Irving Fisher Committee on
Central Bank Statistics

BIS

IFC-Bank of Italy Workshop on "Data science in central banking"

18-20 February 2025

Supporting users in seeking data on the ECB data portal: a use case for retrieval augmented generation[1]

Luca Petracca, Simone De Benedictis,
Thomas Gottron and Zlatina Hofmeister,
European Central Bank

[1] This contribution was prepared for the workshop. The views expressed in this publication are those of the authors and do not necessarily represent the official views of the Committee, its members, or the BIS.

Supporting users in seeking data on the ECB Data Portal: a use case for Retrieval Augmented Generation

Luca Petracca, Simone De Benedictis, Thomas Gottron, Zlatina Hofmeister[1]

Abstract

The European Central Bank (ECB) Data Portal holds vast economic data. However, users occasionally struggle to locate specific information due to inefficient search functionality based on traditional information retrieval methods. This is particularly true for users who are unfamiliar with economic terminology or the dataset structures available on the EDP. To address this, we propose a chatbot interface that leverages Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) techniques. The system can understand complex natural language queries, accurately identify relevant datasets and time series, and provide precise data points along with contextual information and sources. Focusing on high-demand time series, initial evaluations demonstrated high accuracy in dataset identification and series key provision, showcasing the potential to enhance data retrieval on the ECB Data Portal.

Keywords: data portal, artificial intelligence, large language models, retrieval augmented generation, information retrieval

JEL classification: C45, C55, C67, C82

[1] Disclaimer: This paper should not be reported as representing the views of the European Central Bank (ECB). The views expressed are those of the authors and do not necessarily reflect those of the ECB.

Contents

Introduction

The ECB Data Portal[2] hosts vast amounts of economic data, but users often struggle to locate specific information efficiently. Traditional search methods can be inefficient, especially for users unfamiliar with economic terminology or dataset structures. There is a clear need for a more intuitive, user-friendly interface that can understand and respond to queries about economic indicators and time series data using natural language. Enhancing access to ECB data is crucial for researchers, policymakers, and the public to make informed decisions based on accurate economic information. We investigated the option of a chatbot interface that can understand complex queries and provide precise data points, along with contextual information and sources.

Developing an effective chatbot for economic data retrieval poses several challenges. The system must accurately understand the user's information need, identify the right dataset and relevant time series, and retrieve and provide responses that are both precise and contextually relevant. Furthermore, it needs to handle heterogeneous time series data, including various indicators, timeframes, and metadata, while ensuring data accuracy and retrieval.

Previous approaches to improving data portal accessibility have often relied on keyword-based search engines or categorization systems, which lack the flexibility to interpret natural language queries effectively. While Large Language Models (LLMs) have demonstrated promising performance in general question-answering tasks, their application to specialized domains like time series data retrieval has been limited. This project proposes an integration of LLMs and Retrieval-Augmented Generation (RAG) techniques specifically for the ECB Data Portal's requirements.

Our approach combines latest state-of-the-art models for natural language understanding and response generation with RAG techniques to enhance the model's knowledge of ECB datasets and time series. The system processes user queries through several stages: query refinement, query embedding, index search, dataset and time series matching, and response generation with source attribution. We implemented this solution focusing on high-demand time series. Initial evaluations demonstrate high accuracy in dataset identification and series key provision showcasing the potential of this approach to drastically improve economic data retrieval on the ECB Data Portal. Finally, as for all LLM based applications, we looked at safeguarding mechanisms to ensure a safe operation of the system.

In this paper we discuss the technical setup of the overall system, illustrate the design of the RAG component, explain the evaluation methodology and present our insights and findings.

Background and Related Work

Lewis et.al. (Lewis, et al., 2020) introduced the concept of retrieval augmented generation (RAG) as a solution for knowledge intensive NLP tasks that need to

[2] data.ecb.europa.eu/

incorporate specific domain knowledge. Since, the concept has received a lot of attention, especially for use cases where external knowledge needs to be incorporated into the NLP task, but otherwise an out-of-the-box LLM without further fine-tuning is sufficient. Researchers have invested much effort in optimising the naïve RAG approach and designed advanced and modular solutions to address and overcome certain challenges and shortcomings. The survey presented by Gao et.al. (Gao, et al., 2024) provides a good overview of the recent developments.

Interacting AI agents are a common concept for complex solutions. They cover a wide range of use cases and diverse methodological approaches. The idea to base AI agents on LLMs as core method has been investigated for various use cases (Xi, et al., 2023). In the context of combining them for the sake of NLP tasks or, more specifically even, for RAG solutions the AI agents are also referred to as modules (Gao, et al., 2024). An interesting observation is that the combination of multiple modules with specific purposes often results in better system performance, compared to a monolithic LLM. The modules perform tasks like query refinement (Ma, Gong, He, Zhao, & Duan, 2023) or introducing a judge to assess and enhance the generation of the final response (Shao, et al., 2023). Furthermore, small and specialised models with a narrow scope and task in the architecture pipeline seem to provide good contributions in a context of a wider task (Lewis, et al., 2020).

Evaluation of systems based on LLMs and RAG is a complex and evolving topic. Traditional evaluation metrics like recall and precision (Baeza-Yates & Ribeiro-Neto, 1999) are of limited use and commonly applied only to the retrieval component. The overall system performance is sometimes assessed using classical text summarisation metrics like ROUGE (Lin, 2004). More advanced evaluation approaches like RAGAs (Es, James, Espinosa-Anke, & Schockaert, 2024) seek to assess faithfulness, answer relevance and context relevance or involve LLMs as reliable judges for grading the system output (Zheng, et al., 2023).

The architecture for a RAG solution

In this section we describe our approach, the architecture, and technical details. In particular, we motivate certain design decisions and explain how the prototype solution interacts with the users and the information from backend systems.

High-level overview of the architecture of our RAG based solution Figure 1

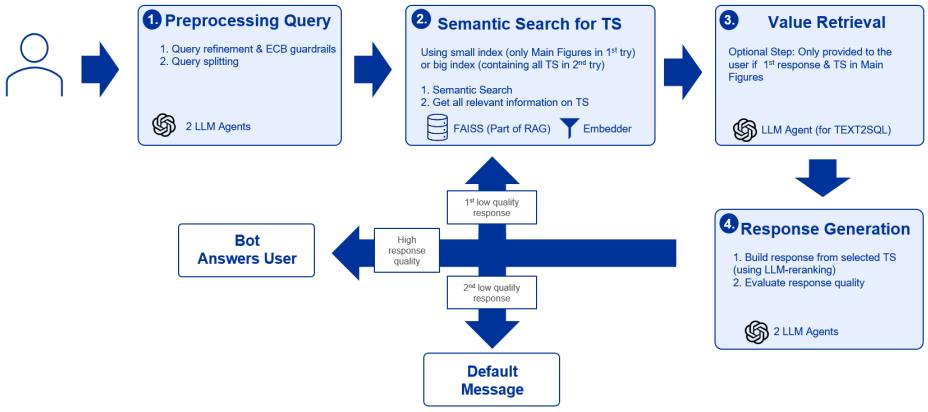


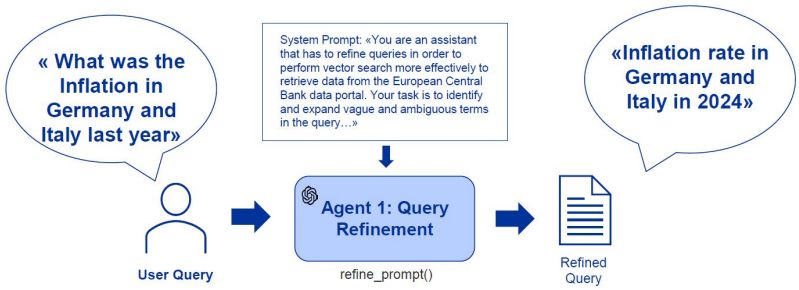
Figure 1 gives a high-level overview of the architecture of our solution. It illustrates how the prototype is leveraging a pipeline of different agents to achieve the intended result and output. Each agent is responsible for a specific task in the overall pipeline. This setup has proved more robust performance compared to simpler and more generic setups we have tested during the development. Several agents within this pipeline leverage LLMs, each configured with specific system prompts that guide their individual task and – whenever possible – examples to implement few-shot learning.

User Query Preprocessing

The first agent in our pipeline is responsible for query refinement. As users interact with the system through natural language, their queries may contain ambiguities, colloquial terms, or lack the precise terminology expected by the data portal's search engine. Therefore, the primary purpose of the *Query Refinement Agent* is to standardize and improve the quality of user-provided input. For example, as illustrated in Figure 2 the agent processes a user query like "*What was the Inflation in Germany and Italy last year?*", by interpreting phrases such as "*Inflation*" as "*Inflation rate*" – the more formally recognized term in economic data contexts. Similarly, relative temporal references, like "*last year*", are converted to specific and absolute time periods. The system prompt driving this agent directs the LLM to act as a helpful expert in preparing queries for optimal data retrieval from the ECB Data Portal. The outcome of this agent is a *Refined Query*, like "*Inflation rate in Germany and Italy in 2024*", that is clearer and more structured. This harmonization step bridges the gap between the various ways users might express their information needs and the system's structured understanding of economic data concepts within the EDP.

Query Refinement Agent

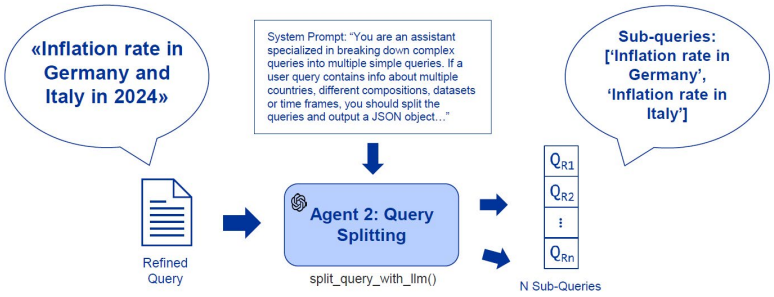
Figure 2



Subsequently, the second agent, takes the *Refined Query* as input and returns a list of multiple, distinct user requests. The key function of this agent is to recognize when a single user question is actually asking for several pieces of information simultaneously, such as when data is requested across multiple geographical regions, for different economic indicators, or across varying time frames.

Query Splitting Agent

Figure 3



For instance, as shown in Figure 3, for the *Refined Query* "*Inflation rate in Germany and Italy in 2024*", the agent's job is to understand that the user is interested in two distinct pieces of information: the inflation rate for Germany and the inflation rate for Italy. To handle these types of queries, the *Query Splitting Agent's* system prompt is instructed to analyse the refined query and, when appropriate, segment it into *n Sub-Queries*. In the given example, the agent effectively breaks down the initial refined query into a list of discrete, simpler sub-queries such as ['Inflation rate in Germany', 'Inflation rate in Italy']. These sub-queries are now easier to manage and address in the following data retrieval steps. Again, in this way we adjust the query in a way that still reflects the user's information need but is conceptually closer to how data is stored in the EDP and captured in our search index.

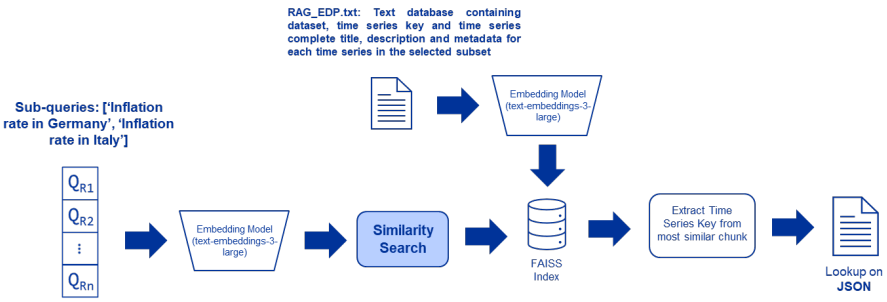
Information Retrieval using Semantic Search

After the query refining and splitting steps, the process continues with the retrieval component, a critical element in our architecture designed to efficiently find the relevant data from the ECB Data Portal. As shown in Figure 4, the initial operation

within the retrieval component involves the transformation of the sub-queries into a vector-based representation. Specifically, each sub-query is embedded via the *text-embedding-3-large* model. This embedding process is crucial for converting natural language text into high-dimensional vectors that numerically encode semantic meaning. The choice of text-embedding-3-large is justified by it being one of the state-of-the-art embedding models available on the market according to several benchmarks (OpenAI, 2024).

Similarity Search

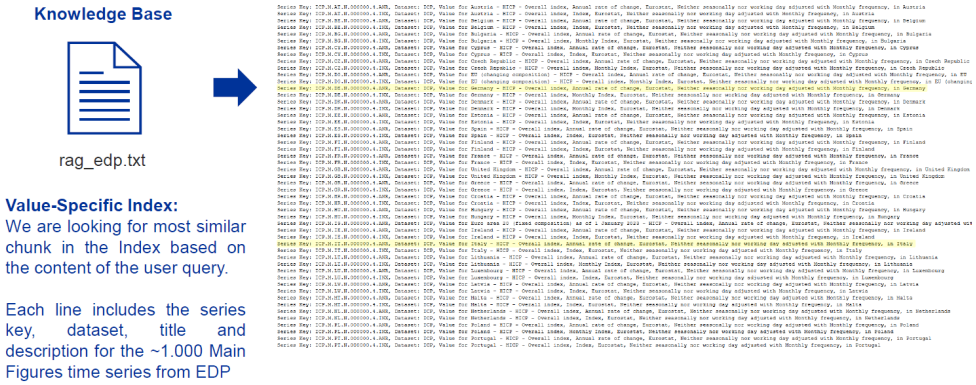
Figure 4



After embedding sub-queries, we perform a similarity search (cf. Figure 4). In a **first-stage search** we use an index built using metadata embeddings derived *exclusively* from time series categorized as "Main Figures" on the ECB Data Portal – representing the top 1000 most frequently demanded and accessed datasets on the EDP website, as shown in Figure 5. The rationale for this focused index is based on the idea that most user queries are likely to be related to the most relevant macroeconomic indicators. By initially searching within this constrained, high-relevance index, the system aims to rapidly identify data for frequent and standard user requests and to provide a better and faster user experience. The similarity search uses cosine similarity to identify the top-k most similar time series keys.

Main Figures Index

Figure 5



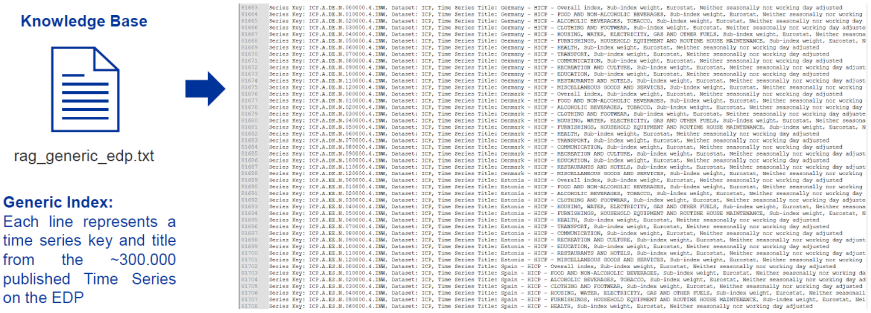
Based on the retrieved time series from the Main Figures, we generate a preliminary answer. This answer is subjected to evaluation by a **Judge Agent**. As elaborated below, the Judge Agent assesses the relevance and quality of the generated response against the original user query. If the Judge Agent's rating is

deemed **satisfactory**, the system proceeds with this initial retrieval set to construct and deliver a "Specific Answer" to the user.

However, if the **Judge Agent assess the answer as not satisfactory**, we conclude that the initial search on Main Figures did not provide relevant results. In this case the system reverts to a **second-stage search**. This secondary search is performed using a broader index. This index is constructed from metadata embeddings representing a much larger subset of the EDP data – encompassing approximately 300,000 time series keys, as shown in Figure 6. This index covers a significantly larger collection of datasets, including those featured in official ECB publications, reports, and analytical articles. Hence, it is designed to capture a more comprehensive representation of the EDP's data offerings, providing a fallback mechanism when user queries extend beyond the most commonly accessed "Main Figures". Technically, this second-stage search follows the same process, using a cosine similarity-based search is performed to retrieve the top-k most similar time series keys.

Generic Index

Figure 6



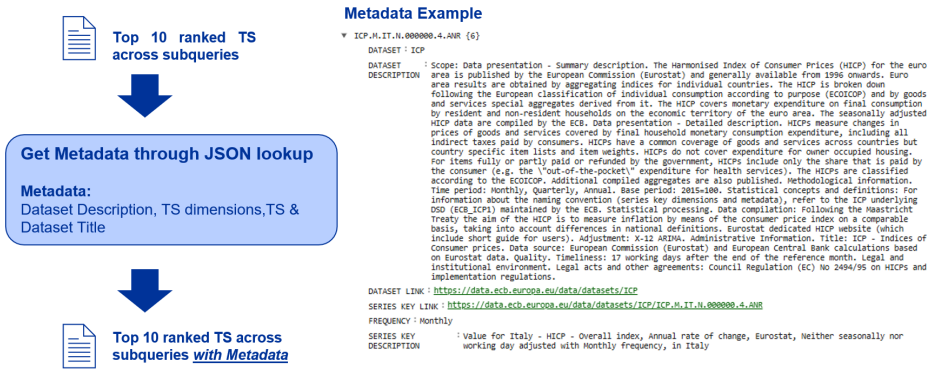
The answer generated from time series retrieved from the *"Generic Index"* is *again* presented to the **Judge Agent** for assessment. As before, if the Judge Agent assigns a **satisfactory rating**, the system proceeds to construct a "Generic Answer" based on this second retrieval set. Conversely, if the Judge Agent's rating remains **unsatisfactory** even after searching the broader index, the system concludes that it cannot provide a satisfactory answer. In this failure scenario, the system defaults to generating a **generic error message**, informing the user that no suitable dataset has been identified. This two-tiered index and judgment-driven retrieval process aims to balance search efficiency and response speed for common queries with the capability to perform more comprehensive data exploration when necessary, guided by intelligent quality assessment via the Judge Agent.

To implement efficient and scalable retrieval, we leverage the **FAISS (Facebook AI Similarity Search) index** (Douze, Guzhva, Deng, Johnson, & Szilvasy, 2024). FAISS is optimized for high-dimensional vector similarity search and provides the necessary infrastructure to rapidly compare query embeddings against the large collection of embeddings representing time series metadata. FAISS supports an efficient computation and retrieval based on **cosine similarity** scores, efficiently identifying the top-k most semantically similar time series metadata embeddings to the input query embedding (Manning, Raghavan, & Schütze, 2008). One of the key differences from a standard Retrieval-Augmented Generation (RAG) approach lies in how we

handle information *after* the initial retrieval step. Most RAG systems incorporate the retrieved top-k text snippets as immediate input for the next stage. However, our priority is to ensure the accuracy of the information and the depth of contextual understanding used to build our responses. Therefore, for each of the identified relevant time series keys we perform an extended **metadata lookup**, as illustrated in Figure 7. In our prototype setup, this is achieved by directly querying a structured JSON file, which serves as our in-house, detailed repository of metadata. Ultimately, this design choice is intentionally aimed at minimizing the risk of introducing incomplete information or hallucinations.

Metadata Retrieval

Figure 7

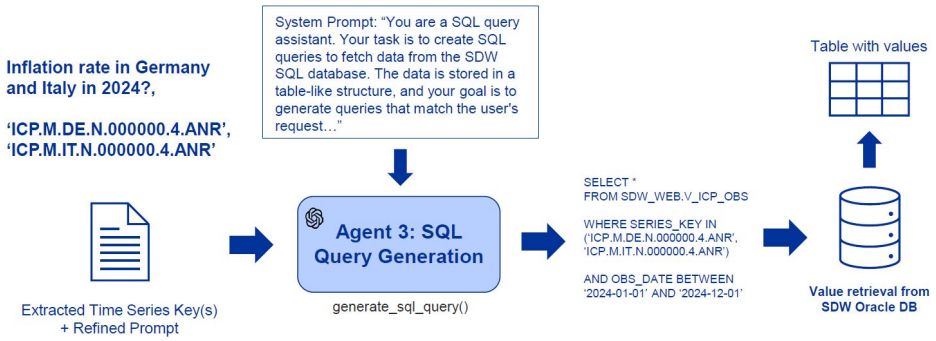


After we have retrieved the relevant series keys and metadata from the JSON, we still need to retrieve values to answer the user's query. The **SQL Query Generation Agent** (Agent 3), as shown in Figure 8, uses the extracted time series key(s) and the refined query prompt(s) as input to produce and execute an SQL query for fetching precise values from the European Central Bank's Secure Data Warehouse (SDW) DB – the backend data repository for the EDP.

To this end, we use a consistent SQL query template. The agent's task is to "fill in the blanks" in the template. The "blanks" correspond to the list of **time series keys** and consider any **temporal boundaries** specified by the user. For example, given the user query "*Inflation rate in Germany and Italy in 2024?*", the agent should extract "*2024*" as the relevant timeframe and refer to the time series keys *ICP.M.DE.N.000000.4.ANR* and *ICP.M.IT.N.000000.4.ANR* as identified in the retrieval phase. In general, the agent translates these elements into appropriate SQL WHERE clause conditions.

SQL Query Generation

Figure 8

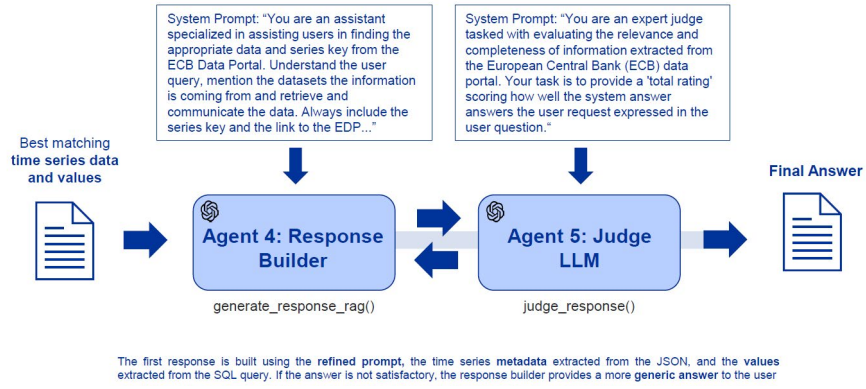


Response Generation

The response builder agent brings together the metadata for the timeseries and the retrieved values to construct the actual response for the user. This agent takes as input the **best matching time series data and values** extracted via the SQL Query Generation Agent, along with the associated time series **metadata**. We instructed the LLM to re-rank the sorted list of extracted time series keys according to the original user request, to add an additional layer of *reasoning* before providing the data to the Response Builder agent. The agent is then tasked to follow a specific template for generating the answer to ensure a harmonised and standardised response type.

Response Generation

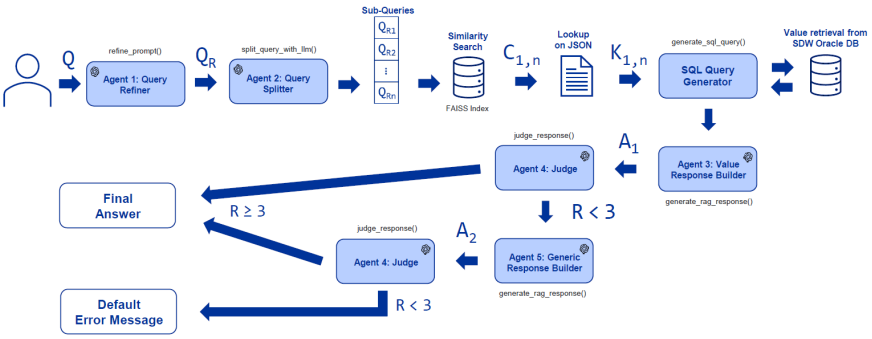
Figure 9



Finally, we incorporated a **Judge Agent** to assess the answer before it is delivered to the user. The purpose of this judge is to check the generated answer and assess how well it reflects the user's initial query and information need. The outcome of this judgement can take several forms. In the best case, the generated answer in considered a good or very good fit to the user's query and is accepted as is. If the answer is considered a mediocre or bad fit, the judge can relate to a modified response builder agent for generating a more generic answer that does not include data and time series values, but only refers to datasets which might be of relevance to the user. If the judge also deems this generic answer not suitable for replying to

the user, a generic error response is produced, informing the user, that the chatbot is incapable of finding a suitable dataset to the query. As indicated in Figure 9, this might involve a few interactions between the response builder and the Judge Agent. Figure 10 summarizes the response generation flow with the rating-based Judge-as-LLM system.

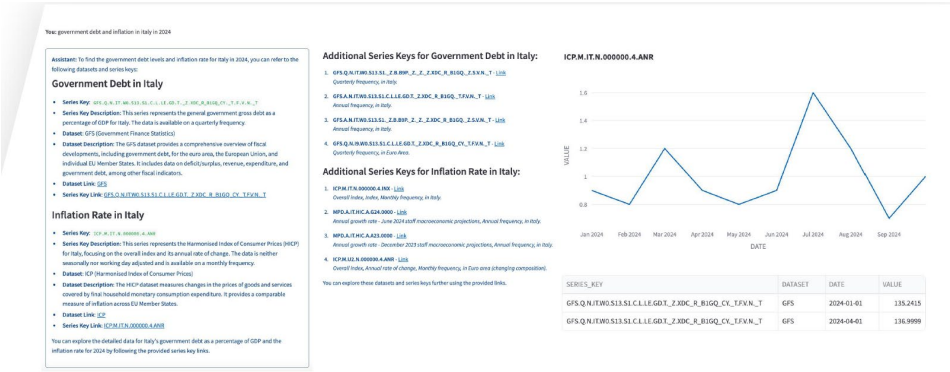
Response Generation Flow Figure 10



Building a User Interface

To provide users with a practical and accessible way to interact with our system, we developed a user-friendly interface using **Streamlit**, an open-source Python library (Streamlit, 2025). This enabled us to rapidly prototype an interactive chatbot interface without extensive front-end development. The resulting interface, as shown in Figure 11, is inspired by a chat-like environment where users can intuitively formulate their natural language queries and receive structured, informative responses.

Chatbot Interface Figure 11



The answers are provided following a consistent and structured format, designed to provide comprehensive information to the user within a single response. First, the response prominently includes the **extracted metadata** for the identified relevant time series. This typically encompasses concise **Dataset and Series Key Descriptions**, offering textual context for the data. Crucially, we also incorporate actionable **hyperlink** elements in the response, enabling users to navigate to the corresponding

pages on the ECB Data Portal for more in-depth exploration and verification. Beyond this core metadata, the interface also presents a list of the **top 5 'Additional Series Keys'** that the system deemed potentially relevant to the user's query. This provides users with alternative data series that might also be of interest and addresses potential ambiguity in the user query. Finally, the interface displays the actual data values in tabular or graphical format, depending on what seems more appropriate given the user query and retrieved data.

Evaluation

To assess the efficacy of our RAG based system in supporting users seeking data on the ECB Data Portal, we designed and conducted a comprehensive evaluation. Our evaluation framework focuses on realistic user needs and is based on quantitative metrics aligned with best practices in information retrieval. This section details the evaluation setup, including the dataset of queries and gold standard answers used, the rationale behind our evaluation methodology, and the specific metrics employed to quantify system performance.

Setup

A key aspect of our evaluation methodology is the creation of a large and diverse test dataset, consisting of **525 Query & Answer (Q&A) labelled pairs**. This dataset was specifically constructed to represent the complexity and diversity of real-world user information needs when interacting with the ECB Data Portal. We leveraged various sources like query logs and hotline requests to compile common user queries. Domain experts with the relevant business knowledge constructed the gold standard of what answer we would expect from an ideal system.

We categorized the 525 queries into three distinct types:

- **Type 1: Retrieving Values (55% of dataset):** These queries represent users explicitly seeking specific numerical values for particular time series. Examples include requests like "What was the inflation rate in Germany and Italy in January 2024?".
- **Type 2: Finding Data (30% of dataset):** These queries represent users aiming to discover relevant datasets and time series, without seeking specific values. Examples include "Where can I find data on the inflation rate in Germany and Italy?".
- **Type 3: Off-Topic (15% of dataset):** This category includes queries that fall outside the intended scope of the EDP chatbot – for instance, general conversational queries, requests unrelated to economic data, non-objective assessments, or questions about non-existent datasets. These 'Off-Topic' queries are crucial for assessing the robustness of the system and its ability to handle out-of-scope user inputs.

This structured distribution across query types is designed to evaluate the system's performance not only in retrieving specific data values but also in guiding users through the data portal, addressing various information seeking behaviours.

The relative proportions of each query type (55%, 30%, 15%) in our test dataset broadly mirror our understanding of typical user interactions and information needs within the EDP context.

Aligned with these distinct user query types, our system is designed to generate correspondingly different types of responses. We aim to deliver both **qualitative** and **quantitative** answers depending on the user's query. Therefore, for each query type, we expect the following system responses:

- **Response Type 1 – Value Response (for Query Type 1):** For users seeking specific data values (Type 1 queries), the expected response is a 'Value Response'. This *full* response type is designed to provide both *qualitative* context and *quantitative* data. It will provide the user with:
    - The *datasets and time series keys* relevant to their query, extracted from the Main Figures index, providing metadata and links.
    - The *extracted values* themselves, extracted via the generated SQL query, directly addressing the user's request for specific numerical data.
- **Response Type 2 – Data Response (for Query Type 2):** When users are exploring the data portal to *find* datasets and time series, or just to ask general questions on methodology and economic indicators (Type 2 queries), the expected system output is a 'Data Response'. This answer type focuses on offering a *qualitative* answer, guiding the user towards relevant data sources. It is designed to deliver:
    - *datasets and time series keys* identified as pertinent to the user's search for data and extracted from the Generic Index.
- **Response Type 3 – Default Message (for Query Type 3):** For queries that are off-topic or inappropriate for the EDP chatbot (Type 3 queries), we expect the system to provide a 'Default Message'. We inform the user that their query is outside the scope of the system and no satisfactory answer can be provided.

This clear mapping between query types and expected response types sets the stage for a precise evaluation of our system's effectiveness in addressing different user needs.

Risk by Query-Response Type

Table 1

	Response Type 1	**Response Type 2**	**Response Type 3**
Query Type 1	Good	Ok	Ok
Query Type 2	Ok	Good	Ok
Query Type 3	**Harmful**	**Harmful**	Good

Beyond evaluating the accuracy of data retrieval and the appropriateness of response types, our evaluation framework also considers the potential risks associated with different combinations of query types and system responses. As shown in Table 1, some mismatches between user queries and system outputs could be considered possibly "harmful", especially when the system incorrectly generates a response to an 'Off-Topic' query (Query Type 3). Specifically, we define a response as possibly "harmful" if either of the following scenarios occur:

1. **Expected Response Type 3 (Default Message) but System Produces Response Type 1 (Value Response).** This indicates a scenario where the system, despite being presented with an out-of-scope query, incorrectly attempts to retrieve and present specific data values.
2. **Expected Response Type 3 (Default Message) but System Produces Response Type 2 (Data Response).** Similarly, this denotes cases where an 'Off-Topic' query inappropriately triggers the system to provide a data-oriented response, even if it is only a qualitative Data Response (datasets and series keys), rather than the correct Default Message indicating inability to handle the query.

These scenarios are categorized as possibly "harmful" because they represent potential misinterpretations of user intent by the system. For our use-case, providing an answer to an off-topic query could mislead the user or suggest capabilities beyond the system's intended scope. For the purpose of our evaluation, the assessment of whether a response to a Type 3 query is indeed possibly "harmful" is conducted via manual human review, allowing for a better judgement of the specific context and potential implications of each response.

Results

Our evaluation framework is based on standard metrics used in information retrieval, focusing on measuring retrieval accuracy at different levels. We mainly looked at three levels of accuracies (a) identifying the relevant dataset, (b) identifying the relevant time series within a dataset and (c) identifying the relevant values within a time series:

- **Dataset Accuracy (Top-5):** This metric measures the percentage of queries for which the *correct dataset* (as defined in the gold standard answer for each query) is present within the top 5 datasets retrieved by the system. "Top-5" accuracy is a commonly used metric in ranked retrieval evaluation, reflecting scenarios where users are presented with a ranked list of results and are likely to examine the top few options (Baeza-Yates & Ribeiro-Neto, 1999). We computed this metric both overall across all query types, and separately for Type 1 and Type 2 queries to analyse performance for different information seeking intents.
- **Series Key Accuracy (Top-5):** Complementary to Dataset Accuracy, Series Key Accuracy focuses on the retrieval of the *correct time series key* – the most granular identifier for a specific data series within the EDP. Series Key Accuracy (Top-5) measures the percentage of queries for which the gold standard *series key* is present within the top 5 series keys retrieved by the system. Similar to Dataset Accuracy, the use of a "Top-5" metric is motivated by user interface considerations and the desire to evaluate performance in a

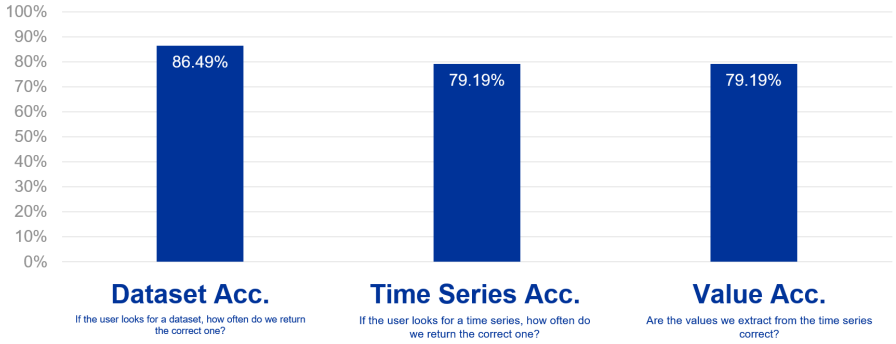
ranked retrieval scenario. This metric is again calculated both overall and per query type (Type 1 and Type 2) to provide a detailed performance breakdown.

- **Value Accuracy:** For queries that refer to a specific time series and pose time constraints (e.g. inflation in 2024), we also assess if the SQL query generation agent was capable to identify the *correct time constraints* from the user query. In this case a result is considered correct if the time constraints precisely match the gold standard provided by human experts and, thus, the also the correct values are retrieved from the SDW backend.

To evaluate the user-perceived benefits of our retrieval system, we analysed the performance metrics specifically related to retrieval accuracy for Query Type 1 and Query Type 2 queries – the query types for which the system is designed to provide responses (Value Responses and Data Responses, respectively). The results for our metrics, evaluated on our dataset of 525 Q&A pairs are summarized in Figure 12.

Top-5 Retrieval Results

Figure 12



Our evaluation reveals strong performance in both Dataset Accuracy and Time Series Key Accuracy. As shown in Figure 12, the **Dataset Accuracy** (Top-5) achieves a score of 86.49%. This indicates that for **86.49%** of all evaluated user queries, the system successfully places the correct dataset within the top 5 datasets retrieved in the ranked list. This high Dataset Accuracy suggests that, in a majority of user interactions, the system effectively directs users to the appropriate broad data categories on the EDP, facilitating data discovery and exploration.

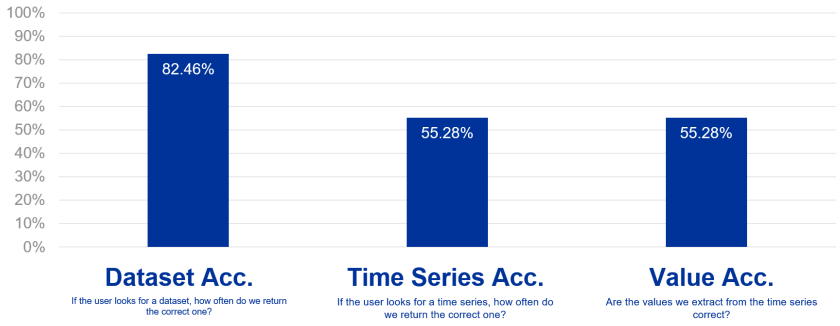
Complementing Dataset Accuracy, **Time Series Key Accuracy** (Top-5) stands at **79.19%.** This metric, representing the ability to retrieve the most specific data identifiers, signifies that for 79.19% of user queries, the specified gold standard time series key is present within the top 5 time series keys returned by the system. This indicates that while the system is highly effective at pinpointing the relevant datasets, narrowing the result down to the single most relevant time series key can present a slightly greater challenge. This is not unexpected, given the subtle semantics that differ closely related time series within our datasets. However, a Top-5 Time Series Key Accuracy of nearly 80% still represents a substantial benefit to users, significantly narrowing their search space within the vast EDP repository.

A particularly noteworthy finding of our evaluation pertains to **Value Accuracy**. As reported in Figure 12, the Value Accuracy score is also **79.19%** – numerically

identical to the Time Series Key Accuracy. This means, that once the time series key is correctly identified, the agent is always capable of extracting the relevant time periods from the user's query. As direct consequence of the 'RAG + Text to SQL' architectural design this directly leads to the system to retrieve and display the correct values for the users. This feature is a significant user benefit, enhancing trust in the reliability and accuracy of the data presented by the system, a crucial factor for the risk assessment of applications involving sensitive economic and financial information like ours.

Top-1 Retrieval Results

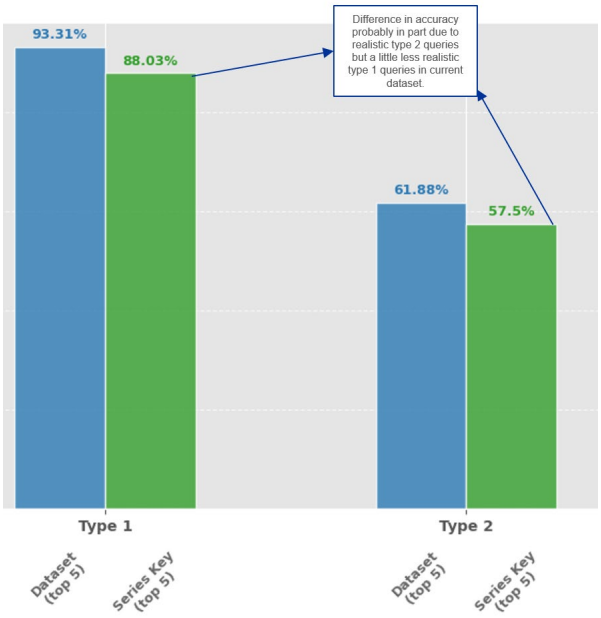
Figure 13



As an additional step, after obtaining these results with a Top 5 retrieval, we wanted to check the granularity of our system's retrieval performance and evaluate a more stringent definition of accuracy. Therefore, we also conducted an additional analysis focusing on **Top 1 retrieval accuracy**. In this case, we consider a retrieval *correct* if and only if the *highest ranked* dataset or time series key returned by the system is the gold standard correct answer. The results for this Top 1 accuracy analysis, also focused on Type 1 and Type 2 queries, are presented in Figure 13. As expected, applying this more stringent criterion results in a decrease in overall accuracy scores compared to the Top 5 metrics. Specifically, **Dataset Accuracy (Top 1)** is measured at **82.46%**, while **Time Series Key Accuracy (Top 1)** and consequently **Value Accuracy (Top 1)** both register at **55.28%**. While these Top 1 accuracy figures are lower than their Top 5 counterparts, they still represent a noteworthy level of performance, especially considering the highly specific nature of the 'correct answer' requirement in this stricter evaluation. A Dataset Accuracy of over 82% at Top 1 rank indicates that, even when demanding pinpoint accuracy in the very first returned result, the system effectively identifies the intended dataset category in the majority of cases.

Model Performance by Query Type

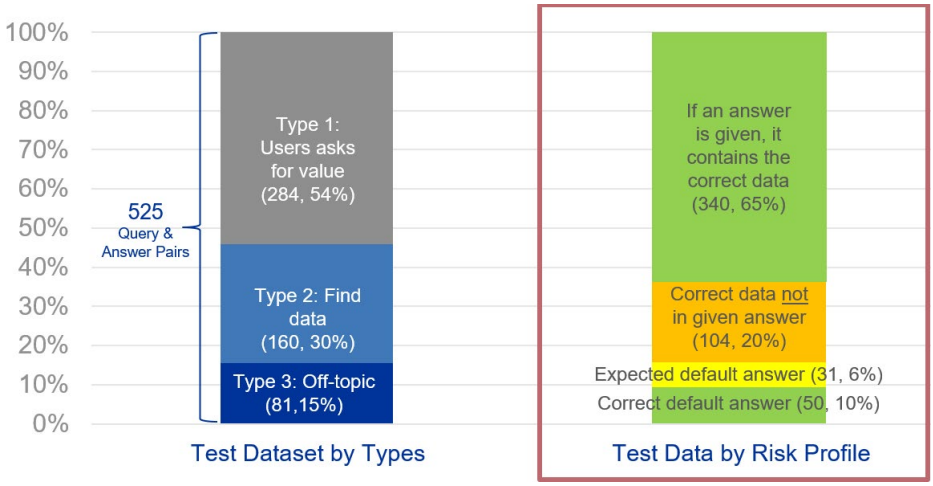
Figure 14



Furthermore, our evaluation reveals a significant distinction in system performance when analysing the results separately for **Type 1 (Retrieving Values) and Type 2 (Finding Data) queries**, as shown in Figure 14. For **Type 1 queries**, we achieve a **Dataset Accuracy (Top 5)** of **93.31%** and a **Time Series Key Accuracy (Top 5)** of **88.03%**. These markedly high accuracy scores indicate that, for queries where users are directly seeking specific data values, our system is highly capable in guiding users to both the correct dataset category and, even more precisely, the intended time series key within the top 5 results. In contrast, for Type 2 queries, the accuracy metrics are lower, yet still respectable. We observe a **Dataset Accuracy (Top 5)** of **61.88%** and a **Time Series Key Accuracy (Top 5)** of **57.50%** for Type 2 queries. This discernible difference in accuracy between Type 1 and Type 2 query types is not unexpected and is likely attributable to the inherent variation in complexity and search scope between these categories. Despite this performance variation between query types, the results robustly demonstrate that our system consistently achieves a meaningful level of accuracy across *both* direct value retrieval (Type 1) and broader data discovery (Type 2) user interactions, adapting effectively to different modes of user engagement with the data portal.

Answer Incorrectness Risk

Figure 15



Beyond assessing user benefits through accuracy metrics, our evaluation framework includes an assessment of various risk categories, particularly focusing on **Answer Incorrectness Risk** and **Hallucination Risk**. The results of this risk-focused analysis, derived from our dataset of 525 Q&A pairs, are presented in , specifically in the "Test Data by Risk Profile" breakdown. Answer Incorrectness Risk, in our context, refers to all those cases where the system's response fails to address the user's information need, or when the system inappropriately attempts to provide an answer for queries it should have declined. We defined Answer Incorrectness based on two primary criteria:

- *Does the returned answer not include the correct Time Series (TS) or Dataset (DS)?* This assesses situations where the system, despite providing a response, fails to identify the expected time series key or dataset relevant to the user's query.
- *Does the bot point towards EDP data for questions it should answer with the default message?* This criterion examines cases where the system incorrectly generates a response instead of defaulting to a generic message when presented with a query that is out of scope or unanswerable using EDP data.

The results of this specific analysis are the following:

- **If an answer is given, it contains the correct data (65%, 340 out of 525 queries):** This majority segment represents queries where, when the system *does* provide a data-driven answer, it successfully includes the correct and expected Time Series or Dataset within its response. This signifies a substantial portion of successful, relevant, and data-accurate answers generated by the system.
- **Correct data not in given answer (20%, 104 out of 525 queries):** In 20% of cases, while the system provides an answer, it *fails to include the specifically correct or expected data series*. This category represents instances of 'incorrect' answers in the sense of failing to perfectly match the gold standard, even though the system is still operating within the EDP data domain.

- **Expected default answer, system gives default answer (10%, 50 out of 525 queries):** For 10% of the test set, corresponding to the 'Off-Topic' (Type 3) queries, the system correctly identifies these queries as outside its scope and appropriately provides the intended *default message*. This indicates successful handling of out-of-scope queries, aligning with expected system behavior.
- **Expected default answer, system does *not* give default answer (6%, 31 out of 525 queries):** A smaller portion, 6% of queries, represents cases where the system *incorrectly* attempts to answer 'Off-Topic' (Type 3) queries with data-driven responses instead of the intended default message. This category highlights the risk of the system misinterpreting or inappropriately responding to out-of-scope user inputs.

Beyond Answer Incorrectness related to data selection, we also investigated **Hallucination Risk**. In our evaluation framework, this analysis focused on assessing whether the system, in its responses, might generate information that is factually incorrect, misrepresents data, or drifts outside the intended scope of the ECB Data Portal's data content. Concretely, our evaluation considered several potential types of hallucinations:

a) *Did the bot mention a topic not relevant to the task?* (Task Irrelevance)
b) *Did the bot make wrong claims (e.g., wrong claims about the ECB's strategy)?* (False Claims)
c) *Does the bot point towards data outside the EDP portal?* (Out-of-Scope Data)
d) *Does the bot invent data and give wrong numbers?* (Invented Data)
e) *Does the bot invent TS titles or data descriptions?* (Invented Metadata)
f) *Does the bot answer questions it should not answer (currently)? E.g., questions about data collection.* (Overreach)

Our analysis for Hallucination Risk was primarily conducted through manual human evaluation of a subset of system responses, particularly focusing on cases flagged for potential answer incorrectness or 'Off-Topic' query handling. Crucially, while a detailed manual assessment for all categories (a-f) remains ongoing, our preliminary analysis has revealed that instances of hallucination of type (c) – pointing towards data outside the EDP portal – were detected in only very few cases. This preliminary finding suggests that our architecture may be effectively mitigating a significant aspect of potential hallucination risk related to data source fidelity. However, a comprehensive quantitative assessment across all hallucination types (a-f) requires further analysis.

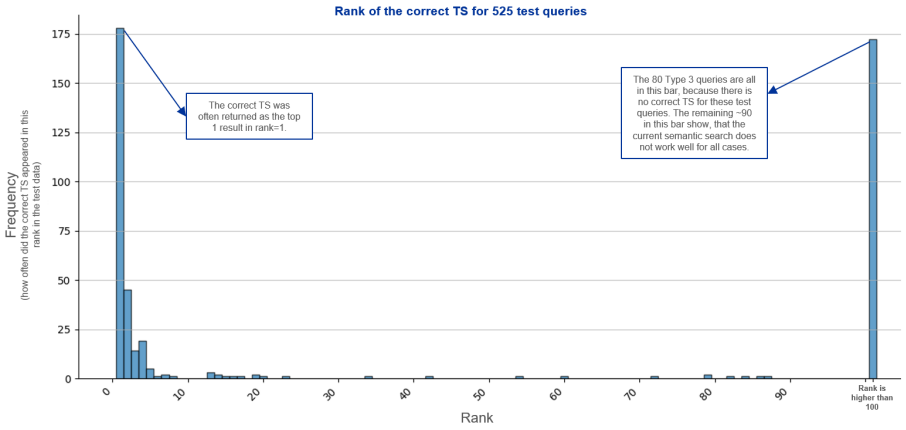
We also acknowledge the presence of **Jailbreak Risk**. This risk category refers to the potential for users to bypass the defined guardrails of the system through adversarial prompting. This could cause the Large Language Models within the system to disregard system prompts, circumvent intended instructions, or potentially produce unintended or inappropriate outputs. While Jailbreak Risk is a recognized concern for LLM-based systems, our evaluation has primarily focused on Answer Incorrectness and Hallucination risks related to core system functionality. A more detailed analysis of Jailbreak Risk, and the implementation of more robust safeguard mechanisms, is an area for future research and development.

Finally, we conducted an analysis of the ranking of the "correct" time series within the retrieved lists. This analysis aims to evaluate how effectively the system ranks the

most relevant time series at the top of the search results. Figure 16 shows a histogram illustrating the distribution of ranks for the gold standard time series across our dataset of 525 test queries. Ideally, for optimal search performance, the "correct" time series should consistently appear at Rank 1 – the very top of the results list.

Analysis of the Search Performance

Figure 16



As depicted in Figure 16, the histogram reveals a highly skewed distribution, strongly concentrated towards the lower ranks. The fullest bar in the histogram is at **Rank 1**, indicating that for a substantial number of queries the gold standard, most relevant time series is indeed retrieved as the top-ranked result by our similarity search process. Following Rank 1, the height of the bars rapidly diminishes as the rank number increases, showing that when the correct time series is not the top result, it is still typically found within the higher ranks (rank 2, 3, and to a lesser extent rank 4, 5 etc.). This distribution pattern is highly encouraging, as it confirms that our semantic search is generally effective at prioritizing and placing the most relevant time series towards the beginning of the ranked result list, aligning well with typical user browsing behaviours where users tend to focus on the top few search results.

However, the results show a noticeable bar on the far right, labelled as "Rank is higher than 100." As annotated on the slide, "The 80 Type 3 queries are all in this bar, because there is no correct TS for these test queries." Indeed, all 80 'Off-Topic' (Type 3) queries are categorized within this "Rank higher than 100" group, as, by definition, there is no 'correct' time series for these out-of-scope requests. However, the remaining count in this far-right bar represents instances, *even for in-scope queries (Type 1 & Type 2)*, where the semantic search failed to rank the correct time series within the top 100 results. This means that for a segment of queries, roughly ~90 out of the 525 in our dataset, our current semantic search approach does not effectively find the intended time series. This highlights an area for further investigation and potential improvement in our semantic search mechanism to enhance recall and ranking robustness across a broader range of user queries and data series within the EDP.

Summary and conclusions

Our experience developing a RAG-based system for the ECB Data Portal yielded several key insights, both regarding the strengths of our current approach and opportunities for further improvement.

A primary observation was the clear advantage of our multi-agent architectural design compared to simpler, single-agent systems. By decomposing the complex data retrieval task into specialized components—such as Query Refinement, Query Splitting, SQL Generation, Response Building, and Judgement—we significantly improved both system robustness and accuracy. This modular design consistently handled complex natural language queries and structured ECB datasets more effectively than the monolithic approaches tested during earlier phases of the project.

Furthermore, our findings strongly support the incorporation of Text-to-SQL mechanisms for extracting numerical data as an essential improvement. By leveraging SQL-based extraction rather than relying exclusively on semantic search for both qualitative and quantitative queries, we significantly reduced the risk of numerical inaccuracies and mitigated potential hallucinations associated with purely language-model-driven data extraction. The evaluation also highlighted our system's effectiveness in handling "off-topic" queries. By designing structured response types, including a clearly defined "Default Message" for queries outside the system's intended scope, we successfully avoided generating misleading or incorrect responses, thereby enhancing the reliability of the system.

Another significant learning was the importance of using realistic evaluation datasets. This approach provided deeper insights into practical system performance and highlighted limitations that might have remained unnoticed otherwise.

Initial experiments on the potential of using an LLM for re-ranking initial semantic search results looked promising. Results indicated that applying LLM-based re-ranking to the top 10 semantic search results significantly improved relevance. However, expanding re-ranking to larger sets (e.g., top 40 results) provided diminishing returns, offering limited improvements relative to computational costs. Thus, we will focus in future work on LLM-based re-ranking strategies on a limited subset of initial top-ranked results, achieving an effective balance between enhanced accuracy and computational efficiency.

Furthermore, semantic search accuracy remains key area for ongoing research and development. A detailed analysis of accuracy of the retrieval component revealed that the current semantic search mechanism struggles to consistently position the correct time series at the top ranks for certain queries. Improving recall and ranking accuracy within the semantic search algorithm will thus be a critical priority for future enhancements.

Finally, our investigations into metadata utilization revealed that simply increasing metadata volume does not necessarily enhance retrieval accuracy. As demonstrated in evaluations using an expanded metadata index, the addition of more metadata fields alone did not significantly improve search performance. This implies that future efforts should focus on more sophisticated metadata strategies, potentially involving LLM-driven metadata re-ranking, dynamic augmentation, or innovative embedding methods, rather than merely expanding metadata quantity.

These insights establish a robust foundation to guide future improvements of our RAG-based system, enhancing both its effectiveness and user-centred design for data discovery on the ECB Data Portal.

References

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval.* New York: ACM press.

Douze, M., Guzhva, A., Deng, C., Johnson, J., & Szilvasy, G. (2024). The Faiss library. *arXiv*(2401.08281).

Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2024). RAGAs: Automated evaluation of retrieval augmented generation. *18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations* (pp. 150-158). Association for Computational Linguistics.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., . . . Wang, H. (2024). *Retrieval-Augmented Generation for Large Language Models: A Survey.*

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., . . . Rocktäschel, T. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems, 33*, 9459--9474.

Liang, L. (2024). *Stacking Small Language Models for Generalizability.* arXiv preprint.

Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. *Text Summarization Branches Out* (pp. 74-81). Barcelona: Association for Computational Linguistics.

Ma, X., Gong, Y., He, P., Zhao, H., & Duan, N. (2023). *Query Rewriting for Retrieval-Augmented Large Language Models.* arXiv preprint.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. In C. D. Manning, *Introduction to Information Retrieval* (p. Chapter 6). Cambridge: Cambridge University Press.

OpenAI. (2024, January 25). *New embedding models and API updates.* Retrieved from OpenAI: https://openai.com/index/new-embedding-models-and-api-updates/

Shao, Z., Gong, Y., Shen, Y., Huang, M., Duan, N., & Chen, W. (2023). *Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy.* arXiv preprint.

Streamlit. (2025). *Streamlit Documentation*. Retrieved from Streamlit: https://docs.streamlit.io/

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., . . . Zhou, E. (2023). *The Rise and Potential of Large Language Model Based Agents: A Survey.* arXiv preprint.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Lin, Z., . . . Xing, E. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems, 36*, 46595-46623.

EUROPEAN CENTRAL BANK

EUROSYSTEM

Supporting users in seeking data on the ECB data portal

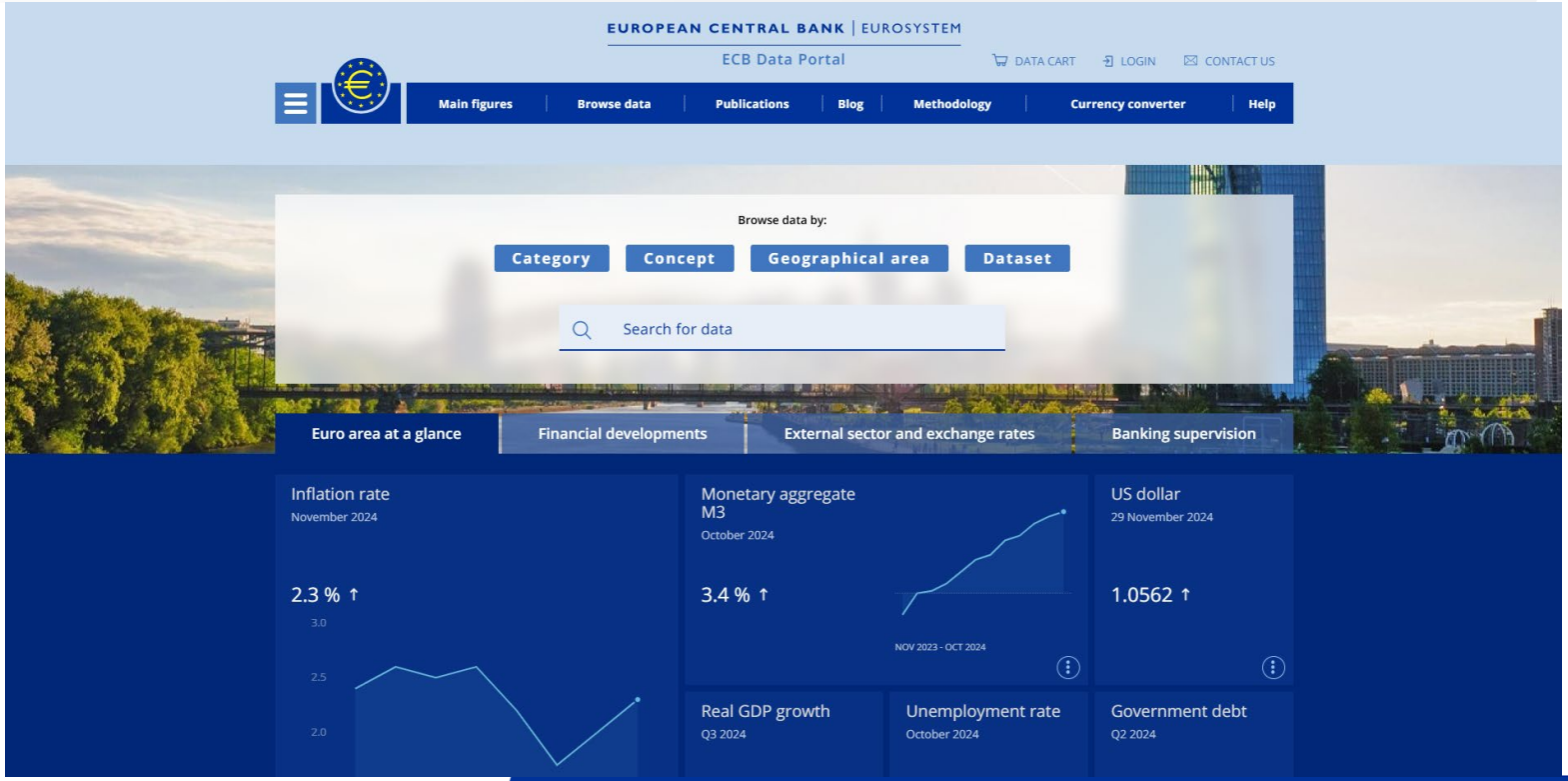
A use case for RAG

The views expressed are those of the authors and do not necessarily reflect those of the ECB.

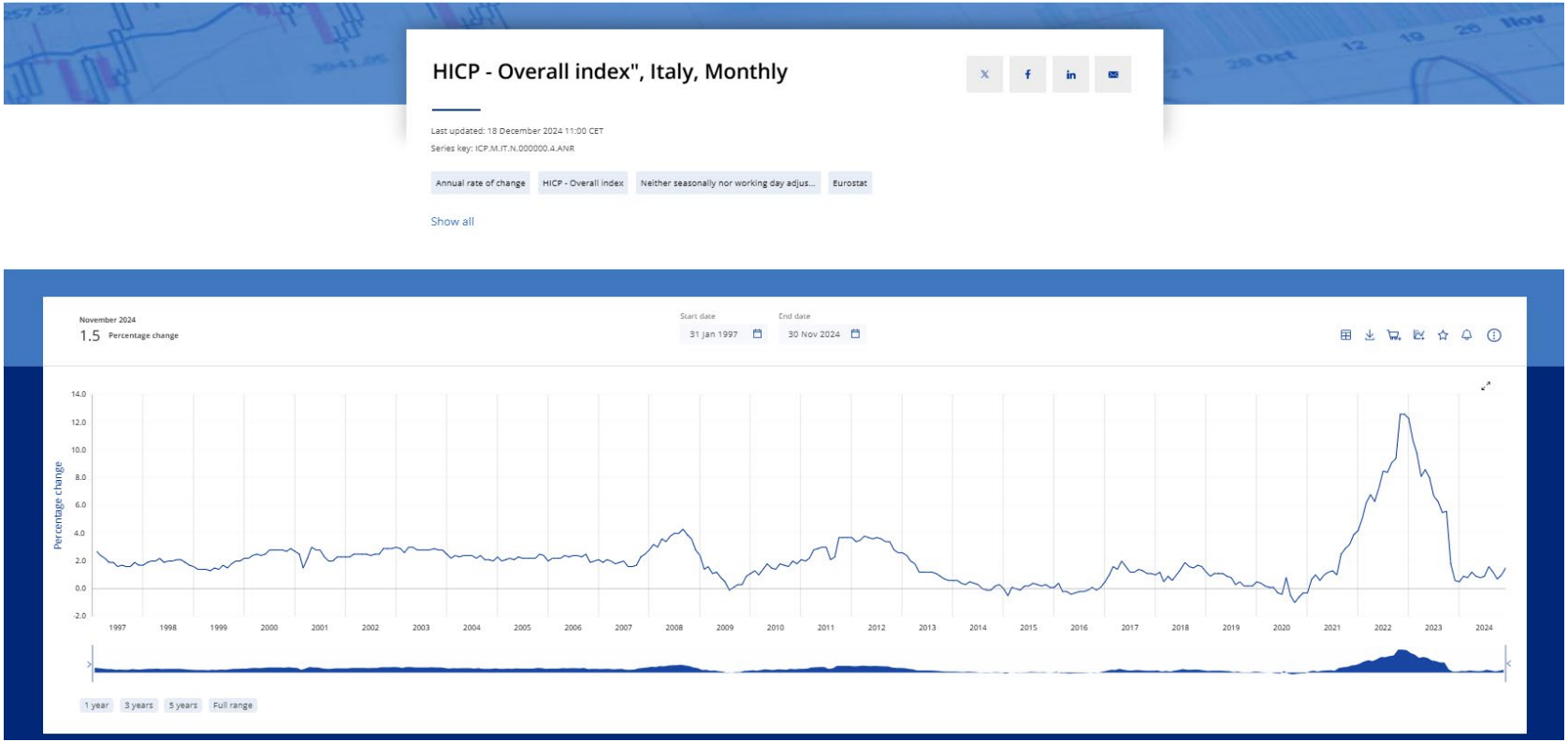
February 2025

Luca Petracca, Simone De Benedictis, Thomas Gottron, Zlatina Hofmeister

The ECB Data Portal



Example of Time Series data on the EDP



The problem

How can we help users finding information in the ECB Data Portal?

- Users report difficulties finding and identifying the right information

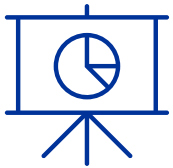
The ECB Data Portal uses “traditional” ways of accessing information

- Browsing by categories, concepts, geographical areas and datasets
- Search by keywords
- Main indicators
- Interactive Publications
- Dashboards

What about providing an interface where users can use their own “natural language”?

Our Idea

- **PoC:** testing **Retrieval Augmented Generation (RAG)** and **Text-to-SQL** techniques to explore how state-of-the-art **LLMs** can be leveraged to enable users to **find data** using **natural language** in a **chat** interface



Understand the user queries

Identify the right dataset and time series key

Retrieve and provide time series data

How does it work?

PoC – Setup

Scope:

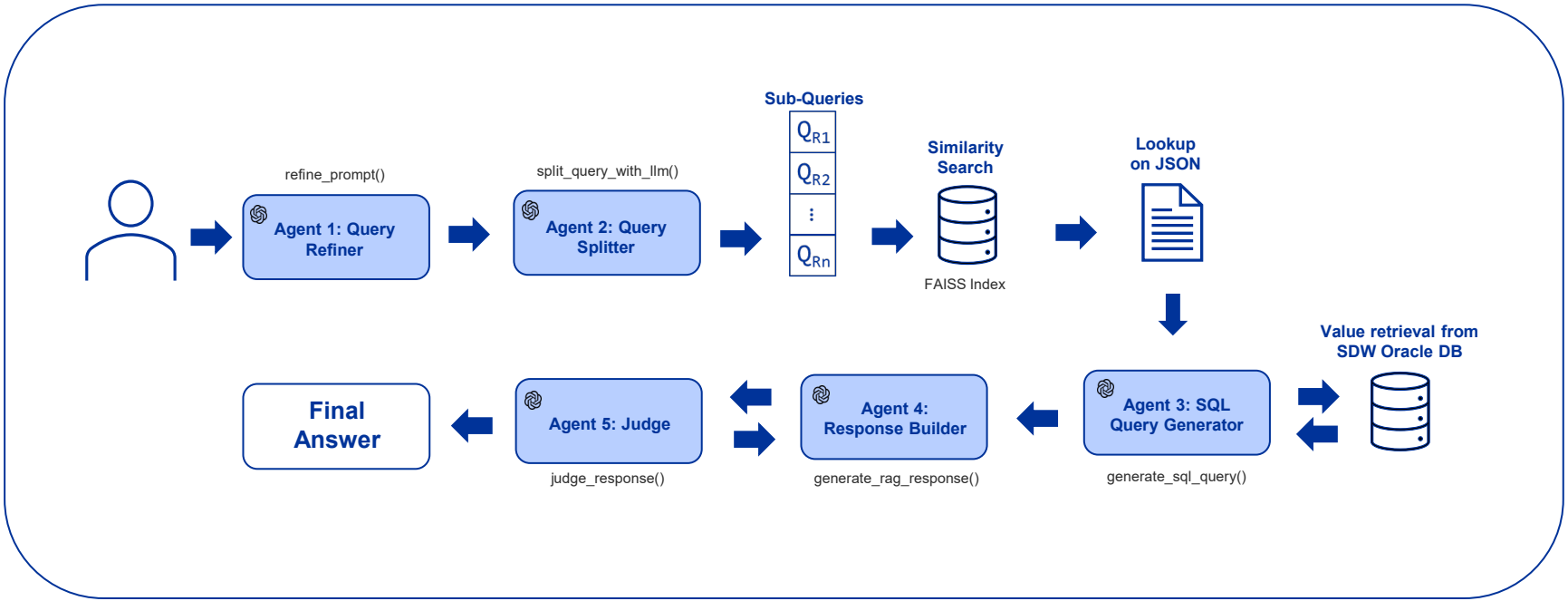
- **Time Series Included:**
  - ~300.000 TS are included (~10% of the EDP)
  - including all TS either in Main Figures* or included in official publications, reports and articles

Technologies & APIs:

- Connection to LLM via **OpenAI API**
- Technologies:
  - **FAISS** (For RAG semantic search)
  - **GPT-4o** (Chat completion model)
  - **Text-embedding-3-large** (Embedder for RAG semantic search)

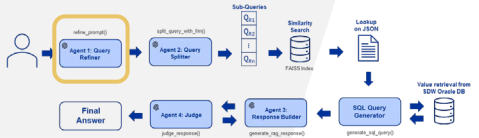
*Main Figures: https://data.ecb.europa.eu/main-figures

Current Architecture



Behind the Scenes

Step 1: Query Refinement



« What was the Inflation in Germany and Italy last year»

System Prompt: «You are an assistant that has to refine queries in order to perform vector search more effectively to retrieve data from the European Central Bank data portal. Your task is to identify and expand vague and ambiguous terms in the query…»

«Inflation rate in Germany and Italy in 2024»

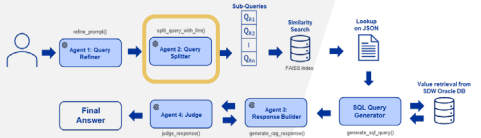
User Query

Agent 1: Query Refinement

refine_prompt()

Refined Query

Step 2: Query Splitting



«Inflation rate in Germany and Italy in 2024»

System Prompt: “You are an assistant specialized in breaking down complex queries into multiple simple queries. If a user query contains info about multiple countries, different compositions, datasets or time frames, you should split the queries and output a JSON object…”

Sub-queries: [‘Inflation rate in Germany’, ‘Inflation rate in Italy’]

Refined Query

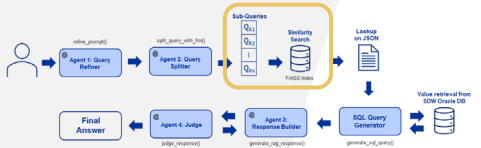
Agent 2: Query Splitting

split_query_with_llm()

Q_{R1}
Q_{R2}
⋮
Q_{Rn}

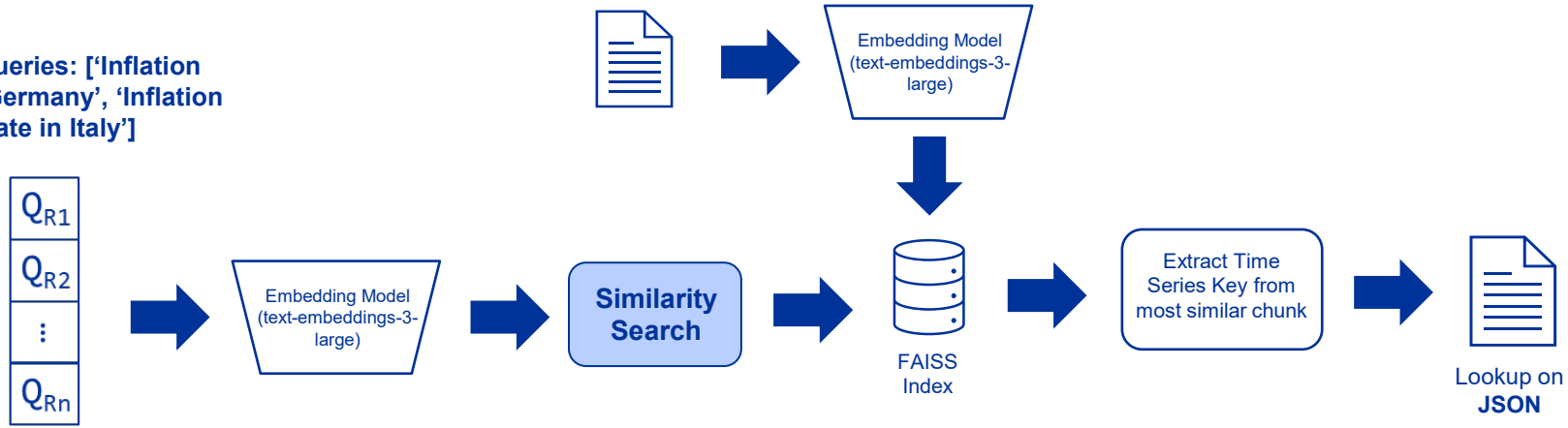
N Sub-Queries

Step 3: Similarity Search

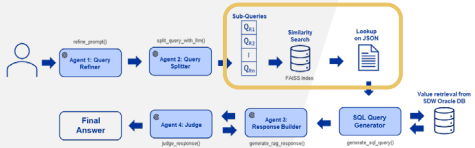


RAG_EDP.txt: Text database containing dataset, time series key and time series complete title, description and metadata for each time series in the selected subset

Sub-queries: ['Inflation rate in Germany', 'Inflation rate in Italy']



Step 4: Information Retrieval



Knowledge Base

FAISS Index

We are looking for most similar chunk in the Index based on the content of the user query. Each line contains the dataset, key and description of a time series

Metadata JSON

Given the extracted time series key, we look up on the JSON to extract the full metadata. **But we still need the values…**

Series Key: ICP.M.IT.N.000000.4.ANR, Dataset: ICP, Value for Italy - HICP - Overall index, Annual rate of change, Eurostat, Neither seasonally nor working day adjusted with Monthly frequency, in Italy

▼ ICP.M.IT.N.000000.4.ANR {6}

DATASET : ICP

DATASET DESCRIPTION : Scope: Data presentation - Summary description. The Harmonised Index of Consumer Prices (HICP) for the euro area is published by the European Commission (Eurostat) and generally available from 1996 onwards. Euro area results are obtained by aggregating indices for individual countries. The HICP is broken down following the European classification of individual consumption according to purpose (ECOICOP) and by goods and services special aggregates derived from it. The HICP covers monetary expenditure on final consumption by resident and non-resident households on the economic territory of the euro area. The seasonally adjusted HICP data are compiled by the ECB. Data presentation - Detailed description. HICPs measure changes in prices of goods and services covered by final household monetary consumption expenditure, including all indirect taxes paid by consumers. HICPs have a common coverage of goods and services across countries but country specific item lists and item weights. HICPs do not cover expenditure for owner occupied housing. For items fully or partly paid or refunded by the government, HICPs include only the share that is paid by the consumer (e.g. the \"out-of-the-pocket\" expenditure for health services). The HICPs are classified according to the ECOICOP. Additional compiled aggregates are also published. Methodological information. Time period: Monthly, Quarterly, Annual. Base period: 2015=100. Statistical concepts and definitions: For information about the naming convention (series key dimensions and metadata), refer to the ICP underlying DSD (ECB_ICP1) maintained by the ECB. Statistical processing. Data compilation: Following the Maastricht Treaty the aim of the HICP is to measure inflation by means of the consumer price index on a comparable basis, taking into account differences in national definitions. Eurostat dedicated HICP website (which include short guide for users). Adjustment: X-12 ARIMA. Administrative Information. Title: ICP - Indices of Consumer prices. Data source: European Commission (Eurostat) and European Central Bank calculations based on Eurostat data. Quality. Timeliness: 17 working days after the end of the reference month. Legal and institutional environment. Legal acts and other agreements: Council Regulation (EC) No 2494/95 on HICPs and implementation regulations.

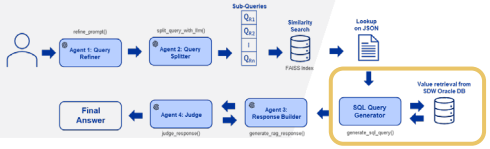
DATASET LINK : https://data.ecb.europa.eu/data/datasets/ICP

SERIES KEY LINK : https://data.ecb.europa.eu/data/datasets/ICP/ICP.M.IT.N.000000.4.ANR

FREQUENCY : Monthly

SERIES KEY DESCRIPTION : Value for Italy - HICP - Overall index, Annual rate of change, Eurostat, Neither seasonally nor working day adjusted with Monthly frequency, in Italy

Step 5: SQL Query Generation



Inflation rate in Germany and Italy in 2024?,

‘ICP.M.DE.N.000000.4.ANR’, ‘ICP.M.IT.N.000000.4.ANR’

System Prompt: “You are a SQL query assistant. Your task is to create SQL queries to fetch data from the SDW SQL database. The data is stored in a table-like structure, and your goal is to generate queries that match the user's request…”

Extracted Time Series Key(s) + Refined Prompt

Agent 3: SQL Query Generation

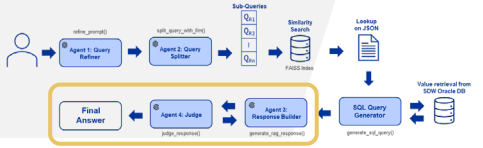
generate_sql_query()

```
SELECT *
FROM SDW_WEB.V_ICP_OBS

WHERE SERIES_KEY IN
(‘ICP.M.DE.N.000000.4.ANR’,
‘ICP.M.IT.N.000000.4.ANR’)

AND OBS_DATE BETWEEN
‘2024-01-01’ AND ‘2024-12-01’
```

Value retrieval from SDW Oracle DB

Table with values

Step 6: Response Generation



System Prompt: “You are an assistant specialized in assisting users in finding the appropriate data and series key from the ECB Data Portal. Understand the user query, mention the datasets the information is coming from and retrieve and communicate the data. Always include the series key and the link to the EDP...”

System Prompt: “You are an expert judge tasked with evaluating the relevance and completeness of information extracted from the European Central Bank (ECB) data portal. Your task is to provide a 'total rating' scoring how well the system answer answers the user request expressed in the user question.“

Best matching **time series data and values**

Agent 4: Response Builder

generate_response_rag()

Agent 5: Judge LLM

judge_response()

Final Answer

The first response is built using the **refined prompt,** the time series **metadata** extracted from the JSON, and the **values** extracted from the SQL query. If the answer is not satisfactory, the response builder provides a more **generic answer** to the user

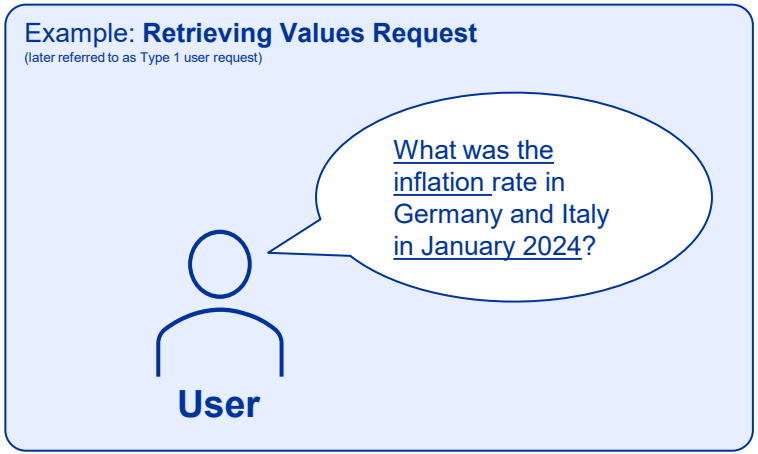
Evaluation

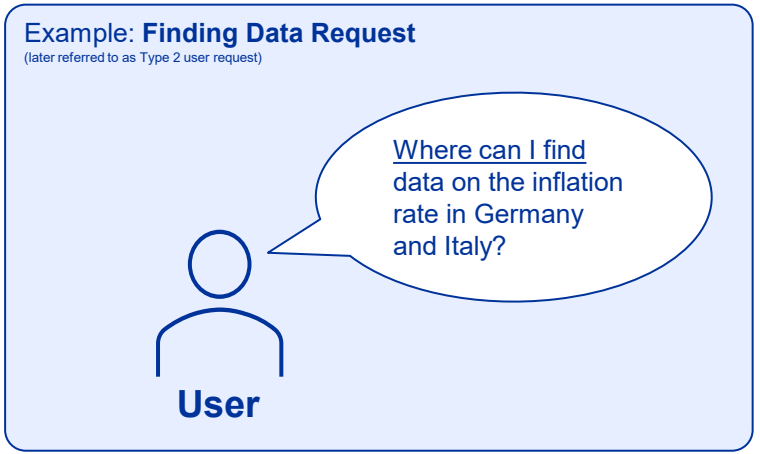
Our Approach

We leverage a mixture of state-of-the-art AI methods based on LLMs (**RAG and TEXT2SQL).**

Our chat assistant supports users with 2 types of requests in natural language:

- ✓ **Type 1 - Retrieving Values:** We retrieve the specific values a user in a time series a user is looking for.
- ✓ **Type 2 - Finding Data:** We identify the correct datasets and time series users are looking for.





Results – Value Accuracy

Test Set: 525 Q&A labelled pairs

Results (overall):

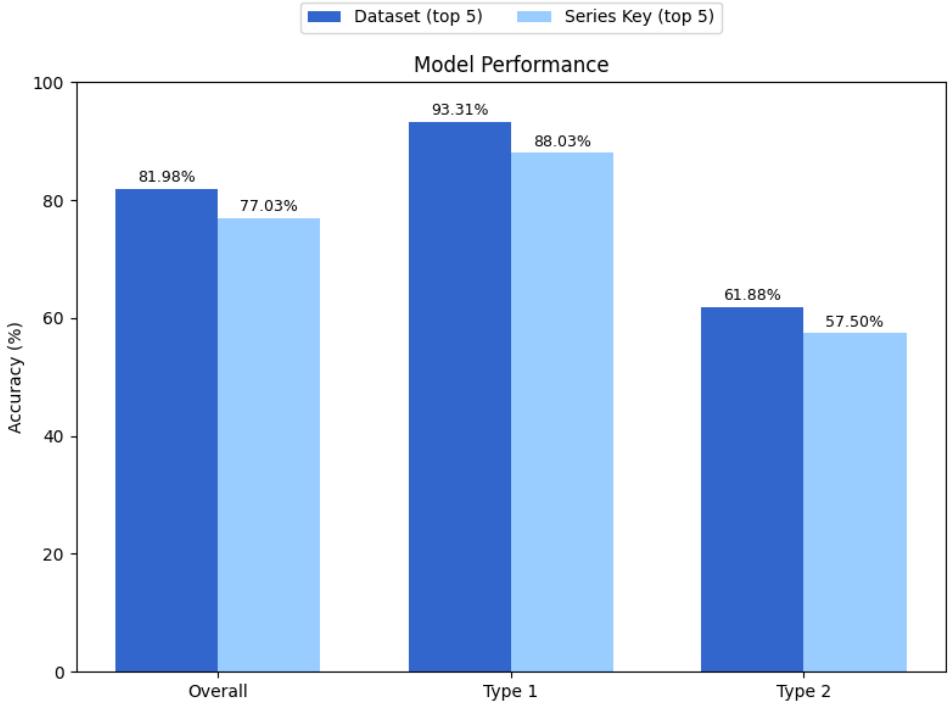
- Dataset Accuracy - **81.98%**
- Series Key Accuracy - **77.03%**

Type 1:

- Dataset Accuracy – **93.31%**
- Series Key Accuracy - **88.03%**

Type 2:

- Dataset Accuracy – **61.88%**
- Series Key Accuracy – **57.50%**



Key Insights and Way Forward

Key Insights and Way Forward

In a nutshell…

- The PoC shows promising results with regards to our goal of **helping users find data on the EDP**
- The only way to increase accuracy and reliability while decreasing costs is… **to avoid LLMs**
  - **Vector search** and fixed **SQL generation** instead of RAG for search and value extraction was crucial for reducing hallucination
  - Judge LLM as the “core” AI-based agent
- **Top-K multiple series key** retrieval drastically improved model accuracy

Next Steps:

- **Improve Accuracy**
  - Exploring new retrieval techniques for the similarity search **(clustering embeddings, hierarchical search…)**
  - Test a **conversational approach** building the SQL query field-by-field with the user
  - Increase the number of questions-answers to better measure accuracy
- **Finding the right balance between cost-benefit-risk**
- **Deploy the first version internally for a selected group of users with the objective of better understanding how users interact with the chatbot**

Thank you!