
AGENTIC RETRIEVAL-AUGMENTED GENERATION: A SURVEY ON AGENTIC RAG

Aditi Singh

Department of Computer Science
Cleveland State University
Cleveland, OH, USA
a.singh22@csuohio.edu

Abul Ehtesham

The Davey Tree Expert Company
Kent, OH, USA
abul.ehtesham@davey.com

Saket Kumar

The MathWorks Inc
Natick, MA, USA
saketk@mathworks.com

Tala Talaei Khoei

Khoury College of Computer Science
Roux Institute at Northeastern University
Portland, ME, USA
t.talaeikhoei@northeastern.edu

ABSTRACT

Large Language Models (LLMs) have revolutionized artificial intelligence (AI) by enabling human-like text generation and natural language understanding. However, their reliance on static training data limits their ability to respond to dynamic, real-time queries, resulting in outdated or inaccurate outputs. Retrieval-Augmented Generation (RAG) has emerged as a solution, enhancing LLMs by integrating real-time data retrieval to provide contextually relevant and up-to-date responses. Despite its promise, traditional RAG systems are constrained by static workflows and lack the adaptability required for multi-step reasoning and complex task management.

Agentic Retrieval-Augmented Generation (Agentic RAG) transcends these limitations by embedding autonomous AI agents into the RAG pipeline. These agents leverage agentic design patterns reflection, planning, tool use, and multi-agent collaboration to dynamically manage retrieval strategies, iteratively refine contextual understanding, and adapt workflows through clearly defined operational structures ranging from sequential steps to adaptive collaboration. This integration enables Agentic RAG systems to deliver unparalleled flexibility, scalability, and context-awareness across diverse applications.

This survey provides a comprehensive exploration of Agentic RAG, beginning with its foundational principles and the evolution of RAG paradigms. It presents a detailed taxonomy of Agentic RAG architectures, highlights key applications in industries such as healthcare, finance, and education, and examines practical implementation strategies. Additionally, it addresses challenges in scaling these systems, ensuring ethical decision-making, and optimizing performance for real-world applications, while providing detailed insights into frameworks and tools for implementing Agentic RAG¹. The GitHub link for this survey is available at: <https://github.com/asinghcsu/AgenticRAG-Survey>.

Keywords Large Language Models (LLMs) · Artificial Intelligence (AI) · Natural Language Understanding · Retrieval-Augmented Generation (RAG) · Agentic RAG · Autonomous AI Agents · Reflection · Planning · Tool Use · Multi-Agent Collaboration · Agentic Patterns · Contextual Understanding · Dynamic Adaptability · Scalability · Real-Time Data Retrieval · Taxonomy of Agentic RAG · Healthcare Applications · Finance Applications · Educational Applications · Ethical AI Decision-Making · Performance Optimization · Multi-Step Reasoning

¹GitHub link: <https://github.com/asinghcsu/AgenticRAG-Survey>

1 Introduction

Large Language Models (LLMs) [1, 2] [3], such as OpenAI’s GPT-4, Google’s PaLM, and Meta’s LLaMA, have significantly transformed artificial intelligence (AI) with their ability to generate human-like text and perform complex natural language processing tasks. These models have driven innovation across diverse domains, including conversational agents [4], automated content creation, and real-time translation. Recent advancements have extended their capabilities to multimodal tasks, such as text-to-image and text-to-video generation [5], enabling the creation and editing of videos and images from detailed prompts [6], which broadens the potential applications of generative AI.

Despite these advancements, LLMs face significant limitations due to their reliance on static pre-training data. This reliance often results in outdated information, hallucinated responses [7], and an inability to adapt to dynamic, real-world scenarios. These challenges emphasize the need for systems that can integrate real-time data and dynamically refine responses to maintain contextual relevance and accuracy.

Retrieval-Augmented Generation (RAG) [8, 9] emerged as a promising solution to these challenges. By combining the generative capabilities of LLMs with external retrieval mechanisms [10], RAG systems enhance the relevance and timeliness of responses. These systems retrieve real-time information from sources such as knowledge bases [11], APIs, or the web, effectively bridging the gap between static training data and the demands of dynamic applications. However, traditional RAG workflows remain limited by their linear and static design, which restricts their ability to perform complex multi-step reasoning, integrate deep contextual understanding, and iteratively refine responses.

The evolution of agents [12] has significantly enhanced the capabilities of AI systems. Modern agents, including LLM-powered and mobile agents [13], are intelligent entities capable of perceiving, reasoning, and autonomously executing tasks. These agents leverage agentic patterns, such as reflection [14], planning [15], tool use, and multi-agent collaboration [16], to enhance decision-making and adaptability.

Furthermore, these agents employ agentic workflow patterns [12, 13], such as prompt chaining, routing, parallelization, orchestrator-worker models, and evaluator-optimizer , to structure and optimize task execution. By integrating these patterns, Agentic RAG systems can efficiently manage dynamic workflows and address complex problem-solving scenarios. The convergence of RAG and agentic intelligence has given rise to Agentic Retrieval-Augmented Generation (Agentic RAG) [14], a paradigm that integrates agents into the RAG pipeline. Agentic RAG enables dynamic retrieval strategies, contextual understanding, and iterative refinement [15], allowing for adaptive and efficient information processing. Unlike traditional RAG, Agentic RAG employs autonomous agents to orchestrate retrieval, filter relevant information, and refine responses, excelling in scenarios requiring precision and adaptability. The overview of Agentic RAG is in figure 1.

This survey explores the foundational principles, taxonomy, and applications of Agentic RAG. It provides a comprehensive overview of RAG paradigms, such as Naïve RAG, Modular RAG, and Graph RAG [16], alongside their evolution into Agentic RAG systems. Key contributions include a detailed taxonomy of Agentic RAG frameworks, applications across domains such as healthcare [17, 18], finance, and education [19], and insights into implementation strategies, benchmarks, and ethical considerations.

The structure of this paper is as follows: Section 2 introduces RAG and its evolution, highlighting the limitations of traditional approaches. Section 3 elaborates on the principles of agentic intelligence and agentic patterns. Section 4 elaborates agentic workflow patterns. Section 5 provides a taxonomy of Agentic RAG systems, including single-agent, multi-agent, and graph-based frameworks. Section 6 examines applications of Agentic RAG, while Section 7 discusses implementation tools and frameworks. Section 8 focuses on benchmarks and dataset, and Section 9 concludes with future directions for Agentic RAG systems.

2 Foundations of Retrieval-Augmented Generation

2.1 Overview of Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) represents a significant advancement in the field of artificial intelligence, combining the generative capabilities of Large Language Models (LLMs) with real-time data retrieval. While LLMs have demonstrated remarkable capabilities in natural language processing, their reliance on static pre-trained data often results in outdated or incomplete responses. RAG addresses this limitation by dynamically retrieving relevant information from external sources and incorporating it into the generative process, enabling contextually accurate and up-to-date outputs.

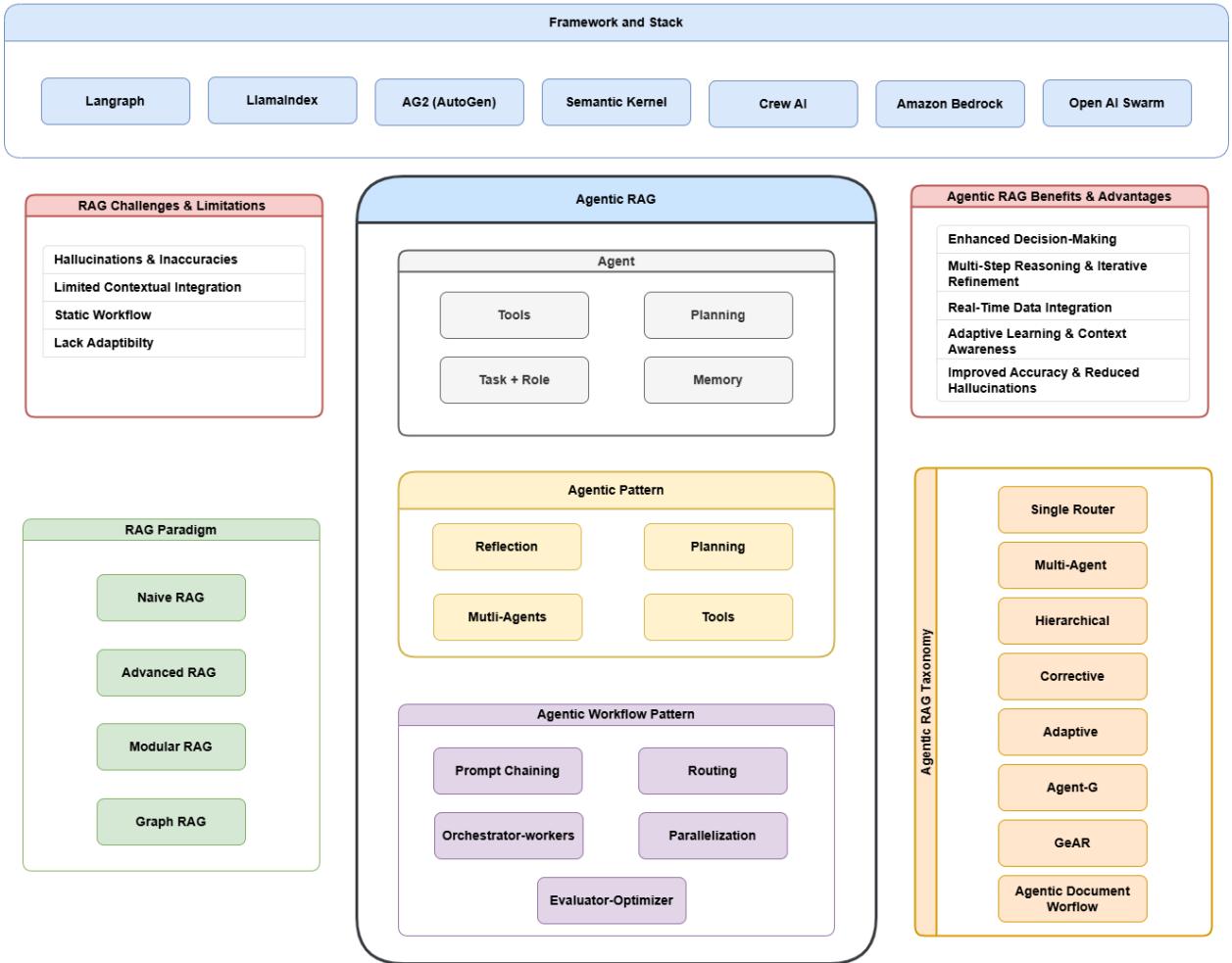


Figure 1: An Overview of Agentic RAG

2.2 Core Components of RAG

The architecture of RAG systems integrates three primary components (Figure2):

- **Retrieveal:** Responsible for querying external data sources such as knowledge bases, APIs, or vector databases. Advanced retrievers leverage dense vector search and transformer-based models to improve retrieval precision and semantic relevance.
- **Augmentation:** Processes retrieved data, extracting and summarizing the most relevant information to align with the query context.
- **Generation:** Combines retrieved information with the LLM's pre-trained knowledge to generate coherent, contextually appropriate responses.

2.3 Evolution of RAG Paradigms

The field of Retrieval-Augmented Generation (RAG) has evolved significantly to address the increasing complexity of real-world applications, where contextual accuracy, scalability, and multi-step reasoning are critical. What began as simple keyword-based retrieval has transitioned into sophisticated, modular, and adaptive systems capable of integrating diverse data sources and autonomous decision-making processes. This evolution underscores the growing need for RAG systems to handle complex queries efficiently and effectively.

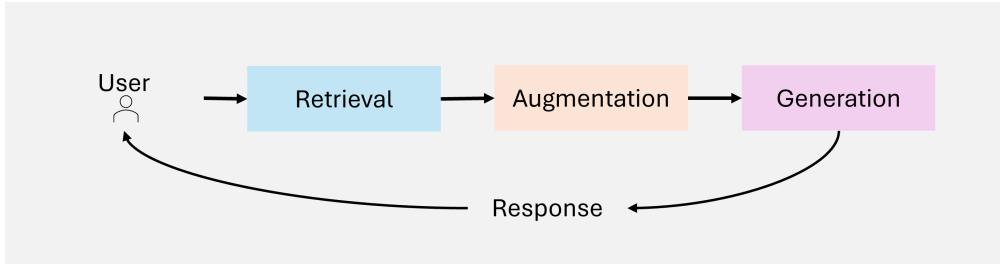


Figure 2: Core Components of RAG

This section examines the progression of RAG paradigms, presenting key stages of development—Naïve RAG, Advanced RAG, Modular RAG, Graph RAG, and Agentic RAG alongside their defining characteristics, strengths, and limitations. By understanding the evolution of these paradigms, readers can appreciate the advancements made in retrieval and generative capabilities and their application in various domains.

2.3.1 Naïve RAG

Naïve RAG [20] represents the foundational implementation of retrieval-augmented generation. Figure 3 illustrates the simple retrieve-read workflow of Naive RAG, focusing on keyword-based retrieval and static datasets.. These systems rely on simple keyword-based retrieval techniques, such as TF-IDF and BM25, to fetch documents from static datasets. The retrieved documents are then used to augment the language model’s generative capabilities.

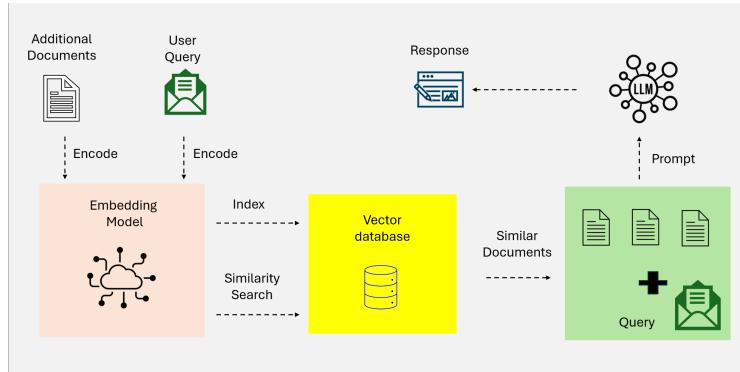


Figure 3: An Overview of Naive RAG.

Naïve RAG is characterized by its simplicity and ease of implementation, making it suitable for tasks involving fact-based queries with minimal contextual complexity. However, it suffers from several limitations:

- **Lack of Contextual Awareness:** Retrieved documents often fail to capture the semantic nuances of the query due to reliance on lexical matching rather than semantic understanding.
- **Fragmented Outputs:** The absence of advanced preprocessing or contextual integration often leads to disjointed or overly generic responses.
- **Scalability Issues:** Keyword-based retrieval techniques struggle with large datasets, often failing to identify the most relevant information.

Despite these limitations, Naïve RAG systems provided a critical proof-of-concept for integrating retrieval with generation, laying the foundation for more sophisticated paradigms.

2.3.2 Advanced RAG

Advanced RAG [20] systems build upon the limitations of Naïve RAG by incorporating semantic understanding and enhanced retrieval techniques. Figure 4 highlights the semantic enhancements in retrieval and the iterative, context-aware pipeline of Advanced RAG. These systems leverage dense retrieval models, such as Dense Passage Retrieval (DPR), and neural ranking algorithms to improve retrieval precision.

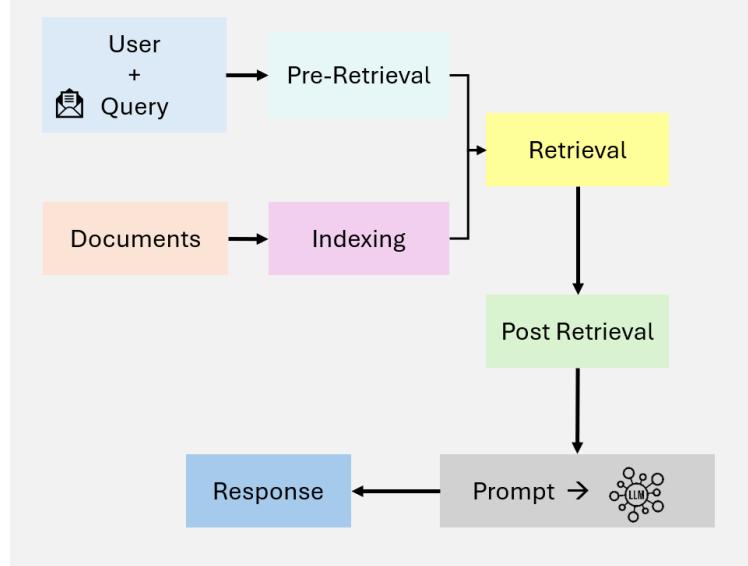


Figure 4: Overview of Advanced RAG

Key features of Advanced RAG include:

- **Dense Vector Search:** Queries and documents are represented in high-dimensional vector spaces, enabling better semantic alignment between the user query and retrieved documents.
- **Contextual Re-Ranking:** Neural models re-rank retrieved documents to prioritize the most contextually relevant information.
- **Iterative Retrieval:** Advanced RAG introduces multi-hop retrieval mechanisms, enabling reasoning across multiple documents for complex queries.

These advancements make Advanced RAG suitable for applications requiring high precision and nuanced understanding, such as research synthesis and personalized recommendations. However, challenges such as computational overhead and limited scalability persist, particularly when dealing with large datasets or multi-step queries.

2.3.3 Modular RAG

Modular RAG [20] represents the latest evolution in RAG paradigms, emphasizing flexibility and customization. These systems decompose the retrieval and generation pipeline into independent, reusable components, enabling domain-specific optimization and task adaptability. Figure 5 demonstrates the modular architecture, showcasing hybrid retrieval strategies, composable pipelines, and external tool integration.

Key innovations in Modular RAG include:

- **Hybrid Retrieval Strategies:** Combining sparse retrieval methods (e.g., a sparse encoder-BM25) with dense retrieval techniques [21] (e.g., DPR - Dense Passage Retrieval) to maximize accuracy across diverse query types.
- **Tool Integration:** Incorporating external APIs, databases, or computational tools to handle specialized tasks, such as real-time data analysis or domain-specific computations.
- **Composable Pipelines:** Modular RAG enables retrievers, generators, and other components to be replaced, enhanced, or reconfigured independently, allowing high adaptability to specific use cases.

For instance, a Modular RAG system designed for financial analytics might retrieve live stock prices via APIs, analyze historical trends using dense retrieval, and generate actionable investment insights through a tailored language model. This modularity and customization make Modular RAG ideal for complex, multi-domain tasks, offering both scalability and precision.

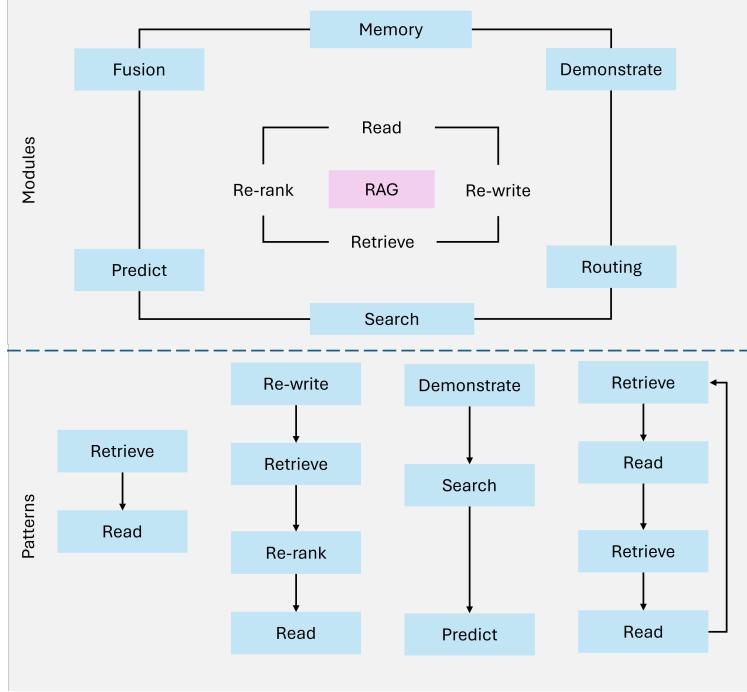


Figure 5: Overview of Modular RAG

2.3.4 Graph RAG

Graph RAG [16] extends traditional Retrieval-Augmented Generation systems by integrating graph-based data structures as illustrated in Figure 6. These systems leverage the relationships and hierarchies within graph data to enhance multi-hop reasoning and contextual enrichment. By incorporating graph-based retrieval, Graph RAG enables richer and more accurate generative outputs, particularly for tasks requiring relational understanding.

Graph RAG is characterized by its ability to:

- **Node Connectivity:** Captures and reasons over relationships between entities.
- **Hierarchical Knowledge Management:** Handles structured and unstructured data through graph-based hierarchies.
- **Context Enrichment:** Adds relational understanding by leveraging graph-based pathways.

However, Graph RAG has some limitations:

- **Limited Scalability:** The reliance on graph structures can restrict scalability, especially with extensive data sources.
- **Data Dependency:** High-quality graph data is essential for meaningful outputs, limiting its applicability in unstructured or poorly annotated datasets.
- **Complexity of Integration:** Integrating graph data with unstructured retrieval systems increases design and implementation complexity.

Graph RAG is well-suited for applications such as healthcare diagnostics, legal research, and other domains where reasoning over structured relationships is crucial.

2.3.5 Agentic RAG

Agentic RAG represents a paradigm shift by introducing autonomous agents capable of dynamic decision-making and workflow optimization. Unlike static systems, Agentic RAG employs iterative refinement and adaptive retrieval strategies to address complex, real-time, and multi-domain queries. This paradigm leverages the modularity of retrieval and generation processes while introducing agent-based autonomy.

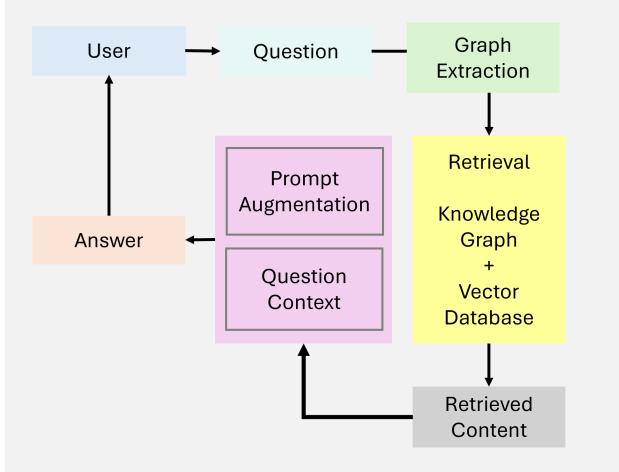


Figure 6: Overview of Graph RAG

Key characteristics of Agentic RAG include:

- **Autonomous Decision-Making:** Agents independently evaluate and manage retrieval strategies based on query complexity.
- **Iterative Refinement:** Incorporates feedback loops to improve retrieval accuracy and response relevance.
- **Workflow Optimization:** Dynamically orchestrates tasks, enabling efficiency in real-time applications.

Despite its advancements, Agentic RAG faces some challenges:

- **Coordination Complexity:** Managing interactions between agents requires sophisticated orchestration mechanisms.
- **Computational Overhead:** The use of multiple agents increases resource requirements for complex workflows.
- **Scalability Limitations:** While scalable, the dynamic nature of the system can strain computational resources for high query volumes.

Agentic RAG excels in domains like customer support, financial analytics, and adaptive learning platforms, where dynamic adaptability and contextual precision are paramount.

2.4 Challenges and Limitations of Traditional RAG Systems

Traditional Retrieval-Augmented Generation (RAG) systems have significantly expanded the capabilities of Large Language Models (LLMs) by integrating real-time data retrieval. However, these systems still face critical challenges that hinder their effectiveness in complex, real-world applications. The most notable limitations revolve around **contextual integration, multi-step reasoning, and scalability and latency issues**.

2.4.1 Contextual Integration

Even when RAG systems successfully retrieve relevant information, they often struggle to seamlessly incorporate it into generated responses. The static nature of retrieval pipelines and limited contextual awareness lead to fragmented, inconsistent, or overly generic outputs.

Example: A query such as, *"What are the latest advancements in Alzheimer's research and their implications for early-stage treatment?"* might yield relevant research papers and medical guidelines. However, traditional RAG systems often fail to synthesize these findings into a coherent explanation that connects the new treatments to specific patient scenarios. Similarly, for a query like, *"What are the best sustainable practices for small-scale agriculture in arid regions?"*, traditional systems might retrieve documents on general agricultural methods but overlook critical sustainability practices tailored to arid environments.

Table 1: Comparative Analysis of RAG Paradigms

Paradigm	Key Features	Strengths
Naïve RAG	<ul style="list-style-type: none"> Keyword-based retrieval (e.g., TF-IDF, BM25) 	<ul style="list-style-type: none"> Simple and easy to implement Suitable for fact-based queries
Advanced RAG	<ul style="list-style-type: none"> Dense retrieval models (e.g., DPR) Neural ranking and re-ranking Multi-hop retrieval 	<ul style="list-style-type: none"> High precision retrieval Improved contextual relevance
Modular RAG	<ul style="list-style-type: none"> Hybrid retrieval (sparse and dense) Tool and API integration Composable, domain-specific pipelines 	<ul style="list-style-type: none"> High flexibility and customization Suitable for diverse applications Scalable
Graph RAG	<ul style="list-style-type: none"> Integration of graph-based structures Multi-hop reasoning Contextual enrichment via nodes 	<ul style="list-style-type: none"> Relational reasoning capabilities Mitigates hallucinations Ideal for structured data tasks
Agentic RAG	<ul style="list-style-type: none"> Autonomous agents Dynamic decision-making Iterative refinement and workflow optimization 	<ul style="list-style-type: none"> Adaptable to real-time changes Scalable for multi-domain tasks High accuracy

2.4.2 Multi-Step Reasoning

Many real-world queries require iterative or multi-hop reasoning—retrieving and synthesizing information across multiple steps. Traditional RAG systems are often ill-equipped to refine retrieval based on intermediate insights or user feedback, resulting in incomplete or disjointed responses.

Example: A complex query like, "*What lessons from renewable energy policies in Europe can be applied to developing nations, and what are the potential economic impacts?*" demands the orchestration of multiple types of information, including policy data, contextualization for developing regions, and economic analysis. Traditional RAG systems typically fail to connect these disparate elements into a cohesive response.

2.4.3 Scalability and Latency Issues

As the volume of external data sources grows, querying and ranking large datasets becomes increasingly computationally intensive. This results in significant latency, which undermines the system's ability to provide timely responses in real-time applications.

Example: In time-sensitive settings such as *financial analytics* or *live customer support*, delays caused by querying multiple databases or processing large document sets can hinder the system's overall utility. For example, a delay in retrieving market trends during high-frequency trading could result in missed opportunities.

2.5 Agentic RAG: A Paradigm Shift

Traditional RAG systems, with their static workflows and limited adaptability, often struggle to handle dynamic, multi-step reasoning and complex real-world tasks. These limitations have spurred the integration of agentic intelligence,

resulting in Agentic RAG. By incorporating autonomous agents capable of dynamic decision-making, iterative reasoning, and adaptive retrieval strategies, Agentic RAG builds on the modularity of earlier paradigms while overcoming their inherent constraints. This evolution enables more complex, multi-domain tasks to be addressed with enhanced precision and contextual understanding, positioning Agentic RAG as a cornerstone for next-generation AI applications. In particular, Agentic RAG systems reduce latency through optimized workflows and refine outputs iteratively, tackling the very challenges that have historically hindered traditional RAG's scalability and effectiveness.

3 Core Principles and Background of Agentic Intelligence

Agentic Intelligence forms the foundation of Agentic Retrieval-Augmented Generation (RAG) systems, enabling them to transcend the static and reactive nature of traditional RAG. By integrating autonomous agents capable of dynamic decision-making, iterative reasoning, and collaborative workflows, Agentic RAG systems exhibit enhanced adaptability and precision. This section explores the core principles underpinning agentic intelligence.

Components of an AI Agent. In essence, an AI agent comprises (Figure. 7):

- **LLM (with defined Role and Task):** Serves as the agent's primary reasoning engine and dialogue interface. It interprets user queries, generates responses, and maintains coherence.
- **Memory (Short-Term and Long-Term):** Captures context and relevant data across interactions. Short-term memory [22] tracks immediate conversation state, while long-term memory [22] stores accumulated knowledge and agent experiences.
- **Planning (Reflection & Self-Critique):** Guides the agent's iterative reasoning process through reflection, query routing, or self-critique[23], ensuring that complex tasks are broken down effectively [24].
- **Tools Vector Search, Web Search, APIs, etc.:** Expands the agent's capabilities beyond text generation, enabling access to external resources, real-time data, or specialized computations.

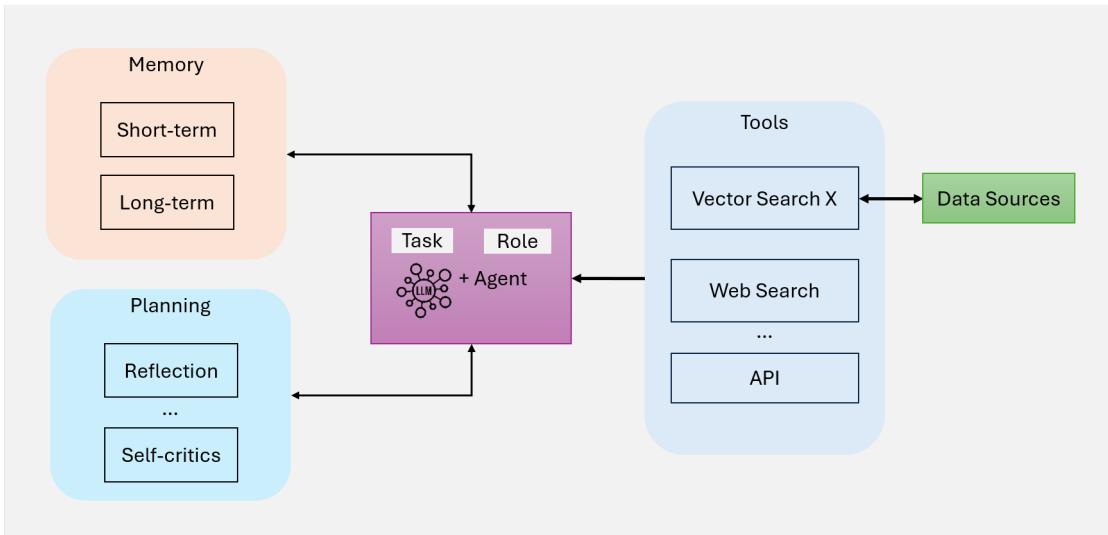


Figure 7: An Overview of AI Agents

Agentic Patterns [25, 26] provide structured methodologies that guide the behavior of agents in Agentic Retrieval-Augmented Generation (RAG) systems. These patterns enable agents to dynamically adapt, plan, and collaborate, ensuring that the system can handle complex, real-world tasks with precision and scalability. Four key patterns underpin agentic workflows:

3.0.1 Reflection

Reflection is a foundational design pattern in agentic workflows, enabling agents to iteratively evaluate and refine their outputs. By incorporating self-feedback mechanisms, agents can identify and address errors, inconsistencies, and areas for improvement, enhancing performance across tasks like code generation, text production, and question answering (as shown in Figure 8). In practical use, Reflection involves prompting an agent to critique its outputs for correctness,

style, and efficiency, then incorporating this feedback into subsequent iterations. External tools, such as unit tests or web searches, can further enhance this process by validating results and highlighting gaps.

In multi-agent systems, Reflection can involve distinct roles, such as one agent generating outputs while another critiques them, fostering collaborative improvement. For instance, in legal research, agents can iteratively refine responses by re-evaluating retrieved case law, ensuring accuracy and comprehensiveness. Reflection has demonstrated significant performance improvements in studies like *Self-Refine* [27], *Reflexion* [28], and *CRITIC* [23].

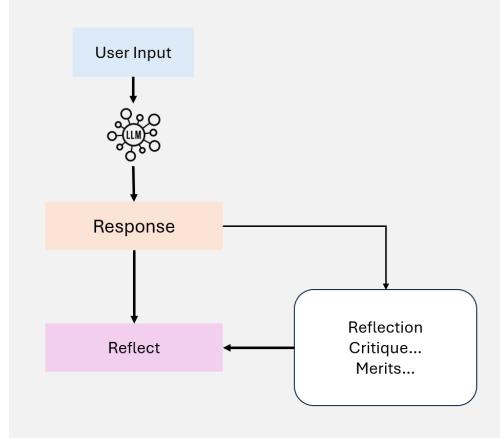


Figure 8: An Overview of Agentic Self- Reflection

3.0.2 Planning

Planning [24] is a key design pattern in agentic workflows that enables agents to autonomously decompose complex tasks into smaller, manageable subtasks. This capability is essential for multi-hop reasoning and iterative problem-solving in dynamic and uncertain scenarios as shown in Figure 9a.

By leveraging planning, agents can dynamically determine the sequence of steps needed to accomplish a larger objective. This adaptability allows agents to handle tasks that cannot be predefined, ensuring flexibility in decision-making. While powerful, Planning can produce less predictable outcomes compared to deterministic workflows like Reflection. Planning is particularly suited for tasks that require dynamic adaptation, where predefined workflows are insufficient. As the technology matures, its potential to drive innovative applications across domains will continue to grow.

3.0.3 Tool Use

Tool Use enables agents to extend their capabilities by interacting with external tools, APIs, or computational resources as illustrated in 9b. This pattern allows agents to gather information, perform computations, and manipulate data beyond their pre-trained knowledge. By dynamically integrating tools into workflows, agents can adapt to complex tasks and provide more accurate and contextually relevant outputs.

Modern agentic workflows incorporate tool use for a variety of applications, including information retrieval, computational reasoning, and interfacing with external systems. The implementation of this pattern has evolved significantly with advancements like GPT-4's function calling capabilities and systems capable of managing access to numerous tools. These developments facilitate sophisticated workflows where agents autonomously select and execute the most relevant tools for a given task.

While tool use significantly enhances agentic workflows, challenges remain in optimizing the selection of tools, particularly in contexts with a large number of available options. Techniques inspired by retrieval-augmented generation (RAG), such as heuristic-based selection, have been proposed to address this issue.

3.0.4 Multi-Agent

Multi-agent collaboration [29] is a key design pattern in agentic workflows that enables task specialization and parallel processing. Agents communicate and share intermediate results, ensuring the overall workflow remains efficient and coherent. By distributing subtasks among specialized agents, this pattern improves the scalability and adaptability of complex workflows. Multi-agent systems allow developers to decompose intricate tasks into smaller, manageable

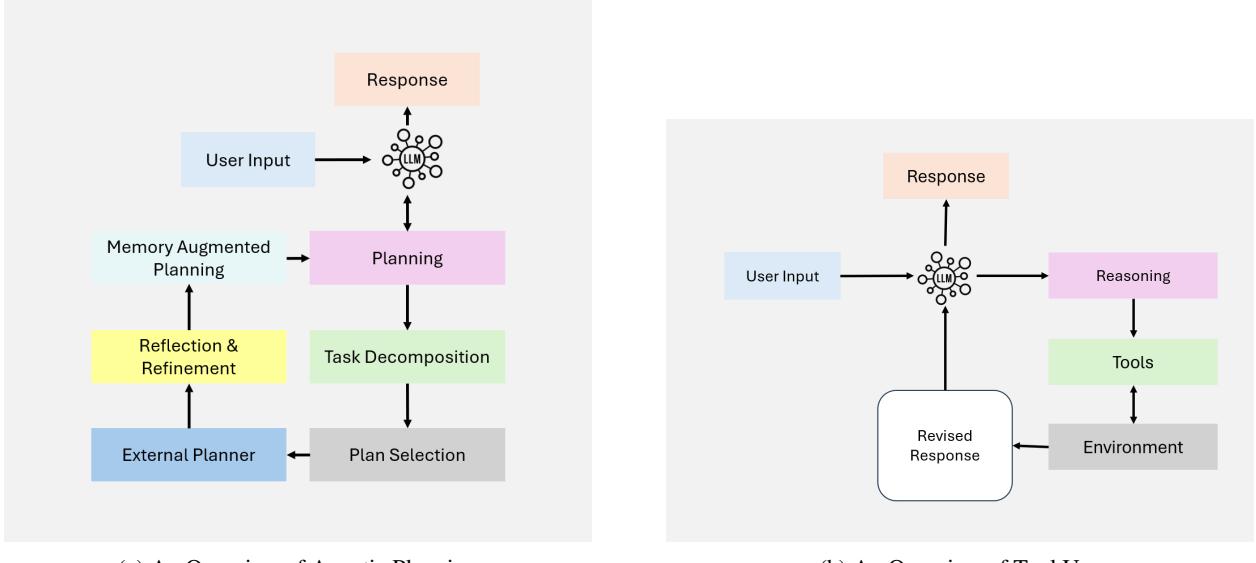


Figure 9: Overview of Agentic Planning and Tool Use

subtasks assigned to different agents. This approach not only enhances task performance but also provides a robust framework for managing complex interactions. Each agent operates with its own memory and workflow, which can include the use of tools, reflection, or planning, enabling dynamic and collaborative problem-solving (see Figure 10).

While multi-agent collaboration offers significant potential, it is a less predictable design pattern compared to more mature workflows like Reflection and Tool Use. Nevertheless, emerging frameworks such as AutoGen, Crew AI, and LangGraph are providing new avenues for implementing effective multi-agent solutions.

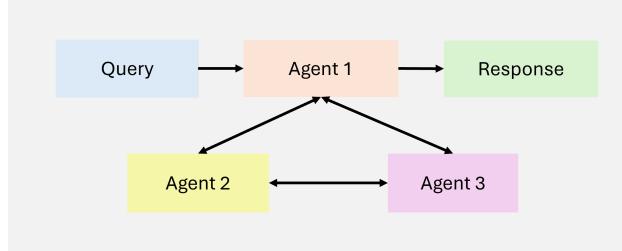


Figure 10: An Overview of MultiAgent

These design patterns form the foundation for the success of Agentic RAG systems. By structuring workflows—from simple, sequential steps to more adaptive, collaborative processes—these patterns enable systems to dynamically adapt their retrieval and generative strategies to the diverse and ever-changing demands of real-world environments. Leveraging these patterns, agents are capable of handling iterative, context-aware tasks that significantly exceed the capabilities of traditional RAG systems.

4 Agentic Workflow Patterns: Adaptive Strategies for Dynamic Collaboration

Agentic workflow patterns, [12, 13] structure LLM-based applications to optimize performance, accuracy, and efficiency. Different approaches are suitable depending on task complexity and processing requirements.

4.1 Prompt Chaining: Enhancing Accuracy Through Sequential Processing

Prompt chaining [12, 13] decomposes a complex task into multiple steps, where each step builds upon the previous one. This structured approach improves accuracy by simplifying each subtask before moving forward. However, it may increase latency due to sequential processing.

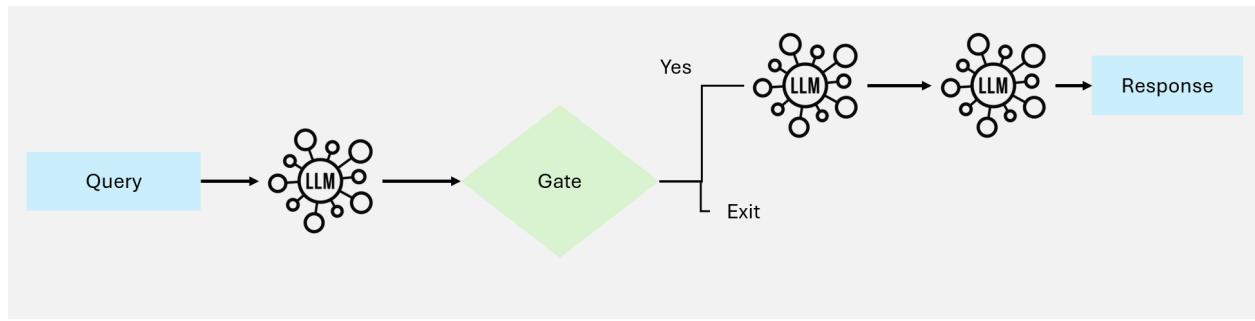


Figure 11: Illustration of Prompt Chaining Workflow

When to Use: This workflow is most effective when a task can be broken down into fixed subtasks, each contributing to the final output. It is particularly useful in scenarios where step-by-step reasoning enhances accuracy.

Example Applications:

- Generating marketing content in one language and then translating it into another while preserving nuances.
- Structuring document creation by first generating an outline, verifying its completeness, and then developing the full text.

4.2 Routing:Directing Inputs to Specialized Processes

Routing [12, 13] involves classifying an input and directing it to an appropriate specialized prompt or process. This method ensures distinct queries or tasks are handled separately, improving efficiency and response quality.

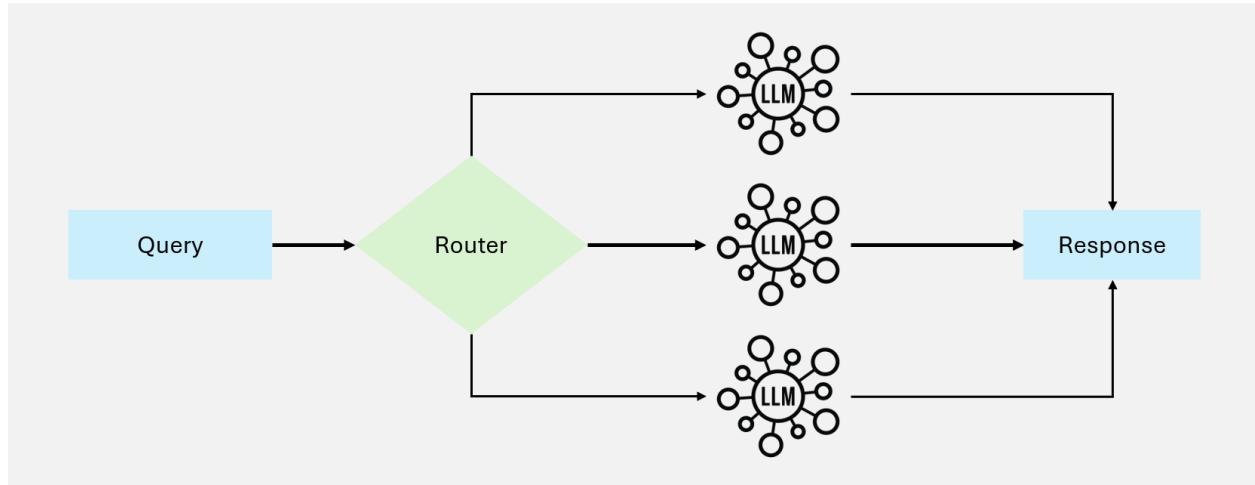


Figure 12: Illustration Routing Workflow

When to Use: Ideal for scenarios where different types of input require distinct handling strategies, ensuring optimized performance for each category.

Example Applications:

- Directing customer service queries into categories such as technical support, refund requests, or general inquiries.
- Assigning simple queries to smaller models for cost efficiency, while complex requests go to advanced models.

4.3 Parallelization: Speeding Up Processing Through Concurrent Execution

Parallelization [12, 13] divides a task into independent processes that run simultaneously, reducing latency and improving throughput. It can be categorized into sectioning (independent subtasks) and voting (multiple outputs for accuracy).

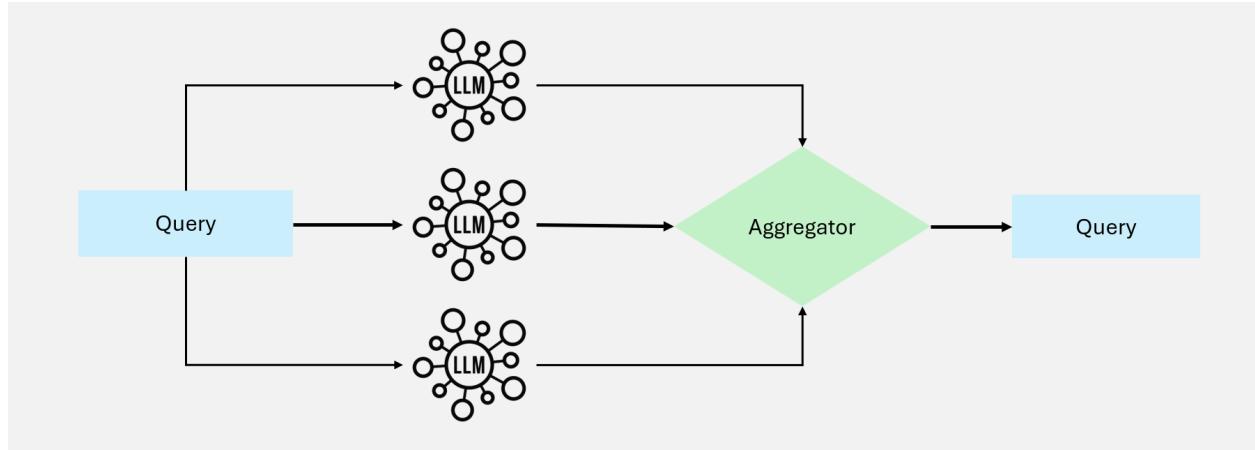


Figure 13: Illustration of Parallelization Workflow

When to Use: Useful when tasks can be executed independently to enhance speed or when multiple outputs improve confidence.

Example Applications:

- **Sectioning:** Splitting tasks like content moderation, where one model screens input while another generates a response.
- **Voting:** Using multiple models to cross-check code for vulnerabilities or analyze content moderation decisions.

4.4 Orchestrator-Workers: Dynamic Task Delegation

This workflow [12, 13] features a central orchestrator model that dynamically breaks tasks into subtasks, assigns them to specialized worker models, and compiles the results. Unlike parallelization, it adapts to varying input complexity.

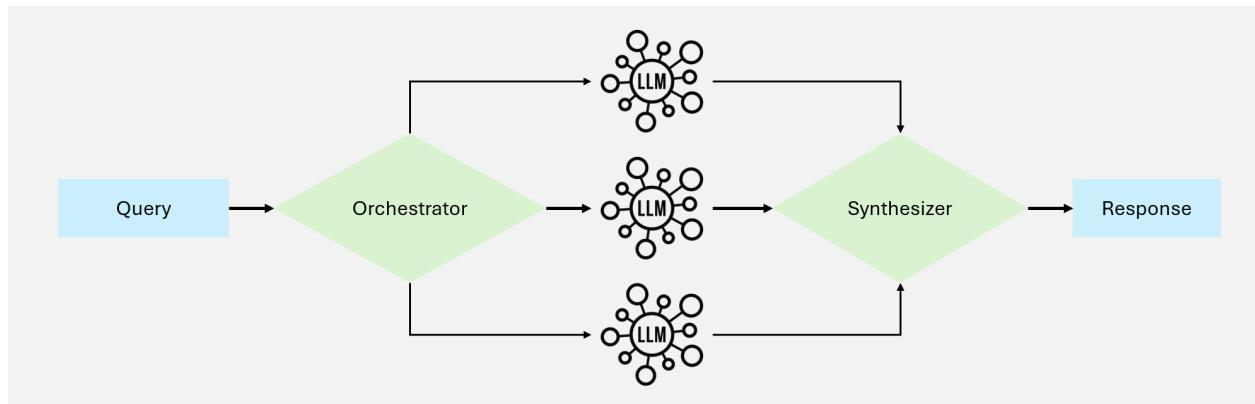


Figure 14: Illustration of Orchestrator-Workers Workflow

When to Use: Best suited for tasks requiring dynamic decomposition and real-time adaptation, where subtasks are not predefined.

Example Applications:

- Automatically modifying multiple files in a codebase based on the nature of requested changes.

- Conducting real-time research by gathering and synthesizing relevant information from multiple sources.

4.5 Evaluator-Optimizer: Refining Output Through Iteration

The evaluator-optimizer [12, 13] workflow iteratively improves content by generating an initial output and refining it based on feedback from an evaluation model.

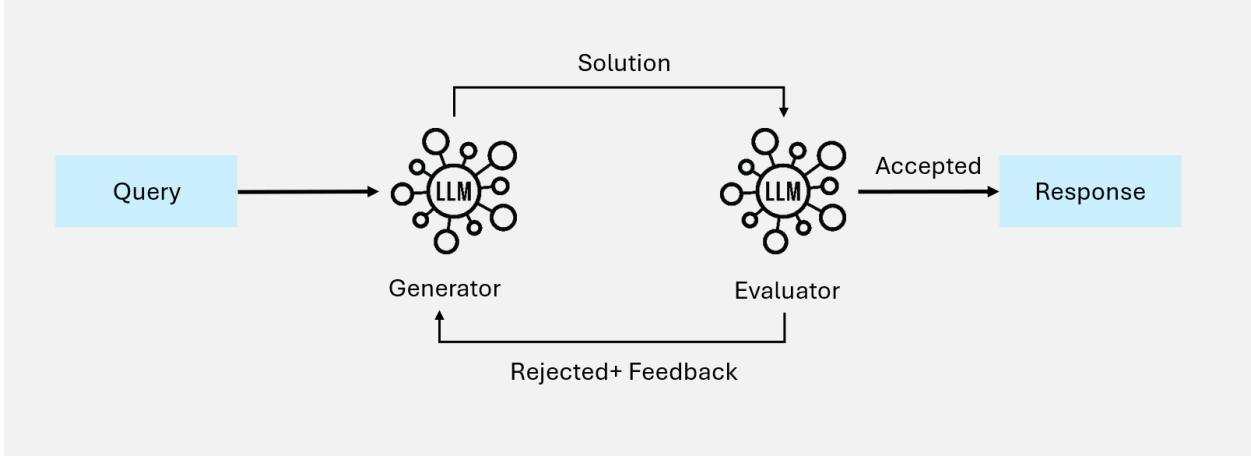


Figure 15: Illustration of Evaluator-Optimizer Workflow

When to Use: Effective when iterative refinement significantly enhances response quality, especially when clear evaluation criteria exist.

Example Applications:

- Improving literary translations through multiple evaluation and refinement cycles.
- Conducting multi-round research queries where additional iterations refine search results.

4.6 Taxonomy of Agentic RAG Systems

Agentic Retrieval-Augmented Generation (RAG) systems can be categorized into distinct architectural frameworks based on their complexity and design principles. These include single-agent architectures, multi-agent systems, and hierarchical agentic architectures. Each framework is tailored to address specific challenges and optimize performance for diverse applications. This section provides a detailed taxonomy of these architectures, highlighting their characteristics, strengths, and limitations.

4.6.1 Single-Agent Agentic RAG: Router

A **Single-Agent Agentic RAG**: [30] serves as a centralized decision-making system where a single agent manages the retrieval, routing, and integration of information (as shown in Figure. 16). This architecture simplifies the system by consolidating these tasks into one unified agent, making it particularly effective for setups with a limited number of tools or data sources.

Workflow

1. **Query Submission and Evaluation:** The process begins when a user submits a query. A coordinating agent (or master retrieval agent) receives the query and analyzes it to determine the most suitable sources of information.
2. **Knowledge Source Selection:** Based on the query's type, the coordinating agent chooses from a variety of retrieval options:
 - **Structured Databases:** For queries requiring tabular data access, the system may use a *Text-to-SQL* engine that interacts with databases like PostgreSQL or MySQL.
 - **Semantic Search:** When dealing with unstructured information, it retrieves relevant documents (e.g., PDFs, books, organizational records) using vector-based retrieval.

- **Web Search:** For real-time or broad contextual information, the system leverages a web search tool to access the latest online data.
 - **Recommendation Systems:** For personalized or contextual queries, the system taps into recommendation engines that provide tailored suggestions.
3. **Data Integration and LLM Synthesis:** Once the relevant data is retrieved from the chosen sources, it is passed to a *Large Language Model (LLM)*. The LLM synthesizes the gathered information, integrating insights from multiple sources into a coherent and contextually relevant response.
4. **Output Generation:** Finally, the system delivers a comprehensive, user-facing answer that addresses the original query. This response is presented in an actionable, concise format and may optionally include references or citations to the sources used.

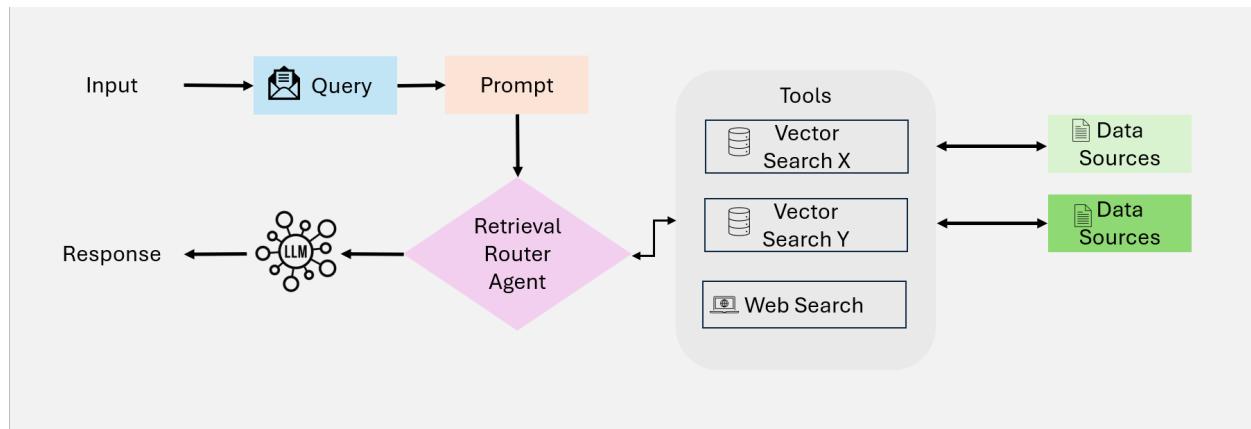


Figure 16: An Overview of Single Agentic RAG

Key Features and Advantages.

- **Centralized Simplicity:** A single agent handles all retrieval and routing tasks, making the architecture straightforward to design, implement, and maintain.
- **Efficiency & Resource Optimization:** With fewer agents and simpler coordination, the system demands fewer computational resources and can handle queries more quickly.
- **Dynamic Routing:** The agent evaluates each query in real-time, selecting the most appropriate knowledge source (e.g., structured DB, semantic search, web search).
- **Versatility Across Tools:** Supports a variety of data sources and external APIs, enabling both structured and unstructured workflows.
- **Ideal for Simpler Systems:** Suited for applications with well-defined tasks or limited integration requirements (e.g., document retrieval, SQL-based workflows).

Use Case: Customer Support

Prompt: Can you tell me the delivery status of my order?

System Process (Single-Agent Workflow):

1. Query Submission and Evaluation:

- The user submits the query, which is received by the coordinating agent.
- The coordinating agent analyzes the query and determines the most appropriate sources of information.

2. Knowledge Source Selection:

- Retrieves tracking details from the order management database.
- Fetches real-time updates from the shipping provider's API.
- Optionally conducts a web search to identify local conditions affecting delivery, such as weather or logistical delays.

3. Data Integration and LLM Synthesis:

- The relevant data is passed to the LLM, which synthesizes the information into a coherent response.

4. Output Generation:

- The system generates an actionable and concise response, providing live tracking updates and potential alternatives.

Response:

Integrated Response: “Your package is currently in transit and expected to arrive tomorrow evening. The live tracking from UPS indicates it is at the regional distribution center.”

4.7 Multi-Agent Agentic RAG Systems:

Multi-Agent RAG [30] represents a modular and scalable evolution of single-agent architectures, designed to handle complex workflows and diverse query types by leveraging multiple specialized agents (as shown in Figure 17). Instead of relying on a single agent to manage all tasks—reasoning, retrieval, and response generation—this system distributes responsibilities across multiple agents, each optimized for a specific role or data source.

Workflow

1. **Query Submission:** The process begins with a user query, which is received by a *coordinator agent* or master retrieval agent. This agent acts as the central orchestrator, delegating the query to specialized retrieval agents based on the query's requirements.
2. **Specialized Retrieval Agents:** The query is distributed among multiple retrieval agents, each focusing on a specific type of data source or task. Examples include:
 - **Agent 1:** Handles structured queries, such as interacting with SQL-based databases like PostgreSQL or MySQL.
 - **Agent 2:** Manages semantic searches for retrieving unstructured data from sources like PDFs, books, or internal records.
 - **Agent 3:** Focuses on retrieving real-time public information from web searches or APIs.
 - **Agent 4:** Specializes in recommendation systems, delivering context-aware suggestions based on user behavior or profiles.
3. **Tool Access and Data Retrieval:** Each agent routes the query to the appropriate tools or data sources within its domain, such as:
 - *Vector Search:* For semantic relevance.
 - *Text-to-SQL:* For structured data.
 - *Web Search:* For real-time public information.
 - *APIs:* For accessing external services or proprietary systems.

The retrieval process is executed in parallel, allowing for efficient processing of diverse query types.

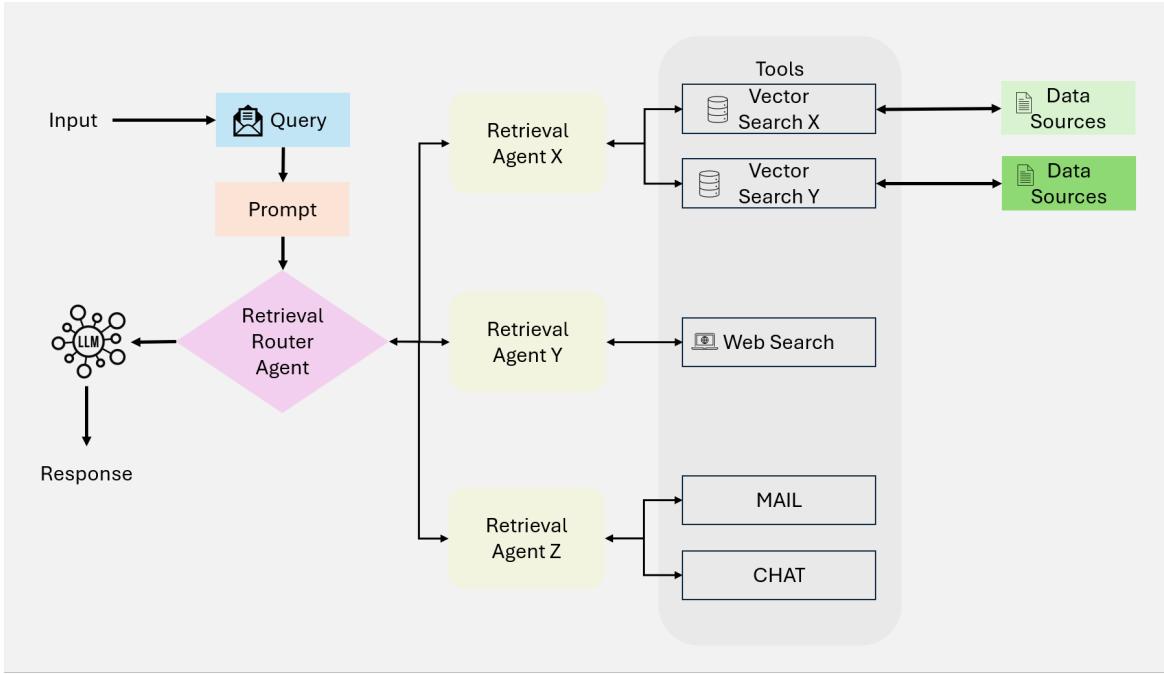


Figure 17: An Overview of Multi-Agent Agentic RAG Systems

4. **Data Integration and LLM Synthesis:** Once retrieval is complete, the data from all agents is passed to a *Large Language Model (LLM)*. The LLM synthesizes the retrieved information into a coherent and contextually relevant response, integrating insights from multiple sources seamlessly.
5. **Output Generation:** The system generates a comprehensive response, which is delivered back to the user in an actionable and concise format.

Key Features and Advantages.

- **Modularity:** Each agent operates independently, allowing for seamless addition or removal of agents based on system requirements.
- **Scalability:** Parallel processing by multiple agents enables the system to handle high query volumes efficiently.
- **Task Specialization:** Each agent is optimized for a specific type of query or data source, improving accuracy and retrieval relevance.
- **Efficiency:** By distributing tasks across specialized agents, the system minimizes bottlenecks and enhances performance for complex workflows.
- **Versatility:** Suitable for applications spanning multiple domains, including research, analytics, decision-making, and customer support.

Challenges

- **Coordination Complexity:** Managing inter-agent communication and task delegation requires sophisticated orchestration mechanisms.
- **Computational Overhead:** Parallel processing of multiple agents can increase resource usage.
- **Data Integration:** Synthesizing outputs from diverse sources into a cohesive response is non-trivial and requires advanced LLM capabilities.

Use Case: Multi-Domain Research Assistant

Prompt: What are the economic and environmental impacts of renewable energy adoption in Europe?

System Process (Multi-Agent Workflow):

- **Agent 1:** Retrieves statistical data from economic databases using SQL-based queries.
- **Agent 2:** Searches for relevant academic papers using semantic search tools.
- **Agent 3:** Performs a web search for recent news and policy updates on renewable energy.
- **Agent 4:** Consults a recommendation system to suggest related content, such as reports or expert commentary.

Response:

Integrated Response: “Adopting renewable energy in Europe has led to a 20% reduction in greenhouse gas emissions over the past decade, according to EU policy reports. Economically, renewable energy investments have generated approximately 1.2 million jobs, with significant growth in solar and wind sectors. Recent academic studies also highlight potential trade-offs in grid stability and energy storage costs.”

4.8 Hierarchical Agentic RAG Systems

Hierarchical Agentic RAG: [14] systems employ a structured, multi-tiered approach to information retrieval and processing, enhancing both efficiency and strategic decision-making as shown in Figure 18. Agents are organized in a hierarchy, with higher-level agents overseeing and directing lower-level agents. This structure enables multi-level decision-making, ensuring that queries are handled by the most appropriate resources.

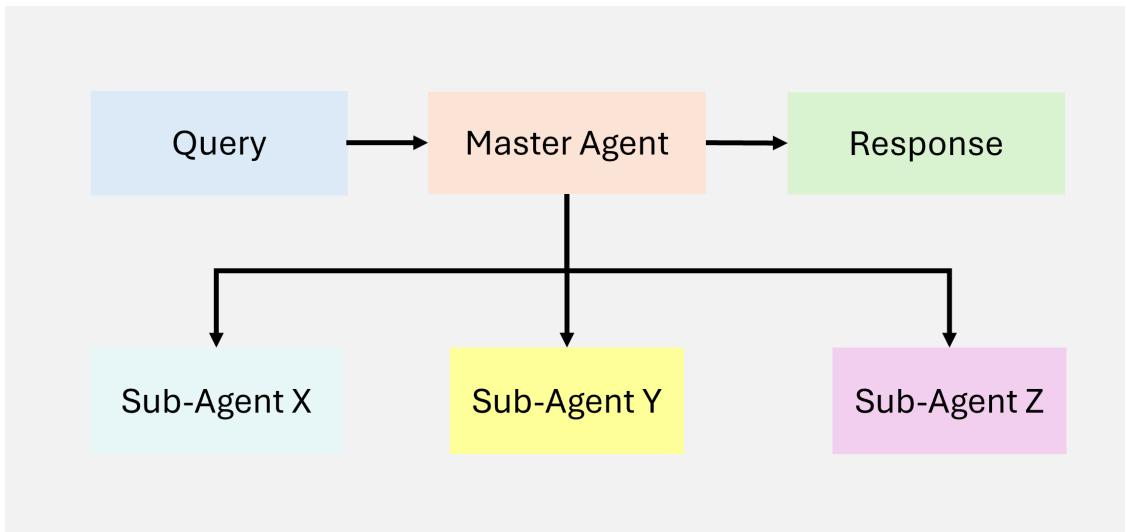


Figure 18: An illustration of Hierarchical Agentic RAG

Workflow

1. **Query Reception:** A user submits a query, received by a *top-tier agent* responsible for initial assessment and delegation.
2. **Strategic Decision-Making:** The top-tier agent evaluates the query's complexity and decides which subordinate agents or data sources to prioritize. Certain databases, APIs, or retrieval tools may be deemed more reliable or relevant based on the query's domain.
3. **Delegation to Subordinate Agents:** The top-tier agent assigns tasks to lower-level agents specialized in particular retrieval methods (e.g., SQL databases, web search, or proprietary systems). These agents execute their assigned tasks independently.

4. **Aggregation and Synthesis:** The results from subordinate agents are collected and integrated by the higher-level agent, which synthesizes the information into a coherent response.
5. **Response Delivery:** The final, synthesized answer is returned to the user, ensuring that the response is both comprehensive and contextually relevant.

Key Features and Advantages.

- **Strategic Prioritization:** Top-tier agents can prioritize data sources or tasks based on query complexity, reliability, or context.
- **Scalability:** Distributing tasks across multiple agent tiers enables handling of highly complex or multi-faceted queries.
- **Enhanced Decision-Making:** Higher-level agents apply strategic oversight to improve overall accuracy and coherence of responses.

Challenges

- **Coordination Complexity:** Maintaining robust inter-agent communication across multiple levels can increase orchestration overhead.
- **Resource Allocation:** Efficiently distributing tasks among tiers to avoid bottlenecks is non-trivial.

Use Case: Financial Analysis System

Prompt: What are the best investment options given the current market trends in renewable energy?

System Process (Hierarchical Agentic Workflow):

1. **Top-Tier Agent:** Assesses the query's complexity and prioritizes reliable financial databases and economic indicators over less validated data sources.
2. **Mid-Level Agent:** Retrieves real-time market data (e.g., stock prices, sector performance) from proprietary APIs and structured SQL databases.
3. **Lower-Level Agent(s):** Conducts web searches for recent policy announcements and consults recommendation systems that track expert opinions and news analytics.
4. **Aggregation and Synthesis:** The top-tier agent compiles the results, integrating quantitative data with policy insights.

Response:

Integrated Response: “Based on current market data, renewable energy stocks have shown a 15% growth over the past quarter, driven by supportive government policies and heightened investor interest. Analysts suggest that wind and solar sectors, in particular, may experience continued momentum, while emerging technologies like green hydrogen present moderate risk but potentially high returns.”

4.9 Agentic Corrective RAG

Corrective RAG : introduces mechanisms to self-correct retrieval results, enhancing document utilization and improving response generation quality as demonstrated in Figure 19. By embedding intelligent agents into the workflow, Corrective RAG [31] [32] ensures iterative refinement of context documents and responses, minimizing errors and maximizing relevance.

Key Idea of Corrective RAG: The core principle of Corrective RAG lies in its ability to evaluate retrieved documents dynamically, perform corrective actions, and refine queries to enhance the quality of generated responses. Corrective RAG adjusts its approach as follows:

- **Document Relevance Evaluation:** Retrieved documents are assessed for relevance by the *Relevance Evaluation Agent*. Documents below the relevance threshold trigger corrective steps.
- **Query Refinement and Augmentation:** Queries are refined by the *Query Refinement Agent*, which leverages semantic understanding to optimize retrieval for better results.

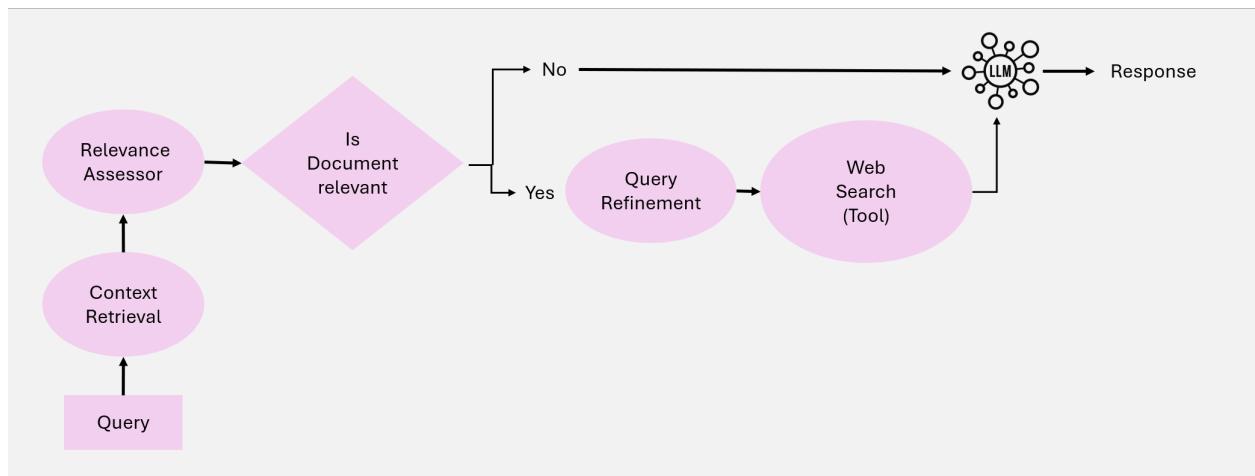


Figure 19: Overview of Agentive Corrective RAG

- **Dynamic Retrieval from External Sources:** When context is insufficient, the *External Knowledge Retrieval Agent* performs web searches or accesses alternative data sources to supplement the retrieved documents.
- **Response Synthesis:** All validated and refined information is passed to the *Response Synthesis Agent* for final response generation.

Workflow: The Corrective RAG system is built on five key agents:

1. **Context Retrieval Agent:** Responsible for retrieving initial context documents from a vector database.
2. **Relevance Evaluation Agent:** Assesses the retrieved documents for relevance and flags any irrelevant or ambiguous documents for corrective actions.
3. **Query Refinement Agent:** Rewrites queries to improve retrieval, leveraging semantic understanding to optimize results.
4. **External Knowledge Retrieval Agent:** Performs web searches or accesses alternative data sources when the context documents are insufficient.
5. **Response Synthesis Agent:** Synthesizes all validated information into a coherent and accurate response.

Key Features and Advantages:

- **Iterative Correction:** Ensures high response accuracy by dynamically identifying and correcting irrelevant or ambiguous retrieval results.
- **Dynamic Adaptability:** Incorporates real-time web searches and query refinement for enhanced retrieval precision.
- **Agentic Modularity:** Each agent performs specialized tasks, ensuring efficient and scalable operation.
- **Factual Assurance:** By validating all retrieved and generated content, Corrective RAG minimizes the risk of hallucination or misinformation.

Use Case: Academic Research Assistant

Prompt: What are the latest findings in generative AI research?

System Process (Corrective RAG Workflow):

1. **Query Submission:** A user submits the query to the system.
2. **Context Retrieval:**
 - The *Context Retrieval Agent* retrieves initial documents from a database of published papers on generative AI.
 - The retrieved documents are passed to the next step for evaluation.
3. **Relevance Evaluation:**
 - The *Relevance Evaluation Agent* assesses the documents for alignment with the query.
 - Documents are classified into relevant, ambiguous, or irrelevant categories. Irrelevant documents are flagged for corrective actions.
4. **Corrective Actions (if needed):**
 - The *Query Refinement Agent* rewrites the query to improve specificity and relevance.
 - The *External Knowledge Retrieval Agent* performs web searches to fetch additional papers and reports from external sources.
5. **Response Synthesis:**
 - The *Response Synthesis Agent* integrates validated documents into a coherent and comprehensive summary.

Response:

Integrated Response: “Recent findings in generative AI highlight advancements in diffusion models, reinforcement learning for text-to-video tasks, and optimization techniques for large-scale model training. For more details, refer to studies published in NeurIPS 2024 and AAAI 2025.”

4.10 Adaptive Agentic RAG

Adaptive Retrieval-Augmented Generation (Adaptive RAG) [33] enhances the flexibility and efficiency of large language models (LLMs) by dynamically adjusting query handling strategies based on the complexity of the incoming query. Unlike static retrieval workflows, Adaptive RAG [34] employs a classifier to assess query complexity and determine the most appropriate approach, ranging from single-step retrieval to multi-step reasoning, or even bypassing retrieval altogether for straightforward queries as illustrated in Figure 20.

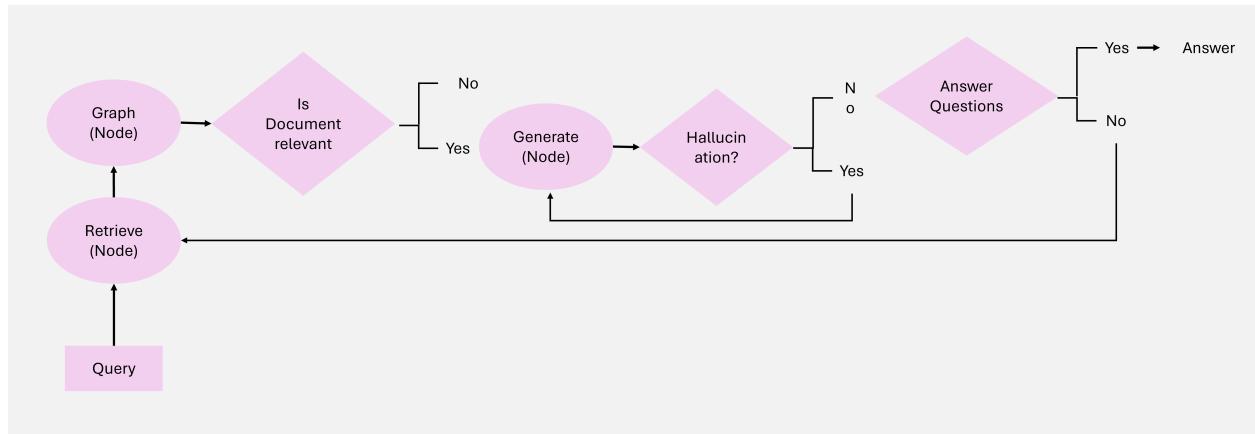


Figure 20: An Overview of Adaptive Agentic RAG

Key Idea of Adaptive RAG The core principle of Adaptive RAG lies in its ability to dynamically tailor retrieval strategies based on the complexity of the query. Adaptive RAG adjusts its approach as follows:

- **Straightforward Queries:** For fact-based questions that require no additional retrieval (e.g., "*What is the boiling point of water?*"), the system directly generates an answer using pre-existing knowledge.
- **Simple Queries:** For moderately complex tasks requiring minimal context (e.g., "*What is the status of my latest electricity bill?*"), the system performs a single-step retrieval to fetch the relevant details.
- **Complex Queries:** For multi-layered queries requiring iterative reasoning (e.g., "*How has the population of City X changed over the past decade, and what are the contributing factors?*"), the system employs multi-step retrieval, progressively refining intermediate results to provide a comprehensive answer.

Workflow: The Adaptive RAG system is built on three primary components:

1. Classifier Role:

- A smaller language model analyzes the query to predict its complexity.
- The classifier is trained using automatically labeled datasets, derived from past model outcomes and query patterns.

2. Dynamic Strategy Selection:

- For straightforward queries, the system avoids unnecessary retrieval, directly leveraging the LLM for response generation.
- For simple queries, it employs a single-step retrieval process to fetch relevant context.
- For complex queries, it activates multi-step retrieval to ensure iterative refinement and enhanced reasoning.

3. LLM Integration:

- The LLM synthesizes retrieved information into a coherent response.
- Iterative interactions between the LLM and the classifier enable refinement for complex queries.

Key Features and Advantages

- **Dynamic Adaptability:** Adjusts retrieval strategies based on query complexity, optimizing both computational efficiency and response accuracy.
- **Resource Efficiency:** Minimizes unnecessary overhead for simple queries while ensuring thorough processing for complex ones.
- **Enhanced Accuracy:** Iterative refinement ensures that complex queries are resolved with high precision.
- **Flexibility:** Can be extended to incorporate additional pathways, such as domain-specific tools or external APIs.

Use Case: Customer Support Assistant

Prompt: Why is my package delayed, and what alternatives do I have?

System Process (Adaptive RAG Workflow):

1. Query Classification:

- The classifier analyzes the query and determines it to be complex, requiring multi-step reasoning.

2. Dynamic Strategy Selection:

- The system activates a multi-step retrieval process based on the complexity classification.

3. Multi-Step Retrieval:

- Retrieves tracking details from the order database.
- Fetches real-time status updates from the shipping provider API.
- Conducts a web search for external factors such as weather conditions or local disruptions.

4. Response Synthesis:

- The LLM integrates all retrieved information, synthesizing a comprehensive and actionable response.

Response:

Integrated Response: “Your package is delayed due to severe weather conditions in your region. It is currently at the local distribution center and will be delivered in 2 days. Alternatively, you may opt for a local pickup from the facility.”

4.11 Graph-Based Agentic RAG

4.11.1 Agent-G: Agentic Framework for Graph RAG

Agent-G [8]: introduces a novel agentic architecture that integrates graph knowledge bases with unstructured document retrieval. By combining structured and unstructured data sources, this framework enhances retrieval-augmented generation (RAG) systems with improved reasoning and retrieval accuracy. It employs modular retriever banks, dynamic agent interaction, and feedback loops to ensure high-quality outputs as shown in Figure 21.

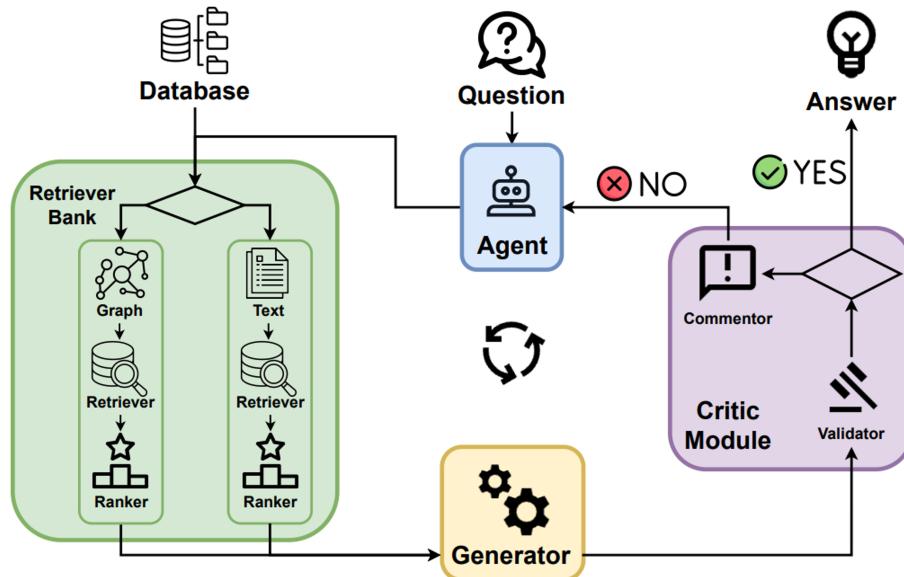


Figure 21: An Overview of Agent-G: Agentic Framework for Graph RAG [8]

Key Idea of Agent-G The core principle of Agent-G lies in its ability to dynamically assign retrieval tasks to specialized agents, leveraging both graph knowledge bases and textual documents. Agent-G adjusts its retrieval strategy as follows:

- **Graph Knowledge Bases:** Structured data is used to extract relationships, hierarchies, and connections (e.g., disease-to-symptom mappings in healthcare).
- **Unstructured Documents:** Traditional text retrieval systems provide contextual information to complement graph data.
- **Critic Module:** Evaluates the relevance and quality of retrieved information, ensuring alignment with the query.
- **Feedback Loops:** Refines retrieval and synthesis through iterative validation and re-querying.

Workflow: The Agent-G system is built on four primary components:

1. Retriever Bank:

- A modular set of agents specializes in retrieving graph-based or unstructured data.
- Agents dynamically select relevant sources based on the query's requirements.

2. Critic Module:

- Validates retrieved data for relevance and quality.
- Flags low-confidence results for re-retrieval or refinement.

3. Dynamic Agent Interaction:

- Task-specific agents collaborate to integrate diverse data types.
- Ensures cohesive retrieval and synthesis across graph and text sources.

4. LLM Integration:

- Synthesizes validated data into a coherent response.
- Iterative feedback from the critic ensures alignment with the query's intent.

Key Features and Advantages

- **Enhanced Reasoning:** Combines structured relationships from graphs with contextual information from unstructured documents.
- **Dynamic Adaptability:** Adjusts retrieval strategies dynamically based on query requirements.
- **Improved Accuracy:** Critic module reduces the risk of irrelevant or low-quality data in responses.
- **Scalable Modularity:** Supports the addition of new agents for specialized tasks, enhancing scalability.

Use Case: Healthcare Diagnostics

Prompt: What are the common symptoms of Type 2 Diabetes, and how are they related to heart disease?

System Process (Agent-G Workflow):

1. **Query Reception and Assignment:** The system receives the query and identifies the need for both graph-structured and unstructured data to answer the question comprehensively.
2. **Graph Retriever:**
 - Extracts relationships between Type 2 Diabetes and heart disease from a medical knowledge graph.
 - Identifies shared risk factors such as obesity and high blood pressure by exploring graph hierarchies and relationships.
3. **Document Retriever:**
 - Retrieves descriptions of Type 2 Diabetes symptoms (e.g., increased thirst, frequent urination, fatigue) from medical literature.
 - Adds contextual information to complement the graph-based insights.
4. **Critic Module:**
 - Evaluates the relevance and quality of the retrieved graph data and document data.
 - Flags low-confidence results for refinement or re-querying.
5. **Response Synthesis:** The LLM integrates validated data from the Graph Retriever and Document Retriever into a coherent response, ensuring alignment with the query's intent.

Response:

Integrated Response: “Type 2 Diabetes symptoms include increased thirst, frequent urination, and fatigue. Studies show a 50% correlation between diabetes and heart disease, primarily through shared risk factors such as obesity and high blood pressure.”

4.11.2 GeAR: Graph-Enhanced Agent for Retrieval-Augmented Generation

GeAR [35]: introduces an agentic framework that enhances traditional Retrieval-Augmented Generation (RAG) systems by incorporating graph-based retrieval mechanisms. By leveraging graph expansion techniques and an agent-based architecture, GeAR addresses challenges in multi-hop retrieval scenarios, improving the system's ability to handle complex queries as shown in Figure 22.

Key Idea of GeAR GeAR advances RAG performance through two primary innovations:

- **Graph Expansion:** Enhances conventional base retrievers (e.g., BM25) by expanding the retrieval process to include graph-structured data, enabling the system to capture complex relationships and dependencies between entities.
- **Agent Framework:** Incorporates an agent-based architecture that utilizes graph expansion to manage retrieval tasks more effectively, allowing for dynamic and autonomous decision-making in the retrieval process.

Workflow: The GeAR system operates through the following components:

1. **Graph Expansion Module:**
 - Integrates graph-based data into the retrieval process, allowing the system to consider relationships between entities during retrieval.
 - Enhances the base retriever's ability to handle multi-hop queries by expanding the search space to include connected entities.
2. **Agent-Based Retrieval:**
 - Employs an agent framework to manage the retrieval process, enabling dynamic selection and combination of retrieval strategies based on the query's complexity.
 - Agents can autonomously decide to utilize graph-expanded retrieval paths to improve the relevance and accuracy of retrieved information.

3. LLM Integration:

- Combines the retrieved information, enriched by graph expansion, with the capabilities of a Large Language Model (LLM) to generate coherent and contextually relevant responses.
- The integration ensures that the generative process is informed by both unstructured documents and structured graph data.

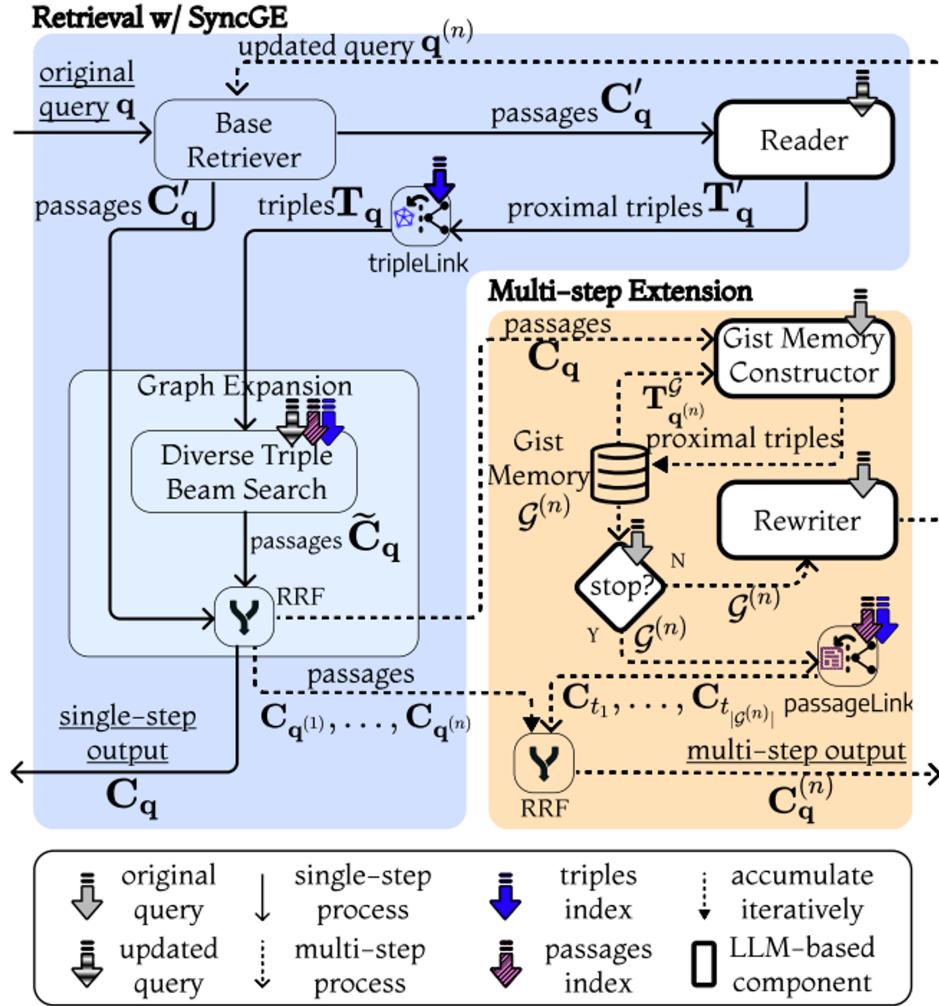


Figure 22: An Overview of GeAR: Graph-Enhanced Agent for Retrieval-Augmented Generation[35]

Key Features and Advantages

- **Enhanced Multi-Hop Retrieval:** GeAR's graph expansion allows the system to handle complex queries that require reasoning over multiple interconnected pieces of information.
- **Agentic Decision-Making:** The agent framework enables dynamic and autonomous selection of retrieval strategies, improving efficiency and relevance.
- **Improved Accuracy:** By incorporating structured graph data, GeAR enhances the precision of retrieved information, leading to more accurate and contextually appropriate responses.
- **Scalability:** The modular nature of the agent framework allows for the integration of additional retrieval strategies and data sources as needed.

Use Case: Multi-Hop Question Answering

Prompt: Which author influenced the mentor of J.K. Rowling?

System Process (GeAR Workflow):

1. **Top-Tier Agent:** Evaluates the query's multi-hop nature and determines that a combination of graph expansion and document retrieval is necessary to answer the question.
2. **Graph Expansion Module:**
 - Identifies that J.K. Rowling's mentor is a key entity in the query.
 - Traces the literary influences on that mentor by exploring graph-structured data on literary relationships.
3. **Agent-Based Retrieval:**
 - An agent autonomously selects the graph-expanded retrieval path to gather relevant information about the mentor's influences.
 - Integrates additional context by querying textual data sources for unstructured details about the mentor and their influences.
4. **Response Synthesis:** Combines insights from the graph and document retrieval processes using the LLM to generate a response that accurately reflects the complex relationships in the query.

Response:

Integrated Response: "J.K. Rowling's mentor, [Mentor Name], was heavily influenced by [Author Name], known for their [notable works or genre]. This connection highlights the layered relationships in literary history, where influential ideas often pass through multiple generations of authors."

4.12 Agentic Document Workflows in Agentic RAG

Agentic Document Workflows (ADW) [36] extend traditional Retrieval-Augmented Generation (RAG) paradigms by enabling end-to-end knowledge work automation. These workflows orchestrate complex document-centric processes, integrating document parsing, retrieval, reasoning, and structured outputs with intelligent agents (see Figure 23). ADW systems address limitations of Intelligent Document Processing (IDP) and RAG by maintaining state, coordinating multi-step workflows, and applying domain-specific logic to documents.

Workflow

1. **Document Parsing and Information Structuring:**
 - Documents are parsed using enterprise-grade tools (e.g., LlamaParse) to extract relevant data fields such as invoice numbers, dates, vendor information, line items, and payment terms.
 - Structured data is organized for downstream processing.
2. **State Maintenance Across Processes:**
 - The system maintains state about document context, ensuring consistency and relevance across multi-step workflows.
 - Tracks the progression of the document through various processing stages.
3. **Knowledge Retrieval:**
 - Relevant references are retrieved from external knowledge bases (e.g., LlamaCloud) or vector indexes.
 - Retrieves real-time, domain-specific guidelines for enhanced decision-making.
4. **Agentic Orchestration:**
 - Intelligent agents apply business rules, perform multi-hop reasoning, and generate actionable recommendations.
 - Orchestrates components such as parsers, retrievers, and external APIs for seamless integration.
5. **Actionable Output Generation:**
 - Outputs are presented in structured formats, tailored to specific use cases.
 - Recommendations and extracted insights are synthesized into concise and actionable reports.

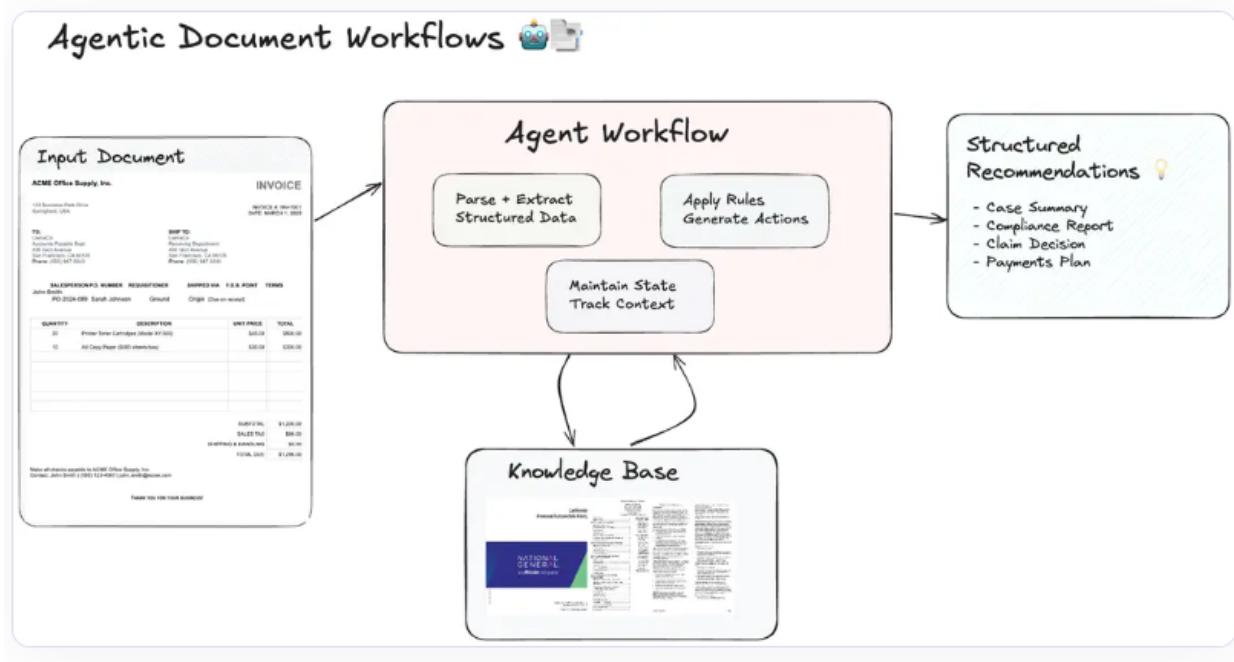


Figure 23: An Overview of Agentic Document Workflows (ADW)
[36]

Use Case: Invoice Payments Workflow

Prompt: Generate a payment recommendation report based on the submitted invoice and associated vendor contract terms.

System Process (ADW Workflow):

1. Parse the invoice to extract key details such as invoice number, date, vendor information, line items, and payment terms.
2. Retrieve the corresponding vendor contract to verify payment terms and identify any applicable discounts or compliance requirements.
3. Generate a payment recommendation report that includes original amount due, potential early payment discounts, budget impact analysis, and strategic payment actions.

Response: Integrated Response: "Invoice INV-2025-045 for \$15,000.00 has been processed. An early payment discount of 2% is available if paid by 2025-04-10, reducing the amount due to \$14,700.00. A bulk order discount of 5% was applied as the subtotal exceeded \$10,000.00. It is recommended to approve early payment to save 2% and ensure timely fund allocation for upcoming project phases."

Key Features and Advantages

- **State Maintenance:** Tracks document context and workflow stage, ensuring consistency across processes.
- **Multi-Step Orchestration:** Handles complex workflows involving multiple components and external tools.
- **Domain-Specific Intelligence:** Applies tailored business rules and guidelines for precise recommendations.
- **Scalability:** Supports large-scale document processing with modular and dynamic agent integration.
- **Enhanced Productivity:** Automates repetitive tasks while augmenting human expertise in decision-making.

4.13 Comparative Analysis of Agentic RAG Frameworks

Table 2 provides a comprehensive comparative analysis of the three architectural frameworks: Traditional RAG, Agentic RAG, and Agentic Document Workflows (ADW). This analysis highlights their respective strengths, weaknesses, and best-fit scenarios, offering valuable insights into their applicability across diverse use cases.

Table 2: Comparative Analysis: Traditional RAG vs Agentic RAG vs Agentic Document Workflows (ADW)

Feature	Traditional RAG	Agentic RAG	Agentic Document Workflows (ADW)
Focus	Isolated retrieval and generation tasks	Multi-agent collaboration and reasoning	Document-centric end-to-end workflows
Context Maintenance	Limited	Enabled through memory modules	Maintains state across multi-step workflows
Dynamic Adaptability	Minimal	High	Tailored to document workflows
Workflow Orchestration	Absent	Orchestrates multi-agent tasks	Integrates multi-step document processing
Use of External Tools/APIs	Basic integration (e.g., retrieval tools)	Extends via tools like APIs and knowledge bases	Deeply integrates business rules and domain-specific tools
Scalability	Limited to small datasets or queries	Scalable for multi-agent systems	Scales for multi-domain enterprise workflows
Complex Reasoning	Basic (e.g., simple Q&A)	Multi-step reasoning with agents	Structured reasoning across documents
Primary Applications	QA systems, knowledge retrieval	Multi-domain knowledge and reasoning	Contract review, invoice processing, claims analysis
Strengths	Simplicity, quick setup	High accuracy, collaborative reasoning	End-to-end automation, domain-specific intelligence
Challenges	Poor contextual understanding	Coordination complexity	Resource overhead, domain standardization

The comparative analysis underscores the evolutionary trajectory from Traditional RAG to Agentic RAG and further to Agentic Document Workflows (ADW). While Traditional RAG offers simplicity and ease of deployment for basic tasks, Agentic RAG introduces enhanced reasoning and scalability through multi-agent collaboration. ADW builds upon these advancements by providing robust, document-centric workflows that facilitate end-to-end automation and integration with domain-specific processes. Understanding the strengths and limitations of each framework is crucial for selecting the most appropriate architecture to meet specific application requirements and operational demands.

5 Applications of Agentic RAG

Agentic Retrieval-Augmented Generation (RAG) systems have demonstrated transformative potential across a variety of domains. By combining real-time data retrieval, generative capabilities, and autonomous decision-making, these systems address complex, dynamic, and multi-modal challenges. This section explores the key applications of Agentic RAG, providing detailed insights into how these systems are shaping industries such as customer support, healthcare, finance, education, legal workflows, and creative industries.

5.1 Customer Support and Virtual Assistants

Agentic RAG systems are revolutionizing customer support by enabling real-time, context-aware query resolution. Traditional chatbots and virtual assistants often rely on static knowledge bases, leading to generic or outdated responses. By contrast, Agentic RAG systems dynamically retrieve the most relevant information, adapt to the user's context, and generate personalized responses.

Use Case: Twitch Ad Sales Enhancement [37]

For instance, Twitch leveraged an agentic workflow with RAG on Amazon Bedrock to streamline ad sales. The system dynamically retrieved advertiser data, historical campaign performance, and audience demographics to generate detailed ad proposals, significantly boosting operational efficiency.

Key Benefits:

- **Improved Response Quality:** Personalized and context-aware replies enhance user engagement.
- **Operational Efficiency:** Reduces the workload on human support agents by automating complex queries.
- **Real-Time Adaptability:** Dynamically integrates evolving data, such as live service outages or pricing updates.

5.2 Healthcare and Personalized Medicine

In healthcare, the integration of patient-specific data with the latest medical research is critical for informed decision-making. Agentic RAG systems enable this by retrieving real-time clinical guidelines, medical literature, and patient history to assist clinicians in diagnostics and treatment planning.

Use Case: Patient Case Summary [38]

Agentic RAG systems have been applied in generating patient case summaries. For example, by integrating electronic health records (EHR) and up-to-date medical literature, the system generates comprehensive summaries for clinicians to make faster and more informed decisions.

Key Benefits:

- **Personalized Care:** Tailors recommendations to individual patient needs.
- **Time Efficiency:** Streamlines the retrieval of relevant research, saving valuable time for healthcare providers.
- **Accuracy:** Ensures recommendations are based on the latest evidence and patient-specific parameters.

5.3 Legal and Contract Analysis

Agentic RAG systems are redefining how legal workflows are conducted, offering tools for rapid document analysis and decision-making.

Use Case: Contract Review [39]

A legal agentic RAG system can analyze contracts, extract critical clauses, and identify potential risks. By combining semantic search capabilities with legal knowledge graphs, it automates the tedious process of contract review, ensuring compliance and mitigating risks.

Key Benefits:

- **Risk Identification:** Automatically flags clauses that deviate from standard terms.
- **Efficiency:** Reduces the time spent on contract review processes.
- **Scalability:** Handles large volumes of contracts simultaneously.

5.4 Finance and Risk Analysis

Agentic RAG systems are transforming the finance industry by providing real-time insights for investment decisions, market analysis, and risk management. These systems integrate live data streams, historical trends, and predictive modeling to generate actionable outputs.

Use Case: Auto Insurance Claims Processing [40]

In auto insurance, Agentic RAG can automate claim processing. For example, by retrieving policy details and combining them with accident data, it generates claim recommendations while ensuring compliance with regulatory requirements.

Key Benefits:

- **Real-Time Analytics:** Delivers insights based on live market data.
- **Risk Mitigation:** Identifies potential risks using predictive analysis and multi-step reasoning.
- **Enhanced Decision-Making:** Combines historical and live data for comprehensive strategies.

5.5 Education and Personalized Learning

Education is another domain where Agentic RAG systems are making significant strides. These systems enable adaptive learning by generating explanations, study materials, and feedback tailored to the learner's progress and preferences.

Use Case: Research Paper Generation [41]

In higher education, Agentic RAG has been used to assist researchers by synthesizing key findings from multiple sources. For instance, a researcher querying, "What are the latest advancements in quantum computing?" receives a concise summary enriched with references, enhancing the quality and efficiency of their work.

Key Benefits:

- **Tailored Learning Paths:** Adapts content to individual student needs and performance levels.
- **Engaging Interactions:** Provides interactive explanations and personalized feedback.
- **Scalability:** Supports large-scale deployments for diverse educational environments.

5.6 Graph-Enhanced Applications in Multimodal Workflows

Graph-Enhanced Agentic RAG (GEAR) combines graph structures with retrieval mechanisms, making it particularly effective in multimodal workflows where interconnected data sources are essential.

Use Case: Market Survey Generation

GEAR enables the synthesis of text, images, and videos for marketing campaigns. For example, querying, "What are the emerging trends in eco-friendly products?" generates a detailed report enriched with customer preferences, competitor analysis, and multimedia content.

Key Benefits:

- **Multi-Modal Capabilities:** Integrates text, image, and video data for comprehensive outputs.
- **Enhanced Creativity:** Generates innovative ideas and solutions for marketing and entertainment.
- **Dynamic Adaptability:** Adapts to evolving market trends and customer needs.

The applications of Agentic RAG systems span a wide range of industries, showcasing their versatility and transformative potential. From personalized customer support to adaptive education and graph-enhanced multimodal workflows, these systems address complex, dynamic, and knowledge-intensive challenges. By integrating retrieval, generation, and agentic intelligence, Agentic RAG systems are paving the way for next-generation AI applications.

6 Tools and Frameworks for Agentic RAG

Agentic Retrieval-Augmented Generation (RAG) systems represent a significant evolution in combining retrieval, generation, and agentic intelligence. These systems extend the capabilities of traditional RAG by integrating decision-making, query reformulation, and adaptive workflows. The following tools and frameworks provide robust support for developing Agentic RAG systems, addressing the complex requirements of real-world applications.

Key Tools and Frameworks:

- **LangChain and LangGraph:** LangChain [42] provides modular components for building RAG pipelines, seamlessly integrating retrievers, generators, and external tools. LangGraph complements this by introducing graph-based workflows that support loops, state persistence, and human-in-the-loop interactions, enabling sophisticated orchestration and self-correction mechanisms in agentic systems.
- **LlamaIndex:** LlamaIndex's [43] Agentic Document Workflows (ADW) enable end-to-end automation of document processing, retrieval, and structured reasoning. It introduces a meta-agent architecture where sub-agents manage smaller document sets, coordinating through a top-level agent for tasks such as compliance analysis and contextual understanding.
- **Hugging Face Transformers and Qdrant:** Hugging Face [44] offers pre-trained models for embedding and generation tasks, while Qdrant [45] enhances retrieval workflows with adaptive vector search capabilities, allowing agents to optimize performance by dynamically switching between sparse and dense vector methods.
- **CrewAI and AutoGen:** These frameworks emphasize multi-agent architectures. CrewAI [46] supports hierarchical and sequential processes, robust memory systems, and tool integrations. AG2 [47] (formerly

knows as AutoGen [48, 49]) excels in multi-agent collaboration with advanced support for code generation, tool execution, and decision-making.

- **OpenAI Swarm Framework:** An educational framework designed for ergonomic, lightweight multi-agent orchestration [50], emphasizing agent autonomy and structured collaboration.
- **Agentic RAG with Vertex AI:** Developed by Google, Vertex AI [51] integrates seamlessly with Agentic Retrieval-Augmented Generation (RAG), providing a platform to build, deploy, and scale machine learning models while leveraging advanced AI capabilities for robust, contextually aware retrieval and decision-making workflows.
- **Semantic Kernel:** Semantic Kernel [52, 53] is an open-source SDK by Microsoft that integrates large language models (LLMs) into applications. It supports agentic patterns, enabling the creation of autonomous AI agents for natural language understanding, task automation, and decision-making. It has been used in scenarios like ServiceNow’s P1 incident management to facilitate real-time collaboration, automate task execution, and retrieve contextual information seamlessly.
- **Amazon Bedrock for Agentic RAG:** Amazon Bedrock [37] provides a robust platform for implementing Agentic Retrieval-Augmented Generation (RAG) workflows.
- **IBM Watson and Agentic RAG:** IBM’s watsonx.ai [54] supports building Agentic RAG systems, exemplified by using the Granite-3-8B-Instruct model to answer complex queries by integrating external information and enhancing response accuracy.
- **Neo4j and Vector Databases:** Neo4j, a prominent open-source graph database, excels in handling complex relationships and semantic queries. Alongside Neo4j, vector databases like Weaviate, Pinecone, Milvus, and Qdrant provide efficient similarity search and retrieval capabilities, forming the backbone of high-performance Agentic Retrieval-Augmented Generation (RAG) workflows.

7 Benchmarks and Datasets

Current benchmarks and datasets provide valuable insights into evaluating Retrieval-Augmented Generation (RAG) systems, including those with agentic and graph-based enhancements. While some are explicitly designed for RAG, others are adapted to test retrieval, reasoning, and generation capabilities in diverse scenarios. Datasets are crucial for testing the retrieval, reasoning, and generation components of RAG systems. Table 3 discusses some key datasets based on the downstream task for RAG Evaluation.

Benchmarks play a critical role in standardizing the evaluation of RAG systems by providing structured tasks and metrics. The following benchmarks are particularly relevant:

- **BEIR (Benchmarking Information Retrieval):** A versatile benchmark designed for evaluating embedding models on a variety of information retrieval tasks, encompassing 17 datasets across diverse domains like bioinformatics, finance, and question answering [55].
- **MS MARCO (Microsoft Machine Reading Comprehension):** Focused on passage ranking and question answering, this benchmark is widely used for dense retrieval tasks in RAG systems [56].
- **TREC (Text REtrieval Conference, Deep Learning Track):** Provides datasets for passage and document retrieval, emphasizing the quality of ranking models in retrieval pipelines [57].
- **MuSiQue (Multihop Sequential Questioning):** A benchmark for multihop reasoning across multiple documents, emphasizing the importance of retrieving and synthesizing information from disconnected contexts [58].
- **2WikiMultihopQA:** A dataset designed for multihop QA tasks over two Wikipedia articles, focusing on the ability to connect knowledge across multiple sources [59].
- **AgentG (Agentic RAG for Knowledge Fusion):** Tailored for agentic RAG tasks, this benchmark assesses dynamic information synthesis across multiple knowledge bases [8].
- **HotpotQA:** A multi-hop QA benchmark requiring retrieval and reasoning over interconnected contexts, ideal for evaluating complex RAG workflows [60].
- **RAGBench:** A large-scale, explainable benchmark featuring 100,000 examples across industry domains, with a TRACe evaluation framework for actionable RAG metrics [61].
- **BERGEN (Benchmarking Retrieval-Augmented Generation):** A library for systematically benchmarking RAG systems with standardized experiments [62].

- **FlashRAG Toolkit:** Implements 12 RAG methods and includes 32 benchmark datasets to support efficient and standardized RAG evaluation [63].
- **GNN-RAG:** This benchmark evaluates graph-based RAG systems on tasks like node-level and edge-level predictions, focusing on retrieval quality and reasoning performance in Knowledge Graph Question Answering (KGQA) [64].

Table 3: Downstream Tasks and Datasets for RAG Evaluation (Adapted from [20])

Category	Task Type	Datasets and References
QA	Single-hop QA	Natural Questions (NQ) [65], TriviaQA [66], SQuAD [67], Web Questions (WebQ) [68], PopQA [69], MS MARCO [56]
	Multi-hop QA	HotpotQA [60], 2WikiMultiHopQA [59], MuSiQue [58]
	Long-form QA	ELI5 [70], NarrativeQA (NQA) [71], ASQA [72], QM-Sum [73]
	Domain-specific QA	Qasper [74], COVID-QA [75], CMB/MMCU Medical [76]
	Multi-choice QA	QuALITY [77], ARC (No reference available), CommonsenseQA [78]
Graph-based QA	Graph QA	GraphQA [79]
	Event Argument Extraction	WikiEvent [80], RAMS [81]
Dialog	Open-domain Dialog	Wizard of Wikipedia (WoW) [82]
	Personalized Dialog	KBP [83], DuleMon [84]
	Task-oriented Dialog	CamRest [85]
Recommendation	Personalized Content	Amazon Datasets (Toys, Sports, Beauty) [86]
Reasoning	Commonsense Reasoning	HellaSwag [87], CommonsenseQA [78]
	CoT Reasoning	CoT Reasoning [88]
	Complex Reasoning	CSQA [89]
Others	Language Understanding	MMLU (No reference available), WikiText-103 [65]
	Fact Checking/Verification	FEVER [90], PubHealth [91]
	Strategy QA	StrategyQA [92]
Summarization	Text Summarization	WikiASP [93], XSum [94]
	Long-form Summarization	NarrativeQA (NQA) [71], QMSum [73]
Text Generation	Biography	Biography Dataset (No reference available)
Text Classification	Sentiment Analysis	SST-2 [95]
	General Classification	VioLens[96], TREC [57]
Code Search	Programming Search	CodeSearchNet [97]
Robustness	Retrieval Robustness	NoMIRACL [98]
	Language Modeling Robustness	WikiText-103 [99]
Math	Math Reasoning	GSM8K [100]
Machine Translation	Translation Tasks	JRC-Acquis [101]

8 Conclusion

Agentic Retrieval-Augmented Generation (RAG) represents a transformative advancement in artificial intelligence, addressing the limitations of traditional RAG systems through the integration of autonomous agents. By leveraging agentic intelligence, these systems introduce capabilities such as dynamic decision-making, iterative reasoning, and collaborative workflows, enabling them to tackle complex, real-world tasks with enhanced precision and adaptability.

This survey explored the evolution of RAG systems, from their initial implementations to advanced paradigms like Modular RAG, highlighting the contributions and limitations of each. The integration of agents into the RAG pipeline has emerged as a pivotal development, resulting in Agentic RAG systems that overcome static workflows and limited contextual adaptability. Applications across healthcare, finance, education, and creative industries demonstrate the transformative potential of these systems, showcasing their ability to deliver personalized, real-time, and context-aware solutions.

Despite their promise, Agentic RAG systems face challenges that require further research and innovation. Coordination complexity in multi-agent architectures, scalability, and latency issues, as well as ethical considerations, must be addressed to ensure robust and responsible deployment. Additionally, the lack of specialized benchmarks and datasets tailored to evaluate agentic capabilities poses a significant hurdle. Developing evaluation methodologies that capture the unique aspects of Agentic RAG, such as multi-agent collaboration and dynamic adaptability, will be crucial for advancing the field.

Looking ahead, the convergence of retrieval-augmented generation and agentic intelligence has the potential to redefine AI's role in dynamic and complex environments. By addressing these challenges and exploring future directions, researchers and practitioners can unlock the full potential of Agentic RAG systems, paving the way for transformative applications across industries and domains. As AI systems continue to evolve, Agentic RAG stands as a cornerstone for creating adaptive, context-aware, and impactful solutions that meet the demands of a rapidly changing world.

References

- [1] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey, 2024.
- [2] Aditi Singh. Exploring language models: A comprehensive survey and analysis. In *2023 International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKIMATE)*, pages 1–4, 2023.
- [3] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2024.
- [4] Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. A complete survey on llm-based ai chatbots, 2024.
- [5] Aditi Singh. A survey of ai text-to-image and ai text-to-video generators. In *2023 4th International Conference on Artificial Intelligence, Robotics and Control (AIRC)*, pages 32–36, 2023.
- [6] Aditi Singh, Abul Ehtesham, Gaurav Kumar Gupta, Nikhil Kumar Chatta, Saket Kumar, and Tala Talaei Khoei. Exploring prompt engineering: A systematic review with swot analysis, 2024.
- [7] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, November 2024.
- [8] Meng-Chieh Lee, Qi Zhu, Costas Mavromatis, Zhen Han, Soji Adeshina, Vassilis N. Ioannidis, Huzefa Rangwala, and Christos Faloutsos. Agent-g: An agentic framework for graph retrieval augmented generation, 2024.
- [9] Penghao Zhao, Hailin Zhang, Qinhuan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey, 2024.
- [10] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation, 2023.
- [11] Yikun Han, Chunjiang Liu, and Pengfei Wang. A comprehensive survey on vector database: Storage and retrieval technique, challenge, 2023.
- [12] Anthropic. Building effective agents, 2024. <https://www.anthropic.com/research/building-effective-agents>. Accessed: February 2, 2025.
- [13] LangChain. Langgraph workflows tutorial, 2025. <https://langchain-ai.github.io/langgraph/tutorials/workflows/>. Accessed: February 2, 2025.
- [14] Chidaksh Ravuru, Sagar Srinivas Sakhinana, and Venkataramana Runkana. Agentic retrieval-augmented generation for time series analysis, 2024.
- [15] Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey, 2023.

- [16] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey, 2024.
- [17] Aditi Singh, Abul Ehtesham, Saifuddin Mahmud, and Jong-Hoon Kim. Revolutionizing mental health care through langchain: A journey with a large language model. In *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0073–0078, 2024.
- [18] Gaurav Kumar Gupta, Aditi Singh, Sijo Valayakkad Manikandan, and Abul Ehtesham. Digital diagnostics: The potential of large language models in recognizing symptoms of common illnesses. *AI*, 6(1), 2025.
- [19] Aditi Singh, Abul Ehtesham, Saket Kumar, Gaurav Kumar Gupta, and Tala Talaei Khoei. Encouraging responsible use of generative ai in education: A reward-based learning approach. In Tim Schlippe, Eric C. K. Cheng, and Tianchong Wang, editors, *Artificial Intelligence in Education Technologies: New Development and Innovative Practices*, pages 404–413, Singapore, 2025. Springer Nature Singapore.
- [20] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.
- [21] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020.
- [22] Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents, 2024.
- [23] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing, 2024.
- [24] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey, 2024.
- [25] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Enhancing ai systems with agentic workflows patterns in large language model. In *2024 IEEE World AI IoT Congress (AIIoT)*, pages 527–532, 2024.
- [26] DeepLearning.AI. How agents can improve llm performance. <https://www.deeplearning.ai/the-batch/how-agents-can-improve-llm-performance/?ref=d1-staging-website.ghost.io>, 2024. Accessed: 2025-01-13.
- [27] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.
- [28] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.
- [29] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges, 2024.
- [30] Weaviate Blog. What is agentic rag? <https://weaviate.io/blog/what-is-agentic-rag#:~:text=is%20Agentic%20RAG%3F-,%E2%80%8B,of%20the%20non%2Dagentic%20pipeline>. Accessed: 2025-01-14.
- [31] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation, 2024.
- [32] LangGraph CRAG Tutorial. Langgraph crag: Contextualized retrieval-augmented generation tutorial. https://langchain-ai.github.io/langgraph/tutorials/rag/langgraph_crag/. Accessed: 2025-01-14.
- [33] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity, 2024.
- [34] LangGraph Adaptive RAG Tutorial. Langgraph adaptive rag: Adaptive retrieval-augmented generation tutorial. https://langchain-ai.github.io/langgraph/tutorials/rag/langgraph_adaptive_rag/. Accessed: 2025-01-14.
- [35] Zhili Shen, Chenxin Diao, Pavlos Vougiouklis, Pascual Merita, Shriram Piramanayagam, Damien Graux, Dandan Tu, Zeren Jiang, Ruofei Lai, Yang Ren, and Jeff Z. Pan. Gear: Graph-enhanced agent for retrieval-augmented generation, 2024.
- [36] LlamaIndex. Introducing agentic document workflows. <https://www.llamaindex.ai/blog/introducing-agentic-document-workflows>, 2025. Accessed: 2025-01-13.
- [37] AWS Machine Learning Blog. How twitch used agentic workflow with rag on amazon bedrock to supercharge ad sales. <https://aws.amazon.com/blogs/machine-learning/how-twitch-used-agentic-workflow-with-rag-on-amazon-bedrock-to-supercharge-ad-sales/>, 2025. Accessed: 2025-01-13.

- [38] LlamaCloud Demo Repository. Patient case summary workflow using llamacloud. https://github.com/run-llama/llamacloud-demo/blob/main/examples/document_workflows/patient_case_summary/patient_case_summary.ipynb, 2025. Accessed: 2025-01-13.
- [39] LlamaCloud Demo Repository. Contract review workflow using llamacloud. https://github.com/run-llama/llamacloud-demo/blob/main/examples/document_workflows/contract_review/contract_review.ipynb, 2025. Accessed: 2025-01-13.
- [40] LlamaCloud Demo Repository. Auto insurance claims workflow using llamacloud. https://github.com/run-llama/llamacloud-demo/blob/main/examples/document_workflows/auto_insurance_claims/auto_insurance_claims.ipynb, 2025. Accessed: 2025-01-13.
- [41] LlamaCloud Demo Repository. Research paper report generation workflow using llamacloud. https://github.com/run-llama/llamacloud-demo/blob/main/examples/report_generation/research_paper_report_generation.ipynb, 2025. Accessed: 2025-01-13.
- [42] LangGraph Agentic RAG Tutorial. Langgraph agentic rag: Nodes and edges tutorial. https://langchain-ai.github.io/langgraph/tutorials/rag/langgraph_agentic_rag/#nodes-and-edges. Accessed: 2025-01-14.
- [43] LlamaIndex Blog. Agentic rag with llamaindex. <https://www.llamaindex.ai/blog/agentic-rag-with-llamaindex-2721b8a49ff6>. Accessed: 2025-01-14.
- [44] Hugging Face Cookbook. Agentic rag: Turbocharge your retrieval-augmented generation with query reformulation and self-query. https://huggingface.co/learn/cookbook/en/agent_rag. Accessed: 2025-01-14.
- [45] Qdrant Blog. Agentic rag: Combining rag with agents for enhanced information retrieval. <https://qdrant.tech/articles/agentic-rag/>. Accessed: 2025-01-14.
- [46] crewAI Inc. crewai: A github repository for ai projects. <https://github.com/crewAIInc/crewAI>, 2025. Accessed: 2025-01-15.
- [47] AG2AI Contributors. Ag2: A github repository for advanced generative ai research. <https://github.com/ag2ai/ag2>, 2025. Accessed: 2025-01-15.
- [48] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. 2023.
- [49] Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. Training language model agents without modifying language models. *ICML'24*, 2024.
- [50] OpenAI. Swarm: Lightweight multi-agent orchestration framework. <https://github.com/openai/swarm>. Accessed: 2025-01-14.
- [51] LlamaIndex Documentation. Agentic rag using vertex ai. https://docs.llamaindex.ai/en/stable/examples/agent/agentic_rag_using_vertex_ai/. Accessed: 2025-01-14.
- [52] Microsoft. Semantic kernel overview, 2025. <https://learn.microsoft.com/en-us/semantic-kernel/overview/>. Accessed: February 2, 2025.
- [53] Microsoft. Semantic kernel github repository, 2025. <https://github.com/microsoft/semantic-kernel>. Accessed: February 2, 2025.
- [54] IBM Granite Community. Agentic rag: Ai agents with ibm granite models. https://github.com/ibm-granite-community/granite-snack-cookbook/blob/main/recipes/AI-Agents/Agentic_RAG.ipynb. Accessed: 2025-01-14.
- [55] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, 2021.
- [56] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco: A human generated machine reading comprehension dataset, 2018.
- [57] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. Overview of the trec 2022 deep learning track. In *Text REtrieval Conference (TREC)*. NIST, TREC, March 2023.
- [58] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition, 2022.
- [59] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps, 2020.

- [60] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018.
- [61] Robert Friel, Masha Belyi, and Atindriyo Sanyal. Ragbench: Explainable benchmark for retrieval-augmented generation systems, 2024.
- [62] David Rau, Hervé Déjean, Nadezhda Chirkova, Thibault Formal, Shuai Wang, Vassilina Nikoulina, and Stéphane Clinchant. Bergen: A benchmarking library for retrieval-augmented generation, 2024.
- [63] Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. Flashrag: A modular toolkit for efficient retrieval-augmented generation research, 2024.
- [64] Costas Mavromatis and George Karypis. Gnn-rag: Graph neural retrieval for large language model reasoning, 2024.
- [65] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.
- [66] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017.
- [67] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.
- [68] Jonathan Berant, Andrew K. Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Conference on Empirical Methods in Natural Language Processing*, 2013.
- [69] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [70] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. Eli5: Long form question answering, 2019.
- [71] Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. 2017.
- [72] Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. Asqa: Factoid questions meet long-form answers, 2023.
- [73] Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. QMSum: A new benchmark for query-based multi-domain meeting summarization. pages 5905–5921, June 2021.
- [74] Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online, June 2021. Association for Computational Linguistics.
- [75] Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. COVID-QA: A question answering dataset for COVID-19. In *ACL 2020 Workshop on Natural Language Processing for COVID-19 (NLP-COVID)*, 2020.
- [76] Xidong Wang, Guiming Hardy Chen, Dingjie Song, Zhiyi Zhang, Zhihong Chen, Qingying Xiao, Feng Jiang, Jianquan Li, Xiang Wan, Benyou Wang, and Haizhou Li. Cmb: A comprehensive medical benchmark in chinese, 2024.
- [77] Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. Quality: Question answering with long input texts, yes!, 2022.
- [78] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

- [79] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering, 2024.
- [80] Sha Li, Heng Ji, and Jiawei Han. Document-level event argument extraction by conditional generation, 2021.
- [81] Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. Multi-sentence argument linking, 2020.
- [82] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents, 2019.
- [83] Hongru Wang, Minda Hu, Yang Deng, Rui Wang, Fei Mi, Weichao Wang, Yasheng Wang, Wai-Chung Kwan, Irwin King, and Kam-Fai Wong. Large language models as source planner for personalized knowledge-grounded dialogue, 2023.
- [84] Xinchao Xu, Zhibin Gou, Wenquan Wu, Zheng-Yu Niu, Hua Wu, Haifeng Wang, and Shihang Wang. Long time no see! open-domain conversation with long-term persona memory, 2022.
- [85] Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. Conditional generation and snapshot learning in neural dialogue systems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2153–2162, Austin, Texas, November 2016. Association for Computational Linguistics.
- [86] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW ’16, page 507–517, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee.
- [87] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics.
- [88] Seungone Kim, Se June Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning, 2023.
- [89] Amrita Saha, Vardaan Pahuja, Mitesh M. Khapra, Karthik Sankaranarayanan, and Sarath Chandar. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. 2018.
- [90] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.
- [91] Neema Kotonya and Francesca Toni. Explainable automated fact-checking for public health claims, 2020.
- [92] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies, 2021.
- [93] Hiroaki Hayashi, Prashant Budania, Peng Wang, Chris Ackerson, Raj Neervannan, and Graham Neubig. Wikiasp: A dataset for multi-domain aspect-based summarization, 2020.
- [94] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don’t give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization, 2018.
- [95] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.
- [96] Sourav Saha, Jahedul Alam Junaed, Maryam Saleki, Arnab Sen Sharma, Mohammad Rashidujjaman Rifat, Mohamed Rahouti, Syed Ishtiaque Ahmed, Nabeel Mohammed, and Mohammad Ruhul Amin. Vio-lens: A novel dataset of annotated social network posts leading to different forms of communal violence and its evaluation. In Firoj Alam, Sudipta Kar, Shammur Absar Chowdhury, Farig Sadeque, and Ruhul Amin, editors, *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 72–84, Singapore, December 2023. Association for Computational Linguistics.

- [97] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search, 2020.
- [98] Nandan Thakur, Luiz Bonifacio, Xinyu Zhang, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Boxing Chen, Mehdi Rezagholizadeh, and Jimmy Lin. "knowing when you don't know": A multilingual relevance assessment dataset for robust retrieval-augmented generation, 2024.
- [99] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.
- [100] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.
- [101] Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, Dan Tuviš, and Dániel Varga. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In Nicoletta Calzolari, Khalid Choukri, Aldo Gangemi, Bente Maegaard, Joseph Mariani, Jan Odijk, and Daniel Tapia, editors, *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May 2006. European Language Resources Association (ELRA).