

What is agentic RAG?

Agentic RAG is the use of [AI agents](#) to facilitate [retrieval augmented generation \(RAG\)](#). Agentic RAG systems add AI agents to the RAG pipeline to increase adaptability and accuracy. Compared to traditional RAG systems, agentic RAG allows [large language models \(LLMs\)](#) to conduct [information retrieval](#) from multiple sources and handle more complex [workflows](#).

What is RAG?

Retrieval augmented generation is an [artificial intelligence \(AI\)](#) application that connects a [generative AI](#) model with an external knowledge base. The data in the knowledge base augments user queries with more context so the LLM can generate more accurate responses. RAG enables LLMs to be more accurate in domain-specific contexts without needing [fine-tuning](#).

Rather than rely solely on training data, RAG-enabled [AI models](#) can access current data in real time through [APIs](#) and other connections to data sources. A standard RAG pipeline comprises two AI models:

- The information retrieval component, typically an [embedding](#) model paired with a [vector database](#) containing the data to be retrieved.
- The [generative AI](#) component, usually an LLM.

In response to [natural language](#) user queries, the embedding model converts the query to a vector embedding, then retrieves similar data from the knowledge base. The AI system combines the retrieved data with the user query for context-aware response generation.

What is agentic AI?

Agentic AI is a type of AI that can determine and carry out a course of action by itself. Most agents available at the time of publishing are LLMs with function-calling capabilities, meaning that they can call tools to perform tasks. In theory, AI agents are LLMs with three significant characteristics:

- They have **memory**, both short and long term, which enables them to plan and execute complex tasks. Memory also allows agents to refer to previous tasks and use that data to inform future workflows. Agentic RAG systems use semantic caching to store and refer to previous sets of queries, context and results.
- They are capable of query **routing**, step-by-step **planning** and decision-making. Agents use their memory capabilities to retain information and plot an appropriate course of action in response to complex queries and prompts.
- They can perform **tool calling** through APIs. More capable agents can choose which tools to use for the workflow they generate in response to user interactions.

Agentic workflows can consist of either one AI agent or multiagent systems that combine several agents together.

Industry newsletter

The latest AI trends, brought to you by experts

Get curated insights on the most important—and intriguing—AI news. Subscribe to our weekly Think newsletter. See the [IBM Privacy Statement](#).

We use your email to validate you are who you say you are, to create your IBMid, and to contact you for account related matters.

Business email

Your subscription will be delivered in English. You will find an unsubscribe link in every newsletter. You can manage your subscriptions or unsubscribe [here](#). Refer to our [IBM Privacy Statement](#) for more information.

Your subscription will be delivered in English. You will find an unsubscribe link in every newsletter. You can manage your subscriptions or unsubscribe [here](#). Refer to our [IBM Privacy Statement](#) for more information.

Agentic RAG vs. traditional RAG systems

Agentic RAG brings several significant improvements over traditional RAG implementation:

- **Flexibility:** Agentic RAG applications pull data from multiple external knowledge bases and allow for external tool use. Standard RAG pipelines connect an LLM to a single external dataset. For example, many enterprise RAG systems pair a [chatbot](#) with a knowledge base containing proprietary organization data.
- **Adaptability:** Traditional RAG systems are reactive data retrieval tools that find relevant information in response to specific queries. There is no ability for the RAG system to adapt to changing contexts or access other data. Optimal results often require extensive [prompt engineering](#).

Meanwhile, agentic RAG is a transition from static rule-based querying to adaptive, intelligent problem-solving. Multiagent systems encourage multiple AI models to collaborate and check each other's work.

- **Accuracy:** Traditional RAG systems do not validate or optimize their own results. People must discern whether the system is performing at an acceptable standard. The system itself has no way of knowing whether it is finding the right data or successfully incorporating it to facilitate context-aware generation. However, AI agents can iterate on previous processes to optimize results over time.
- **Scalability:** With networks of RAG agents working together, tapping into multiple external data sources and using tool-calling and planning capabilities, agentic RAG has greater scalability. Developers can construct flexible and scalable RAG systems that can handle a wide range of user queries.
- **Multimodality:** Agentic RAG systems benefit from recent advancements in multimodal LLMs to work with a greater range of data types, such as images and audio files. Multimodal models process multiple types of [structured, semistructured and unstructured data](#). For example, several recent [GPT](#) models can generate visual and audio content in addition to standard text generation.

Consider several employees working in an office. A traditional RAG system is the employee who performs well when given specific tasks and told how to accomplish them. They are reluctant to take initiative and feel uncomfortable going outside explicit instructions.

In comparison, an agentic RAG system is a proactive and creative team. They are also good at following directions but love to take initiative and solve challenges on their own. They are unafraid to come up with their own solutions to complex tasks that might stump or intimidate their coworkers.

Is agentic RAG better than traditional RAG?

While agentic RAG optimizes results with function calling, multistep reasoning and multiagent systems, it isn't always the better choice. More agents at work mean greater expenses, and an agentic RAG system usually require paying for more tokens. While agentic RAG can increase speed over traditional RAG, LLMs also introduce latency because it can take more time for the model to generate its outputs.

Lastly, agents are not always reliable. They might struggle and even fail to complete tasks, depending on the complexity and the agents used. Agents do not always collaborate smoothly and can compete over resources. The more agents in a system, the more complex the collaboration becomes, with a higher chance for complications. And even the most airtight RAG system cannot eliminate the potential for hallucinations entirely.

How does agentic RAG work?

Agentic RAG works by incorporating one or more types of AI agents into RAG systems. For example, an agentic RAG system might combine multiple information retrieval agents, each specialized in a certain domain or type of data source. One agent queries external databases while another can comb through emails and web results.

[Agentic AI frameworks](#), such as [LangChain](#) and [LlamaIndex](#), and the orchestration framework [LangGraph](#) can be found on GitHub. With them, it is possible to experiment with [agentic architectures](#) for RAG at minimal costs. If using [open source](#) models such as [Granite™](#) or Llama-3, RAG system designers can also mitigate the fees demanded by other providers such as OpenAI while enjoying greater [observability](#).

Agentic RAG systems can contain one or more types of AI agents, such as:

- Routing agents
- Query planning agents
- ReAct agents
- Plan-and-execute agents

Routing agents

Routing agents determine which external knowledge sources and tools are used to address a user query. They process user prompts and identify the RAG pipeline most likely to result in optimal response generation. In a single-agent RAG system, a routing agent chooses which data source to query.

Query planning agents

Query planning agents are the task managers of the RAG pipeline. They process complex user queries to break them down into step-by-step processes. They submit the resulting subqueries to the other agents in the RAG system, then combine the responses for a cohesive overall response. The process of using one agent to manage other AI models is a type of [AI orchestration](#).

ReAct agents

ReAct (reasoning and action) is an agent framework that creates [multiagent systems](#) that can create and then act on step-by-step solutions. They can also identify appropriate tools that can help. Based on the results of each step, ReAct agents can dynamically adjust subsequent stages of the generated workflow.

Plan-and-execute agents

Plan-and-execute agent frameworks are a progression from ReAct agents. They can execute multistep workflows without calling back to the primary agent, reducing costs and increasing efficiency. And because the planning agent must reason through all the steps needed for a task, completion rates and quality tend to be higher.

Agentic RAG use cases

While agentic RAG can suit any traditional RAG application, the greater compute demands make it more appropriate for situations that require querying multiple data sources. Agentic RAG applications include:

- **Real-time question-answering:** Enterprises can deploy RAG-powered chatbots and FAQs to provide employees and customers with current, accurate information.
- **Automated support:** Businesses wanting to streamline customer support services can use automated RAG systems to handle simpler customer inquiries. The agentic RAG system can escalate more demanding support requests to human personnel.
- **Data management:** RAG systems make it easier to find information within proprietary data stores. Employees can quickly get the data they need without having to sort through databases themselves.