DataGEMS

Data Discovery Platform with Generalized Exploratory, Management, and Search Capabilities



REPORT

D4.1

User Assistance Architecture Report

PU

31/10/2025

Disclaimer

This document contains description of the DataGEMS project findings, work and products. Certain parts of it might be under partner Intellectual Property Right (IPR) protection rules so, prior to using its content please consult carefully the licensing information of this document and/or contact the Project and the Editor for any further information.

In case you believe that this document harms in any way IPR held by you as a person or as a representative of an entity, please do notify the Project and the Editor immediately.

The editor and authors of this document have taken any available measure in order to ensure that its content is accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content. The information in this document is provided “as is” without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability.

This project has received funding from the European Union’s Horizon Europe framework programme under grant agreement No. 101188416.



Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency, neither of which can be held responsible for those views and opinions and are not liable for any use that may be made of the information contained therein.

Document Information

Type	REPORT
Identifier	D4.1
Title	User Assistance Architecture Report
Dissemination level	PU
Due Date	31/10/2025
Actual Delivery Date	31/10/2025
Lead Beneficiary	CNRS
Other beneficiaries	UNIVR, ZHAW, ARC
Editor	Silviu Maniu
Authors	Stavroula Eleftheraki, Silviu Maniu, Niccolo Marastoni, Elisa Quintarelli, Kurt Stockinger, Yi Zhang
Contributors	Mike Xydas
Reviewer	Themis Palpanas (UPCité)
Workpackage(s)	WP4
Keywords	Assistance, query and dataset recommendation and disambiguation, explanations
Version	1.0
Version information	Number of pages: 27 Last file save: 31/10/2025 6:26 PM File revision: File name: D4.1_User_Assistance_Architecture_Report.docx

Copyright Information

License: Creative Commons Attribution (CC-BY)
This document is released under the Creative Commons Attribution (CC-BY) license, which allows for the free use, distribution, and adaptation of the work, provided proper attribution is given to the original author(s). By accessing or using this report, you acknowledge and agree to comply with the terms and conditions of the CC-BY license. For the full text of the license, please visit: https://creativecommons.org/licenses/by/4.0/legalcode

Document Version

Version	Date	Change Description	Editor
1.0	31/10/2025	Submitted to EU	Silviu Maniu

Document Tables

Table of Figures

Figure 1 User Assistance components in the DataGEMS Architecture 8
Figure 2 The components of the Query Disambiguation Service. 15
Figure 3 The components of the Dataset-to-Dataset Recommendation Service 18
Figure 4 AP Explanations architecture 22

Table of Contents

Copyright Information 3
1 General Overview of User Assistance 8
1.1 Introduction 8
1.2 DataGEMS Platform Requirements Overview 9
1.3 Use Case Specific Requirements Overview 10
1.4 Overview of Components 11
2 User Assistance Components 12
2.1 Query Recommendation 12
2.1.1 Introduction 12
2.1.2 Component Architecture 12
2.1.3 Integration with the DataGEMS platform 13
2.1.4 Technologies 13
2.1.5 Security policies 13
2.1.6 Monitoring and Accounting 13
2.2 Query Disambiguation 14
2.2.1 Introduction 14
2.2.2 Component Architecture 14
2.2.3 Integration with the DataGEMS platform 15
2.2.4 Technologies 15
2.2.5 Security Policies 16
2.2.6 Monitoring and Accounting 16
2.3 Dataset Recommendation 17
2.3.1 Introduction 17
2.3.2 Component Architecture 17
2.3.3 Integration with the DataGEMS platform 19
2.3.4 Technologies 19
2.3.5 Security Policies 19
2.3.6 Monitoring and Accounting 20
2.4 Analytical Pattern Explanations 20
2.4.1 Introduction 20
2.4.2 Component Architecture 21
2.4.3 Integration with the DataGEMS platform 23
2.4.4 Technologies 23
2.4.5 Security policies 23
2.4.6 Monitoring and Accounting 24
3 User Assistance for Use Cases 25
3.1 Introduction 25
3.2 Query Recommendation in Action 25
3.3 Dataset Recommendation in Action 26
3.4 Query Disambiguation in Action 26
3.5 Analytical Pattern Explanations in Action 27

Abstract

This document provides an overview of the architecture of the components present in the User Assistance category of functionalities of the DataGEMS platform, corresponding to the tasks in WP4 of the project. User Assistance components are Query Recommendation, Query Disambiguation, Dataset Recommendation, and Analytical Pattern Explanations. They intervene in the user flows related to searching of datasets, by providing explanations and dataset recommendations, and related to executing tasks on datasets, by providing dataset recommendation, query disambiguation and recommendation, and explanations. They provide functionalities for the pilot use cases of the project, in particular Higher Education, Weather, and Language. Internally, the components are implemented as services that are accessible via API endpoints and communicate directly with the MoMa Object Catalog and primary data stored in datasets. The interactions between each component and the overall DataGEMS architecture are detailed in this document, along with the technology they use and how they implement monitoring and accounting metrics.

1 General Overview of User Assistance

1.1 Introduction

User Assistance in the DataGEMS platform is tasked with accompanying the user all along their discovery and querying process. The assistance takes the form of query assistance (reformulating and decomposing queries into tasks, but also recommending new or follow-up queries), recommending new or related datasets, and explaining results to users.

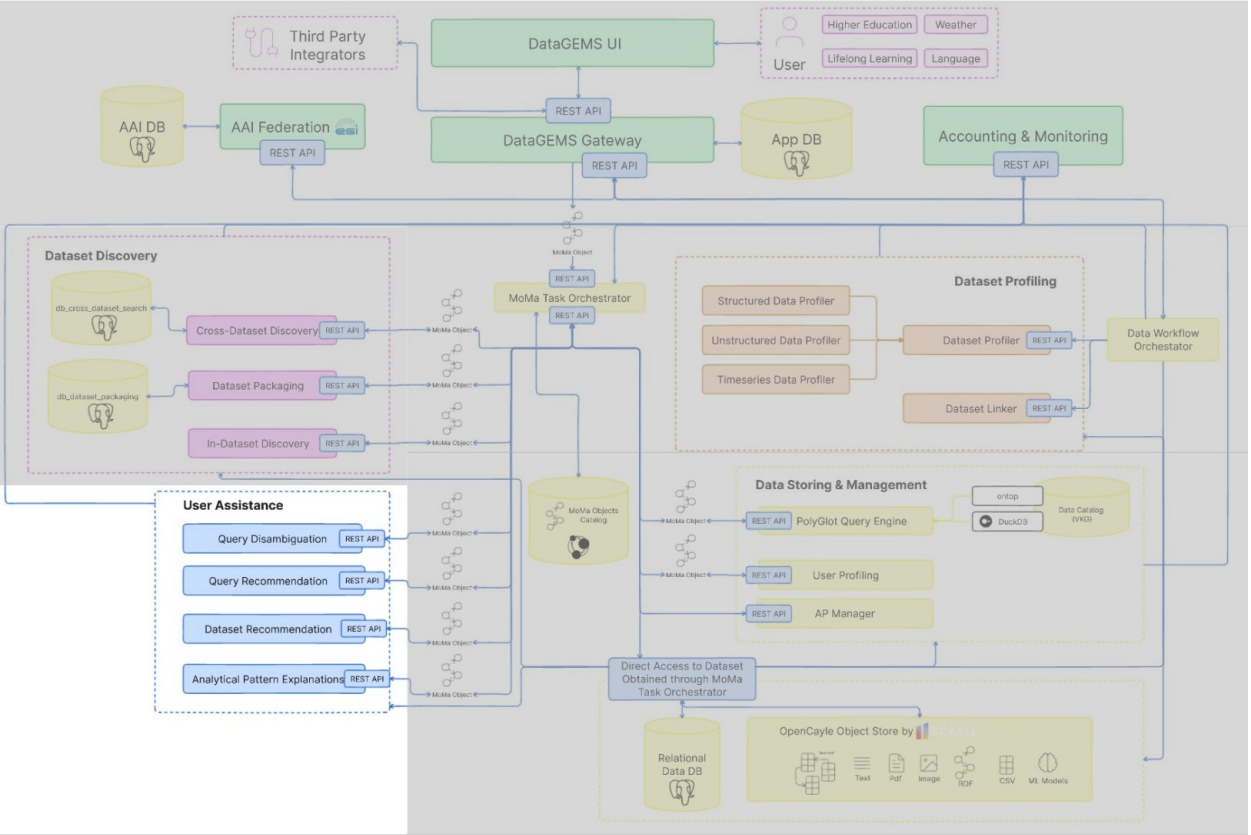


Figure 1 User Assistance components in the DataGEMS Architecture

The User Assistance architecture is organized into 4 main components, as highlighted in Figure 1, which details the overall DataGEMS architecture. Each component is implemented as a REST API providing methods that are generic operators in the MoMa data model (see detailed description in Deliverable 5.1) and can have specific implementations for the pilot use cases in the project. The four components are independent, but communicate with the DataGEMS platform via two main pathways:

1. They all *request and store data in the MoMa Object Catalog*, by retrieving user tasks and returning results; these results are also MoMa objects and can represent new queries, links to existing datasets, or can be enriched with explanations.

2. Some components also need *direct access to datasets*, by interacting with the Dataset Profiler (see Deliverable D2.1) and the dataset storage, and by modifying the datasets themselves.

All components must provide outputs that can be interpreted and shown on the User Interface: new queries expressed into natural language, lists or summaries of datasets, tabular explanation of results of user querying tasks. This is indirectly linked via the MoMa Task orchestrator, that sends MoMa objects to the gateway and the UI.

Section DataGEMS Platform Requirements Overview presents the DataGEMS platform requirements that User Assistance components implement, Section Use Case Specific Requirements Overview describes the overall use case capabilities and functionalities of the components, and Section Overview of Components gives a summary of the generic functionalities of each component.

1.2 DataGEMS Platform Requirements Overview

The DataGEMS platform allows users to fulfill two main user needs, as defined in D6.2 Platform Architecture Design Report. The User Assistance components are an integral part of the successful execution of the user needs, as described below.

The first user need is to discover datasets from a starting query representing a data need (e.g., weather information from stations in Athens, Greece). In this task, the requirement is that the platform discovers datasets that fulfill the user need, populate the results and present them. *Optionally*, at the user's request, the platform should be able to call the User Assistance components for:

- Providing explanations on why the result datasets were obtained, by calling the **Explanation** component.
- Finding and recommending other datasets related to the user need, by calling the **Dataset Recommendation** component.

The second user need is to perform tasks on the datasets that are discovered or provided, by asking queries or questions on these datasets. The platform should be able to interpret the query, find the relevant data via in-dataset discovery (see Deliverable 3.1), interpret and execute the task as an Analytical Pattern (see Deliverable 5.1) and present the results of the task to the user. Importantly, at the query stage, the **Query Disambiguation** component is called to clarify the user need by removing any ambiguities in natural language queries. Moreover, two other components are actively involved in this user need flow:

- The **Query Recommendation** component provides optional follow-up queries to the user; in case they want to continue their exploration of the task.
- The **Explanation** component can provide indication on the provenance and how the data was used to generate the response.

1.3 Use Case Specific Requirements Overview

User Assistance is involved in all use cases of the project, in a variety of tasks, spanning recommendations, explanations and query refinements. The tasks described below are a subset of the tasks described in deliverable D7.1 Pilot Requirements Report. The details on how these tasks will be handled by each component are described in Section User Assistance for Use Cases.

The **Query Recommendation** component will be able to suggest the next natural language query considering the user profile and their current natural language query.
In the *Higher Education* use case, the components operate in the following tasks:

- UC1T2 (Student-level adaptive quiz recommendation), suggesting natural language queries to retrieve exercises personalized based on the student profile and history of interaction with DataGEMS.
- UC1T4 (Teacher dashboard and pattern analysis), recommending the best types of visualizations given the query result.
- UC1T5 (Teacher-driven quiz composition), by interpreting the teacher's intent, suggests alternative queries depending on the topic of the current queries and the past class performance.

The **Query Disambiguation** component enhances data exploration by accurately interpreting user intent and clarifying natural language queries in relation to the dataset. It is a generic component. Example use cases in Weather and Language are:

- UC3T1 (Natural Language Querying Interpretation) and UC4T2 (Natural Language Querying), by providing highly relevant answers to natural language questions.

The **Dataset Recommendation** component will be able to suggest additional resources related to the dataset or material a user is currently exploring.

In the project use cases, the component operates in the following tasks:

- UC1T3 (Pedagogical resource creation/recommendation, Higher Education), by recommending complementary pedagogical resources (e.g., PDFs, videos).
- UC4T5 (Semantic and Metadata-Based Retrieval, Language), by recommending related corpora or documents.

The **Analytical Pattern (AP) Explanation** component will be able to provide functionality at the level of analysis of results. Some example use cases are Higher Education and Weather.
In the *Higher Education* use case, the component operates in the following tasks:

- UC1T4 (Teacher Dashboard and Pattern Analysis), by providing the provenance of the original quiz questions/answers that contributed to an outcome.

In the *Weather* use case, the component can provide support for the following tasks:

- UC3T5 (Result Export and Reusability) and UC3T3 (Data Processing), by providing provenance of the SQL queries transformed from the natural language queries asked by the users.

1.4 Overview of Components

We provide below short summaries of the components that are part of the User Assistance and their functionalities. The full details for each component are given in Section User Assistance Components.

The **Query Recommendation** component helps users refine or broaden their searches by suggesting relevant follow-up queries. It leverages on the user's profile, current query, and—when available—past query history with related retrieved data to generate tailored suggestions. By integrating with Cross-Dataset Discovery and Dataset Packaging services (see Deliverable D3.1), it can recommend semantically related queries based on the content of connected datasets. This enables users to find information more efficiently and uncover related topics they might not have considered on their own.

The **Query Disambiguation** component enhances data exploration by accurately interpreting user intent and clarifying natural language queries in relation to the dataset. The service leverages LLMs for automatic disambiguation as well as Human-in-the-Loop approaches, where users can disambiguate queries. The service is tightly integrated with In-Dataset and Cross-Dataset Discovery (see Deliverable D3.1) to enable highly relevant natural language exploration.

The **Dataset Recommendation** component helps users discover additional datasets relevant to their current exploration. By suggesting related resources from the data lake, it supports dataset discovery and reuse across the platform.

The **AP Explanation** component is an optional component that complements the user flow when requesting, querying, and analyzing datasets. The AP Explainer requires the MoMa object to retrieve the task-specific analytical patterns (AP) and the datasets that are involved as inputs in the AP. It uses two main steps: first, the datasets are annotated so that the explanations can be computed. On request explanations are run alongside the AP to combine initial annotations into an explanation. Final explanations are stored in MoMa as datasets and can be retrieved on demand in future interactions.

2 User Assistance Components

2.1 Query Recommendation

2.1.1 Introduction

Query recommendation provides users with explicit suggestions for the next query, to facilitate their data exploration journey. To better pinpoint the intention and goals of specific users, these suggestions will be guided by the users' previous queries and their personal user profile. The next query recommended can also be a refinement of the previous query, where the component suggests a similar, but possibly restricted in scope and more precise, query.

The query recommendations shown to the users by the module can encompass different modalities, e.g.:

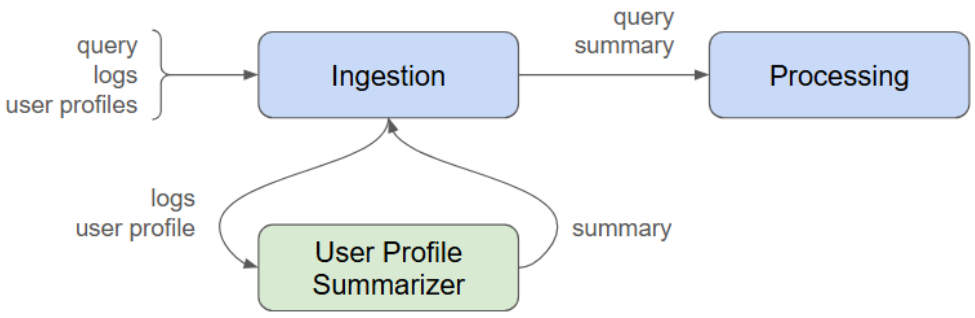
1. Natural language: guiding intuitive data exploration via new queries that are related to the previous ones in the same session.
2. Structured queries: more advanced users might prefer structured queries.
3. Different visualization types: the system can recommend better ways to visualize the requested data.

The component will be first evaluated offline via state-of-the-art metrics and benchmarks, and then through rigorous testing with a selected user base.

2.1.2 Component Architecture

The preliminary design of the Query Recommendation component leverages three interconnected modules:

1. **Ingestion**: takes data from the user query, past logs, user profiles and makes it into a coherent data structure.
2. **User profile summarizer**: a sub module that summarizes the user profile and its query log to condense enough relevant information for the recommender.
3. **Processing**: a deep learning approach that generates the next query by processing the previous one along with its results, and the behaviors/preferences inferred from the user profile summary.



Input and output: The component takes as input the previous query (both in natural language and structured) and its results, i.e. the data and the related datasets. Additionally, the user profile and its query logs are also taken into account. The output is either a new query that follows logically from the previous, or a refinement of it (restricting and better specifying its scope).

2.1.3 Integration with the DataGEMS platform

The main input to the recommender is the previous query by the user, which means that this model will receive this data from the user interface. In addition to this, and as described in the previous section, the Query recommender requires access to the user profiles, which are provided by the MoMa Object Catalog (specifically Task 5.2). The query logs are also obtained via connection to the same component. Lastly, the system requires the results of the previous query, thus it will be connected to all the systems that provide such results (e.g. In-Dataset Discovery, Dataset Packaging).

Dependency	Dependency reason
MoMa Objects Catalog	Require access to the user profiles to provide personalized recommendations.
User Interface	The recommender needs the current user query.

2.1.4 Technologies

A first prototype of the system is being built in Python, leveraging several common libraries for Data Science such as Pandas, Numpy, sklearn, and Keras, while also accessing an LLM through native API calls. The connection to the DataGEMS ecosystem will be provided through FastAPI.

2.1.5 Security policies

No specific security policies are expected for this module; it will follow the global security policies of DataGEMS.

2.1.6 Monitoring and Accounting

In order to ensure the proper utilization of this module and increase its efficiency, we will keep track of the suggestions that are selected by the user. This will allow the recommender to improve over time and offer a more personalized experience to the user.

User feedback in general will be important for ascertaining the performance of the Query Recommender.

Monitoring	
Check	Description
Health API	Routinely check for API health

Accounting	
Metric/Indicator	Description
User Feedback	We will keep track of the user feedback to improve the quality of the recommendations

2.2 Query Disambiguation

2.2.1 Introduction

Query disambiguation enhances data exploration by accurately interpreting user intent and clarifying natural language queries in relation to the dataset. For instance, the question, "*What is the average temperature of Zurich over the last ten years?",* should the average be calculated annually, monthly, or otherwise? Additionally, does "Zurich" refer to the city or the canton, and what specific start and end years define the ten-year years?

This process serves as a critical upstream optimization for both in-dataset and cross-dataset data discovery (WP3). The system employs an LLM-based approach, human-in-the-loop feedback, and data-specific suggestions to resolve ambiguities, while decomposing complex queries into simpler sub-questions and providing explainable disambiguation through self-reasoning. This ensures precise, contextually relevant query interpretations tailored to the dataset and user needs.

The query disambiguation component will undergo thorough evaluation, starting with offline assessments using state-of-the-art metrics and benchmarks, followed by user-based testing to validate its effectiveness in real-world scenarios.

2.2.2 Component Architecture

The design of query disambiguation leverages LLMs and human-in-the-loop (HITL) feedback to generate context-driven disambiguated options, empowering the user with complete control over the final output (see Figure 2). Each option is annotated with a transparent analysis of the self-reasoning process for enhanced explainability. A potential ambiguity is analyzed by an LLM considering the respective data profile. Next, the system provides disambiguation suggestions with explanations that can be verified via a human-in-the-loop approach. If the advice is accepted by the human, the query is modified and the result is generated and formatted accordingly. For more details on query disambiguation, also consider the examples of in-dataset discovery of Deliverable D3.1.

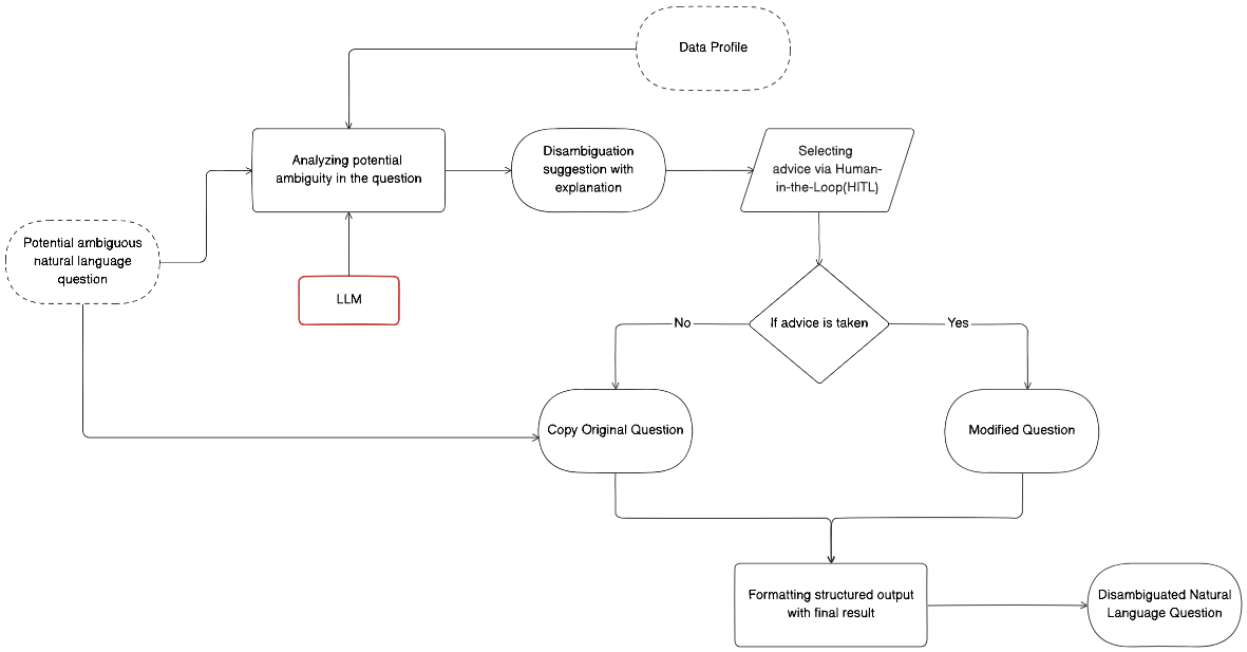


Figure 2 The components of the Query Disambiguation Service.

2.2.3 Integration with the DataGEMS platform

The query disambiguation component is a vital upstream component for data discovery, which refines user intent by utilizing contextual dataset profiles. During the integration, the dataset profile is required as a mandatory input and ensures accessibility through the user interface.

Dependency	Dependency reason
MoMa Objects Catalog	Requires access to the data profiles to provide the meta data of the dataset.
User Interface	The access to current user question is mandatory.

2.2.4 Technologies

By resolving ambiguities in natural language queries, the query disambiguation component ensures the system correctly understands the user's goals, enabling precise and relevant data retrieval. This is achieved through a combination of advanced techniques implemented via Python, including:

- **LLM-based approach**: The component leverages LLMs to interpret and disambiguate natural language queries. By utilizing the contextual understanding and semantic reasoning capabilities of LLMs, the system identifies potential ambiguities in user queries and generates clarified versions that align with the dataset's content and structure.

- **Human-in-the-loop (HITL) disambiguation**: To enhance accuracy and user trust, the system incorporates a human-in-the-loop approach. When the system detects high uncertainty in query interpretation, it prompts users with clarification options or alternative query formulations, allowing them to guide the disambiguation process. This iterative feedback loop ensures that the system aligns closely with user intent.
- **Explainable disambiguation by self-reasoning**: The system employs self-reasoning to provide transparent explanations for its disambiguation decisions. By articulating the steps taken to resolve ambiguities—such as identifying keywords and analyzing dataset context—the component ensures users understand how their query was interpreted and refined, fostering trust and usability.
- **Dataset-specific suggestion**: Query disambiguation relies on data profiling from WP2 to generate contextually relevant suggestions. By understanding the dataset's schema, relationships, and metadata, the system tailors disambiguation outputs to the specific characteristics of the data, ensuring that clarified queries are both precise and aligned with the dataset's structure and content.

2.2.5 Security Policies

The query disambiguation component conforms to the global service security framework of DataGEMS.

2.2.6 Monitoring and Accounting

Monitoring. The query disambiguation service will monitor two critical components to ensure reliable performance: the responsiveness of its API endpoints and the continuous connectivity to the MoMa objects catalog for accessing the data profile.

Accounting. The token length and elapsed time per query serve as critical indicators of service quality. Collecting data on user actions is essential for informing future optimizations.

Monitoring	
Check	Description
Health API	Verify that the disambiguation API endpoints are responsive and return valid responses.
MoMA connection	Ensure uninterrupted connectivity to the data profile

Accounting	
Metric/Indicator	Description
User Actions	We will keep track of the user actions to improve the query disambiguation.
Average token counts per query	The count of prompting tokens and completion tokens on the LLM.
Average querying time	The elapsed time of querying, i.e., the period from receiving request to sending out the response.

2.3 Dataset Recommendation

2.3.1 Introduction

Within the DataGEMS ecosystem, dataset recommendation services are designed to assist users in discovering datasets more efficiently by reducing the effort required to identify resources relevant to their current needs. As a first step toward this goal, a *dataset-to-dataset recommendation service* is being developed, which recommends related resources based on the dataset a user is actively exploring[1]. For example, when viewing a dataset on climate time series, the service may recommend other datasets related to weather forecasting or environmental monitoring. In its first stage, the recommender will be relying primarily on dataset metadata (i.e., title, description), with plans to incorporate additional signals (such as dataset links and exploration patterns) as the project evolves.

An offline evaluation of this component is planned to assess its effectiveness in supporting dataset discovery. The evaluation will focus on accuracy, measured with standard recommendation metrics (e.g., Precision, Recall, NDCG). Experiments will be conducted using project's data as well as established benchmarks in the dataset recommendation domain (e.g., DataFinder[2]), and results will be compared against strong baseline methods for the same task.

2.3.2 Component Architecture

The dataset-to-dataset recommender is designed as a modular pipeline that transforms dataset metadata into meaningful representations, generates candidate recommendations, and re-ranks them to provide the most relevant results to the user. Figure 3 illustrates the main architectural components.

[1] Wang, Xu, Frank van Harmelen, and Zhisheng Huang. "Biomedical Dataset Recommendation." DATA. 2021.
[2] Viswanathan, Vijay, et al. "DataFinder: Scientific dataset recommendation from natural language descriptions." arXiv preprint arXiv:2305.16636 (2023).

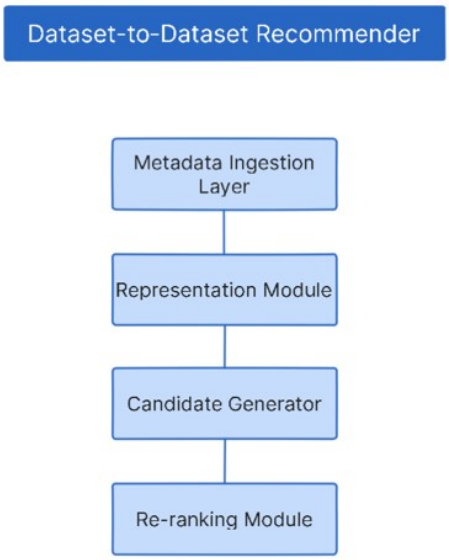


Figure 3 The components of the Dataset-to-Dataset Recommendation Service

Input and output. The component receives as input the metadata of a query dataset that the user is actively exploring within the DataGEMS platform. The output is a ranked list of similar datasets stored in the data lake, presented directly on the dataset's page to assist user discovery. Since all datasets follow a common representation (metadata and, in future, links between them) no special handling based on their format is needed.

Internal Modules. The recommender consists of the following modules:

- **Metadata Ingestion Layer.** This module collects and normalizes dataset metadata (e.g., titles, descriptions, task/domain annotations, etc.) from the DataGEMS data lake. Both structured attributes (e.g., task type) and unstructured text (e.g., descriptions) are ingested and pre-processed.
- **Representation Module.** Metadata is transformed into vector representations suitable for similarity computation. The plan is to employ a neural encoder such as Specter[3], which is well suited for scientific metadata. In later stages of the project, representations are expected to also incorporate dataset links, so that structural relationships between datasets complement the contextual information extracted from metadata.
- **Candidate Generator.** Given a query dataset, this module retrieves a pool of potentially relevant datasets from the database (e.g., using a nearest-neighbor search in the embedding space).
- **Re-ranking Module.** The candidate datasets are re-ordered based on additional criteria such as relevance based on user's exploration history or diversity of information, producing the final ranked list presented to the user.

Offline Processes. The recommender relies on offline indexing, where dataset metadata is processed, transformed into embeddings, and used to precompute the top-N related datasets for

[3] Cohan, Arman, et al. "Specter: Document-level representation learning using citation-informed transformers." arXiv preprint arXiv:2004.07180 (2020).

each entry. These recommendation pairs are stored in a PostgreSQL database, ensuring that results can be retrieved efficiently at query time without requiring on-the-fly computation.

2.3.3 Integration with the DataGEMS platform

The dataset-to-dataset recommendation service will be integrated into the DataGEMS platform as part of the dataset discovery workflow. It is invoked whenever a user explores a dataset within the platform. On each dataset's profile page, the recommender provides a ranked list of related datasets from the data lake, allowing users to quickly identify additional resources relevant to their task or domain of interest. In this way, the component acts as an assistant for dataset exploration, enhancing the discoverability and reuse of resources available in DataGEMS.

The component depends on several key services of the platform to function effectively:

- **MoMa Objects Catalog.** The catalog provides structured dataset descriptions that are required to build contextual embeddings. In later stages, the recommender will also leverage dataset links, created by the *dataset linking service* (WP2) and exposed through MoMa, as additional signals to enrich and refine recommendations.
- **User Profiling Service.** If personalization is introduced in the future, the recommender will also depend on the *user profiling service* to draw information from user exploration history (e.g., viewed datasets, search activity, etc.).

Dependency	Dependency reason
MoMa Objects Catalog	Provides access to dataset metadata and links for building representations.
User Profiling Service	Provides information from user exploration history (subject to explicit user consent).

2.3.4 Technologies

The dataset recommendation service is being implemented in Python. Core components rely on scikit-learn and Hugging Face Transformers, with pandas and Numpy supporting data handling. Recommendations will be stored and retrieved through a PostgreSQL database, which maintains dataset identifiers and their related recommendations. For integration within DataGEMS, the service will be exposed as web API though FastAPI.

2.3.5 Security Policies

The dataset recommendation service will comply with the security policies of the broader DataGEMS platform. In its first stage, recommendations will be generated solely from dataset metadata and will not include any form of personalization. This means that all users with access to the DataGEMS platform can view recommendations directly on dataset pages without requiring additional permissions. If, in future stages of the project, the service incorporates personalized recommendations based on user exploration history, explicit consent will be requested from the

user before any personal activity data is processed. This ensures compliance with data protection regulations and aligns with best practices for user privacy[4].

2.3.6 Monitoring and Accounting

Monitoring. The dataset recommendation service will monitor two key aspects to ensure reliable operation: the responsiveness of its API endpoints and the uninterrupted connectivity to the PostgreSQL database where dataset-recommendation pairs are stored. These checks guarantee that the service remains accessible and able to always return recommendations.

Monitoring	
Check	Description
Health API	Verify that the recommendation API endpoints are responsive and return valid responses.
PostgreSQL connection	Ensure uninterrupted connectivity to the database where dataset-recommendation pairs are stored.

Accounting. The dataset recommendation service will track three key indicators of its coverage, usage, and performance: the number of datasets indexed in the PostgreSQL database (coverage), the number of recommendation queries served through the API (usage), and the latency of responses (performance). Together, these indicators provide insight into both the adoption of the service and its operational performance.

Accounting	
Metric/Indicator	Description
Number of datasets indexed	Total number of datasets stored in PostgreSQL with recommendations.
Number of queries served	Total recommendation requests processed by the API.
Query latency	Average response time for serving recommendations.

2.4 Analytical Pattern Explanations

2.4.1 Introduction

The Analytical Pattern Explanation component is an optional component in the user flow. Its main objective is to enable users to get insights on how and why Analytical Patterns (AP) compute their answers, by using data provenance and interpretability models. To this end, the AP Explanation component needs to use the MoMa data model to retrieve the analytical patterns and the datasets that are used in the process.

Explanations are computed based on the sequence of operations that the APs contain. To enable tracking of dataset provenance and explanation, the component must interact with the MoMa data model and catalog to add provenance/explanation annotations to the dataset and its data fragments. Once datasets are annotation-enabled, explanations are provenance information that can be used for any AP that uses these datasets.

[4] https://gdpr-info.eu

Example user-facing functionalities enabled by the AP Explanation component are: (1) asking from which datasets the answer was derived, (2) obtaining all data fragments that were used for the calculation of an answer, (3) enabling what-if analysis by recomputing the answers in case some information is added or removed.

The AP Explanation component relies on the following main functionalities for computing explanations:

- *Dataset and Data Fragment Annotation*, where datasets are data fragments annotated with explanations and provenance-enabled tokens.
- *Computation of Explanations*, by parallelly running the APs on the annotated datasets and data fragments.
- *Storage and Retrieval on Explanations,* by adding the explanations as special types of datasets in MoMa for fast retrieval without the need to recompute explanations.

The resulting explanation is **evaluated on two complementary dimensions**:

- the performance and overhead of the computation and storage of explanations, by measuring the added time for query answers when explanations are also needed and by measuring the extra storage space needed by the
- the user feedback, by assessing whether the explanations are intuitive and/or useful; this can be done at the UI level.

2.4.2 Component Architecture

The AP Explanation component contains an engine that is tasked with computing explanations. The flow of this engine depends on what type of explanation is requested, and a specific computation case is that of SQL query provenance computation. The main flow is given in Figure 4.

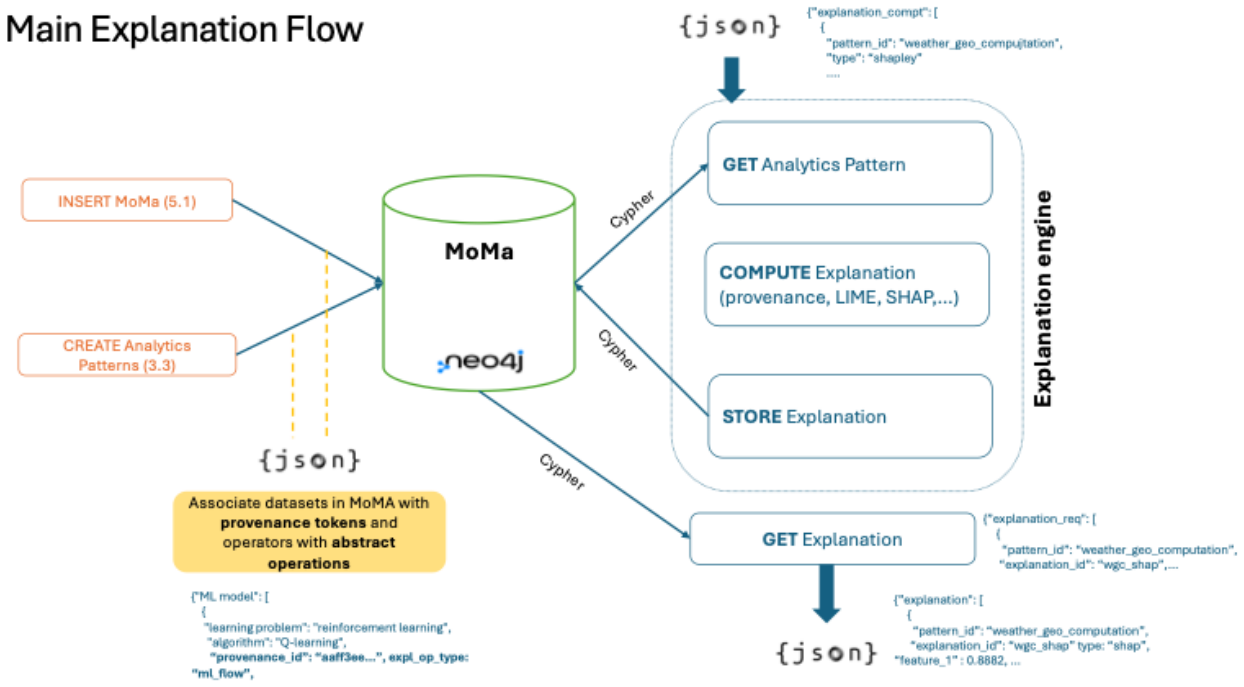


Figure 4 AP Explanations architecture

Inputs and Outputs. The main input of the component are the Analytical Patterns that need explanations. These are retrieved from the MoMa object store and represent the entire flow of the task being executed and the input datasets. The input datasets are useful for provenance annotations and explainability. The outputs are Analytical Patterns modified to show the explanations that are computed, along with the resulting explanation ids (stored as nodes in MoMa) and the modified datasets, depending on the type of explanation that is computed.

Explanation Engine. The internal engine provides the main computation of explanations. It consists of several algorithms and methods, depending on what type of explanation is desired:

- For *structured relational data*, the engine will compute provenance information for SQL[5], queries that can result from NL-to-SQL operators for instance. The provenance algorithm uses the open-source PostgreSQL plug-in, ProvSQL[6], which operates in two steps: (1) adds provenance tokens to any relational dataset present in the DataGEMS store, in an *offline manner;* (2) at the time of any query on the plugin, a *provenance circuit* is computed and stored in the PostgreSQL store, and (3) the provenance circuit can be used to show to users which data fragments (tuples) were used in the computation.
- For *AP-level explanations,* the engine will output the datasets used at each step, by using provenance algorithms at the *dataset level*, and by using graph provenance approaches[7].

[5] Aryak Sen, Silviu Maniu, Pierre Senellart "ProvSQL: A General System for Keeping Track of the Provenance and Probability of Data". CoRRabs/2504.12058 (2025)
[6] https://provsql.org/
[7] Yann Ramusat, Silviu Maniu, Pierre Senellart "Provenance-Based Algorithms for Rich Queries over Graph Databases". EDBT 2021: 73-84

- Optionally, operators that use ML models can request off-the-shelf state of the art explanation algorithms, such as LIME or SHAP, for explanation analysis of the features used in the models' predictions.

2.4.3 Integration with the DataGEMS platform

The component integrates into the DataGEMS platform into the two main workflows of the platform: the dataset search workflow and the user query processing workflow. In both cases, it is an optional module, only being executed when the user requests it.

In the **dataset search workflow**, the AP Explanation component is used when datasets are found by retrieving related analytical patterns and related explanations, in the form of example summaries of how the execution of the AP is performed on the data present in the retrieved datasets, by directly accessing the MoMa Object Catalog and, if needed, the data present in the dataset original storage.

In the case of the **user query workflow**, once the user request is disambiguated and translated into a structured language (such as SQL) and then executed, the user can request explanations of the executed analytical pattern. The output explanation will take the form of a workflow containing data fragments for the input datasets that details the running of the analytical pattern. An example are SQL queries on tabular datasets where the explanation will take the form of the summary of a data provenance algorithm where the inputs are specific tuples and the operations are the SQL operators (select, project, join).

To enable explanation computation, the component needs access to the components detailed in the following table.

Dependency	Dependency reason
MoMa Objects Catalog	Access to Analytical Patterns and Dataset metadata to enable explanation and provenance computation and storage of explanations.
Direct Data Access	The component requires direct access to the dataset storage to update and change data fragments.
PostgreSQL Direct Access	The component needs direct access to the PostgreSQL server for implementing and calling the ProvSQL plugin for provenance computation.

2.4.4 Technologies

The component is implemented as a FastAPI service providing access methods for creating, computing, and retrieving explanations. It is based on code implemented in Python. Additionally, for explanations given as data provenance in SQL, the main functionality is given by a PostgreSQL plugin, ProvSQL, which is implemented in SQL and C++.

2.4.5 Security policies

The AP Explanation component service generates explanations only when users have access to the data relevant to the explanation. Specifically, explanations will only be available as an option

in the dataset search and query workflows and only if the datasets security and privacy clearances allow unrestricted access to the underlying data. The reason is that for computing reliable outputs, the component needs access to all datasets and data fragments; outputting explanation on limited or restricted data would not return useful insight. Other than these cases, the component follow the DataGEMS security requirements.

2.4.6 Monitoring and Accounting

Monitoring The AP explanation component needs to continually monitor the status of the main providers of data: the PostgreSQL server and the MoMa Object Store. To enable this, frequent heartbeat checks will be sent to the two servers; if no connection is available, the component will not run its functionalities. Optionally, the explanation component may monitor the primary data stores where the datasets that may be used are kept. Internally, the API service will provide API health status messages via endpoints for the use of other downstream components.

Monitoring	
Check	Description
Health API	Check that the API endpoints are responding correctly and timely.
PostgreSQL server	Check the health of the PostgreSQL server containing the provenance and explanation plugin.
MoMa object server	Check the health of MoMa, necessary for AP retrieval.
Dataset connections	Check that connections to base dataset storage is available.

Accounting For fine-tuned analysis of the performance and adoption of the components, the AP Explanation component will keep track of the number of explanations that have been computed, the number of explanations that have been retrieved, the size of the explanations (number of operations, number of data fragments), statistics on the type of explanations, statistics on the datasets used (number, proportion, types), and meta-information about the API (number of calls, average response time, number of failures/timeouts).

Accounting	
Metric/Indicator	Description
Number of explanation APs computed	Total number of explanations APs
Number of explanation APs retrieved	Total number of explanations requested in the dataset search and user query workflows.
Explanation statistics	Types of explanations, types of datasets used, number of datasets used, etc.
API Latency	Average response time of the API
API Calls	Total number of API calls.
Accepted Explanations	Proportion of explanations accepted by the users
Storage overhead	The extra storage space needed by the explanations.

3 User Assistance for Use Cases

3.1 Introduction

The User Assistance components will implement functionalities at least in three use cases:

- ***Higher Education*** (personalized student- and teacher-driven quiz recommendation via query recommendation, recommending additional resources via dataset recommendations, and providing explanation of quiz outcomes),
- ***Weather*** (query reformulation via disambiguation, explanation of outputs for queries asked on weather data), and
- ***Language*** (enriching datasets with additional relevant texts via dataset recommendation).

3.2 Query Recommendation in Action

For the ***Higher Education*** Use Case, Task UC1T2 (student-level adaptive quiz recommendation), the Query Recommendation component will suggest queries to retrieve exercises personalized based on the student profile and history of interaction with DataGEMS.

For example, imagine Bob, a high-school student, who enters the query "Function domain exercise". Based on his past interactions, DataGEMS recognizes that:

- He previously struggled with rational functions involving square roots.
- He recently completed several basic polynomial domain exercises with high accuracy.
- He spent extra time on logarithmic domain rules but eventually improved.

Given this history, instead of simply retrieving a generic list of “function domain” exercises, the system adapts the recommendation:

- It suggests slightly more advanced queries such as "Domain of rational functions with square roots" or "Logarithmic function domain practice".
- It offers more precise queries to retrieve a mixed sequence of MCQs, personalized to Bob’s learning path, ensuring that the exercises are both relevant and progressively adaptive.

Thus, while a new student entering "Function domain exercise" might be guided toward general introductory tasks, Bob is instead nudged toward targeted practice that reflects his unique journey, accelerating his mastery without redundancy.

For the Use Case Task UC1T5 (teacher-driven quiz composition), suppose Ms. Green, a math teacher, enters the query “Function domain quiz” into DataGEMS. The system interprets her role and intent: DataGEMS looks at class-level profiles, aggregate student performance data, and curriculum alignment. For example, if most of her students are struggling with rational functions but have already mastered polynomial domains, the query recommendation engine adapts accordingly.

Therefore, when the teacher requests “Function domain quiz” the module acts with an adaptive goal as follows:

- it suggests alternative queries such as “Introductory function domain quiz for beginners” or “Rational function domain exercises” depending on class performance.
- with analytics-driven insights, the system may highlight “70% of your class struggled with logarithmic domains last week. Would you like to include a review section?”

3.3 Dataset Recommendation in Action

In the ***Higher Education*** use case, Task UC1T3 (pedagogical resource creation and recommendation), the dataset recommendation service may support students by suggesting additional educational resources related to the material they are viewing.

For instance, imagine Valerios, a primary-school student, who is working on a quiz about fractions. Alongside the quiz, the system recommends a PDF guide on fraction rules and a video lecture on fraction-to-decimal conversions.

In this scenario, the recommendation service operates not on datasets but on educational resources, guiding students to complementary material that reinforces their learning path and helps them better understand challenging concepts.

In the ***Language*** Use Case, Task UC4T5 (semantic and metadata-based retrieval), the same service may support retrieval of relevant texts based on semantic similarity.

For example, imagine Maria, a linguistics researcher studying a corpus of Spanish parliamentary debates. While examining this dataset, the system recommends other Spanish political discourse corpora from different time periods, as well as parallel corpora of parliamentary debates in English and French. These additional resources allow Maria to expand her analysis across languages and time, uncovering comparative insights that would otherwise remain hidden.

In this scenario, the recommendation service operates on textual corpora and documents.

It is important to note that in all these scenarios the representation module of the dataset-to-dataset service must adapt to the type of “item” being recommended (i.e., datasets, educational material, documents, etc.).

3.4 Query Disambiguation in Action

Focusing on the ***Weather*** use case and task UC3T1 (NL Querying Interpretation) where the query disambiguation service provides correct interpretation of the users’ intent.

For instance, let us revisit the question, “*What is the average temperature of Zurich over the last ten years?”.* By leveraging LLMs, the service identifies that Zurich could either be a city or a canton. Thus, it asks the user for the intended meaning.

Moreover, the system also asks the user if the average should be over ten years or if the average should be per year. In the first case, the result would be one value, while in the second case the result would be ten values.

Finally, the system asks the user if the 10-year observation window should start as of today, i.e. September 24, 2025, or with respect to the end of 2024.

3.5 Analytical Pattern Explanations in Action

For the ***Higher Education*** use case, the component operates in the UC1T4 (Teacher Dashboard and Pattern Analysis), by providing the provenance of the original quiz questions/answers that contributed to an outcome.

An example of explanation functionality in the case of the Higher Education use case is when a teacher wants to analyze the outcomes of a quiz. Explanations in this case can provide the provenance of the outcome at a finer level than just the answer of the query. For instance, imagine a lecture asks the question "*What are the topics for questions of level 4 that are wrongly answered?"*. The answer to this query can be, e.g., "*Complex Numbers"* but the answer can be enriched with finer details. In this case, they represent the original questions that contributed to this outcome.

The dashboard could then contain the answer to the question *"Complex Numbers"* and detail the questions and the answers that contributed to it. For example, it could contain: "*The answers occurs because question 349 was answered wrongly in answers 785, 11438, and 8764"*. This explanation can help teachers gain insights into the difficulty of their quizzes and take action to correct it if needed.

In the ***Weather*** use case, the component can provide support for the UC3T5 (Result Export and Reusability) and UC3T3 (Data Processing), by providing provenance of the SQL queries transformed from the NL queries asked by the users.

One example is when an analysis of temperature patterns in a location, for example Athens, shows some outliers in the temperature readings. Since the temperature patterns are aggregated via SQL queries from reading of individual weathers stations, the SQL provenance computation can keep pointers to the stations that contributed to the results and show them. This allows weather analysts to see whether some weather stations output incorrect data and remove them if this is the case.