Department for Science, Innovation & Technology

Government Digital Service

AI Insights

Agentic RAG

What is RAG?

Retrieval Augmented Generation (RAG) is a mechanism used to make additional data available to a large language model (LLM) so that data can be incorporated into queries.

This is a very common use case in both public and private sector systems. For more information, see our Insight article on RAG [LINK].

In brief, RAG works by slicing documents into chunks, converting those chunks into a numerical format (known as embeddings), and storing them in a special kind of database called a vector database. This vector database is optimised for high-speed similarity searches so that you can quickly find data that matches your queries.

If you wanted to query your own data, you could use those embeddings to quickly locate matching information within your vector database and then include those results in the prompt sent to the LLM.

The LLM is a natural language processing expert, but it has no knowledge apart from its training data, so providing this data in this manner allows you to use the power of the LLM to examine and analyse that information.

While this is a simple overview of RAG that does not cover all of its processes, this is how most RAG systems work in general.

What is agentic RAG?

Agentic RAG transforms the traditional RAG process by introducing a novel degree of autonomy. This allows the artificial intelligence (AI) to plan, reflect and iterate over the data, and integrate retrieval updates until it is satisfied.

The traditional RAG is a process which will retrieve data, embed a query and send it to the LLM. An example of this would be generating a report from data it has retrieved from the vector store.

Agentic RAG, on the other hand, is an iterative, goal-driven refinement process. The agent may start with a report template and then iterate over it, populating and refining sections until it is complete.

If there are any gaps or obsolete information, agentic RAG can issue more specific and nuanced instructions to the LLM, along with the conversation history and the missing or obsolete information required. The agentic system will then incorporate the response into subsequent drafts of the report. This is a process of iterative refinement.

How does agentic RAG work?

If we examine the above scenario which uses the report template - for example, describing trends in renewable energy - the process might take the following steps:

1. Initial Draft

The agent begins by retrieving relevant documents from the database based on an initial query which it retrieves from the template. It then generates a first draft of the report, covering key areas like market growth, technology trends, and policy impacts.

Initially, simple validation functions, which do not rely on the LLM, are executed. These ensure that mandatory fields are populated, or that certain fields are detected as specific data types such as dates or numbers, and so on. If any of these rules fail, a validation error is generated.

2. Gap Identification

As the agent reviews the draft, it notices that the section on government policy is sparse and potentially obsolete. It is able to do this because more elaborate analysis functions have been registered with the report template. The most appropriate ones for each section are selected by the LLM as it reviews the document.

3. Iterative Refinement

Recognising this gap, the agentic RAG system autonomously issues a follow-up query, now specifically targeting recent policy updates. It retrieves the latest documents and integrates this new information into the report.

4. Finalisation

The system continues this cycle, evaluating the output for each section, querying for missing or updated data, and refining the report as necessary until it is confident that the final version is complete.

Risks

Agentic RAG systems introduce some additional, sometimes novel, risks. As with any new technology, these systems bring their own challenges. Awareness is key, and knowing how to manage and mitigate these issues is essential.

Among these risks are:

Unintended Iterative Loops

The autonomous, iterative process could potentially lead the system into feedback loops that reinforce errors or generate outputs that meander around the intended objective without converging.

To manage this scenario, it is important to implement robust termination criteria and monitoring mechanisms to detect when iterations are no longer improving the output. Maintaining state and passing audit history between invocations can help to prevent this by detecting unnecessary invocations.

Complexity and Debugging Challenges

Debugging is the process of removing errors from software processes. As agentic AI systems become more complex by integrating more powerful tools, functions, and even multi-agent scenarios, debugging becomes increasingly challenging.

In this case, it is important to design these systems with this complexity in mind. We mention in our agentic AI Insight, in a very similar scenario, that passing an audit object among all functions, tools, and agents can clarify the execution order and parameter values at each step, aiding in detailed logging and transparent debugging.

Data Quality and Bias Issues

Iterative retrieval can compound biases or misinformation present in the source data, leading to unreliable outputs. As the agent iterates over a section - for example, the government policy as above - it becomes vulnerable to steering the response based on the context.

In this case, extensive testing, data validation techniques, and regular updates of the knowledge base with high-quality, verified information is essential.

LLM Risks

If any parts of our agentic RAG system utilise an LLM to generate content, and not just to plan and route the flow of control, we are subject to the usual concerns regarding confabulation and bias.

It is also essential to implement bias detection and correction protocols as a final quality assurance arbiter. Different underlying models will have different bias profiles, so if we change the agent model, we will likely have to re-evaluate our agentic RAG system.

Testing is vital, especially as systems become more complex. This applies to agentic RAG as much as any other system. It is important to know and quantify, where possible, the limitations of any system.

In our agentic AI Insight we further discuss risks and mitigations of agentic systems in detail.

Is agentic RAG the future?

The straightforward answer is that it is not likely we will rely solely on agentic RAG in the future. Traditional RAG systems work extremely well over a great many use cases. They are simple, fast and, as they involve a single query to a retrieval component (like a vector database or search index), they incur relatively low token costs. will continue to play a crucial role alongside agentic approaches.

Agentic RAG is a more contemplative approach involving multiple rounds of retrieval and improvement. It will take longer, it may also be harder to optimise and refine. It may be more difficult to analyse and debug, and will cost more to execute overall.

Agentic RAG is another intelligent instrument in our toolset. It solves problems that traditional RAG simply cannot. We can see from the example above that it is an extremely capable system, when used in the right areas of a system.

The challenge for system designers will come when their RAG solutions intersect both traditional and agentic types. For example, if you find yourself making your vanilla RAG system more elaborate in order to handle rare or complex queries, that might be a reason to investigate an agentic solution for those special cases.

Ultimately, the choice between traditional and agentic RAG should be driven by the specific needs of the use case, balancing speed and simplicity against depth and iterative refinement.