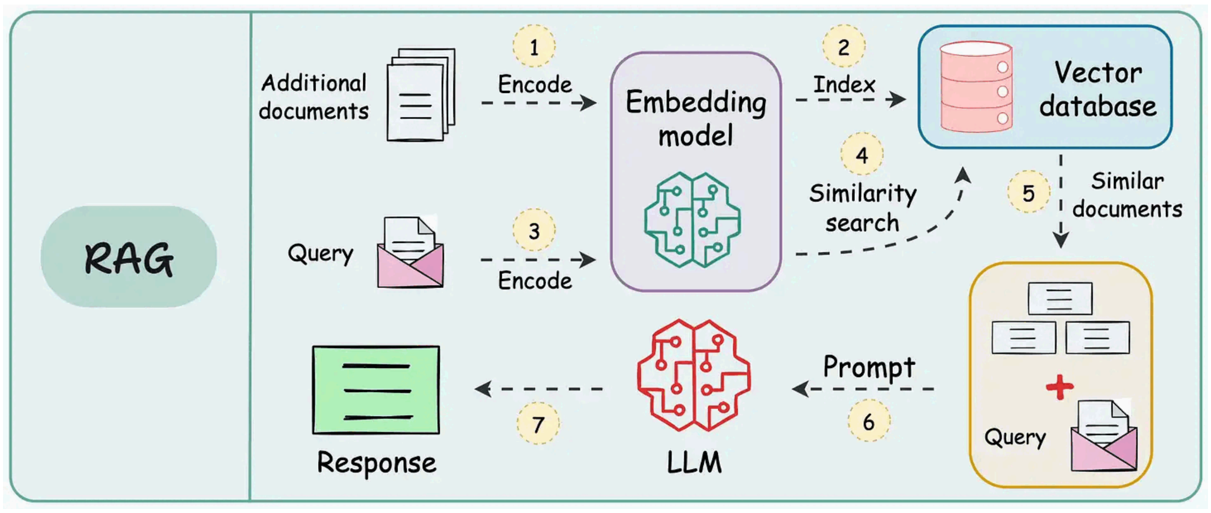
Zaid Ahmed Khan

15th December 2024

Basic RAG Architecture

Before we move on to complex RAG architecture let's have a look at the basic RAG architecture in brief.

The basic RAG architecture utilises a dual-component(retriever+generator) strategy that markedly enhances the text generation process through the integration of retrieval capabilities directly into the model. This approach ensures that responses are more informed and contextually relevant by basing the content on verified data. The retriever and generator components of the RAG architecture collaborate synergistically to refine the text generation process. This collaborative functionality permits the production of high-quality, accurate, and contextually appropriate responses, rendering the RAG architecture a potent tool in applications such as chatbots, question-answering systems, and other AI-driven applications that demand a sophisticated level of language understanding and response generation.



(Gif Src: https://www.dailydoseofds.com/a-crash-course-on-building-rag-systems-part-1-with-implementations/)
In the above gif, Parts 1 to 5 are basically the retrieval part that is retrieving the relevant documents and Parts 6 to 7 are the generator part.

Type of RAG Architectures :

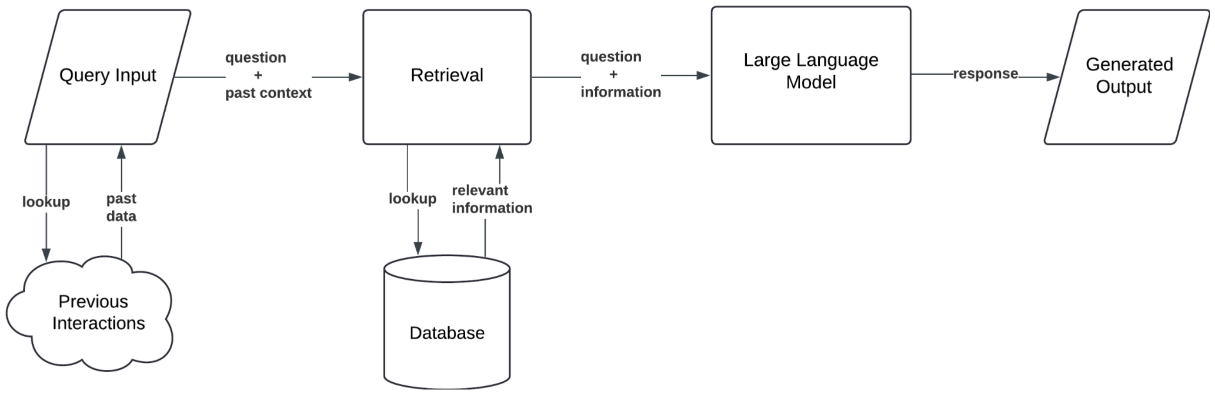
There exist many modifications to RAG that can be done, I have covered some of the major and most used types.

1: Simple/Basic RAG

The basic RAG which we discussed above.

2. Simple RAG with Memory

The simple RAG can be improved by introducing a storage component that allows the model to retain information from previous interactions. This addition makes it more powerful for continuous conversations or tasks that require contextual awareness across multiple queries.

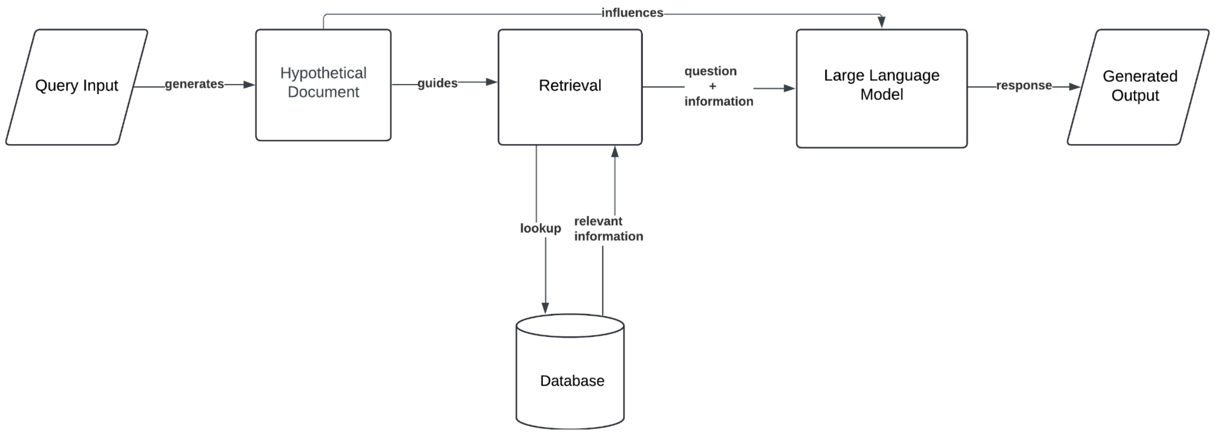


Workflow:

- **Query Input**: The user submits a query or prompt.
- **Memory Access**: The model retrieves past interactions or data stored in its memory.
- **Document Retrieval**: It searches the external database for new relevant information.
- **Generation**: The model generates a response by combining retrieved documents with the stored memory.

3. HyDe (Hypothetical Document Embedding)

HyDe (Hypothetical Document Embedding) is a unique RAG variant that generates hypothetical documents based on the query before retrieving relevant information. Instead of directly retrieving documents from a database, HyDe first creates an embedded representation of what an ideal document might look like, given the query. It then uses this hypothetical document to guide retrieval, improving the relevance and quality of the results.

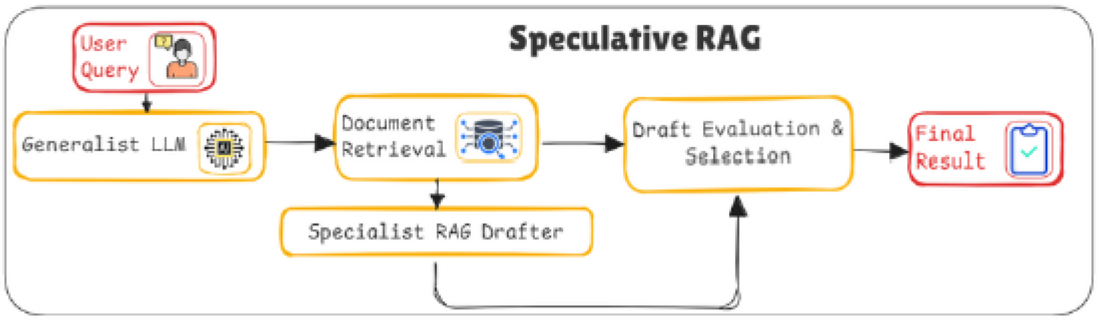


Workflow:

- **Query Input**: The user provides a prompt or question.
- **Hypothetical Document Creation**: The model generates an embedded representation of an ideal response.
- **Document Retrieval**: Using the hypothetical document, the model retrieves actual documents from a knowledge base.
- **Generation**: The model generates an output based on the retrieved documents, influenced by the hypothetical document.

4. Speculative RAG

Speculative RAG takes a different approach by encouraging the model to make educated guesses or speculative responses when the retrieved data is insufficient or ambiguous. This model is designed to handle scenarios where complete information may not be available, yet the system still needs to provide a useful response. The speculative aspect allows the model to generate plausible conclusions based on patterns in the retrieved data and the broader knowledge embedded in the language model.

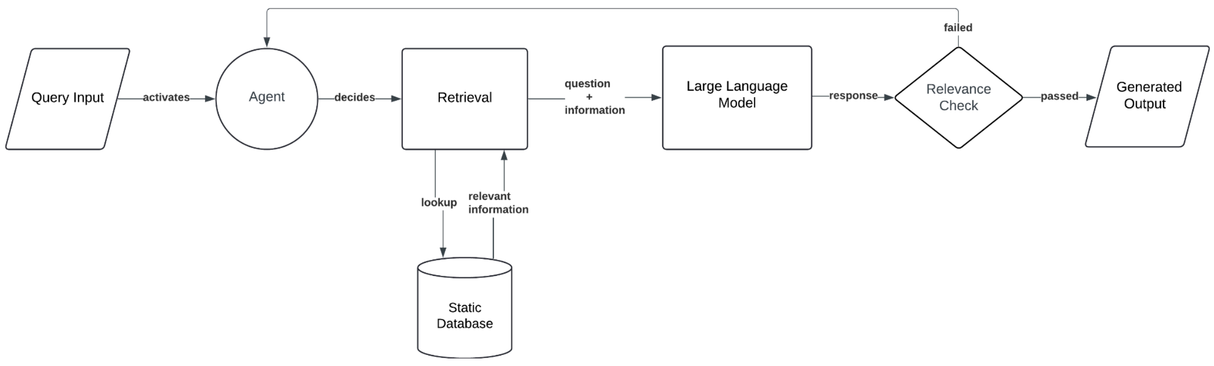


Workflow:

1. **User Query**: The user begins entering a query.
2. **Contextual Analysis**: The system analyzes the user's behavior and context.
3. **Predictive Data Retrieval**: Relevant data is pre-retrieved based on predicted needs.
4. **Speculative Response Generation**: Generates a tentative response.
5. **User Feedback Collection**: User interacts with the generated response.
6. **Refined Response Generation**: Response is updated based on feedback.
7. **Final Output to User**: The user receives a quick, relevant answer.

5. Agentic RAG

Agentic RAG introduces a more autonomous, agent-like behaviour in the retrieval and generation process. Here the model acts as an "agent" that can perform complex, multi-step tasks, proactively interacting with multiple data sources or APIs to gather information. What sets Agentic RAG apart is its ability to assign Document Agents to each individual document and orchestrate their interactions through a meta-agent. This system allows for more sophisticated decision-making, enabling the model to determine which retrieval strategies or external systems to engage with based on the complexity of the query.



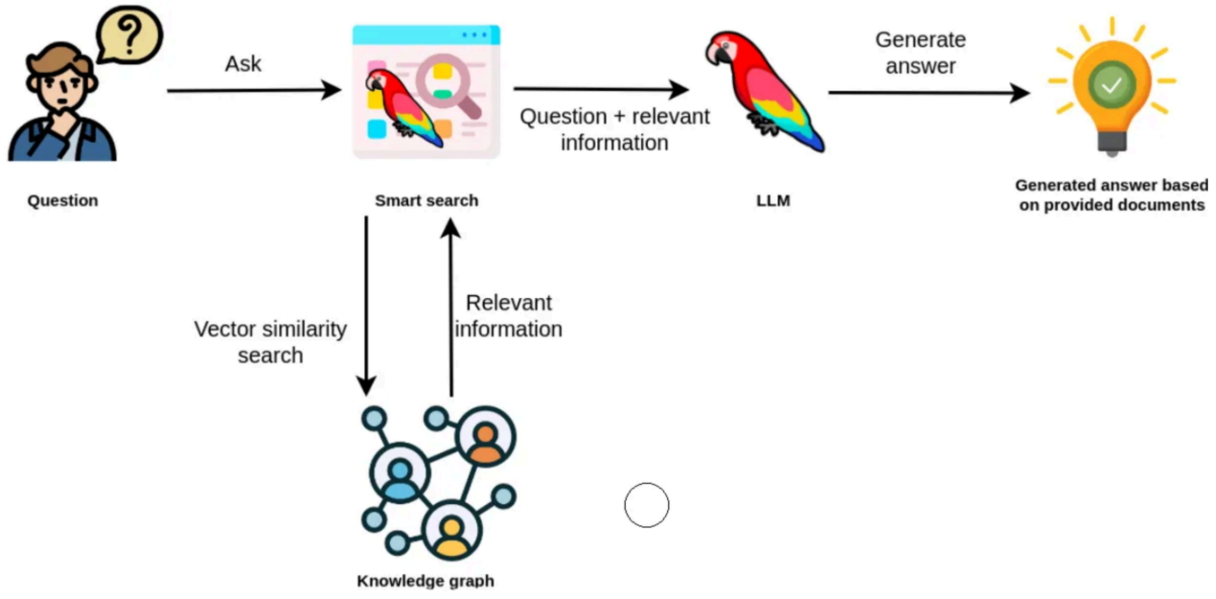
Workflow:

- **Query Input**: The user submits a complex query or task.
- **Agent Activation**: The model activates multiple agents. Each Document Agent is responsible for a specific document, capable of answering questions and summarising information from that document.
- **Multi-step Retrieval**: The Meta-Agent manages and coordinates the interactions between the various Document Agents, ensuring the most relevant information is retrieved.
- **Synthesis and Generation**: The Meta-Agent integrates the outputs from the individual Document Agents and generates a comprehensive, coherent response based on the collective insights gathered.

6. Graph RAG

Graph RAG incorporates graph-based data structures into the retrieval process, allowing the model to retrieve and organize information based on entity relationships. It is particularly useful in contexts where the data structure is crucial for understanding, such as knowledge graphs, social networks, or semantic web applications.

By leveraging graphs, the model can retrieve isolated information and their connections. For example, in a legal context, Graph RAG could retrieve relevant case law and the precedents that connect those cases, providing a more nuanced understanding of the topic.



Workflow:

- **Query Input**: The user submits a complex query or task that requires understanding intricate relationships and contextual connections.
- **Agent Activation**: The model activates a Graph Agent capable of mapping and traversing the graph-based data structure, identifying key entities and their relationships.
- **Multi-step Retrieval**: The Graph Agent manages a comprehensive traversal of the knowledge graph, exploring direct and indirect connections to retrieve the most relevant and contextually rich information.
- **Synthesis and Generation**: The Graph Agent integrates the retrieved information from various graph segments, generating a comprehensive response that maintains the semantic relationships and provides a nuanced understanding of the topic.

Some References :

- https://mlubbad.medium.com/top-6-different-rag-architectures-2efed3f8a868
- https://www.bluetickconsultants.com/blogs/from-rag-to-graphrag-transforming-information-retrieval-with-knowledge-graphs.html
- https://medium.com/@zbabar/design-variations-of-the-rag-architecture-9ff5d1d5b4de
- https://medium.com/@rupeshit/mastering-the-25-types-of-rag-architectures-when-and-how-to-use-each-one-2ca0e4b944d7