and generate correct rankings. In detail, for 24 out of 33 correct rankings (73%), all the evaluators agreed with its ranking, while in 9 rankings out of 33 (27%) one evaluator proposed a different solution.

After determining the best model, a second experiment was designed to evaluate the impact of the information coming from the Knowledge Graph on the final ranking. To this purpose, we removed such an information from the prompts and we sent them again to the best performing model, namely GPT-4o. The results are shown in Table 3. It is evident that the information coming from the Knowledge Graph significantly improves the results: in fact, the number of incorrect rankings increases from 6 to 15, while the score is reduced from 203 to 166 (−18.2%). It is also interesting to notice that, in one case (i.e., prompt #15), the ranking proposed by GPT-4o without KG is incomplete, as only the first position is generated.

5.4 Evaluation of Explainability

In order to evaluate the explainability capacity of the three considered LLMs, the generated responses were evaluated on a scale from 1 to 5 by three human operators with respect to two different aspects:

- the *coherence* of the response, i.e., how accordant the explanation is with prior knowledge and beliefs [35];
- the *overall quality* of the response, taking into account all aspects (correctness, completeness and comprehensibility of the explanation)

The results of the evaluation are shown in Table 4. GPT-4o gets the best scores in terms of both quality and coherence, as its explanations resulted to be the most exhaustive and correct. Slightly lower scores are given to DeepSeek-R1's responses while the explanations given by Llama-70b have the worst coherence and overall quality on average. A common problem in the responses generated by Llama-70b was that they were not able to clearly explain the criteria used for the ranking and, sometimes, the explanations provided were not consistent with the data provided as input.

Furthermore, we evaluated the *compactness* of the explanations given by the three LLMs, i.e. the size (expressed in terms of number of characters) of the answers, as described in Nauta et al. [35]. The results are shown in Table 5, where also the standard deviations are reported. Coupled with the

Table 4 Evaluation of quality and coherence of the answers given by the LLMs

Prompt	Quality			Coherence		
	GPT-4o	DeepSeek-R1	Llama-70b	GPT-4o	DeepSeek-R1	Llama-70b
1	4,67	3,00	3,00	5,00	2,33	3,33
2	5,00	3,00	4,50	5,00	4,00	5,00
3	5,00	4,67	3,00	5,00	5,00	4,33
4	5,00	5,00	3,33	4,67	5,00	4,00
5	5,00	5,00	1,67	4,67	4,67	2,67
6	4,67	5,00	2,67	5,00	5,00	3,00
7	5,00	3,33	2,00	5,00	4,33	2,67
8	5,00	4,67	5,00	5,00	4,67	5,00
9	5,00	4,67	5,00	5,00	4,67	4,00
10	4,33	2,67	2,33	5,00	3,33	3,00
11	4,33	3,67	4,00	5,00	4,33	4,33
12	5,00	5,00	5,00	5,00	4,67	4,67
13	4,33	4,67	4,67	5,00	4,33	5,00
14	3,00	5,00	2,33	4,33	5,00	4,00
15	4,67	4,67	3,00	4,33	5,00	3,00
16	5,00	4,67	4,33	4,67	4,67	4,67
17	4,00	4,00	3,00	4,67	4,33	4,00
18	4,33	3,67	3,00	4,67	5,00	3,33
19	5,00	3,67	2,00	5,00	4,33	2,33
20	4,67	5,00	4,00	5,00	5,00	3,67
21	4,67	4,33	4,00	4,67	4,33	4,00
22	4,00	5,00	4,00	4,00	5,00	4,00
23	5,00	5,00	3,33	5,00	5,00	3,00
24	5,00	5,00	5,00	5,00	5,00	5,00
25	4,33	4,33	5,00	4,33	4,33	5,00
26	4,67	4,67	2,33	4,67	4,67	2,67
27	5,00	4,67	3,67	5,00	4,67	4,00
28	4,67	4,67	4,33	5,00	5,00	4,33
29	5,00	4,67	4,67	5,00	4,33	4,00
30	4,67	4,67	4,67	4,33	4,00	4,33
31	5,00	4,67	4,67	5,00	4,67	5,00
32	4,67	4,67	4,33	5,00	4,67	3,67
33	5,00	4,67	3,33	5,00	5,00	3,00
34	4,67	4,67	2,67	4,67	4,67	2,67
35	5,00	5,00	5,00	5,00	5,00	5,00
36	4,67	4,67	3,33	4,67	4,67	3,67
37	5,00	4,67	3,33	5,00	5,00	3,33
38	3,67	4,00	4,00	4,00	4,00	4,00
39	5,00	5,00	5,00	5,00	5,00	5,00
40	4,00	3,67	3,67	3,67	3,67	3,67
Average	**4,67**	**4,44**	**3,70**	**4,78**	**4,56**	**3,88**

Table 5 Compactness of the explanations given by the three LLMs

	GPT-4o	DeepSeek-R1	Llama-70b
Average length	2147,05	2962,4	1445,15
Std. Dev	536,93	782,42	416,43

coherence, the compactness allows to better evaluate the explainability performance: in fact, answers that are both short (i.e., high compactness) and coherent are an indicator of a model's high explainability capacity, as it is more difficult to give convincing explanations by using few words. DeepSeek-R1 produced the longest outputs on average (avg_length=2962.40), followed by GPT-4o (avg_length=2147.05), with Llama-70b generating the shortest responses (avg_length=1445.15). This suggests that DeepSeek-R1 is significantly more verbose in its responses, which could reflect its internal design choices that may encourage a broader elaboration of explanations. The standard deviation (SD) values provide insights into response variability. DeepSeek-R1 not only generated the longest outputs but also had the highest variability (SD=782.42), indicating less

consistency in output length. Conversely, GPT-4o displayed moderate variability (SD=536.93), while Llama-70b was the most consistent (SD=416.43), aligning with its generally shorter and more uniform output. However, in this case a high compactness is not correlated with high coherence, as Llama-70b showed the lowest values for that metric. As a consequence, GPT-4o is confirmed as the best option for correctly generating a ranking and its explanation.

6 Conclusion

In this work, we introduced a novel Graph Retrieval-Augmented Generation (Graph RAG) framework to enhance the explainability and effectiveness of dataset discovery. By integrating Knowledge Graphs and Large Language Models (LLMs), the approach addresses challenges associated with interpreting heterogeneous data sources and providing user-centric explanations for dataset recommendations. The proposed methodology combines enriched query formulation, leveraging contextual information from Knowledge Graphs, with advanced LLM capabilities for query generation, solution ranking, and explainable result presentation.

Our experimental evaluation, which compared GPT-4o, DeepSeek-R1 and Llama-70b, demonstrated the system's ability to generate well-formed queries, rank solutions effectively based on user preferences and produce explainable results, especially for GPT-4o. The results highlighted its robustness in handling vague or incomplete requests, as well as its capacity to enhance user understanding through clear explanations of ranking decisions. The integration of approximate data profiling further ensures scalability, making the framework suitable for large and complex data ecosystems like Data Lakes and Data Spaces.

This research advances the state-of-the-art in dataset discovery by addressing the often-overlooked aspect of explainability, which is critical for fostering trust and informed decision-making among users. Future work will focus on extending the framework to incorporate dynamic user feedback, enhancing the semantic parsing capabilities for query generation with more advanced prompting techniques and exploring the applicability of the approach in further real-world domains.

Acknowledgements Authors would like to thank the evaluators for their assessment of the results.

Author Contributions Emanuele Storti and Alex Mircoli contributed to the study conception and design. Data collection and experimentation were performed by all authors. The first draft of the manuscript was written by Emanuele Storti, Alex Mircoli, Cristina Rossetti and Alessandro Mele. All authors reviewed and edited previous versions of the manuscript, read and approved the final manuscript.

Funding Cristina Rossetti has received funding from the MUR – DM 118/2023 as part of the project PNRR-NGEU. Alex Mircoli has received funding from the project Vitality – Project Code ECS00000041, CUP I33C22001330007 - funded under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5 - 'Creation and strengthening of innovation ecosystems,' construction of 'territorial leaders in R&D' – Innovation Ecosystems - Project 'Innovation, digitalization and sustainability for the diffused economy in Central Italy – VITALITY' Call for tender No. 3277 of 30/12/2021, and Concession Decree No. 0001057.23-06-2022 of Italian Ministry of University funded by the European Union – NextGenerationEU.

Availability of data and materials Prompts and results of the evaluation are available at https://github.com/KDMG/graph_RAG_dataset_discovery.

Declarations

Competing interests All authors declare that they have no competing interests.

References

1. Abedjan Z, Grütze T, Jentzsch A et al (2014) Profiling and mining rdf data with prolod++. In: 2014 IEEE 30th International Conference on Data Engineering, IEEE, pp 1198–1201
2. Auer S, Demter J, Martin M et al (2012) Lodstats–an extensible framework for high-performance dataset analytics. In: Knowledge Engineering and Knowledge Management: 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings 18, Springer, pp 353–362
3. Bahr L, Wehner C, Wewerka J et al (2024) Knowledge graph enhanced retrieval-augmented generation for failure mode and effects analysis. arXiv preprint arXiv:2406.18114
4. Banerjee D, Nair PA, Kaur JN et al (2022) Modern baselines for sparql semantic parsing. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 2260–2265
5. Bernard N, Balog K (2025) A systematic review of fairness, accountability, transparency, and ethics in information retrieval. ACM Comput Surv 57(6):1–29
6. Bogatu A, Fernandes AA, Paton NW et al (2020) Dataset discovery in data lakes. In: 2020 IEEE 36th international conference on data engineering (icde), IEEE, pp 709–720
7. Broder AZ (1997) On the resemblance and containment of documents. In: Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171), IEEE, pp 21–29
8. Castelo S, Rampin R, Santos A et al (2021) Auctus: a dataset search engine for data discovery and augmentation. Proc VLDB

Endow 14(12):2791–2794. https://doi.org/10.14778/3476311.3476346
9. Castro Fernandez R, Abedjan Z, Koko F et al (2018) Aurum: A data discovery system. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp 1001–1012, https://doi.org/10.1109/ICDE.2018.00094
10. Chapman A, Simperl E, Koesten L et al (2020) Dataset search: a survey. VLDB J 29(1):251–272
11. Chazette L, Brunotte W, Speith T (2021) Exploring explainability: a definition, a model, and a knowledge catalogue. In: 2021 IEEE 29th international requirements engineering conference (RE), IEEE, pp 197–208
12. Diamantini C, Potena D, Storti E (2018) Multidimensional query reformulation with measure decomposition. Inf Syst 78:23–39
13. Diamantini C, Potena D, Storti E (2024) Analytic processing in data lakes: a semantic query-driven discovery approach. Inf Syst Front 14:1–9
14. Diamantini C, Lo Giudice P, Potena D et al (2021) An approach to extracting topic-guided views from the sources of a data lake. Inf Syst Front 23:243–262
15. Edge D, Trinh H, Cheng N et al (2024) From local to global: A graph rag approach to query-focused summarization. arXiv preprint arXiv:2404.16130
16. Fernandez RC, Min J, Nava D et al (2019) Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, pp 1190–1201
17. Hai R, Geisler S, Quix C (2016) Constance: An intelligent data lake system. In: Proceedings of the 2016 international conference on management of data, pp 2097–2100
18. He X, Tian Y, Sun Y et al (2024) G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. arXiv preprint arXiv:2402.07630
19. Helal A, Helali M, Ammar K et al (2021) A demonstration of kglac: a data discovery and enrichment platform for data science. Proc VLDB Endow 14(12):2675–2678
20. Honovich O, Scialom T, Levy O et al (2023) Unnatural instructions: Tuning language models with (almost) no human labor. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 14409–14428
21. Hoseini S, Theissen-Lipp J, Quix C (2024) A survey on semantic data management as intersection of ontology-based data access, semantic modeling and data lakes. J Web Semant 81:100819
22. Hu EJ, Shen Y, Wallis P et al (2022) LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations, https://openreview.net/forum?id=nZeVKeeFYf9
23. Hu Y, Lei Z, Zhang Z et al (2024) Grag: Graph retrieval-augmented generation. arXiv preprint arXiv:2405.16506
24. Huang L, Yu W, Ma W et al (2023) A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. ACM Transactions on Information Systems
25. Jia M, Tang L, Chen BC et al (2022) Visual prompt tuning. In: European Conference on Computer Vision, Springer, pp 709–727
26. Kimball R (1996) The data warehouse toolkit: practical techniques for building dimensional data warehouses. John Wiley & Sons Inc
27. Krippendorff K (2011) Computing krippendorff's alpha-reliability. Computing 1
28. Langegger A, Woss W (2009) Rdfstats-an extensible rdf statistics generator and library. In: 2009 20th International Workshop on Database and Expert Systems Application, IEEE, pp 79–83
29. Leis V, Radke B, Gubichev A et al (2017) Cardinality estimation done right: Index-based join sampling. In: Cidr
30. Lewis P, Perez E, Piktus A et al (2020) Retrieval-augmented generation for knowledge-intensive nlp tasks. Adv Neural Inf Process Syst 33:9459–9474
31. Marchionini G (2006) Exploratory search: from finding to understanding. Commun ACM 49(4):41–46
32. Mavromatis C, Karypis G (2024) Gnn-rag: Graph neural retrieval for large language model reasoning. arXiv preprint arXiv:2405.20139
33. Nagel L, Hierro JJ, Perea E et al (2021) Design principles for data spaces: Position paper. Tech. rep., E. ON Energy Research Center
34. Nargesian F, Zhu E, Pu KQ et al (2018) Table union search on open data. Proc VLDB Endow 11(7):813–825
35. Nauta M, Trienes J, Pathak S et al (2023) From anecdotal evidence to quantitative evaluation methods: a systematic review on evaluating explainable ai. ACM Comput Surv. https://doi.org/10.1145/3583558
36. Nguyen LM, Le NK, Anh KQ et al (2024) Semantic parsing for question and answering over scholarly knowledge graph with large language models. In: JSAI International Symposia on AI, https://api.semanticscholar.org/CorpusID:270227673
37. Oram A (2015) Managing the Data Lake: Moving to Big Data Analysis. O'Reilly Media
38. Ouyang L, Wu J, Jiang X et al (2022) Training language models to follow instructions with human feedback. Adv Neural Inf Process Syst 35:27730–27744
39. Peng B, Zhu Y, Liu Y et al (2024) Graph retrieval-augmented generation: A survey. arXiv preprint arXiv:2408.08921
40. Procko TT, Ochoa O (2024) Graph retrieval-augmented generation for large language models: A survey. In: 2024 Conference on AI, Science, Engineering, and Technology (AIxSET), IEEE, pp 166–169
41. Rafailov R, Sharma A, Mitchell E et al (2024) Direct preference optimization: Your language model is secretly a reward model. Advances in Neural Information Processing Systems 36
42. Rosenfeld A, Richardson A (2019) Explainability in human-agent systems. Auton Agent Multi-Agent Syst 33:673–705
43. Sanmartin D (2024) Kg-rag: Bridging the gap between knowledge and creativity. arXiv preprint arXiv:2405.12035
44. Santos A, Bessa A, Musco C et al (2022) A sketch-based index for correlated dataset search. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE), IEEE, pp 2928–2941
45. Sawadogo P, Darmont J (2021) On data lake architectures and metadata management. J Intell Inf Syst 56:97–120
46. Schneider P, Afzal A, Vladika J et al (2023) Investigating conversational search behavior for domain exploration. In: European Conference on Information Retrieval, Springer, pp 608–616
47. Sha Y, Feng Y, He M et al (2023) Retrieval-augmented knowledge graph reasoning for commonsense question answering. Mathematics 11(15):3269
48. Shrivastava A, Li P (2015) Asymmetric minwise hashing for indexing binary inner products and set containment. In: Proceedings of the 24th international conference on world wide web, pp 981–991
49. Solmaz G, Cirillo F, Fürst J et al (2022) Enabling data spaces: Existing developments and challenges. In: Proceedings of the 1st International Workshop on Data Economy, pp 42–48
50. Song Y, Wang T, Cai P et al (2023) A comprehensive survey of few-shot learning: evolution, applications, challenges, and opportunities. ACM Comput Surv 55(13s):1–40
51. Wei J, Bosma M, Zhao VY et al (2022) Finetuned language models are zero-shot learners. In: ICLR 2022 - 10th International Conference on Learning Representations. International Conference on Learning Representations, ICLR
52. Wei J, Wang X, Schuurmans D et al (2022) Chain-of-thought prompting elicits reasoning in large language models. Adv Neural Inf Process Syst 35:24824–24837

53. Wilkinson MD, Dumontier M, Aalbersberg IJ et al (2016) The fair guiding principles for scientific data management and stewardship. scientific data 3 (2016). Number 1:160018
54. Xu Z, Cruz MJ, Guevara M et al (2024) Retrieval-augmented generation with knowledge graphs for customer service question answering. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 2905–2909
55. Yao S, Zhao J, Yu D et al (2023) React: Synergizing reasoning and acting in language models. In: International Conference on Learning Representations (ICLR)
56. Zhao Y, Ravat F, Aligon J et al (2021) Analysis-oriented metadata for data lakes. In: Proceedings of the 25th International Database Engineering & Applications Symposium, pp 194–203
57. Zhu E, Nargesian F, Pu KQ et al (2016) Lsh ensemble: internet-scale domain search. Proc VLDB Endow 9(12):1185–1196
58. Zhu E, Deng D, Nargesian F et al (2019) Josie: Overlap set similarity search for finding joinable tables in data lakes. In: Proceedings of the 2019 International Conference on Management of Data, pp 847–864