Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Exploring Agentic RAG for Metadata Question Answering in Enterprise Data Models

Ramin Allakhyarov
Adem Alparslan
adem.alparslan@fom.de

FOM University of Applied Sciences

Research Article

Keywords:

Posted Date: May 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9679011/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Abstract

This paper explores the design and cost-performance trade-offs of agentic retrieval-augmented generation (RAG) architectures for metadata question answering (QA) over structured enterprise data models. While large language models (LLMs) have demonstrated strong capabilities in interpreting structured metadata, their practical deployment is constrained by context window limitations, token costs, and limited transparency in reasoning processes. To address these challenges, we investigate an agentic RAG framework that decomposes metadata QA into multiple coordinated steps, combining category-specific retrieval with a multi-agent architecture operating on declarative data model definitions. Rather than aiming to outperform existing approaches, the proposed system is designed to investigate how modular decomposition and selective retrieval influence token consumption and answer quality. We evaluate the approach against a standalone LLM baseline using a controlled question catalog with varying levels of reasoning complexity. The results indicate that the agentic system reduces input token consumption by approximately 33% on average, while maintaining comparable answer completeness. At the same time, completion token usage increases due to coordination overhead, highlighting a structural trade-off between input efficiency and generation cost. Further analysis indicates that token consumption scales primarily with the number of reasoning steps and the volume of retrieved context, suggesting that the performance of agentic RAG depends strongly on query complexity. These findings provide empirical insights into the design space of agentic LLM systems and contribute to a better understanding of when modular, RAG-based architectures may be beneficial for enterprise metadata analysis.

1 Introduction

In today's increasingly competitive business landscape, data has emerged as a fundamental strategic resource that enables organizations to generate valuable insights and support informed decision-making processes (Arslan et al., 2024, p. 4723; Mohammed et al., 2023, p. 1628). This dependency on data becomes even more pronounced as organizations progressively adopt artificial intelligence (AI) as a driver of competitive advantages and innovation (van der Meulen, 2020; Yang et al., 2025, p. 2). Accordingly, effective management of data as an asset is essential for organizations to position themselves as truly data- and AI-driven enterprises (Mohammed et al., 2023, p. 1628; Yang et al., 2025, p. 2). In this context, metadata management arises as a key discipline that allows organizations to attain full transparency on the structure, dependencies, and characteristics of their data landscape (DAMA, 2017, pp. 417–418; Hechler et al., 2023, pp. 293–294). This transparency ensures that enterprise data is not only available, but can also be properly discovered and utilized to drive value creation throughout the organization (Hechler et al., 2023, p. 293; Yang et al., 2025, p. 1). However, despite its recognized importance, maintaining adequate metadata documentation remains a labor-intensive task in light of the ever-growing volume and complexity of data (M. Singh et al., 2025, p. 1; Yang et al., 2025, p. 18).

Consequently, an increasing number of research efforts have focused on metadata management in recent years, reflecting its growing importance in organizational practice (Eichler et al., 2021, p. 1). In this

regard, the relationship between AI and metadata management is bidirectional, as metadata not only enables AI-driven data products, but can also be enhanced through AI (Yang et al., 2025, p. 11). Fueled by their recent surge in popularity and proficiency across various domains (Ozdemir, 2023, p. 3; Pan et al., 2024, p. 64), large language models (LLMs) have emerged as a key enabler of AI-driven metadata management, holding the potential to bring transformative changes to the field (Mohammed et al., 2023, pp. 1629–1631; Yang et al., 2025, p. 8). The advanced capabilities of contemporary LLMs in analyzing metadata from various sources and formats enable these models for a wide range of potential applications, including the intelligent exploration and classification of existing as well as the creation of new and enriched metadata (Yang et al., 2025, pp. 11, 16–17). Yet, despite this symbiotic relationship between AI and metadata, researchers note that it has primarily been explored on a conceptual level, with empirical studies and real-world implementations remaining scarce (Eichler et al., 2021, p. 10) and mostly limited to vendor-based solutions (Yang et al., 2025, pp. 5–6).

This paper addresses this research gap by exploring the potential of commercially available LLMs to improve metadata understanding through the development of a question answering (QA) system. Designed to streamline metadata analysis, the proposed framework operates directly on the declarative source code of a sample data model, thereby avoiding reliance on a specific platform and extensive prior curation steps. While the general proficiency of LLMs in understanding structured metadata is widely acknowledged (Mohammed et al., 2023, p. 1629), critical aspects such as the scalability and token efficiency of such approaches remain underexplored. Consequently, this paper adopts an exploratory approach by leveraging the increased capabilities (Han et al., 2024, p. 1) and task flexibility (Talebirad & Nadiri, 2023, p. 2) of collaboratively combining multiple LLMs, along with the scalability and factual grounding provided by retrieval-augmented generation (RAG) (Jeong, 2024, p. 2; Peng et al., 2024, p. 2).

This paper makes the following contributions:

First, it conceptualizes and instantiates an agentic RAG architecture for metadata QA that operates directly on declarative data model definitions. Rather than proposing a performance-optimized solution, the architecture serves as an experimental artifact to explore how modular decomposition and category-specific retrieval influence system behavior in structured metadata contexts.

Second, the paper provides an empirical exploration of cost-performance trade-offs in agentic LLM systems. In particular, it analyzes how input and completion token consumption evolve under multi-step reasoning and selective retrieval, thereby exposing structural shifts in cost distribution introduced by agent-based coordination.

Third, the study investigates the relationship between query complexity and system behavior. By modeling token usage as a function of reasoning depth and retrieved context, it derives insights into how different dimensions of complexity affect both efficiency and response generation dynamics.

Fourth, the paper contributes a structured experimental setup for metadata QA, including a categorized chunking strategy and a controlled question catalog. This setup is intended to support reproducible

exploration of agentic RAG designs rather than to serve as a definitive benchmark.

2 Theoretical Foundation

2.1 Large Language Models

LLMs are a class of pre-trained neural language models designed to process and generate natural language. They build on advances in neural machine translation, particularly the introduction of attention mechanisms and the Transformer architecture in the mid-2010s, which enabled scalable sequence modeling. (Alammar & Grootendorst, 2024, pp. 1–6, 13–15)

Modern LLMs can be broadly categorized into encoder-based models, which focus on language understanding, and decoder-based models, which are optimized for text generation (Alammar & Grootendorst, 2024, pp. 20–25; Ozdemir, 2023, pp. 11–12). In recent years, decoder-only architectures such as the *Generative Pre-trained Transformer* (GPT) have become dominant due to their strong generative capabilities and ease of interaction (Alammar & Grootendorst, 2024, pp. 28–29; Foster, 2022, p. 396). At a technical level, LLMs generate text by sequentially predicting the probability distribution over the next token based on the preceding context (Alammar & Grootendorst, 2024, pp. 54, 97–103; Ozdemir, 2023, pp. 7–10; Sanseviero et al., 2024, pp. 23, 28). This probabilistic formulation enables coherent text generation but also introduces variability depending on decoding strategies (Alammar & Grootendorst, 2024, pp. 221–222).

Despite their strong performance, LLMs exhibit several fundamental limitations. First, their context window restricts the amount of information that can be processed at once, which becomes problematic for large inputs such as enterprise data models (Alammar & Grootendorst, 2024, pp. 26, 106; Mohammed et al., 2023, p. 1629; Ozdemir, 2023, p. 5; Sanseviero et al., 2024, pp. 89, 737–738). Second, their knowledge is primarily derived from pre-training data and is therefore static in nature, limiting their ability to access up-to-date or domain-specific information (Alammar & Grootendorst, 2024, p. 294; Peng et al., 2024, p. 2; Sanseviero et al., 2024, p. 489). Third, these constraints can lead to hallucinations, where models generate plausible but factually incorrect outputs (Alammar & Grootendorst, 2024, pp. 294, 327; Du et al., 2023, p. 1; Foster, 2022, p. 399; Njeh et al., 2025, p. 202).

These limitations motivate the integration of external knowledge sources, as realized in RAG approaches.

2.2 Retrieval-Augmented Generation

RAG has emerged as a key approach to mitigate fundamental limitations of LLMs, including their reliance on static parametric knowledge, sensitivity to training data quality, and limited context windows. Popularized by Lewis et al. (2020), RAG integrates external retrieval mechanisms into the generation process, enabling models to incorporate domain-specific information without requiring additional

training. (Alammar & Grootendorst, 2024, p. 327; Arslan et al., 2024, p. 4724; Gao et al., 2024, p. 1; Jeong, 2024, p. 2; Lewis et al., 2020, p. 2; Masoudifard et al., 2024, p. 5; Njeh et al., 2025, p. 205; Peng et al., 2024, pp. 2, 4; Sanseviero et al., 2024, p. 489)

The corresponding workflow of a typical RAG pipeline consists of three main stages (see Figure 1). First, external documents are segmented into smaller units and encoded into vector representations, which are stored in a vector index. Second, upon receiving a user query, the system retrieves the most relevant segments by comparing vector representations of the query and the indexed content. Third, these retrieved segments are provided to the LLM as additional context for response generation, ultimately improving factual grounding and extending the effective context available to the model. (Alammar & Grootendorst, 2024, pp. 327–328; Arslan et al., 2024, p. 4724; Jeong, 2024, pp. 2–3; Kim et al., 2024, p. 3; Lewis et al., 2020, p. 2; Njeh et al., 2025, pp. 205–206; Peng et al., 2024, pp. 2, 4; Sanmartin, 2024, p. 2; Sanseviero et al., 2024, pp. 489–490; A. Singh et al., 2025, pp. 2–4)

More recent approaches adopt modular architectures that decompose retrieval and reasoning into multiple coordinated steps. In this context, research explores the integration of RAG with multi-agent systems, where specialized components iteratively retrieve, evaluate, and synthesize information. Such architectures aim to improve both scalability and interpretability, particularly in settings that require structured reasoning over complex data representations. (A. Singh et al., 2025, pp. 5–8)

2.3 Metadata

Metadata plays a central role in effective data management by providing structured descriptions of data assets and their characteristics (DAMA, 2017, p. 420; Yang et al., 2025, p. 1). It enables transparency into data structures, quality, and usage, thereby supporting data governance, regulatory compliance, and consistent data understanding across an organization (DAMA, 2017, p. 417; Mohammed et al., 2023, p. 1628). As a result, an organization’s ability to operate in a data-driven manner depends largely on how effectively its metadata is managed (DAMA, 2017, pp. 417–418; Eichler et al., 2021, p. 1; Mohammed et al., 2023, pp. 1628–1629; Yang et al., 2025, p. 25).

Metadata is commonly classified into three categories: technical, operational, and business metadata (DAMA, 2017, pp. 422, 425–426; Hechler et al., 2023, p. 295). Technical metadata describes structural aspects of data assets, including schemas, data types, and relationships, as well as data lineage, which captures the origin and transformation of data (DAMA, 2017, p. 441; Eryurek et al., 2021, pp. 46–47, 180–181; Hechler et al., 2023, pp. 296, 306; Mohammed et al., 2023, pp. 1628–1629; Yang et al., 2025, p. 5). Operational metadata refers to system-generated information such as logs and runtime parameters, which require further processing to become analytically useful (DAMA, 2017, p. 425; Hechler et al., 2023, p. 296; Mohammed et al., 2023, p. 1628). Business metadata, by contrast, provides semantic and organizational context, including definitions, business rules, and key performance indicators (Hechler et al., 2023, pp. 295–296; Mohammed et al., 2023, p. 1628). However, despite its importance, many organizations struggle to establish effective metadata management

practices (Bandara et al., 2024, p. 269). This often results in fragmented knowledge, limiting shared understanding, consistent governance, and clearly defined data responsibilities (Gröger, 2021, pp. 103–104).

In light of the critical implications of rigid metadata and the growing complexity of data, metadata management is an area that has experienced significant evolution in recent years (Yang et al., 2025, p. 5). For most organizations, structured metadata management initiatives are commonly led by the introduction of a centralized data catalog which documents and organizes the various types of metadata, enabling its accessibility and searchability across the organization (Eichler et al., 2021, pp. 3–4; Gröger, 2021, p. 105; Yang et al., 2025, pp. 19–20). Despite advancements in organizing, standardizing, and partially automating metadata (e.g., ingestion), traditional data catalogs remain reliant on significant manual effort and domain expertise while lacking the flexibility and contextual understanding to manage complex data environments (Hechler et al., 2023, p. 298; M. Singh et al., 2025, pp. 1, 6; Yang et al., 2025, p. 8).

With the growing adoption of AI, recent approaches increasingly use it to improve metadata management and address the limitations of traditional solutions (Yang et al., 2025, pp. 8, 25). A precursor to more advanced automated approaches is presented by Berti et al. (2021), who employ process mining techniques to analyze operational metadata of *SAP* enterprise resource planning (ERP) systems, enabling insights into the underlying data structures of key enterprise processes (pp. 1–2). Seeking to overcome the historical reliance on commercial vendors and domain experts to provide such analytics, their work utilizes a graph-based modeling of the corresponding database tables and their interconnections, allowing for a more holistic, modular, and efficient exploration of the associated metadata (Berti et al., 2021, pp. 1–2, 4–5, 11).

A language model-based approach to metadata management is provided by Suhara et al. (2022) through their framework *DoDuo*, which leverages *BERT*-based models to analyze table contents and metadata, effectively predicting the semantic column types as well as inter-column relationships (p. 1494). However, to accomplish this task, also known as column type annotation, *DoDuo* relies on resource-intensive fine-tuning of its underlying LLMs (Suhara et al., 2022, p. 1496). Korini & Bizer (2023) address this limitation by employing the pre-trained *ChatGPT* for column type annotation, which yields improved results over preceding approaches while simultaneously being much less reliant on task-specific training and fine-tuning (pp. 8–9). Similarly exploring the interplay of LLMs and tabular data, Lobo et al. (2023) employ an LLM-enabled matching of table columns to business glossary entries, thereby stressing the integration of technical and business metadata to simplify data discovery (p. 2). Notably, the authors' paper demonstrates that, even when metadata is only scarcely maintained, LLMs manage to accurately analyze it and predict its implications for business domains (Lobo et al., 2023, p. 2). This principle is further expanded in the work of M. Singh et al. (2025) and scaled to an enterprise-wide data catalog (p. 2). The authors develop a RAG-based pipeline that enhances the metadata of catalog entries by incorporating contextually relevant information drawn from other entries (M. Singh et al., 2025, p. 2). In this regard, it is particularly emphasized that the contextual understanding abilities of LLMs allow their

framework to draw such supporting context even when table and column names are only present in abbreviated form (M. Singh et al., 2025, p. 2). Ultimately, the LLM-generated metadata descriptions only need to be reviewed by qualified data stewards, significantly reducing the effort compared to manual creation (M. Singh et al., 2025, p. 2).

Hayashi et al. (2024), in similar fashion, utilize RAG in their paper on metadata-based data exploration (p. 6575). Through the encoding of dataset metadata into vector embeddings, the authors present a framework that, given a user-provided dataset, can recommend both similar and complementary datasets that align with user-specific analytical needs (Hayashi et al., 2024, pp. 6576, 6582). Extending beyond tabular formats, the true flexibility of LLMs in the management of metadata is addressed in the work of Hoseini et al. (2024, pp. 1–2). Recognizing that data within organizations is scattered across multiple departments where it is present in manifold formats alongside tables, including XML and JSON, it is demonstrated that a comprehensive and overarching understanding of the associated metadata becomes critical for organizations to ensure data usability (Hoseini et al., 2024, p. 2). The authors explore the semantic data management capabilities of LLMs, highlighting their advanced proficiency in extracting and integrating metadata from heterogeneous sources and formats into an enterprise-wide ontology-aligned schema (Hoseini et al., 2024, pp. 3–4, 12).

3 Research Design

Building on the theoretical foundations and identified research gap, this study adopts an empirical design to evaluate the proposed agentic RAG architecture for metadata QA. To guide the empirical investigation, the following research questions are addressed:

RQ1: How does an agentic RAG architecture affect answer quality in metadata QA compared to a standalone LLM?

RQ2: What impact does the proposed multi-agent system have on token efficiency relative to conventional direct LLM inference?

RQ3: How does query complexity influence the performance and efficiency of the agentic system?

To structure the empirical exploration, the following expectations are examined:

E1: The agentic RAG architecture is expected to reduce input token consumption relative to a full-context baseline.

E2: The multi-agent architecture is expected to increase completion token usage due to coordination overhead.

E3: The answer quality of the agentic system is expected to be comparable to that of a standalone LLM.

E4: The efficiency gains of the agentic system are expected to decrease with query complexity.

In line with the exploratory character of this study, these statements are framed as expectations rather than as formal hypotheses, since the study design does not aim at confirmatory hypothesis testing but at structured empirical exploration.

Against this background, although LLMs show considerable potential for metadata management (Hayashi et al., 2024, p. 6575; Mohammed et al., 2023, p. 1629), their limited scalability and operational transparency highlight the need for architectures that enable practical and efficient deployment (Yang et al., 2025, pp. 18–19). In response, this paper examines a modular QA system that leverages LLMs and RAG to support metadata understanding in an efficient and adaptable manner.

The implementation targets a publicly available sample data model provided by *SAP* in their blog series introducing the *Datasphere Analytic Model* (jaigupta, 2023). While prior work has explored graph-based representations of metadata (Berti et al., 2021, p. 5; Schmid et al., 2019, pp. 263–264), their integration into RAG pipelines (Gao et al., 2024, p. 5; Masoudifard et al., 2024, pp. 2–3; Peng et al., 2024, p. 2), and their interpretability by LLMs (Guo et al., 2023, pp. 1–2; Pan et al., 2024, pp. 64–65), the present approach deliberately adopts a simplified ingestion of metadata in its native declarative format. For *SAP*-based data models, such definitions are commonly expressed using *Core Schema Notation* (CSN), a JSON-like format that encodes entities, attributes, and associations in a structured and compact representation (*Core Schema Notation (CSN)*, 2024). Leveraging the capability of LLMs to interpret such code-like structures (Foster, 2022, p. 400; Ozdemir, 2023, p. 3; Pan et al., 2024, p. 64), the proposed architecture enables a lean, domain-agnostic approach that can be transferred across heterogeneous metadata environments.

To address the inherent complexity of analyzing structured metadata, the system adopts a multi-agent design. Prior research has shown that LLM performance on complex reasoning tasks improves when intermediate reasoning steps are explicitly modeled, as in Chain-of-Thought (CoT) prompting (Wei et al., 2022, p. 9). Extending beyond such prompt-based approaches, multi-agent systems distribute reasoning across specialized agents that collaboratively solve tasks through coordinated interaction (Talebirad & Nadiri, 2023, pp. 1–2). Following this paradigm, the proposed QA system employs a set of agents with distinct roles, including planning, retrieval, validation, and summarization. Worker agents specialize in retrieving metadata from different categories, while supervisory agents coordinate reasoning, refine intermediate outputs, and synthesize final responses. This modular decomposition enables scalable and flexible metadata analysis.

Given that CSN-structured metadata can quickly exceed practical context limits, the use of RAG in the system serves to selectively retrieve relevant metadata segments during inference. This design avoids the need for computationally expensive model fine-tuning (Lobo et al., 2023, p. 5) and instead focuses on achieving two primary objectives:

Improved token efficiency: By retrieving only relevant metadata chunks instead of processing the entire data model, the system reduces input token consumption and associated costs. This is particularly relevant for API-based LLMs with token-based pricing models.

Improved answer quality: By limiting the amount of processed context to relevant information, the system mitigates known limitations of LLMs when handling long inputs, such as attention bias toward the beginning and end of sequences (Liu et al., 2023, p. 1). This supports more accurate and contextually grounded responses.

To evaluate these objectives, the proposed multi-agent system is benchmarked against a standalone LLM baseline with full access to the CSN-structured data model. The evaluation is conducted using a manually curated question catalog that captures varying levels of query complexity. Performance is assessed in terms of answer quality and token efficiency, enabling a systematic comparison between monolithic and agentic LLM architectures.

4 Empirical Study Design

4.1 Sample Data Model

4.1.1 Characteristics

As discussed in section 3, the data model employed by the QA system in this paper is adopted from *SAP*'s blog series introducing the *Datasphere Analytic Model* – a cloud-based warehousing solution that aims to streamline data integration, modeling, and analytics within enterprises (jan_fetzer, 2023; Mueller, 2023). The specific CSN file employed in the presented approach is obtained from the corresponding *GitHub* repository which is referenced in said blog series (Eichmann, 2024). Architecturally, the data model is centered on a view which contains enterprise sales opportunity data and sources information from a total of seven dimension tables, three text tables, and one central fact table (jaigupta, 2023). In addition, a time-related view, comprising one temporal dimension table and three temporal text tables, allows the analytic model to incorporate time-based drill-down functionality (jaigupta, 2023). As outlined in section 2.3, the intermediary connections between entities, referred to as the data lineage, constitute another substantial part of the data model. They are illustrated in Figure 2.

4.1.2 Chunking

As discussed in section 2.2, a fundamental prerequisite for any document that is utilized in a RAG system is its chunking, followed by the embedding of the resulting chunks in a vector store for later retrieval. While the embedding of an entire document as a single chunk is technically possible, the resulting need to process its full content is likely to summon many of the drawbacks associated with long context processing by LLMs, as elaborated in section 3, thereby defeating the purpose of RAG (Alammar & Grootendorst, 2024, p. 309; Ozdemir, 2023, pp. 68–69). As an alternative, the segmentation of documents by imposing a fixed token length for each chunk enables lower token consumption during the generation stage of the RAG pipeline (Ozdemir, 2023, pp. 60, 70). While the implementation simplicity of this method is comparable to that of the single-chunk embedding, such arbitrary segmentation tends to disrupt semantic continuity and dilute relevant information through added noise, leading to an inherently restricted semantic consistency of the resulting chunks (Ozdemir,

2023, pp. 60, 70). To ensure high retrieval accuracy, however, RAG systems rely on chunks that are semantically coherent and contextually meaningful (Alammar & Grootendorst, 2024, pp. 310–311). Contemporary approaches to achieve just that have focused on the incorporation of semantic embeddings and machine learning techniques to cluster segments into semantically aligned chunks (Ozdemir, 2023, pp. 66, 70; Qu et al., 2024, p. 2). Despite the marginal performance increase observed in such novel approaches to chunking, the complexity of their development and their computational overhead imply that their application is not necessarily the most practical option (Ozdemir, 2023, pp. 66, 70; Qu et al., 2024, pp. 2, 4).

Seeking to establish a balance between chunk-level semantic coherence and implementation efficiency, the present approach employs a set of *Python*-based parsing functions to analyze the hierarchical key-value structures of the data model. The resulting chunks are grouped into the following metadata categories:

- Tables,
- Views,
- Columns,
- Associations.

This categorical distinction, in combination with the semantic self-containment of each CSN-structured chunk (Figure 3), ensures that relevant context can be accurately retrieved and interpreted during inference of the RAG-based QA system. Particularly in cases involving polysemous terms (e.g., when a table and a column share the same description), such discernment between metadata categories is critical. The resulting chunk counts yield 15 tables, 2 views, 114 columns, and 24 associations across the respective metadata categories.

4.1.3 Anonymization

To accurately measure the performance of the presented agentic system with regard to its ability to analyze CSN-structured data models, it is imperative to reduce the risk of data leakage from the employed LLM's parametric knowledge. This is particularly relevant given that the present approach employs a publicly available dataset, as elaborated in section 3. The fact that the publication date of the data model precedes the knowledge cutoff of *ChatGPT-4.1*, which is the LLM employed in this paper, further amplifies this concern (*GPT-4.1 - OpenAI API*, n.d.; jaigupta, 2023). As a counteractive measure, the descriptions of the tables, views, columns, and associations entailed in the data model undergo an anonymization procedure wherein they are masked with abstract identifiers. In addition to minimizing the potential influence of pre-trained knowledge on the QA system's generated responses, this step reduces the likelihood that the LLM relies on suggestive naming patterns simply based on the names of tables, views, columns, or associations. For instance, the technical names *V_MCT_OpportunityItems_v2* and *MCT_OpportunityItems* alone hold a strong indication that the view is consuming its data from the fact table. Similarly, even without prior analysis of the column-level metadata, it can be presumed with high

confidence that columns such as *FirstName*, *LastName*, and *Gender* are likely assigned to the table *MCT_Employees*.

Throughout the CSN-structured data model, there are numerous nested levels where such descriptive information occurs. Consequently, deliberate parsing is employed to thoroughly identify existing instances of technical and user-facing labels so they can be anonymized using 16-character hashes. This anonymization procedure increases the token count of the data model from 11,840 to 15,299.

4.2 Multi-Agent Framework

Alongside the preprocessing procedures applied to the data model, the LLM-based multi-agent system constitutes another vital component of the proposed QA architecture. Recognizing the advantages offered by the collaboration of multiple agents, several frameworks have recently emerged to streamline the development of agentic architectures (Arslan et al., 2024, p. 4724). Following the selection by Arslan et al. (2024, p. 4724), this paper employs the *CrewAI* framework to facilitate the LLM-based collaboration that powers the QA-based understanding of the CSN-structured metadata. The operational principle of the framework is defined by the following core components:

- **Agents**, which are powered by an LLM of choice and can be assigned distinct roles;
- **Tasks**, which specify the agents' objectives and deliverable outputs;
- **Crews**, which function as a wrapper for the execution of one or more agents. (*CrewAI - Introduction*, n.d.)

The fundamental principle of *CrewAI* is built on the assignment of a clearly defined role to each agent within the system, thereby guiding the course of their decision-making and the resulting outputs (Arslan et al., 2024, p. 4725; *CrewAI - Introduction*, n.d.; A. Singh et al., 2025, p. 9). Furthermore, individual agents can be equipped with specialized tools, enabling them to expand their contributions to the multi-agent system by, for instance, conducting web searches, connecting to databases, or performing RAG using documents in various formats (*CrewAI - Tools Overview*, n.d.). As the QA system presented in this paper focuses on analyzing CSN-structured metadata, the retrieval of the previously chunked data model segments during user inference emerges as the main required capability. Consequently, the chosen approach synthesizes the RAG-based expansion of parametric LLM knowledge along with the collaborative and modular problem-solving capabilities of multi-agent systems to harness the principle of what is formally referred to as agentic RAG (A. Singh et al., 2025, pp. 2–3, 7–9).

Due to the inherently structured nature of analyzing a CSN-structured data model, the QA system utilizes *CrewAI*'s flow functionality to manage agent interactions and ensure precision and token efficiency during the metadata analysis. Doing so, it employs a total of eight agents, each covering a distinct aspect necessary for generating answers to the user queries. They are specified as follows:

Planning Agent: The Planning Agent is the first component of the multi-agent system which is exposed to incoming user queries. Drawing on the principle of planning, as outlined by A. Singh et al. (2025, p. 10), this agent decomposes a given user query into agent-specific instructions and develops a multi-step execution plan according to which the remaining agents in the system must act to generate an accurate response in a token-efficient manner.

Search Mode Determination Agent: Supporting the token-efficient retrieval of the metadata chunks while supplying the multi-agent system with sufficient context to provide accurate responses constitutes the primary objective of the Search Mode Determination Agent. To do so, during each iteration of the execution plan provided by the Planning Agent, this agent reasons over the optimal number of CSN chunks to retrieve and forwards the resulting value as a parameter to the subsequently conducted RAG search.

Table, View, Column, and Association Agents: Respectively specializing in each of the four metadata categories discussed in section 4.1.2, these four worker agents use a custom-developed RAG search tool to retrieve the most relevant CSN-structured metadata chunks in accordance with the execution plan crafted by the Planning Agent and the scope determined by the Search Mode Determination Agent. Ultimately, the retrieved information fuels the QA system's ability to generate factually grounded responses.

Quality Control Agent: The Quality Control Agent integrates conceptual elements that stem from both planning and reflection (A. Singh et al., 2025, pp. 9–10). By evaluating the outputs of its preceding agents, this agent concludes each iteration of the agentic system by analyzing the respectively retrieved CSN-structured metadata chunks. Based on the obtained insights, it formulates natural-language interpretations of the retrieved content and additionally refines the Planning Agent's initial execution plan to enhance the quality of downstream iteration outputs.

Summarizing Agent: As the terminal component in the multi-agent workflow, the Summarizing Agent is tasked with synthesizing the intermediate interpretations of the Quality Control Agent into a coherent and accurate answer to the initially submitted user query. While this agent does not source additional information or utilize tools to complete its objective, its generated output constitutes the final answer of the QA system presented to the user. This underscores the critical need for the agent's contextual awareness as its performance ultimately represents the effectiveness of the agentic system as a whole.

Given the elaborated structure of agent interactions, as illustrated in Figure 4, the overall workflow strategy of the presented multi-agent system partly reflects the "Orchestrator-Workers" pattern, with its main characteristic being reflected in the task decomposition and delegation performed by the Planning Agent (A. Singh et al., 2025, p. 13). Simultaneously, by iteratively refining outputs based on the Quality Control Agent's intermediately provided insights, the QA system holds structural resemblance to the "Evaluator-Optimizer" pattern (A. Singh et al., 2025, p. 14) and can thus be characterized as a hybrid that combines the strengths of both patterns.

5 Prototype Implementation

5.1 Retrieval-Augmented Generation Pipeline Setup

To set up the RAG pipeline and ensure accurate chunk retrieval for the multi-agent QA system, *OpenAI*'s "most capable embedding model" *text-embedding-3-large* (*Text-Embedding-3-Large - OpenAI API*, n.d.) is employed to generate vector embeddings for each of the metadata chunks. Simultaneously, using *CrewAI*'s custom tool functionality (*CrewAI - Create Custom Tools*, n.d.), a RAG search tool is set up to provide the Table, View, Column, and Association Agents with access to the metadata chunks of their respective categories.

5.2 Multi-Agent System

5.2.1 Large Language Model Configuration

As outlined in section 4.2, the agents that make up a multi-agent system are powered by LLMs, giving them the ability to process language-based inputs and generate contextually relevant outputs. The presented QA system, in this regard, distinguishes between two classes of employed LLMs:

API-based LLMs: Models within this category possess flagship-level capabilities and are therefore used for tasks which require complex reasoning and logical problem-solving proficiency. As a result, their operation necessitates computational resources far beyond the means of ordinary consumer-grade hardware. Their integration within the multi-agent system is therefore facilitated through a paid API access, with costs calculated per one million input or completion tokens, respectively. Consequently, token efficiency, for these models, represents a core aspect which is deliberately managed in the present context.

Local LLMs: In contrast to flagship LLMs, the capabilities within this class are much less sophisticated, making them primarily suitable for simpler tasks. Simultaneously, their basic makeup renders them far less demanding for computational resources, allowing them to run on most consumer-grade devices. Due to the models' open-source nature and their on-premises deployment, no direct financial costs are associated with their operation.

To address the more complex tasks in the agentic system, *OpenAI*'s *ChatGPT-4.1* is employed. Given that the QA system's performance will be evaluated on the basis of multiple curated queries, consistent performance of the underlying LLMs upon repeated exposure is favorable. Therefore, following the approaches by Korini & Bizer (2023, p. 2) and Gamage et al. (2024, p. 5), the proposed agentic system draws on the deterministic nature of its main task and employs a lower temperature setting of 0.1. The remaining configurable parameters of the model are kept at their default values. On the other hand, as local LLMs in the present context are used exclusively to invoke the RAG tool described in section 5.1, the selected model for this class is the compact 3B version of *Meta*'s *Llama 3.2*. Its modest hardware

requirements, combined with built-in support for tool calling (*Llama 3.2*, 2024), make it particularly well suited for this purpose.

5.2.2 Agentic Flow Configuration

5.2.2.1 Planning Agent

To match the complexity of developing an execution plan with sufficient reasoning capability, the Planning Agent is configured to be API-based and is used only in the initial iteration. As illustrated in Algorithm 1, the agent's task commences with an assessment of the user query to determine the metadata categories and, thereby, the worker agents that are necessary to address it in adequate fashion. Subsequently, following the principle of multi-hop RAG (Alammar & Grootendorst, 2024, p. 334), the agent is prompted to serialize a set of instructions into an iterative execution plan which governs the sequence of the worker agents' ensuing actions, as illustrated in Figure 5. To maintain a manageable level of complexity throughout the course of the multi-agent system, the task's rules prevent the Planning Agent from employing more complex agentic workflow patterns such as parallelization (A. Singh et al., 2025, p. 13), thereby limiting the number of permissible instructions per agent to one.

Algorithm 1: Planning Agent Task Prompt (pseudocode)

```
[...]

1 YOUR TASK (MUST FOLLOW THE RULES STATED BELOW):
 2 1. Think deep and determine, which of the 4 worker Agents, based on their capabilities, are relevant to answer the user query and which are not relevant.
 3 2. Based on the respective capabilities of the worker Agents you have deemed relevant, think deep and determine the order in which the relevant worker Agents must act to precisely answer the user query.
 4 3. Formulate a detailed execution plan which gives an instruction to each relevant worker Agent and provides a sequence number for each instruction in ascending order. The instructions for not relevant Agents must remain empty.
 5
 6 RULES FOR YOUR TASK (YOUR ANSWER MUST STRICTLY ADHERE TO EACH RULE):
 7 - You can give up to 1 instruction to each of the 4 worker Agents.
 8 - You MUST give each iteration a sequence number as a single digit value. The sequence number of the first iteration MUST be 1. All subsequent iterations' sequence numbers ascend by 1.

[...]
```

Resulting from an iterative experimentation and refinement process, as common practice for well-designed prompts (Alammar & Grootendorst, 2024, p. 230), Algorithm 1 additionally reveals the prompt engineering techniques employed to optimize the Planning Agent's output. As an initial step, the CoT technique is utilized to decompose the agent's main task into three modular components. Nevertheless, a certain degree of complexity remains inherent in all three. Consequently, a technique known as zero-shot CoT (Alammar & Grootendorst, 2024, pp. 242–243) is additionally leveraged to force the Planning Agent into further reasoning over each of the subtasks by using key phrases such as 'Think deep' and 'Think in multiple iterations'. While alternatively including desired output examples in the prompt, known

as few-shot learning (Besta et al., 2024, p. 17682; Masoudifard et al., 2024, p. 12), has demonstrated favorable outcomes (Du et al., 2023, p. 1), its application during the experimental development of the present QA system led many of the agents to adhere to the provided examples too closely, limiting their ability to generalize.

5.2.2.2 Search Mode Determination Agent

The role assigned to the subsequent agent, the API-based Search Mode Determination Agent, is to iteratively analyze the Planning Agent's execution plan and determine the appropriate number of metadata chunks to be retrieved by the worker agent designated for execution in the current iteration. Traditionally, within conventional RAG implementations, the number of chunks to retrieve is manually defined, wherein it is critical that the chosen value is well-balanced to ensure the accuracy of downstream processes (Njeh et al., 2025, pp. 205–206). Setting the value too low can lead to the exclusion of relevant context, thus supplying the LLM with insufficient context to generate an accurate response, while an overly large number tends to dilute the response accuracy by incorporating irrelevant information (Njeh et al., 2025, p. 206). Recent research efforts have addressed this challenge by, for instance, employing confidence score-based thresholds rather than fixed chunk counts, resulting in the amount of retrieved context to be adaptively determined based on its semantic relevance to a given query (Njeh et al., 2025, p. 206). In the context of the present QA system, however, fixed numbers of chunks and a similarity score threshold alike pose inherent challenges, especially considering the vast discrepancy in the number of chunks across the four metadata categories, ranging from merely 2 views to 114 columns. Instead, drawing on the Auto-RAG model (Yu et al., 2024, p. 2), the inherent reasoning abilities of LLMs to dynamically optimize the parameters applied during the performed RAG search are leveraged, as illustrated in Figure 6.

As shown in Algorithm 2, the Search Mode Determination Agent's task closely follows the prompting principles of its predecessor and utilizes a combination of CoT and zero-shot CoT to invoke a reasoned response. However, whereas the Planning Agent's output inherently mirrors its underlying reasoning process through the design of the execution plan, the number of metadata chunks to retrieve constitutes the only tangible output of the Search Mode Determination Agent. To nonetheless reinforce the zero-shot CoT principle, a reasoning statement, which reflects on the actual process of determining the appropriate value, is additionally enforced.

Algorithm 2: Search Mode Determination Agent Task Prompt (pseudocode)

```
[...]

1 CONTEXT FOR YOUR TASK:
 2 - There is a total of 15 Tables, 2 Views, 114 Columns, and 24 Associations in the data model (SAP Datasphere).
 3 - You are currently working on iteration number: {self.state.iteration} out of {self.state.max_iterations}
 4 - The retrieving Agent in this Iteration is therefore: {current_agent}
 5 - The instruction for the current iteration is: {current_instruction}
 6
 7 YOUR TASK (MUST FOLLOW THE RULES STATED BELOW):
 8 1. You must provide a precise deep reasoning statement that elaborates how many entities must be returned to give a satisfactory answer to the instruction in a fact-based manner.
 9 2. Based on your result in step 1, provide the exact number of entities (i.e., Tables, Views, Columns, or Associations) that the respective Agent must return to fulfill the current instruction (ONLY Integer values are acceptable).

[...]
```

5.2.2.3 Table, View, Column, and Association Agents

Following the activities of the Planning Agent and the Search Mode Determination Agent, a conditional activation logic is configured for each of the four specialized worker agents, ensuring that the agent identified as relevant for the current iteration proceeds to execute its RAG search. Supplied with the previous agents' contributions as parameters, the tool output of the RAG search is simultaneously returned as the output of the respective worker agent itself (Figure 7). This design choice effectively strips the worker agents of the need to contribute through their generative abilities, allowing them to be powered by the locally hosted *Llama 3.2 3B*. In addition, it decouples the RAG pipeline by assigning full responsibility for the retrieval stage to the worker agents, while delegating the augmentation and generation stages to the subsequently acting Quality Control Agent.

5.2.2.4 Quality Control Agent

Once the RAG search of the current iteration's worker agent has been completed, the Quality Control Agent is introduced into the agentic workflow. Given this agent's critical role within the multi-agent system and the required reasoning depth of its task, the agent is configured to be API-based. The corresponding task specifies numerous instructions while adopting a prompt structure similar to those of the Planning Agent's and the Search Mode Determination Agent's tasks. As outlined in Algorithm 3, to supply the Quality Control Agent with sufficient contextual awareness, its prompt details out the originally submitted user query as well as a comprehensive summary of the QA system's state.

Once again incorporating the principles of CoT and zero-shot CoT, the agent is prompted to provide an analysis of the current iteration's retrieved CSN-structured metadata chunks and subsequently synthesize the resulting insights to modify the Planning Agent's execution plan for upcoming iterations (Figure 8). In doing so, the Quality Control Agent not only ensures the proper conclusion of each iteration, but also establishes the foundation for subsequent ones, which reflects the agent's manifold responsibilities.

Algorithm 3: Conveying Contextual Awareness to the Quality Control Agent (pseudocode)

```
[...]

1 CONTEXT FOR YOUR TASK:
 2 - The general goal of your multi-Agent team is to provide a precise answer to the following user query about the data model (SAP Datasphere) in an enterprise: {self.state.user_query}
 3 - The execution plan developed by the supervising Planning Agent is given as follows: {self.state.user_query_agent}
 4 - You are currently working on iteration number: {self.state.iteration} out of {self.state.max_iterations}
 5 - The active Agent in this iteration was therefore: {current_agent}
 6 - The {current_agent}, following its instruction, provided the following CSN-structured metadata chunks during this iteration:
 7 {self.state.current_RAG}

[...]
```

Building on Shi et al.'s (2023) observation that explicitly prompting an LLM to ignore irrelevant passages of retrieved context may yield more precise outputs compared to the standard practice of directly augmenting and generating a response in conventional RAG pipelines (p. 9), a corresponding instruction is incorporated into the Quality Control Agent's prompt. Furthermore, the agent is prevented from adjusting the responsibilities as well as the sequential order of instructions, hence narrowing its permissible range of actions to the sole refinement of existing instructions. It is thereby accepted that this design choice benefits the QA system's reliability at the expense of its flexibility.

5.2.2.5 Summarizing Agent

Once all instructions of the multi-agent system's execution plan have been completed, the Summarizing Agent is activated as the concluding step of the workflow. The agent is API-based since, as stated in section 4.2, its output is representative of the entire QA system's work. Given that the agent's output is not utilized in further downstream processes, its task prompt is structured in a relatively concise fashion compared to the tasks of preceding agents. Mainly, the agent is instructed to reflect on the initially submitted user query as well as the list of natural-language interpretations generated during preceding iterations by the Quality Control Agent to ultimately provide an answer to the user query (Figure 9).

6 Prototype Evaluation

6.1 Evaluation Metrics

Although RAG-based LLM applications have experienced a surge in popularity, no universally accepted gold standard has yet been established for their objective evaluation, spurring ongoing research efforts (Alammar & Grootendorst, 2024, p. 336). Aiming to streamline the objective assessment of various types of RAG systems, the *RAGAs* framework introduces a set of LLM-assisted metrics which respectively address a variety of quality dimensions along the entire RAG pipeline process (Alammar & Grootendorst, 2024, p. 336; Es et al., 2024, p. 156). As the performance of the agentic system is ultimately

benchmarked against the baseline of a standalone LLM that generates responses without the implementation and coordination overhead of multiple agents, the agentic system is evaluated based on its ability to comparatively improve the answer quality and token efficiency.

To achieve this, the *RAGAs* framework's Factual Correctness metric is employed. The metric focuses on the holistic evaluation of a QA system's output through a multi-stage process that begins with the decomposition of each given response into a series of modular claims. Subsequently, an evaluating LLM is employed to determine the degree to which the claims implied by the QA systems' outputs semantically overlap with a human-annotated reference (Fig. 10). By applying such binary classification of claim presence, both the multi-agent system's and standalone LLM's performances are assessed by establishing confusion matrices that incorporate the following metrics:

True positive (**TP**): The number of claims provided within the QA system's response that are simultaneously present in the human-annotated reference.

False positive (**FP**): The number of claims that are returned by the QA system whilst not being part of the human-annotated reference.

False negative (**FN**): The number of claims that are missing in the QA system's response but are present in the human-annotated reference.

(*RAGAs - Factual Correctness*, 2025)

Based on these counts, standard evaluation metrics such as Precision, Recall, and F1 score can be calculated. In the present context, however, Recall is employed as the primary metric, as the focus is on ensuring that the system meets the minimum requirements specified by the reference answers rather than penalizing the inclusion of supplementary or redundant information. It is important to note that the LLM-as-a-judge principle employed by many of the *RAGAs* metrics (Alammar & Grootendorst, 2024, p. 336) is susceptible to the very same limitations as the LLMs that power the QA systems. Acknowledging the significance of human evaluation in mitigating these limitations (Alammar & Grootendorst, 2024, p. 336), this paper therefore derives its ultimate conclusions through manual review and critical interpretation of the results, rather than solely quantitative metrics.

6.2 Question Catalog

The evaluation of the presented QA system prototype utilizes a manually crafted question catalog, as elaborated in section 3. To holistically assess the QA system's performance, all four metadata categories are addressed within the question catalog, whereas each category is represented by a set of queries with varying degrees of complexity. Following the work of Gabburo et al. (2024, pp. 2, 6), the present evaluation approach models the concept of query complexity as a two-dimensional measure defined by multi-hop complexity and compositional complexity. Multi-hop complexity denotes the number of reasoning steps that must be taken to provide an accurate answer to a given user question (Gabburo et al., 2024, p. 2). In the present context, this corresponds to the number of iterations in the execution plan

and thereby the number of worker agents required to formulate an accurate answer to a given user query. Conversely, compositional complexity refers to the amount of contextual information pieces that must be drawn together to provide a complete and precise response (Gabburo et al., 2024, p. 2). This complexity dimension is represented by the number of CSN-structured chunks that the worker agents must collectively retrieve to satisfy the contextual needs of a given user query.

Building on these definitions, the evaluation question catalog addresses the multi-hop complexity dimension with questions that are designed to require between one and three iterations to be accurately answered. To reflect typical real-world query patterns, a pyramid-like structure (Fig. 11) is employed, where 22 questions (55%) require one reasoning step, 12 questions (30%) require two reasoning steps, and six questions (15%) require a total of three reasoning steps. As a result, the question catalog entails a total of 40 questions. Within each of the multi-hop complexity levels, the questions' compositional complexity alternates between low and high, as determined by the number of CSN chunks associated with the metadata categories addressed. It is also worth noting that, for questions explicitly referencing table, view, column, or association names from the data model, these occurrences are replaced with their corresponding anonymized hash values.

To enable a precise evaluation of the respective LLM setups' performances, the user queries are deliberately designed to employ clear and simple instructions that aim to minimize ambiguity and lead to consistent and reproducible results. Furthermore, the human-annotated reference answers are precisely formulated to strictly address the requirements of the user query, excluding any supplementary details. Consequently, they serve as minimum sufficiency criteria that must be met for the respective answer of the multi-agent system or the standalone LLM to be regarded as correct.

6.3 Results

6.3.1 Result Generation

To comparatively benchmark the performance of the agentic system against a control baseline, a standalone LLM is initialized. Ensuring fairness in the comparative analysis, the standalone setup is powered by the same configuration of *ChatGPT* as the multi-agent system. As shown by Algorithm 4, the standalone LLM's prompt closely adheres to the structure employed in the multi-agent system. Since the Summarizing Agent is never directly exposed to metadata in its native CSN-structured format but instead formulates its final answers based on the Quality Control Agent's preceding natural-language interpretations, the agentic system's responses inherently adopt a user-understandable tone through semantic abstraction from the technical notation. The standalone LLM, by contrast, directly works with the CSN-structured data model and therefore exhibits a tendency to respond to user queries by directly quoting the CSN file. To counteract this, the task rules instruct the standalone LLM to adopt a similar user-friendly tone when formulating its answers. Lastly, the standalone LLM's prompt is concluded with the provision of the corresponding user query as well as the CSN-structured data model.

Algorithm 4

Standalone LLM Task Prompt (pseudocode)

```
1 BACKSTORY:

2 - You are an expert in understanding the semantic meaning of CSN-structured metadata for an enterprise data model (SAP Datasphere).

3 - There is a total of 15 Tables, 2 Views, 114 Columns, and 24 Associations in the data model (SAP Datasphere).

4

5 YOUR TASK (MUST FOLLOW THE RULES STATED BELOW):

6 - Provide the user with a detailed and precise final answer to the user query based on the CSN-structured definition listed below.

7

8 RULES FOR YOUR TASK (YOUR ANSWER MUST STRICTLY ADHERE TO EACH RULE):

9 - The answer has to be provided in natural language and direct references to the CSN-structured data model have to be kept to a minimum.

10

11 USER QUERY:

12 {User Query}

13

14 DATA MODEL (CSN):

15 {DATAMODEL_ANONYMIZED with indentation level 2}
```

Once set up, both architectures are exposed to all 40 user queries in the catalog, while their corresponding responses and token usage metrics are systematically recorded for subsequent evaluation.

6.3.2 Result Analysis

6.3.2.1 Token Efficiency

The result analysis commences with an assessment of the agentic system's capability to adequately reflect the anticipated multi-hop and compositional complexity of the catalog questions in its iteration and retrieval planning. With the exception of one user query, the ongoing consistency between the questions' planned and actual multi-hop complexity indicates that the Planning Agent is able to identify the specialized worker agents relevant to a given user query by correctly setting the number of necessary iterations. A similarly low deviation between predicted and actual values is observed for the compositional complexity, which differ in only two out of 40 cases. This primarily underscores the

effectiveness of the Search Mode Determination Agent in adequately determining the required number of CSN-structured metadata chunks to accurately follow the system's execution plan.

In line with the principle that the standalone LLM fully processes the entire CSN-structured data model for each of its generated answers, its input token consumption adheres closely to the data model's size of 15,299 tokens, with respective markups being applied for the token size of each user query. Across the complete set of 40 user queries, the input token cost for the standalone LLM solution ranges between 15,482 and 15,529 (mean = 15,502; SD = 12.7; median = 15,500; IQR = 15,491 – 15,512). By comparison, the multi-agent system's token consumption reflects its scalability and ability to dynamically react to multi-hop and compositional complexity through intelligent delegation and iteration planning. In this setup, the input token consumption ranges from values as low as 4,679 to 23,873 (mean = 10,321; SD = 5,661.3; median = 8,231; IQR = 6,501 – 12,904). A further distinguishing factor in the input token consumption of the RAG-based agentic system is the additional embedding token overhead which occurs for each of the retrieving agent's input queries. To assess this markup, the tokenizer utilized by *text-embedding-3-large* (*cl100k_base*) is employed. With the input token costs of *text-embedding-3-large* amounting to 6.5% of those of *ChatGPT-4.1*, the embedding token volume is expressed relative to the chat input, ultimately raising the multi-agent system's mean input token consumption to 10,326. Furthermore, the agentic system introduces additional token consumption by embedding the data model itself, which is a prerequisite for the subsequent RAG pipeline. In the present case, combined tokenization of all metadata chunks using *cl100k_base* yields a token count of 18,274. Normalized against the input token cost of *ChatGPT-4.1*, the embedding overhead corresponds to a one-time addition of 1,188 input tokens, bringing the mean input token consumption across 40 user queries to 10,356 and hence making the multi-agent system approximately 33% less token-demanding for the employed question catalog than the standalone LLM's mean of 15,502 tokens.

The observed query-wise variability of the multi-agent system's token use is assumed to depend on the number of iterations and retrieved CSN chunks. To assess the statistical significance and magnitude of these relationships, the following linear regression model is fitted:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

with y denoting the input token usage per query, x_1 representing the number of iterations, and x_2 representing the number of retrieved CSN chunks (Montgomery et al., 2013, pp. 67–70). Both x_1 and x_2 appear highly significant ($p < 0.001$) and coefficients indicate that each additional iteration increases token usage by approximately 3,743 tokens, while each additional retrieved chunk adds approximately 91 tokens. The model explains 97.1% of the variance ($R^2 = 0.971$), suggesting that these two features capture most of the query-wise variability. Based on the fitted regression model, the agentic system's expected input token usage for a given user query can be estimated via:

$$\hat{y} = 1{,}622 + 3{,}743x_1 + 91x_2$$

Table 1 illustrates a token efficiency matrix, with each cell reporting the absolute estimated input token consumption and its relative deviation from the baseline, which is given by the standalone LLM's mean consumption of 15,502 input tokens.

Table 1
Token Efficiency Matrix Estimated via Multiple Linear Regression

CSN chunks/ iterations	**1**	**10**	**50**	**100**	**150**
1	5,456 (-64.8%)	6,275 (-59.5%)	9,915 (-36%)	14,465 (-6.7%)	19,015 (+ 22.7%)
2	9,199 (-40.7%)	10,018 (-35.4%)	13,658 (-11.9%)	18,208 (+ 17.5%)	22,758 (+ 46.8%)
3	12,942 (-16.5%)	13,761 (-11.2%)	17,401 (+ 12.3%)	21,951 (+ 41.6%)	26,501 (+ 71%)
4	16,685 (+ 7.6%)	17,504 (+ 12.9%)	21,144 (+ 36.4%)	25,694 (+ 65.8%)	30,224 (+ 95.1%)

It should additionally be noted that, to fully assess the token efficiency of both architectures, completion token efficiency must also be considered, particularly given their significantly higher associated API costs (*GPT-4.1 - OpenAI API*, n.d.). Upon assessment, it becomes apparent that the agentic setup, on average, has a higher consumption of completion tokens than the standalone LLM. This observation is primarily attributed to the collaborative exchanges between the agents and the overhead resulting from each agent's response generation. Consequently, with a mean of 1,170 completion tokens per user query, the multi-agent system consumes around 269% more than the standalone LLM with its average of 317 tokens per query. However, contrary to the completion token usage of the multi-agent system, which scales with the amount of agent interaction, the completion token count of the standalone LLM does not adhere to any apparent architectural factors. With an observed minimum of 33, a maximum of 1,579, and a resulting SD of 351 tokens, the completion token count of the standalone LLM reveals the highly variable nature of its answer verbosity. In this regard, deliberate prompting for more concise or more detailed responses may improve completion token consistency of both the agentic system and the standalone LLM. However, such restrictions, in turn, may cause the systems to omit important details from their responses or, conversely, dilute their answers with incoherent information to attain a desired level of verbosity. Consequently, while acknowledging the completion token overhead inherently linked to hosting multiple collaborating LLMs instead of a single LLM, this paper prioritizes the input token efficiency as the more predictable and optimizable parameter.

6.3.2.2 Answer Quality

Given the susceptibility of the LLM-as-a-judge principle employed by the *RAGAs* framework to incorrectly interpret the responses of the multi-agent system as well as the standalone LLM, the resulting Recall scores undergo a manual review prior to their inclusion in a quantitative evaluation. With a mean of 0.91 (min = 0; max = 1; SD = 0.22), the standalone LLM surpasses the multi-agent system's adjusted Recall score of 0.89 (min = 0.12; max = 1; SD = 0.24) by approximately 2.2%, suggesting slightly superior completeness in capturing reference answer content. However, it must be noted that in all instances where the agentic system did not manage to achieve a complete Recall score of 1, its failures to generate factual responses appear to be rooted in the RAG search's limitation to selectively retrieve information in low-compositional-complexity scenarios where the Search Mode Determination agent requires a smaller number of metadata chunks to be retrieved. Yet, this limitation is not observed consistently as the system, on certain occasions, accurately retrieves information for questions that are similar in structure to those where retrieval failures previously occurred.

7 Discussion

7.1 Interpretation and Implications

The empirical results presented in section 6.3 allow the three guiding research questions to be addressed and the four expectations stated in section 3 to be assessed. Given the exploratory character of the study, the limited sample size of 40 queries, and the reliance on a single data model, the following observations should be read as indications rather than as confirmatory evidence. The corresponding constraints are discussed in detail in section 7.2.

RQ1 – Answer Quality

The agentic RAG architecture achieves an answer quality that appears broadly comparable to the standalone LLM baseline within the evaluated setting, with a mean Recall of 0.89 against 0.91. The minor difference of approximately 2.2% suggests that selective retrieval may lead to occasional information gaps, with their impact being particularly pronounced for queries of low compositional complexity, where missing or incorrectly retrieved context more strongly affects the final answer. Beyond this retrieval-side effect, the modularization of reasoning across specialized agents, in combination with the chunked representation of the CSN-structured metadata, may also fragment globally relevant information that would be more readily accessible in a monolithic setup. This points to a possible structural limitation of the retrieval and decomposition approach rather than a deficiency of the reasoning pipeline itself.

RQ2 – Token Efficiency

Within the evaluated catalog, the multi-agent system reduces input token consumption by approximately 33% on average compared to the standalone baseline, indicating that selective retrieval reduces the amount of processed context. This effect is not uniform across all queries, as input token usage varies considerably depending on the number of reasoning steps and retrieved metadata chunks. The reduction

in input tokens is furthermore accompanied by an increase in completion token usage of approximately 269%, primarily driven by inter-agent coordination during planning, validation, and summarization. Taken together, these observations indicate a trade-off where RAG-based multi-agent architectures appear to shift cost from input to completion tokens rather than producing a uniform efficiency gain. This is consistent with a structural shift from monolithic inference toward distributed reasoning, where efficiency gains in one part of the pipeline are partially offset by increased complexity in another. Whether this redistribution is economically favorable depends on the relative pricing of input and completion tokens in the deployed model and should therefore be evaluated holistically rather than on the basis of a single metric.

RQ3 – Query Complexity

The regression results reported in section 6.3 indicate that, within the evaluated catalog, input token usage scales primarily with the number of reasoning steps and retrieved CSN chunks, suggesting that the efficiency characteristics of the agentic system are not inherent but contingent on query complexity. As illustrated in the token efficiency matrix (Table 1), the relative advantage of the multi-agent system declines as iteration count and retrieved context volume increase, with the break-even point shifting jointly with iteration count and retrieved context volume rather than along a single complexity dimension. Accordingly, the suitability of an agentic QA system for metadata analysis seems to depend on the expected query profile and should not be assumed a priori.

Building on these observations, the four expectations stated in section 3 can be assessed as summarized in Table 2.

Table 2
Expectation Assessment

Expectation	Statement	Empirical observation	Assessment
E1	Reduced input token consumption vs. full-context baseline	-33% mean input tokens	Consistent
E2	Increased completion tokens due to coordination overhead	+ 269% mean completion tokens	Consistent
E3	Comparable answer quality to standalone LLM	Recall 0.89 vs. 0.91 (-2.2%)	Tentatively consistent; retrieval-related caveats
E4	Efficiency gains decrease with query complexity	Regression indicates scaling; break-even shifts with complexity	Consistent

From a practical perspective, these observations have potential implications for organizational use of RAG-based multi-agent systems for metadata analysis, particularly in cost-sensitive environments where token-based pricing models prevail. Optimizing for input token efficiency may lead to substantial cost

reductions when processing large and complex metadata repositories, but practitioners should account for the additional overhead introduced by agent coordination and carefully balance efficiency gains against increased system complexity. In the context of metadata management, the proposed architecture suggests a scalable alternative to traditional approaches by enabling flexible, on-demand analysis without extensive preprocessing or manual curation, while the identified trade-offs underscore that more complex architectures cannot be assumed to universally outperform simpler baselines.

The boundary conditions of the presented architecture follow directly from this complexity dependence. The potential benefits of agentic RAG appear most pronounced in scenarios with limited iteration counts and constrained retrieval scope, whereas in settings with higher iteration counts and larger retrieved context volumes the coordination overhead may outweigh the gains from selective retrieval. Accordingly, the architecture is best understood as a design option whose suitability should be aligned with the expected query profile, the relative pricing of input and completion tokens, and the volume and heterogeneity of the metadata to be analyzed, rather than as a universally superior replacement for monolithic inference.

7.2 Limitations

Despite the demonstrated strengths, certain limitations of the presented approach must be acknowledged.

First, the observed performance bottleneck resulting from queries with low compositional complexity appears to be localized to the use of *text-embedding-3-large* for dense retrieval and does not correspondingly compromise *ChatGPT-4.1*'s response quality upon augmentation. In this regard, it should be noted that the chosen chunking method prioritizes the modular encoding and encapsulation of metadata in separate, independently retrievable embeddings. This procedure, however, departs from the native CSN format where e.g., column and association metadata is embedded in a broader context of surrounding definitions and dependencies, possibly resulting in the loss of valuable contextual cues of the original data model structure. This finding echoes the observation made by Qu et al. (2024), according to whom a simplified fixed-size chunking approach may provide a more efficient alternative to computationally expensive semantic chunking, while achieving comparable results in RAG applications (p. 4).

Second, this paper is correspondingly constrained by its reliance on preexisting knowledge of the data model to implement semantic chunking through deliberate parsing, which restricts the generalizability of the findings to other domains and metadata structures.

Third, the diversity of user queries employed in the evaluation question catalog constitutes another limitation. Even though the agentic system was developed and tested independently of the user queries employed for its evaluation, it is unavoidable that a portion of the evaluating queries shares a semantic overlap with those that shaped the architecture and prompts at the base of the multi-agent system. As a

result, the evaluation may not fully capture the system's performance and robustness across a broader spectrum of potential user queries.

Fourth, despite the application of manual result validation, the reliance on LLM-assisted evaluation metrics remains a potential source of bias.

Further testing is therefore needed to evaluate alternative chunking and retrieval strategies, validate the approach across different data models, assess its performance on more diverse question sets, and apply complementary evaluation frameworks. Accordingly, the reported results should be interpreted as exploratory rather than definitive.

8 Conclusion

In response to the evolving challenges of metadata management, this paper explores a lightweight, domain- and platform-agnostic approach for analyzing metadata in its native structured text-based format. By harnessing the inherent reasoning and code-understanding capabilities of LLMs, a multi-agent QA system is used as an experimental artifact. Compared to a traditional setup in which a single LLM processes an entire data model, thus being limited by its context window and imposing substantial costs upon wide-scale processing of extensive metadata structures, this agentic RAG framework selectively analyzes relevant passages. It thereby allows the investigation of how selectively retrieved context influences response generation and token consumption. The results indicate that response completeness remains broadly comparable to a standalone LLM baseline, while input token consumption is lower in the evaluated setting, with an average reduction of approximately 33%, though these savings are accompanied by substantially higher completion token usage due to inter-agent coordination. By focusing on selective retrieval for metadata exploration and addressing the challenges posed by long and complex metadata inputs for LLMs, these findings are relevant for the design of flexible and cost-aware metadata analysis approaches in enterprise contexts.

Ultimately, the results suggest that agentic systems have the potential to enhance metadata understanding, while their efficiency and scalability remain dependent on query complexity, retrieval scope, and coordination overhead. Particularly in enterprise environments with large user bases and heterogeneous metadata formats, the adaptive nature of the presented framework may reduce barriers in complex query scenarios, thereby supporting greater data maturity. Although multiple research efforts have explored the synergetic relationship of LLMs and metadata management, to the best of current knowledge, limited work has empirically examined the use of agentic RAG for analyzing data model metadata in its native declarative format through AI-assisted methods. This paper therefore provides exploratory insights into the potential of integrating agentic LLMs and RAG in this domain.

Certain limitations are nevertheless acknowledged for the presented approach. To a large extent, these manifest in the retrieval stage of the RAG pipeline, particularly when fine-grained nuances play a decisive role in identifying relevant metadata segments. To improve the semantic discernability of metadata chunks and retrieval performance, the development of a less implementation-heavy chunking strategy

that preserves the informational content of each chunk represents a key factor that warrants more systematic investigation. Furthermore, as LLMs are primarily trained on natural language, representing and chunking data model metadata in natural language rather than structured code may constitute a worthwhile direction for future research. Lastly, the modular decomposition of tasks, which is central to the present agentic system implementation, opens the possibility of strategically delegating an even larger share of less complex subtasks to locally hosted LLMs, thereby highlighting opportunities for future work on cost-efficient hybrid deployments.

Declarations

Conflict of interest:

The authors declare no conflict of interest;

Consent to Participate

declaration: not applicable;

Consent to Publish

declaration: not applicable;

Ethics declaration

not applicable.

Funding:

The authors did not receive any funding for this research;

Author Contribution

R.A. and A.A. jointly conceptualized the study, designed the methodology, and developed the agentic RAG architecture. R.A. implemented the prototype, conducted the experiments, and prepared the figures and tables. A.A. supervised the research and provided critical revisions. Both authors wrote and reviewed the manuscript.

References

1. Alammar, J., & Grootendorst, M. (2024). *Hands-on Large Language Models: Language Understanding and Generation*. O'Reilly Media, Inc.
2. Arslan, M., Munawar, S., & Cruz, C. (2024). Sustainable Digitalization of Business with Multi-Agent RAG and LLM. *Procedia Computer Science*, *246*, 4722–4731.
3. Bandara, F., Jayawickrama, U., Subasinghage, M., Olan, F., Alamoudi, H., & Alharthi, M. (2024). Enhancing ERP Responsiveness Through Big Data Technologies: An Empirical Investigation. *Information Systems Frontiers*, *26*(1), 251–275.
4. Berti, A., Park, G., Rafiei, M., & van der Aalst, W. (2021). An Event Data Extraction Approach from SAP ERP for Process Mining. *arXiv Preprint arXiv:2110.03467*.
5. Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., & Hoefler, T. (2024). Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, *38*(16), 17682–17690.
6. *Core Schema Notation (CSN)*. (2024). Retrieved August 29, 2025, from https://cap.cloud.sap/docs/cds/csn
7. *CrewAI - Create Custom Tools*. (n.d.). CrewAI. Retrieved August 29, 2025, from https://docs.crewai.com/en/learn/create-custom-tools
8. *CrewAI - Introduction*. (n.d.). CrewAI. Retrieved August 28, 2025, from https://docs.crewai.com/en/introduction
9. *CrewAI - Tools Overview*. (n.d.). CrewAI. Retrieved August 28, 2025, from https://docs.crewai.com/en/tools/overview
10. DAMA. (2017). *DAMA-DMBOK: Data Management Body of Knowledge* (2nd ed.). Technics Publications, LLC.
11. Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., & Mordatch, I. (2023). Improving Factuality and Reasoning in Language Models through Multiagent Debate. *Forty-First International Conference on Machine Learning*, 1–31.
12. Eichler, R., Giebler, C., Gröger, C., Hoos, E., Schwarz, H., & Mitschang, B. (2021). Enterprise-Wide Metadata Management—An Industry Case on the Current State and Challenges. *Proceedings of the 24th International Conference on Business Information Systems (BIS 2021)*.
13. Eichmann, G. (2024). *Analytic Model Sample Data Model*. GitHub. https://github.com/SAP-samples/analytics-cloud-datasphere-community-content/releases/tag/DSPSampleAM_1.0.0
14. Eryurek, E., Gilad, U., Lakshmanan, V., Kibunguchy-Grant, A., & Ashdown, J. (2021). *Data Governance: The Definitive Guide*. O'Reilly Media, Inc.
15. Es, S., James, J., Espinosa Anke, L., & Schockaert, S. (2024). RAGAs: Automated Evaluation of Retrieval Augmented Generation. In N. Aletras & O. De Clercq (Eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations* (pp. 150–158). Association for Computational Linguistics. https://aclanthology.org/2024.eacl-demo.16/

16. Foster, D. (2022). *Generative Deep Learning*. O'Reilly Media, Inc.
17. Gabburo, M., Jedema, N. P., Garg, S., Ribeiro, L. F., & Moschitti, A. (2024). Measuring Retrieval Complexity in Question Answering Systems. *arXiv Preprint arXiv:2406.03592*.
18. Gamage, G., Mills, N., De Silva, D., Manic, M., Moraliyage, H., Jennings, A., & Alahakoon, D. (2024). Multi-Agent RAG Chatbot Architecture for Decision Support in Net-Zero Emission Energy Systems. *2024 IEEE International Conference on Industrial Technology (ICIT)*, 1–6.
19. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2024). Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv Preprint arXiv:2312.10997*, *2*(1).
20. *GPT-4.1—OpenAI API*. (n.d.). Retrieved August 23, 2025, from https://platform.openai.com/docs/models/gpt-4.1
21. Gröger, C. (2021). There Is No AI Without Data. *Communications of the ACM*, *64*(11), 98–108.
22. Guo, J., Du, L., Liu, H., Zhou, M., He, X., & Han, S. (2023). Gpt4Graph: Can Large Language Models Understand Graph-Structured Data? An Empirical Evaluation and Benchmarking. *arXiv Preprint arXiv:2305.15066*.
23. Han, S., Zhang, Q., Yao, Y., Jin, W., Xu, Z., & He, C. (2024). LLM Multi-Agent Systems: Challenges and Open Problems. *arXiv Preprint arXiv:2402.03578*.
24. Hayashi, T., Sakaji, H., Dai, J., & Goebel, R. (2024). Metadata-Based Data Exploration with Retrieval-Augmented Generation for Large Language Models. *2024 IEEE International Conference on Big Data (BigData)*, 6574–6583.
25. Hechler, E., Weihrauch, M., & Wu, Y. (Catherine). (2023). Intelligent Cataloging and Metadata Management. In *Data Fabric and Data Mesh Approaches with AI: A Guide to AI-based Data Cataloging, Governance, Integration, Orchestration, and Consumption* (pp. 293–310). Apress.
26. Hoseini, S., Burgdorf, A., Paulus, A., Meisen, T., Quix, C., & Pomp, A. (2024). Towards LLM-Augmented Creation of Semantic Models for Dataspaces. *The Second International Workshop on Semantics in Dataspaces, Co-Located with the Extended Semantic Web Conference*.
27. jaigupta. (2023). *SAP Datasphere Analytic Model Series—Data Model Introduction*. SAP Community. Retrieved August 29, 2025, from https://community.sap.com/t5/technology-blogs-by-sap/sap-datasphere-analytic-model-series-data-model-introduction/ba-p/13550812
28. jan_fetzer. (2023). *Introducing the Analytic Model in SAP Datasphere*. SAP Community. https://community.sap.com/t5/technology-blogs-by-sap/introducing-the-analytic-model-in-sap-datasphere/ba-p/13568591
29. Jeong, C. (2024). A Study on the Implementation Method of an Agent-Based Advanced RAG System Using Graph. *arXiv Preprint arXiv:2407.19994*.
30. Kim, D., Kim, B., Han, D., & Eibich, M. (2024). AutoRAG: Automated Framework for Optimization of Retrieval Augmented Generation Pipeline. *arXiv Preprint arXiv:2410.20878*.
31. Korini, K., & Bizer, C. (2023). Column Type Annotation Using ChatGPT. *arXiv Preprint arXiv:2306.00745*.

32. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., & Rocktäschel, T. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, *33*, 9459–9474.
33. Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2023). Lost in the Middle: How Language Models Use Long Contexts. *arXiv Preprint arXiv:2307.03172*.
34. *Llama 3.2*. (2024). Meta AI. Retrieved August 30, 2025, from https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/
35. Lobo, E., Hassanzadeh, O., Pham, N., Mihindukulasooriya, N., Subramanian, D., & Samulowitz, H. (2023). Matching Table Metadata with Business Glossaries Using Large Language Models. *arXiv Preprint arXiv:2309.11506*.
36. Masoudifard, A., Sorond, M. M., Madadi, M., Sabokrou, M., & Habibi, E. (2024). Leveraging Graph-RAG and Prompt Engineering to Enhance LLM-Based Automated Requirement Traceability and Compliance Checks. *arXiv Preprint arXiv:2412.08593*.
37. Mohammed, M., Talburt, J. R., Syed, H., & Mehjabeen. (2023). Metadata: An Integral Component of the Modern Data Strategy. *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, 1628–1631.
38. Montgomery, D. C., Peck, E. A., & Vining, G. G. (2013). *Introduction to Linear Regression Analysis* (5th ed.). Wiley.
39. Mueller, J. (2023). *A Deeper Look Into How SAP Datasphere Enables a Business Data Fabric*. SAP News Center. Retrieved August 31, 2025, from https://news.sap.com/2023/03/sap-datasphere-business-data-fabric/
40. Njeh, C., Nakouri, H., & Jaafar, F. (2025). Enhancing RAG-Retrieval to Improve LLMs Robustness and Resilience to Hallucinations. In H. Quintián, E. Corchado, A. Troncoso Lora, H. Pérez García, E. Jove Pérez, J. L. Calvo Rolle, F. J. Martínez de Pisón, P. García Bringas, F. Martínez Álvarez, Á. Herrero, & P. Fosci (Eds.), *Hybrid Artificial Intelligent Systems* (pp. 201–213). Springer Nature Switzerland.
41. Ozdemir, S. (2023). *Quick Start Guide to Large Language Models: Strategies and Best Practices for using ChatGPT and other LLMs*. Addison-Wesley Professional.
42. Pan, S., Zheng, Y., & Liu, Y. (2024). Integrating Graphs with Large Language Models: Methods and Prospects. *IEEE Intelligent Systems*, *39*(1), 64–68.
43. Peng, B., Zhu, Y., Liu, Y., Bo, X., Shi, H., Hong, C., Zhang, Y., & Tang, S. (2024). Graph Retrieval-Augmented Generation: A Survey. *arXiv Preprint arXiv:2408.08921*.
44. Qu, R., Tu, R., & Bao, F. (2024). Is Semantic Chunking Worth the Computational Cost? *arXiv Preprint arXiv:2410.13070*.
45. *RAGAs—Factual Correctness*. (2025). Retrieved October 20, 2025, from https://docs.ragas.io/en/latest/concepts/metrics/available_metrics/factual_correctness/
46. Sanmartin, D. (2024). KG-RAG: Bridging the Gap Between Knowledge and Creativity. *arXiv Preprint arXiv:2405.12035*.

47. Sanseviero, O., Cuenca, P., Passos, A., & Whitaker, J. (2024). *Hands-On Generative AI with Transformers and Diffusion Models*. O'Reilly Media, Inc.
48. Schmid, S., Henson, C., & Tran, T. (2019). Using Knowledge Graphs to Search an Enterprise Data Lake. In P. Hitzler, S. Kirrane, O. Hartig, V. de Boer, M.-E. Vidal, M. Maleshkova, S. Schlobach, K. Hammar, N. Lasierra, S. Stadtmüller, K. Hose, & R. Verborgh (Eds.), *The Semantic Web: ESWC 2019 Satellite Events* (pp. 262–266). Springer International Publishing.
49. Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E. H., Schärli, N., & Zhou, D. (2023). Large Language Models Can Be Easily Distracted by Irrelevant Context. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *Proceedings of the 40th International Conference on Machine Learning* (Vol. 202, pp. 31210–31227). PMLR.
50. Singh, A., Ehtesham, A., Kumar, S., & Khoei, T. T. (2025). Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG. *arXiv Preprint arXiv:2501.09136*.
51. Singh, M., Kumar, A., Donaparthi, S., & Karambelkar, G. (2025). Leveraging Retrieval Augmented Generative LLMs for Automated Metadata Description Generation to Enhance Data Catalogs. *arXiv Preprint arXiv:2503.09003*.
52. Suhara, Y., Li, J., Li, Y., Zhang, D., Demiralp, Ç., Chen, C., & Tan, W.-C. (2022). Annotating Columns with Pre-trained Language Models. *Proceedings of the 2022 International Conference on Management of Data, SIGMOD '22*, 1493–1503.
53. Talebirad, Y., & Nadiri, A. (2023). Multi-Agent Collaboration: Harnessing the Power of Intelligent LLM Agents. *arXiv Preprint arXiv:2306.03314*.
54. *Text-Embedding-3-Large—OpenAI API*. (n.d.). Retrieved August 29, 2025, from https://platform.openai.com/docs/models/text-embedding-3-large
55. van der Meulen, R. (2020). *Understand the 4th Era of ERP*. Gartner. Retrieved September 15, 2025, from https://www.gartner.com/smarterwithgartner/understand-the-4th-era-of-erp
56. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., & others. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, *35*, 24824–24837.
57. Yang, W., Fu, R., Amin, M. B., & Kang, B. (2025). The Impact of Modern AI in Metadata Management. *Human-Centric Intelligent Systems*, *5*(3).
58. Yu, T., Zhang, S., & Feng, Y. (2024). Auto-RAG: Autonomous Retrieval-Augmented Generation for Large Language Models. *arXiv Preprint arXiv:2411.19443*.

Figures

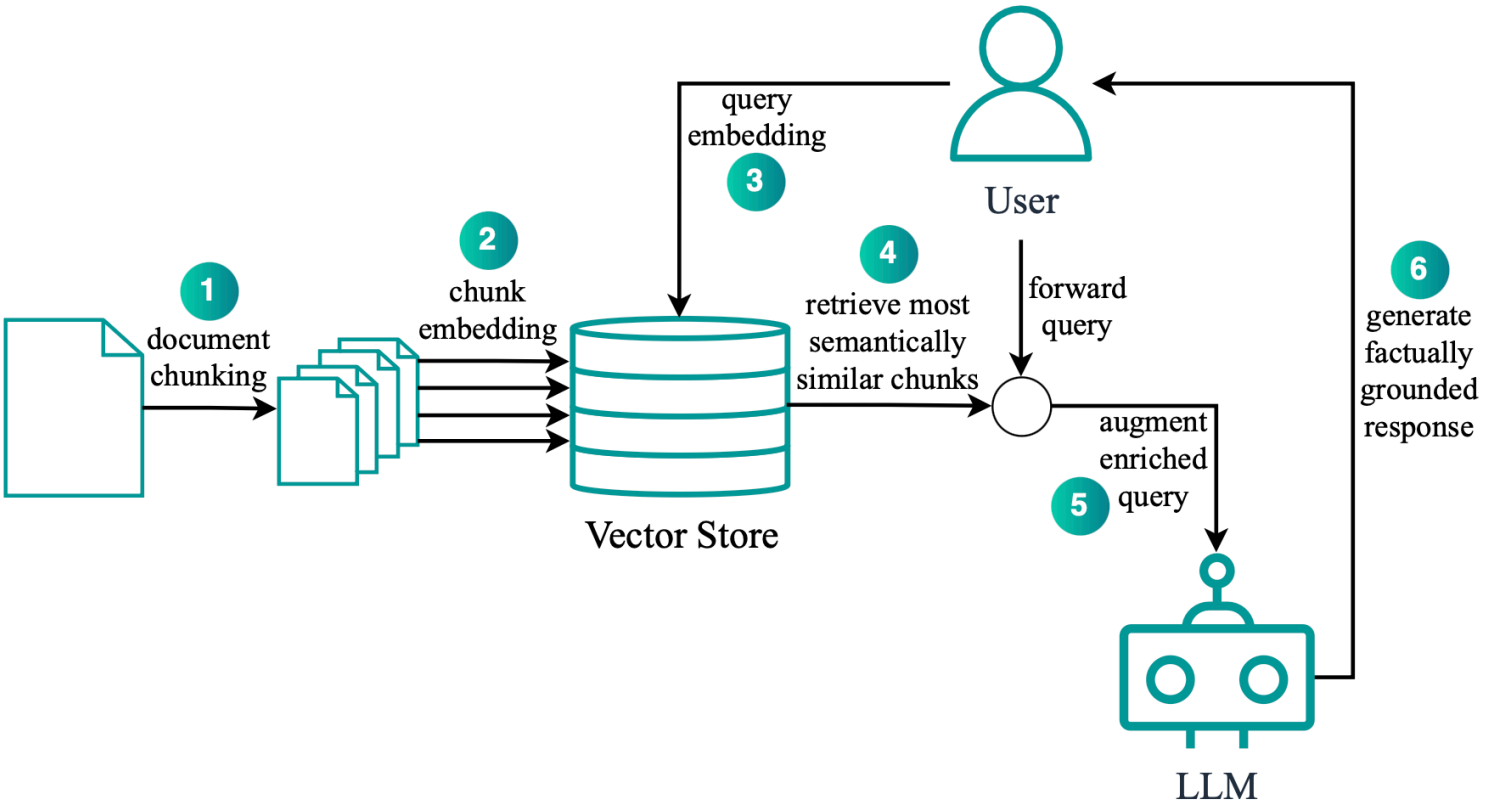


Figure 1

RAG Workflow (Alammar & Grootendorst, 2024, pp. 300, 312, 314, 328–329)

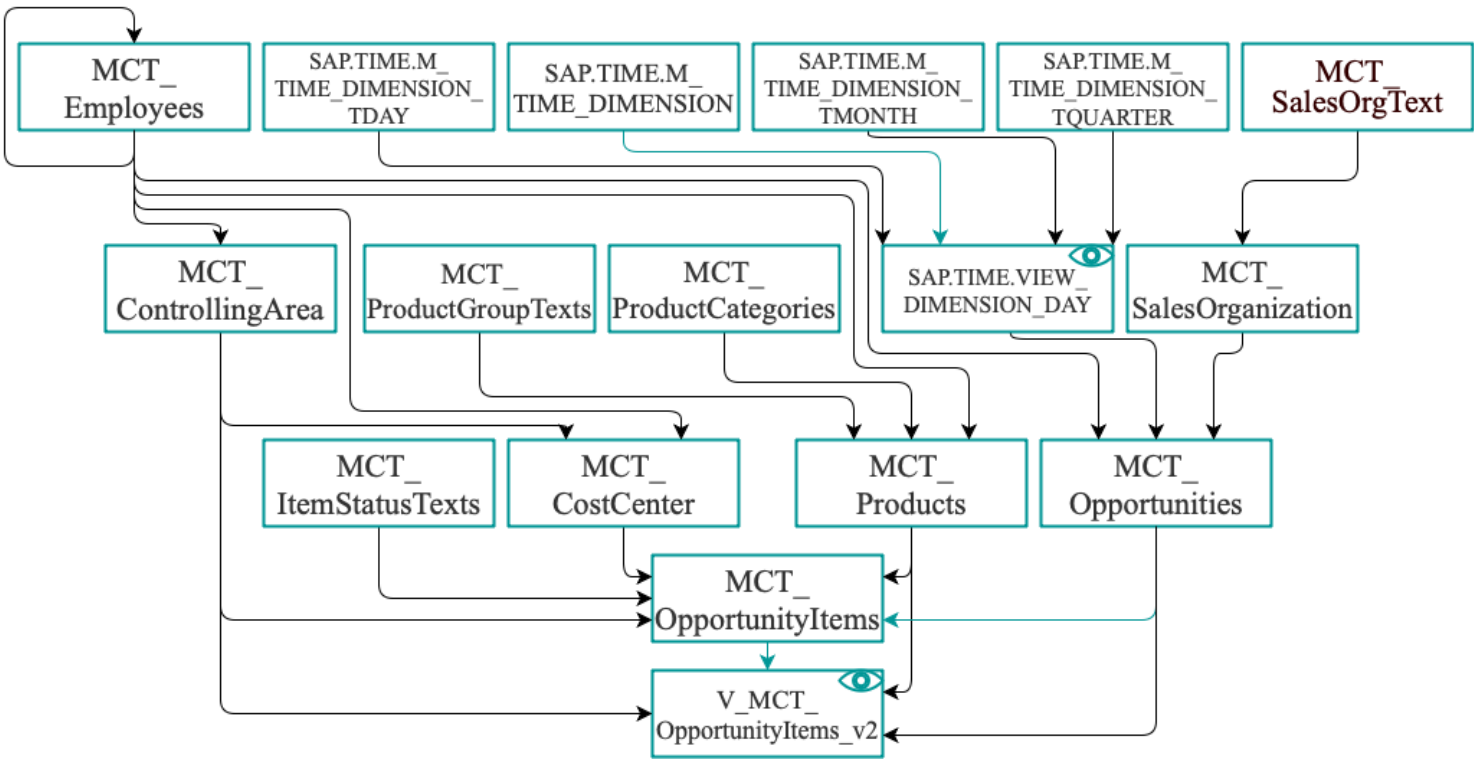


Figure 2

Data Lineage Diagram of the Sample Data Model (Eichmann, 2024; jaigupta, 2023)

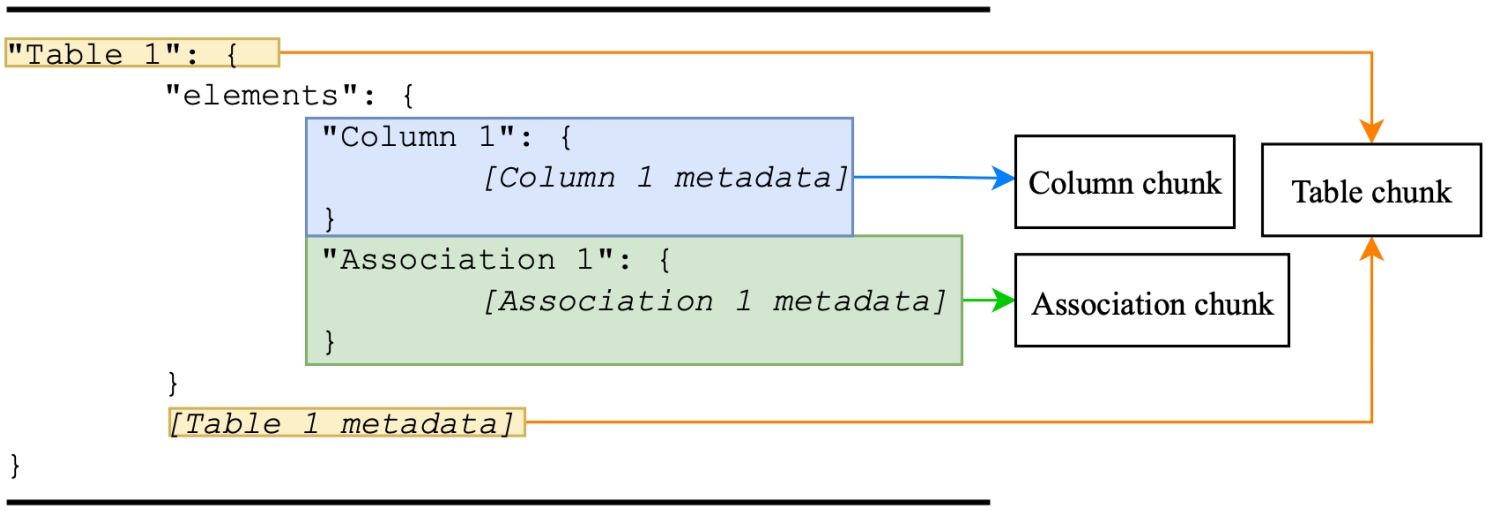


Figure 3

Data Model Chunking Centered on Self-Containment

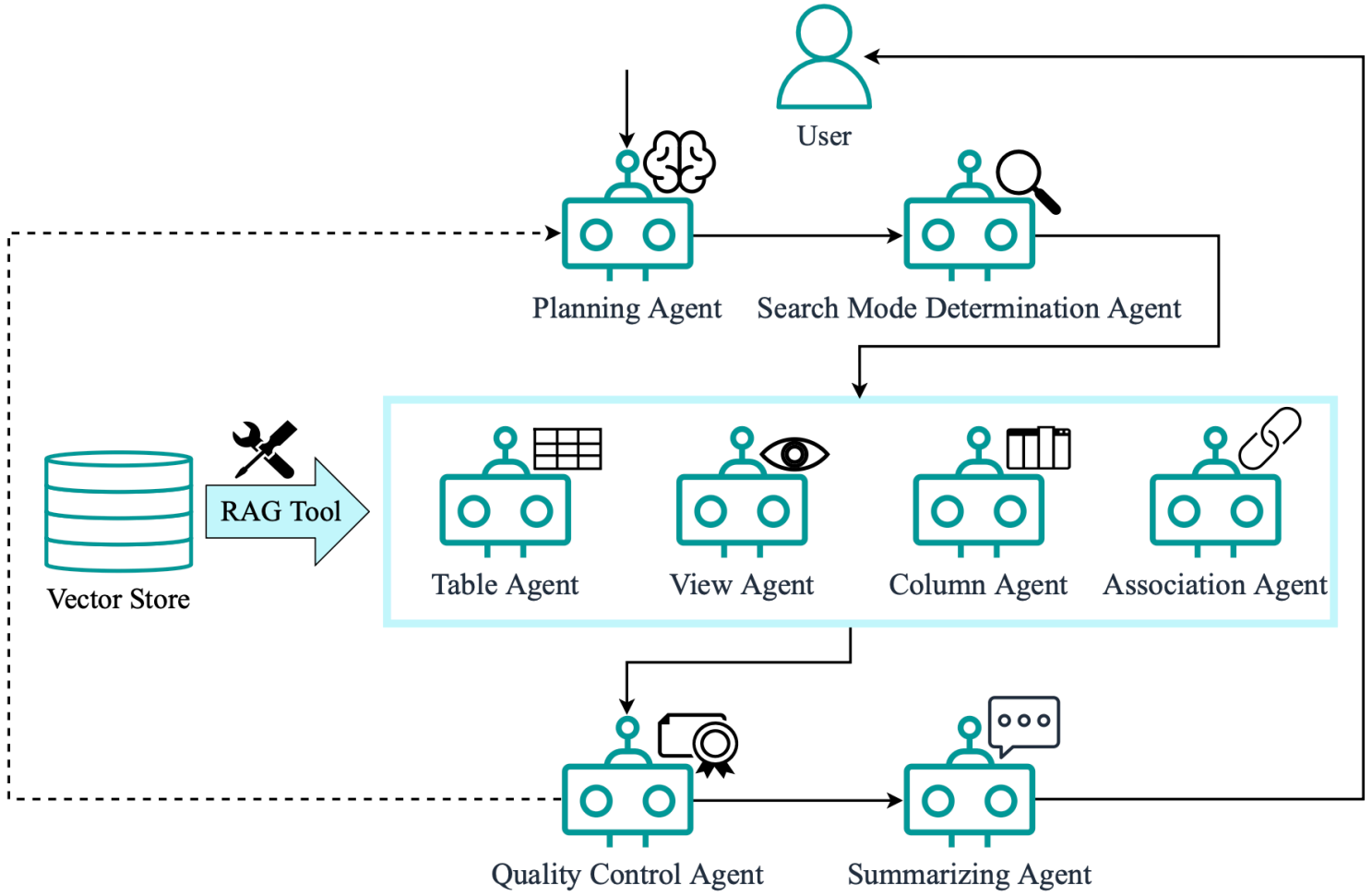


Figure 4

Working Structure of the Multi-Agent System

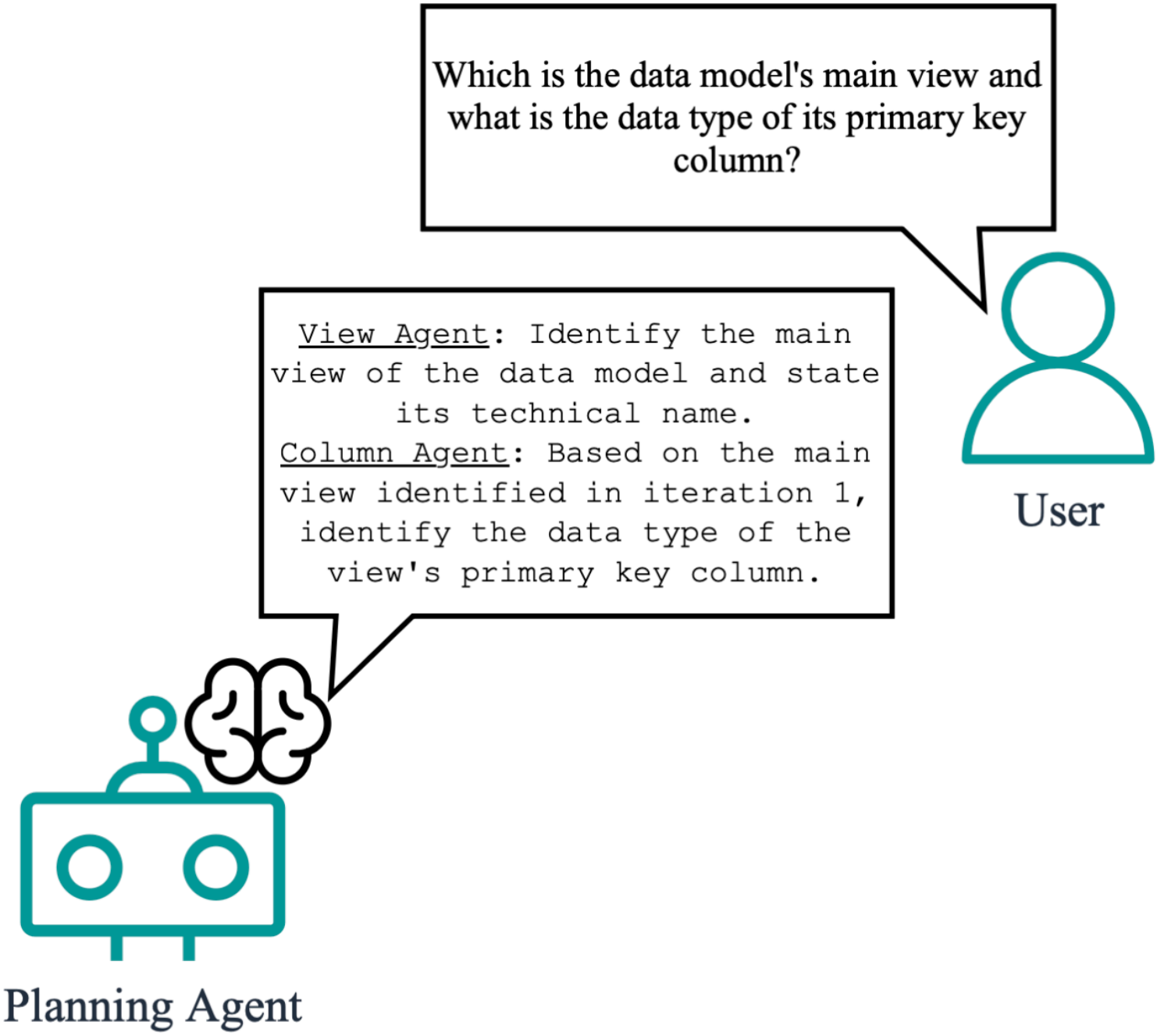


Figure 5

Functional Principle of the Planning Agent

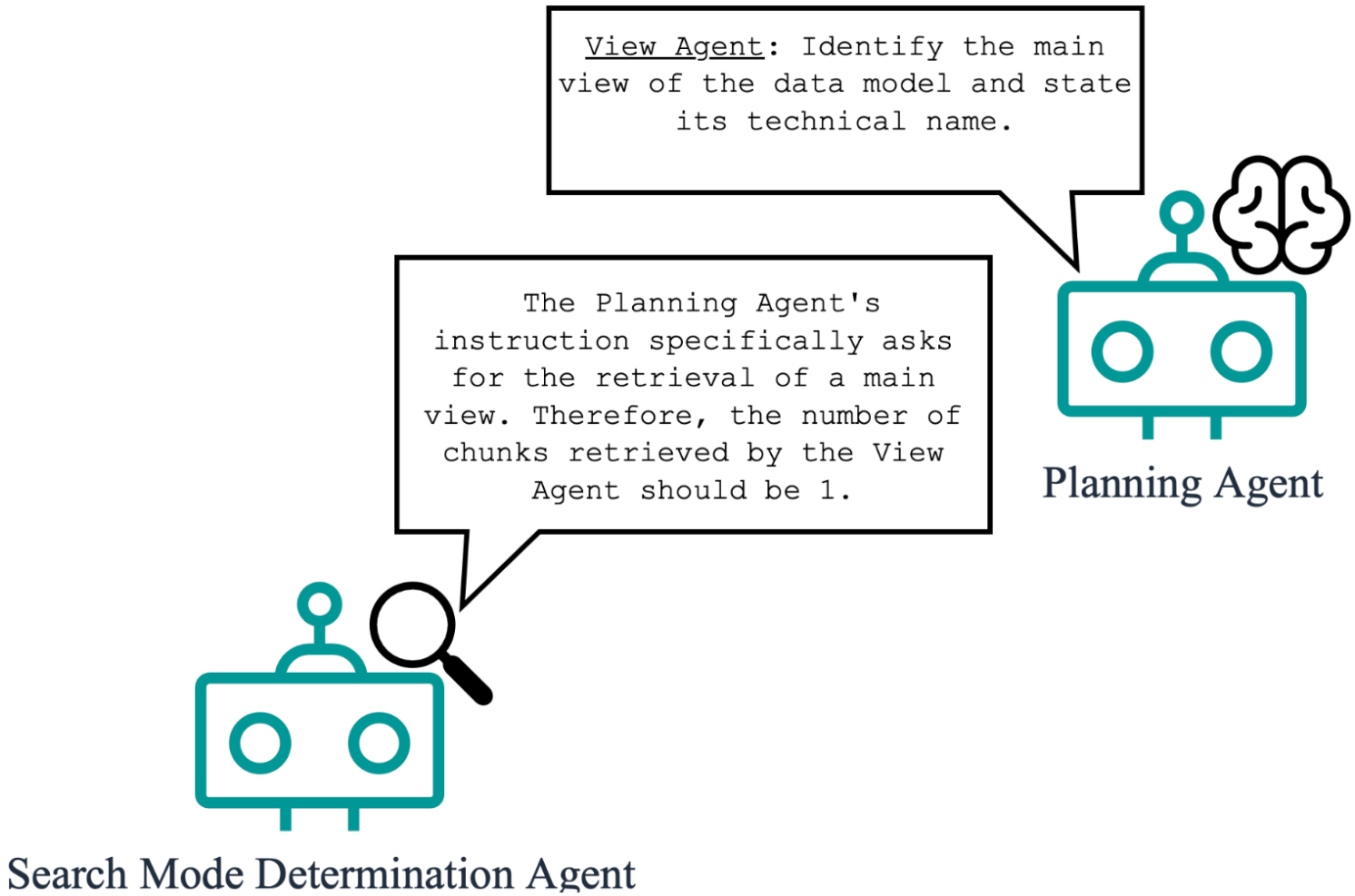


Figure 6

Functional Principle of the Search Mode Determination Agent

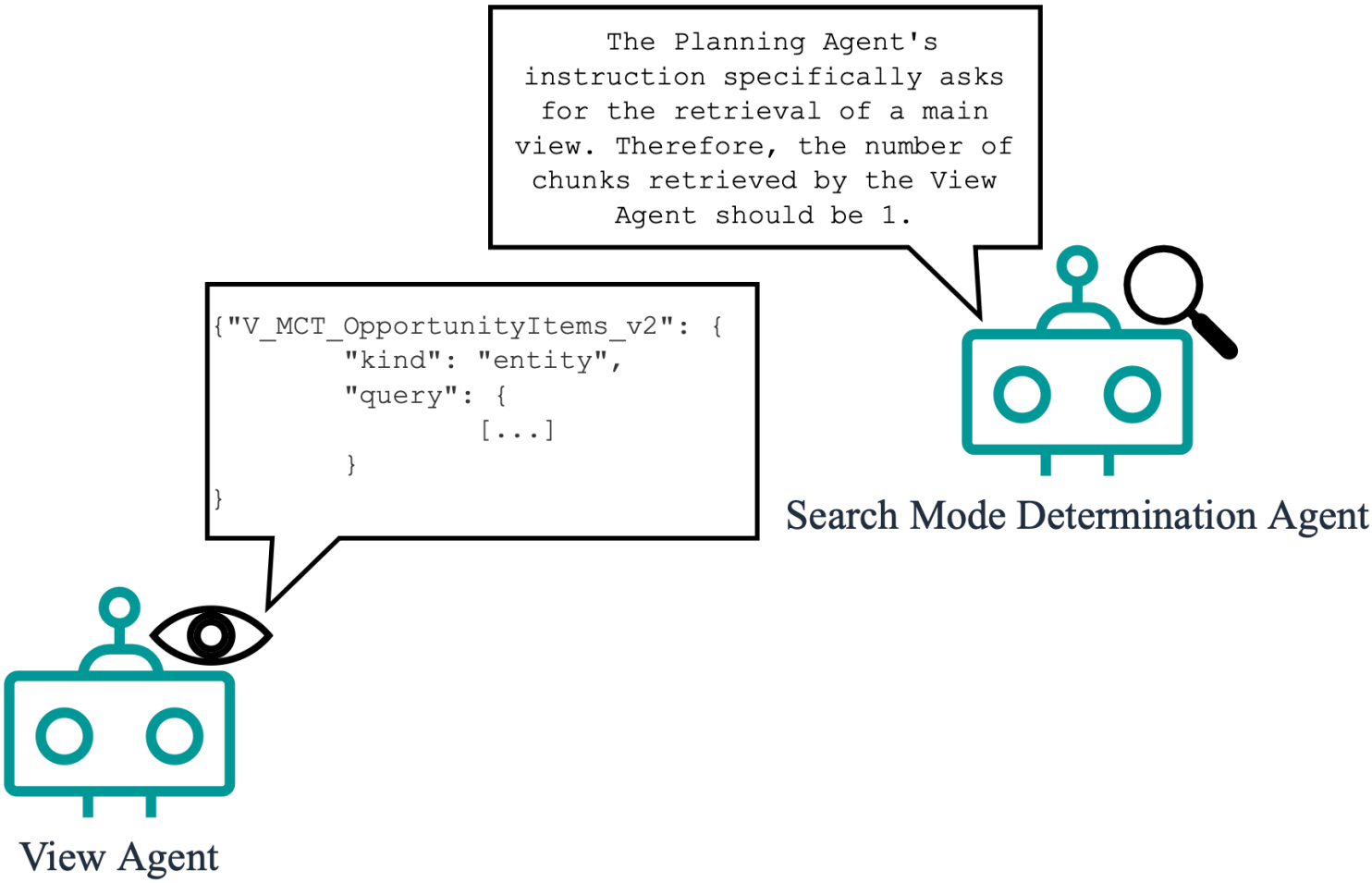


Figure 7

Functional Principle of the Worker Agents

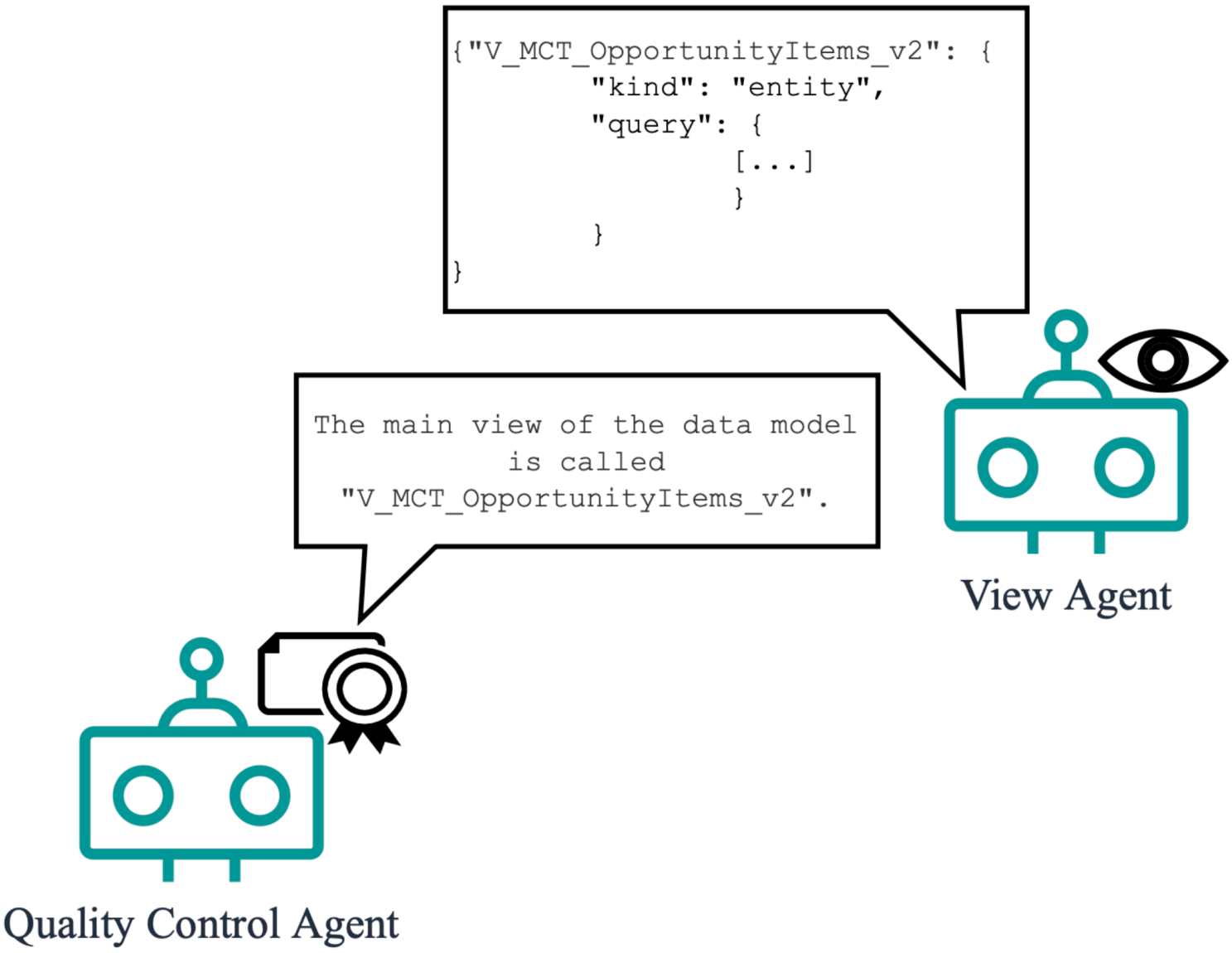


Figure 8

Functional Principle of the Quality Control Agent

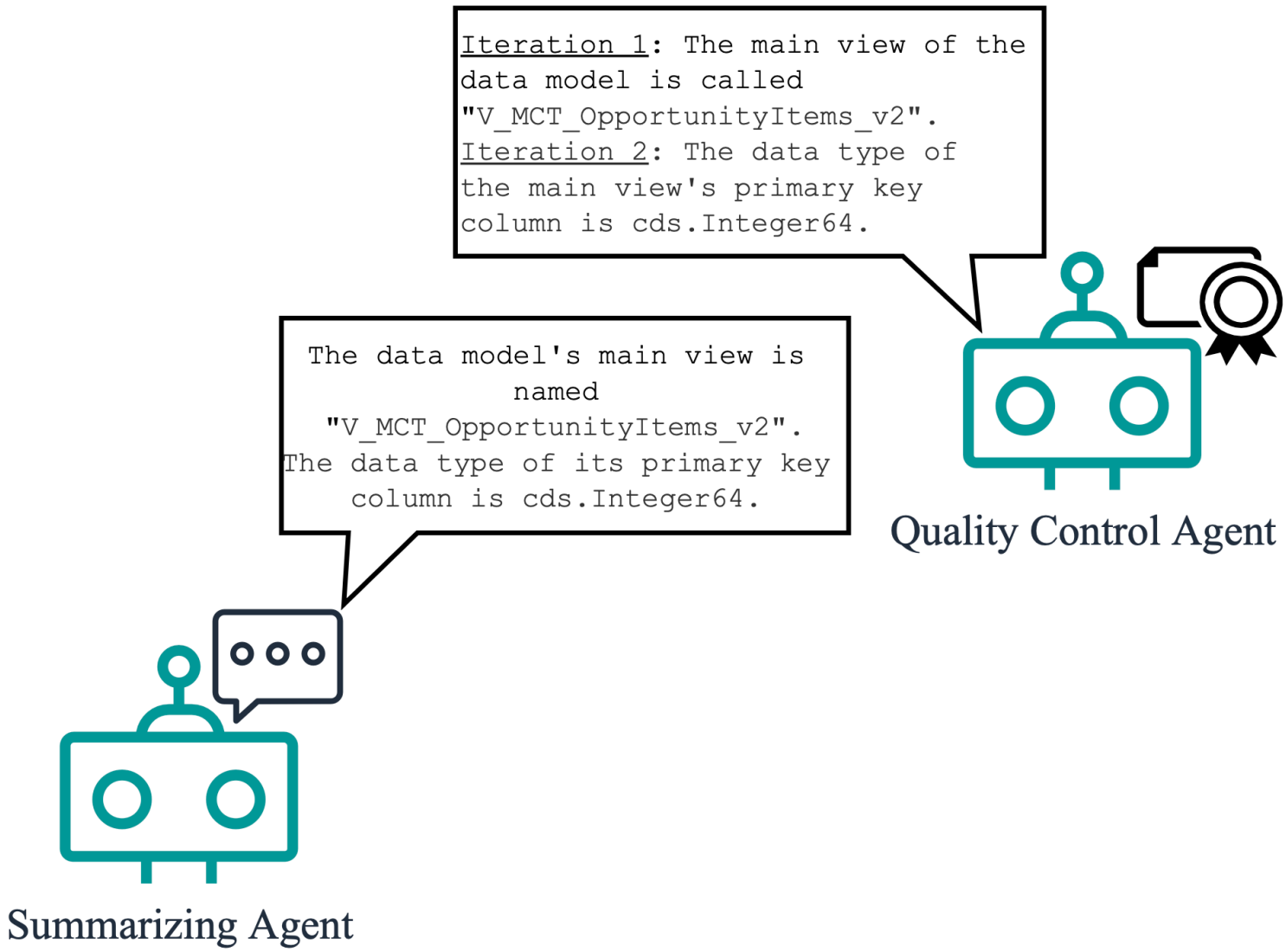


Figure 9

Functional Principle of the Summarizing Agent

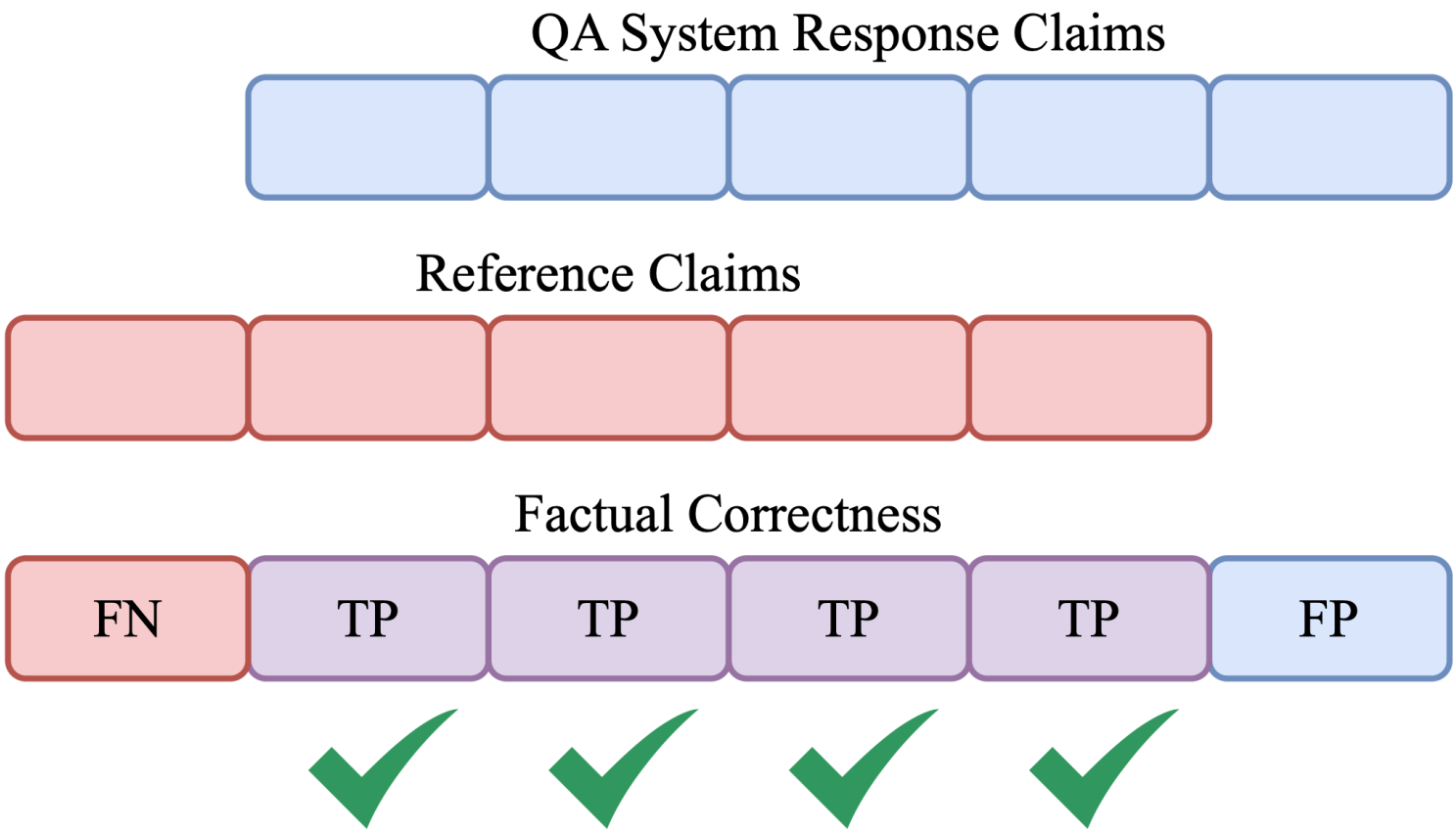


Figure 10

Determination of Factual Correctness (*RAGAs - Factual Correctness*, 2025)

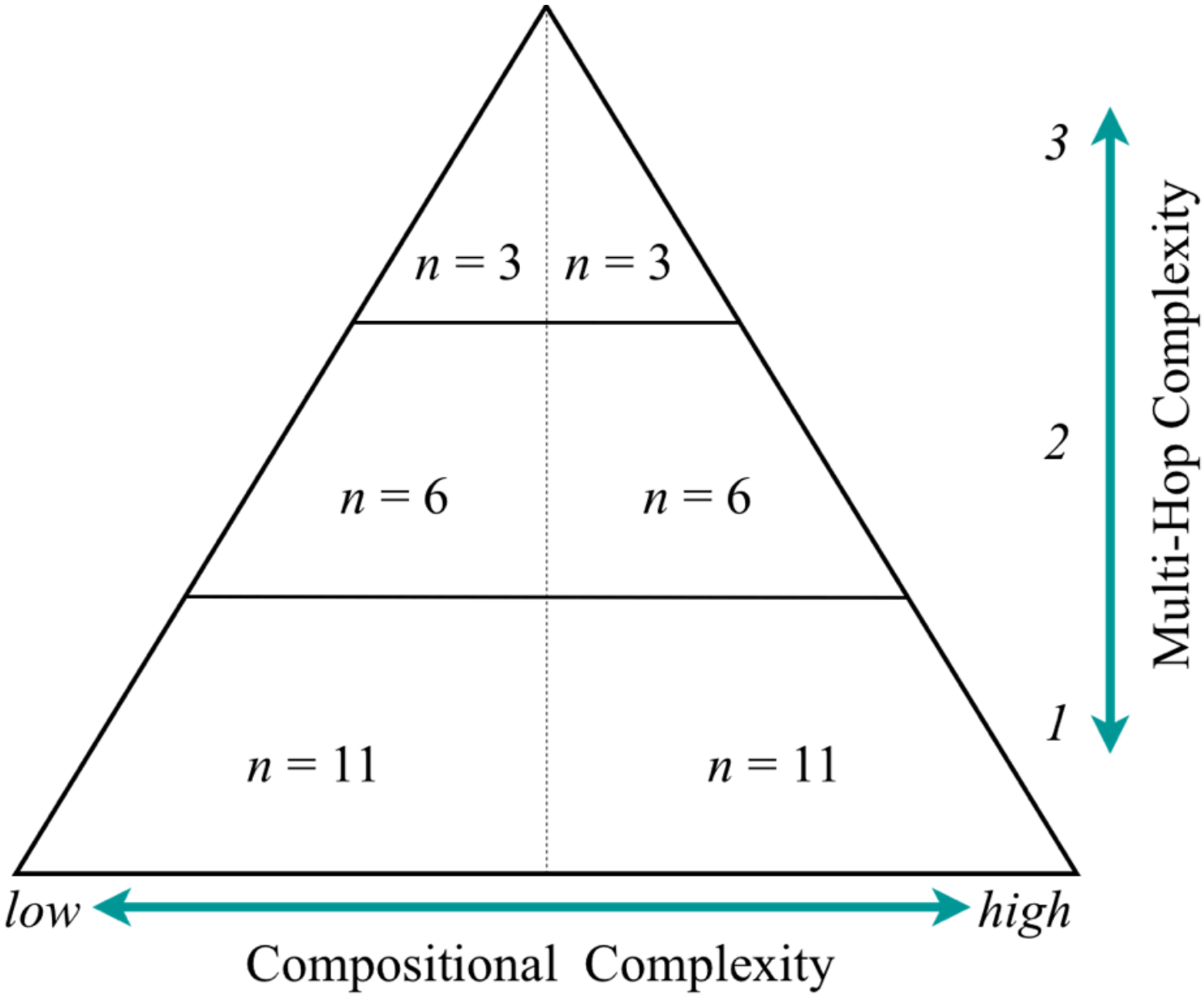


Figure 11

Complexity Dimensions of Evaluation Question Catalog