Department for
Science, Innovation
& Technology

Government
Digital Service

AI Insights

RAG Systems

Introduction

Retrieval-augmented generation (RAG) is an AI Framework that complements AI models, and more specifically language models, by combining information retrieval techniques for context-specific information from varied external sources with the generation of multi-model content. This framework allows the models to improve the accuracy and the new generated results to increase relevance, reduce hallucinations and extend their capabilities beyond their trained data. RAG framework does not change the underlying learnt model and its weights; instead, it supplements the learnt model with additional external knowledge.

By combining retrieval (R) and generation (G) into a unified system, the RAG framework enables large language models (LLMs) to access a larger pool of knowledge beyond their static training data. The framework adds a semantic search capability by dynamically fetching data from relevant sources at query time using search algorithms, which serves as contextual information for the AI models. This approach mitigates knowledge limitations, improves factual accuracy, and enhances the adaptability of AI models, such as LLMs. We will focus on the RAG systems for LLMs for this article.

As LLMs continue to evolve, RAG represents a fundamental shift in how language models interact with knowledge, particularly for systems that are relied upon for precise, reliable, and domain-specific responses, for example government information websites.

What is RAG?

AI models are trained on static datasets, so they suffer from an inherent limitation of knowledge newness and domain-specific context. RAG framework is a powerful approach to enhancing the customisation and transparency of LLMs with the contextual relevance of AI-generated responses. RAG also complements LLMs by seamlessly integrating real-time information retrieval with their dynamic text generation capabilities.

RAG allows the underlying AI model to produce answers that are not only coherent and contextually relevant but also grounded in up-to-date, accurate information. Most importantly, because RAG retrieves new data dynamically while using it during the model inference, such as in the process of using a trained LLM to generate new text or other outputs based on a given input, this reduces the need to retrain the entire model whenever new information becomes available. As a result, the system is more adaptable, maintainable and cost effective.

RAG operates by combining user queries and documents into a shared semantic space, allowing the efficient retrieval of relevant external information through similarity matching using 3 basic principles.

Retrieval (or R)

The Retrieval functions by obtaining relevant information from a knowledge repository, such as data sources and documents, in response to a specific query which is usually applicable to the business processes or internal organisation data. This data is converted into numerical representations called embeddings or vector embeddings using an embedding model. Embeddings are represented as points in a multi-dimensional vector space, where similar data points are closer together to capture the semantic meaning and relationships within the data. Embeddings link the data's semantic similarity to the user query, allowing AI models to understand and process it effectively.

Augmentation (or A)

Augmentation improves the input query or prompt by incorporating relevant semantic details collected from the retrieved sources to enrich the model's understanding with additional context for the inference and generation of the output response.

Generation (or G)

Generation creates a more informed, contextually relevant, and precise response by utilising the generative capabilities of the model enhanced by the augmented input.

However, on top of these 3 basic principles, RAG-based systems involve additional steps while implementing effective and engaging AI-powered solutions, such as data preprocessing, vectorisation and indexing, query preprocessing, query embedding, and post-processing.

Retrieved data undergoes pre-processing, such as data tokenisation and stemming for LLMs. Data tokenisation is the process of breaking down retrieved text data into smaller, manageable units called tokens used for embeddings, and stemming, is a natural language technique used to reduce words to their base form, also known as the root form, to prepare it for the LLM.

LLMs typically have token limits, such as 35,000 tokens in GPT-4o, for the maximum number of combined input and output tokens known as context windows. This restricts the amount of text that the underlying model can process at one time. To tackle this, you need to break down significant raw text inputs into smaller, manageable pieces using a technique called chunking. Chunking is crucial for token management, semantic accuracy, context preservation for coherence, and processing efficiency.

Chunks are converted into embeddings and stored in the vector database. This enables the retrieval system to compare and retrieve relevant information quickly.

The benefits of RAG

One of the most significant advantages of RAG is its ability to reduce hallucinations. Since traditional models operate on fixed training data, they may fabricate facts when faced with unfamiliar queries, producing fully coherent and believable responses which are factually incorrect. The RAG system improves upon this issue by grounding the LLM's responses in authoritative knowledge sources without human annotation or intervention.

Conventional LLMs require frequent retraining to incorporate new knowledge, a process that is computationally expensive and time consuming. RAG eliminates this bottleneck by fetching relevant information on demand and allowing AI systems to process vast amounts of data without extensive retraining. This makes RAG an ideal solution for applications where information is constantly evolving, as it can seamlessly adapt to new data sources and ensure that its responses remain relevant over time, capturing real-time use cases. Cost-effective scalability is another key benefit of RAG.

RAG enables quick experimentation, benchmarking and comparison of various LLMs for your specific use case and data, eliminating the need for initial data training. This also helps avoid the expense and intricacy of pre-training or fine-tuning the LLMs while exposing them to your own context and proprietary data in a controlled environment.

RAG is particularly valuable in domain-specific applications. General-purpose LLMs often struggle to provide precise or reliable answers in specialised fields such as law, medicine, finance, and research due to their limited exposure to expert knowledge.

By enabling real-time access to domain-specific knowledge bases, RAG empowers LLMs to generate responses that align with expert insights and adhere to industry standards. This capability enhances the overall trust and reliability of LLM-based systems, which makes them more effective in complex and domain-specific environments.

RAG offers better control over the organisation's data and knowledge that the model is exposed to. You can easily accommodate your company's changing data policies and customers data changes. Unlearning a piece of data from a pre-trained model is expensive. With RAG, it is much easier to remove data points from the knowledge your LLM is exposed to.

RAG System Workflow

The standard workflow for a basic RAG system has 10 steps.

1. Data ingestion

Data (structured or unstructured) is ingested using a Data pipeline. The ingested data is preprocessed to encode it as text, if required. Preprocessing steps in the data pipeline are highly dependent on the underlying data type. For instance, audio data needs a transcription pipeline to convert the audio data into text, while ingestion of PDF documents or image files requires corresponding preprocessing techniques.

2. Index process

   Data is chunked and converted into embeddings. The original chunks and embeddings are then indexed and saved in a vector database. Splitting methods and the chunk size may affect the performance and context relevance, while the underlying data type drives the chunking strategy. During this step, the selection of the underlying embedding method is also crucial, as altering the chunking as well as the embedding strategy necessitates re-indexing all chunks. The embedding should be chosen for its capacity to retrieve accurate responses semantically, a decision influenced by chunk size, expected question types, content structure, and the application domain.

   Initial input (prompt + query)A user provides an initial prompt + query to the system. This is the starting point for the user interaction.

3. Query rewriter

   The query is preprocessed, or rewritten, to include a general context specific to the user, including their previous questions and specific interactions using an LLM.

4. Query embedding for relevant information

   An embedding is calculated for the rewritten query part of the initial input and is sent to a ‘search relevant information retriever component.

5. Retriever and reranker

   The retriever queries knowledge sources, which are represented by the organisations' internal documents and a database or data store saved in the form of a vector database. Relevant information for enhanced context is retrieved based on the query. Top-k similar documents are retrieved using a similarity method such as cosine similarity. Top-k then instructs the model to consider the top k most likely tokens when making a prediction, as opposed to the entire database vocabulary. Retrieved documents are re-ranked to prioritise the most relevant chunks.

6. Consolidator

   The consolidator processes these chunks to overcome limitations of LLMs, such as the token limit and rate limit. Due to token limits, a reduction strategy might be used to chain prompts and extract an answer. Rate limits on online services like OpenAI can impact system latency, prompting engineers to consider trade-offs in RAG system design.

7. Enriched input for LLM

   The original prompt + rewritten query is combined with the enhanced context (the relevant information retrieved in the previous step). This creates a more comprehensive input for the LLM.

8. LLM processing and response generation

   The combined prompt + rewritten query + enhanced context is sent to a large language model endpoint. The LLM processes this enriched input and generates a generated text response.

9. Response

   This generated text response is then presented back to the user or the originating system.

Challenges and limitations

While RAG systems complement LLMs, they also introduce several challenges and limitations. These factors impact the RAG system’s effectiveness, efficiency, and reliability in real-world applications.

Understanding and addressing these limitations is essential for developing robust and scalable systems.

Retrieval quality

One of the primary challenges of RAG is the quality of the retrieved knowledge. The system's ability to fetch relevant and accurate information depends on the effectiveness of the retriever.

Poorly-indexed documents, noisy data, or weak similarity measures can lead to the retrieval of irrelevant or misleading information. If the retrieved content lacks precision, the model generates incorrect responses, despite its overall generative strengths.

Hallucinations

Hallucinations remain a concern even in RAG-enhanced models. If the retriever returns incomplete or contradictory sources, the generative component may still fabricate details to produce a coherent response.

This issue undermines the reliability of LLM-driven responses, particularly in domains where factual accuracy is crucial, such as medicine and finance.

Model alignment

Model alignment and bias present additional challenges. The retrieved knowledge must align with the intended use case, but existing biases in training data or indexed sources can lead to skewed or unfair outputs.

If the retriever model prioritises sources with specific perspectives, the generated responses may reinforce pre-existing biases rather than provide balanced, objective insights. Addressing these biases requires careful curation of data sources and ongoing monitoring of LLM behaviour.

Latency and scalability

Latency and scalability present another significant limitation. RAG requires real-time information retrieval, which can cause delays in generating responses.

Querying large datasets and ranking relevant content require computational resources, which makes the system slower in comparison to LLM models alone. As the system's external knowledge grows, maintaining efficiency without sacrificing response times becomes increasingly complex, particularly for large-scale deployments.

Security and privacy

Security and privacy concerns can arise with RAG implementations. Since the system retrieves external knowledge dynamically, as with any other database, there is a risk of exposing sensitive or proprietary information.

If retrieval sources include unverified or malicious content, this may lead to the generation of harmful or deceptive responses. Protecting user data, ensuring secure access to knowledge bases, and minimising the risk of data leakage are crucial considerations for a safe deployment.

Advanced RAG systems

The transition towards the latest version of RAG (also known as RAG 2.0) reflects a broader trend in LLM development, where system components become more dynamic, context-aware, and capable of integrating structured knowledge.

By incorporating fully trainable RAG systems, contextual retrieval, and Hypothetical Document Embeddings (HyDE), which is a retrieval method that uses fake documents to improve generated answers, advanced RAG methods promise to elevate the performance and reliability of AI systems.

Multimodal RAG systems

Multimodal RAG expands traditional RAG systems by integrating multiple data types, including text, images, audio, video, and structured data.

While classic RAG systems combine text retrieval with language generation, the multimodal approach enables the LLM to generate richer and more informative responses by leveraging a diverse array of knowledge sources.

Unlike text, multimedia inputs require specialised processing techniques, such as image recognition, audio transcription, and video analysis.

During the retrieval, the system evaluates which modalities are most relevant to the query and combines information from multiple sources to construct a comprehensive response that maintains consistency and relevance.

This component offers significant opportunities for enhancing AI-driven interactions by making responses more dynamic and versatile. In practical applications, such as virtual assistants or professional tools, the system can provide multimedia-rich responses that improve accessibility and understanding.

RAG evaluation

Evaluating RAG systems involves assessing both the retrieval and generation components separately, as well as the overall performance of the integrated system. RAG systems present unique testing challenges. Often, there's a lack of existing test data designed explicitly for RAG evaluation. Testing typically involves an experimental approach, with the discovery of appropriate test cases happening through 2 different methods. The generation of synthetic data enables the system to be piloted with minimal initial testing and iteratively refined based on the results. At the same time, retrieval accuracy is measured through metrics such as precision and recall. It focusses on how well the retriever identifies relevant documents from the knowledge base.

In contrast, the accuracy of the LLM can be evaluated using generation metrics such as bilingual evaluation understudy (BLEU), Recall-Oriented Understudy for Gisting Evaluation (ROUGE) or Metric for Evaluation of Translation with Explicit Ordering (METEOR), which assess the quality, coherence, and contextual relevance of the AI-generated responses based on the retrieved information.
Retrieval benchmarks such as RAGBench, CLAPNQ, and TriviaQA provide standardised datasets that help compare RAG systems across different scenarios.

These benchmarks highlight how effectively the retriever component performs in sourcing relevant data across varied query types and domains.

Conclusion

RAG is more than just a helpful function - it empowers language models to dynamically adapt responses based on external insights to ensure domain-specific accuracy and relevance.

By dynamically retrieving information at query time, RAG complements the LLM's capacity to provide contextually aware answers that adapt to evolving knowledge landscapes without the need for frequent retraining.

While RAG introduces its own set of challenges, such as retrieval quality, latency, model alignment, and security considerations, most of these obstacles can be effectively mitigated through careful system design and implementation.

Furthermore, the development of advanced RAG systems enables models to maintain factual accuracy, reduce hallucinations, and ensure that an AI systems' output is aligned with recent and relevant information.

As LLMs continue to evolve, RAG systems are poised to play an essential role in delivering more grounded and trustworthy AI systems.