Retrieval-augmented generation (**RAG**) is a technique that enables large language models (LLMs) to retrieve and incorporate new information.[1] With RAG, LLMs do not respond to user queries until they refer to a specified set of documents. These documents supplement information from the LLM's pre-existing training data.[2] This allows LLMs to use domain-specific and/or updated information that is not available in the training data.[2][3] For example, this helps LLM-based chatbots access internal company data or generate responses based on authoritative sources.

RAG improves large language models (LLMs) by incorporating information retrieval before generating responses.[4] Unlike traditional LLMs that rely on static training data, RAG pulls relevant text from databases, uploaded documents, or web sources.[1] According to *Ars Technica*, "RAG is a way of improving LLM performance, in essence by blending the LLM process with a web search or other document look-up process to help LLMs stick to the facts." This method helps reduce AI hallucinations,[4][5] which have caused chatbots to describe policies that don't exist, or recommend nonexistent legal cases to lawyers that are looking for citations to support their arguments.[6]

RAG also reduces the need to retrain LLMs with new data, saving on computational and financial costs.[1] Beyond efficiency gains, RAG also allows LLMs to include sources in their responses, so users can verify the cited sources. This provides greater transparency, as users can cross-check retrieved content to ensure accuracy and relevance.

The term RAG was first introduced in a 2020 research paper[4] from Meta.[7][3]

RAG and LLM Limitations

LLMs can provide incorrect information. For example, when Google first demonstrated its LLM tool "Google Bard", the LLM provided incorrect information about the James Webb Space Telescope. This error contributed to a $100 billion decline in the company's stock value.[6] RAG is used to prevent these errors, but it does not solve all the problems. For example, LLMs can generate misinformation even when pulling from factually correct sources if they misinterpret the context.[8] *MIT Technology Review* gives the example of an AI-generated response stating, "The United States has had one Muslim president, Barack Hussein Obama." The model retrieved this from an academic book rhetorically titled *Barack Hussein Obama: America's First Muslim President?* The LLM did not "know" or "understand" the context of the title, generating a false statement.[2]

LLMs with RAG are programmed to prioritize new information. This technique has been called "prompt stuffing." Without prompt stuffing, the LLM's input is generated by a user; with prompt stuffing, additional relevant context is added to this input to guide the model's response. This approach provides the LLM with key information early in the prompt, encouraging it to prioritize the supplied data over pre-existing training knowledge.[9]

Process

Retrieval-augmented generation (RAG) enhances large language models (LLMs) by incorporating an information-retrieval mechanism that allows models to access and utilize additional data beyond their original training set. AWS states, "RAG allows LLMs to retrieve relevant information from external data sources to generate more accurate and contextually relevant responses" ("indexing").[10] This approach reduces reliance on static datasets, which can quickly become outdated. When a user submits a query, RAG uses a document retriever to search for relevant content from available sources before incorporating the retrieved information into the model's response ("retrieval").[11] *Ars Technica* notes that "when new information becomes available, rather than having to retrain the model, all that's needed is to augment the model's external knowledge base with the updated information" ("augmentation").[6] By dynamically integrating relevant data, RAG enables LLMs to generate more informed and contextually grounded responses ("generation").[5] IBM states that "in the generative phase, the LLM draws from the augmented prompt and its internal representation of its training data to synthesize an engaging answer tailored to the user in that instant.[1]

RAG key stages

Indexing

Typically, the data to be referenced is converted into LLM embeddings, numerical representations in the form of a large vector space.[8] RAG can be used on unstructured (usually text), semi-structured, or structured data (for example knowledge graphs).[12] These embeddings are then stored in a vector database to allow for document retrieval.[13]

Overview of RAG process, combining external documents and user input into an LLM prompt to get tailored output

Retrieval

Given a user query, a document retriever is first called to select the most relevant documents that will be used to augment the query.[2][4] This comparison can be done using a variety of methods, which depend in part on the type of indexing used.[1][12]

Augmentation

The model feeds this relevant retrieved information into the LLM via prompt engineering of the user's original query.[10][14] Newer implementations (as of 2023) can also incorporate specific augmentation modules with abilities such as expanding queries into multiple domains and using memory and self-improvement to learn from previous retrievals.[12]

Generation

Finally, the LLM can generate output based on both the query and the retrieved documents.[2][15] Some models incorporate extra steps to improve output, such as the re-ranking of retrieved information, context selection, and fine-tuning.[12]