Advanced RAG Techniques

Learn advanced RAG methods like dense retrieval, reranking, or multi-step reasoning to tackle issues like hallucination or ambiguity.

Sep 30, 2024 · 12 min read

Stanislav Karzhev
Research Intern at UCL

TOPICS

Artificial Intelligence

Large Language Models

Retrieval-augmented generation (RAG) combines document retrieval with natural language generation, creating more accurate and contextually aware responses.

While basic RAG is effective, it struggles with complex queries, hallucinations, and maintaining context in multi-turn conversations.

In this blog, I'll explore advanced techniques that address these challenges by improving retrieval accuracy, generation quality, and overall system performance.

If you're reading this in preparation for an interview, be sure to check out the Top 30 RAG Interview Questions and Answers article.

Limitations of Basic RAG Systems

While basic RAG implementations can be useful, they do have their limitations, especially when applied in more demanding contexts.

Hallucination

One of the most prominent issues is hallucination, where the model generates content that is factually incorrect or unsupported by the retrieved documents. This can undermine the system's reliability, especially in fields requiring high accuracy, such as medicine or law.

Lack of domain specificity

Standard RAG models may struggle when dealing with domain-specific queries. Without tailoring retrieval and generation processes to the nuances of specialized domains, the system risks retrieving irrelevant or inaccurate information.

Handling complex or multi-turn conversations

Another challenge is managing complex, multi-step queries or multi-turn conversations. Basic RAG systems often struggle to maintain context across interactions, leading to disjointed or incomplete answers. As user queries grow more complex, RAG systems need to evolve to handle this growing complexity.

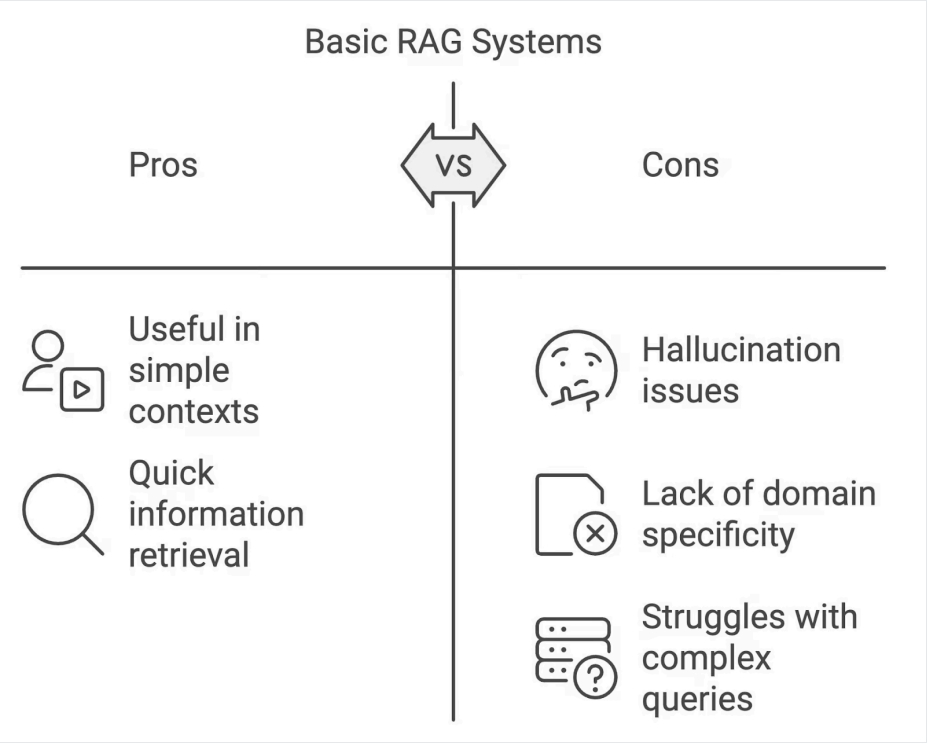


Diagram generated with napkin.ai

Advanced Retrieval Techniques

Advanced retrieval techniques focus on enhancing both the relevance and scope of retrieved documents. These techniques, which include dense retrieval, hybrid search, reranking, and query expansion, address the limitations of keyword-based retrieval.

Dense retrieval and hybrid search

Dense retrieval and hybrid search are key techniques for improving retrieval accuracy and relevance. Methods like TF-IDF or BM25 often struggle with semantic understanding when queries are phrased differently from the documents.

Dense retrieval, such as DPR (Dense Passage Retrieval), uses deep learning to map queries and documents into dense vector representations, capturing the meaning of text beyond exact keywords.

Hybrid search blends sparse and dense retrieval, balancing precision and recall by combining keyword-based matching with semantic similarity, making it effective for more complex queries.

Reranking

Reranking is another advanced technique used to refine the list of retrieved documents before they are passed to the generation component. In a typical RAG system, the initial retrieval phase might produce a large set of documents that vary in relevance.

The role of reranking is to reorder these documents so that the most relevant ones are prioritized for use by the language model. Reranking can be achieved from a simple scoring based on query-document similarity to more complex machine learning models trained to predict the relevance of each document.

You can learn how to implement reranking in this tutorial on reranking with RankGPT.

Query expansion

Query expansion involves enriching the user's query with additional terms that increase the chances of retrieving relevant documents. It can be achieved through:

- **Synonym expansion:** Adding synonyms or closely related terms to the original query

For example, if the original query is “artificial intelligence in healthcare,” query expansion might include related terms like “AI,” “machine learning,” or “health tech,” ensuring a wider retrieval net.

Optimizing Relevance and Quality in RAG Systems

In RAG systems, simply retrieving documents is not enough, ensuring the relevance and quality of these documents is key to improving the final output. To this end, advanced techniques that refine and filter the retrieved content are vital.

These methods work to reduce noise, boost relevance, and focus the language model on the most important information during the generation process.

Advanced filtering techniques

Advanced filtering techniques use metadata or content-based rules to exclude irrelevant or low-quality documents, ensuring that only the most relevant results are passed on.

- **Metadata-based filtering**: Documents may be filtered based on metadata like date, author, domain, or document type. In legal or medical applications this can ensure that only the most recent or authoritative sources are used.
- **Content-based filtering**: This evaluates the content of the documents themselves, applying rules to exclude those that don't meet certain relevance thresholds. It could also involve filtering out documents with low semantic similarity to the query or documents that don’t include key phrases or terms related to the query.

Context distillation

Context distillation is the process of summarizing or condensing retrieved documents to focus the language model on the most important pieces of information. This is useful when the retrieved documents contain too much irrelevant content or when the query involves complex, multi-step reasoning.

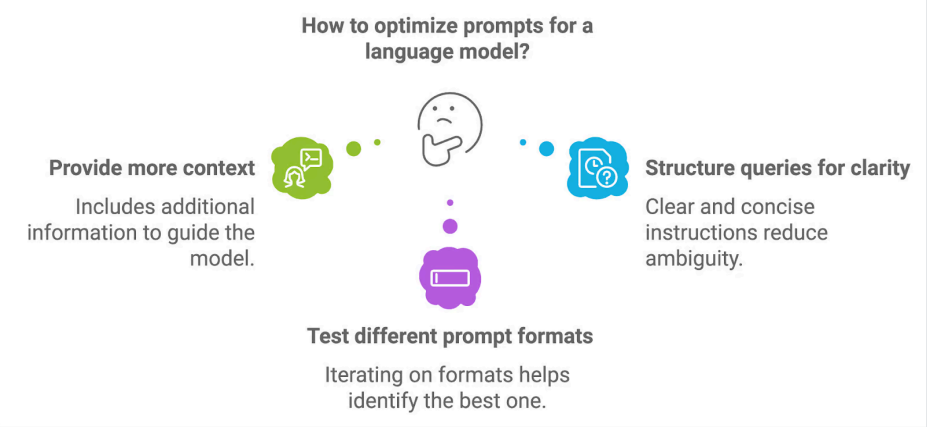
By distilling the context, the system extracts the key insights and most relevant passages from the retrieved documents, ensuring that the language model has the clearest and most pertinent information to work with.

Generation Process Optimization

Once relevant documents have been retrieved and refined, the next step in a RAG system is the generation process. Optimizing how the language model generates responses is essential to establishing accuracy, coherence, and relevance.

Prompt engineering

Prompt engineering refers to the process of designing and structuring the prompts that are fed into the language model. The quality of the prompt directly impacts the quality of the model’s generated output, as the prompt provides the initial instructions or context for the generation task.



Including additional information, such as explicit instructions or key terms, can guide the model toward more accurate and contextually relevant responses. For example, in a medical RAG system, a prompt might explicitly request that the model provide a diagnosis summary based on retrieved documents.

Structuring queries for clarity

Well-structured prompts, with clear and concise instructions, help reduce ambiguity and lead to more focused generation results. Phrasing the prompt as a direct question or request can often yield better outcomes.

Testing different prompt formats

Iterating on prompt formats such as rephrasing questions, adjusting the level of specificity, or providing examples can help identify the format that delivers the best results for a specific use case.

Learn more in this blog: Prompt Optimization Techniques.

Multi-step reasoning

Many queries particularly in areas like research, law, or technical support, involve multiple steps or require complex reasoning. Multi-step reasoning is the process by which a system breaks down a complex query into manageable sub-tasks and processes them sequentially to arrive at a comprehensive answer.

We can implement multi-step reasoning within a RAG system in numerous ways:

- **Chaining retrieval and generation**: In some cases, multi-step reasoning can be achieved by chaining retrieval and generation steps together. After processing an initial query, the system might generate a follow-up query or request additional information before generating the final answer.
- **Incorporating intermediate steps**: For queries requiring reasoning across multiple documents or topics, the system might retrieve different sets of documents for each step, gradually building up a more nuanced and complete answer.
- **Multi-hop question answering**: This approach allows the system to make logical connections across different pieces of retrieved information, enabling it to handle more sophisticated queries that involve relationships between various facts or data points.

Addressing hallucination

As briefly mentioned, one of the main challenges in generation models, including those used in RAG systems, is hallucination. Several techniques can help mitigate hallucinations in RAG systems:

- **Grounding on retrieved documents**: One of the most effective ways to reduce hallucinations is to ensure that the generation model is tightly grounded in the retrieved content. This means conditioning the model to only generate responses based on the actual content in the documents, rather than relying on external, pre-trained knowledge.
- **Context conditioning**: By refining how context is provided to the model, developers can better control the generation process. This might involve filtering out irrelevant parts of the retrieved documents before passing them to the model or providing specific instructions that guide the model to focus on key information.
- **Feedback loops**: Implementing a feedback mechanism where the system checks generated outputs against the retrieved documents for accuracy can help catch hallucinations before they reach the user. This additional verification step can significantly improve the reliability of the system.

Handling Complex Queries and Conversations

As RAG systems are increasingly applied to real-world tasks, they must be able to handle

One of the primary challenges in conversational RAG systems is managing the flow of information across multiple interactions. In many everyday scenarios, such as customer support or ongoing technical discussions users often engage in multi-turn conversations where the context must be maintained across several exchanges.

Having the system track and remember relevant parts of the conversation is key to providing coherent and consistent responses. To handle multi-turn conversations effectively, RAG systems can use the following techniques:

- **Conversation history tracking:** Maintaining a structured representation of the conversation's history plays an important role. This can involve saving key interactions, such as previous queries and generated responses, to be used as context in subsequent turns.
- **Context windowing:** Using a context window that dynamically updates as the conversation progresses allows the system to focus on the most relevant parts of the interaction. By limiting the scope of the conversation history to the most recent or critical exchanges, the system can remain focused without overwhelming the generation model with excessive information.
- **Retrieval-based memory:** For particularly complex or long conversations, RAG systems can implement a retrieval-based memory mechanism. This approach allows the system to selectively retrieve relevant parts of the conversation history when needed, ensuring that only the most pertinent context is passed to the language model.

Handling ambiguous or complex queries

User queries are not always straightforward, many times, they can be vague, ambiguous, or involve complex reasoning that challenges a RAG system's capabilities.

Disambiguation through clarification

One way to address ambiguity is to prompt the system to seek clarification from the user. For instance, if the query is too vague, the system can generate a follow-up question asking for more specifics. This interactive process helps narrow down the user's intent before proceeding with the retrieval and generation phases.

Versatile query processing

For complex queries that involve multiple aspects or subtopics, the system can break down the query into smaller, more manageable parts. This involves retrieving information in stages, where each stage tackles a specific aspect of the query. The final output is then synthesized from multiple retrieval and generation steps, ensuring that all components of the query are addressed.

Using contextual clues

To handle ambiguity, the system can use contextual clues from the query or conversation history. By analyzing previous interactions or related topics, the RAG system can infer the user's intent more accurately, reducing the likelihood of generating irrelevant or incorrect responses.

Advanced retrieval techniques for complex queries

For particularly challenging queries, RAG systems can implement advanced retrieval methods, such as multi-hop question answering, where the system retrieves information from multiple documents and makes logical connections across them to answer complex queries.

Addressing Common RAG Challenges

While RAG systems offer powerful solutions for information retrieval and text generation, they also introduce specific challenges that need to be addressed.

Dealing with bias in generation

Bias in language models, including those used in RAG systems, is a well-known issue that

To mitigate bias in RAG systems, we can apply several strategies:

- **Bias-aware retrieval**: Biases in the retrieval phase can arise when the documents retrieved disproportionately represent certain viewpoints, demographics, or perspectives. By applying filtering techniques to ensure diversity in retrieved documents, such as balancing sources based on authorship, date range, or geography, RAG systems can reduce the likelihood of biased retrieval.
- **Fairness in generation**: Bias in the generation phase can occur if the language model is trained on data that contains biased content or if the model amplifies certain perspectives over others. One approach to mitigating this is to fine-tune the model on curated datasets designed to minimize bias, ensuring that the generated responses are as neutral and fair as possible.
- **Post-generation filtering**: Implementing post-processing steps where the generated output is analyzed for biased or harmful content can further reduce bias. These filters can flag or modify problematic outputs before they are presented to the user, ensuring that the final output meets fairness criteria.

Computational overheads

As RAG systems become more complex with the integration of advanced retrieval and generation techniques, the computational demands also increase. This challenge manifests in areas such as model size, processing speed, and latency, all of which can affect the system's efficiency and scalability.

To manage computational overheads, developers can employ the following optimizations:

- **Efficient retrieval techniques**: Optimizing the retrieval phase by using more efficient indexing and search algorithms (such as approximate nearest neighbors) can greatly reduce the time and resources needed to locate relevant documents.
- **Model compression and optimization**: The language models used in RAG systems can be computationally intensive, especially when dealing with large-scale or domain-specific queries. Techniques like model distillation, quantization, and pruning can be used to reduce the size and computational cost of these models without sacrificing too much performance.

Data limitations

RAG systems are heavily dependent on the quality and scope of the data they retrieve and generate. In domain-specific applications, data limitations can be a major challenge, particularly when the available training data is insufficient, outdated, or of low quality.

We can address data limitations in RAG systems with a few approaches.

Data augmentation

When domain-specific training data is limited, data augmentation techniques can help artificially expand the dataset. This can include generating synthetic data, paraphrasing existing documents, or using external sources to complement the original dataset. Data augmentation ensures that the model has access to a broader range of examples, improving its ability to handle diverse queries.

Domain adaptation

Fine-tuning pre-trained language models on small, domain-specific datasets can help RAG systems adapt to specialized use cases, even with limited data. Domain adaptation allows the model to better understand industry-specific terminology and nuances, improving the quality of generated responses.

Active learning

In cases where high-quality training data is scarce, active learning can be employed to improve the dataset iteratively. By identifying the most informative data points and focusing annotation efforts on those, developers can gradually enhance the dataset without requiring

Implementing advanced techniques in RAG systems requires a solid understanding of the tools, frameworks, and strategies available. As these techniques become more complex, leveraging specialized libraries and frameworks simplifies integrating sophisticated retrieval and generation workflows.

Tools and libraries

Many frameworks and libraries have emerged to support the implementation of advanced retrieval, ranking, filtering, and generation.

LangChain is a popular framework designed specifically for working with language models and integrating them with external data sources. It supports advanced retrieval-augmented techniques, including document indexing, querying, and chaining different processing steps (retrieval, generation, and reasoning).

LangChain also offers out-of-the-box integrations with vector databases and various retrievers, making it a versatile option for building custom RAG systems.

Learn more about LangChain and RAG in this course: Build RAG Systems with LangChain

Haystack

Haystack is an open-source framework that specializes in building RAG systems for production use. It provides tools for dense retrieval, document ranking, and filtering, as well as natural language generation.

Haystack is particularly powerful in applications that require domain-specific search, question answering, or document summarization. With support for a variety of backends and integration with popular language models, Haystack simplifies the deployment of RAG systems in real-world scenarios.

OpenAI API

The OpenAI API allows developers to integrate powerful language models, such as GPT-4, into RAG workflows. While not specific to retrieval-augmented tasks, OpenAI's models can be used in conjunction with retrieval frameworks to generate responses based on retrieved information, enabling advanced generation capabilities.

Implementation strategies

To integrate advanced techniques into an existing RAG system, it's essential to follow a structured approach.

Choose the right framework

Start by selecting a framework or library that aligns with your use case. For example, if you need a highly scalable system with dense retrieval capabilities, frameworks like LangChain or Haystack are ideal.

Set up document retrieval

The first step is to set up the retrieval component, which involves indexing your data source and configuring the retrieval method. Depending on your use case, you might choose dense retrieval (using vector embeddings) or hybrid search (combining sparse and dense methods). For instance, LangChain or Haystack can be used to create the retrieval pipeline.

Implement reranking and filtering

Once the retrieval system is operational, the next step is to enhance relevance through reranking and filtering techniques. This can be done using the built-in reranking modules in Haystack or by customizing your reranking models based on your specific query types.

Incorporate advanced generation techniques

If hallucination is an issue, focus on grounding the generation in retrieved documents, ensuring the model generates output based on the content of those documents.

Test and evaluate

Regular testing is crucial for refining the performance of the RAG system. Use evaluation metrics like accuracy, relevance, and user satisfaction to assess the effectiveness of advanced techniques such as reranking and context distillation. Run A/B tests to compare different approaches and fine-tune the system based on feedback.

Optimize for scalability

As the system grows, computational overheads may become a concern. To manage this, employ optimization techniques like model distillation or quantization, and ensure that retrieval processes are efficient. Utilizing GPU acceleration or parallelization can also help maintain performance at scale.

Monitor and update

RAG systems need to evolve to adapt to new queries and data. Set up monitoring tools to track the system's performance in real-time, and continuously update the model and retrieval index to handle emerging trends and requirements.

Evaluating Advanced RAG Techniques

Implementing advanced techniques in a RAG system is only the beginning. By using appropriate evaluation metrics and testing methods like A/B testing, we can assess how well the system responds to user queries and refines over time.

Accuracy

Accuracy measures how often the system retrieves and generates the correct or relevant response. For question-answering systems, this could involve a direct comparison of the generated answers with ground-truth data. Increased accuracy indicates that the system is accurately interpreting queries and delivering precise results.

Relevance

This metric evaluates the relevance of retrieved documents and the quality of the generated response based on how well they answer the user's query. Metrics like Mean Reciprocal Rank (MRR) or Precision@K are commonly used to quantify relevance, assessing how high in the ranking the most relevant document appears.

Latency

While accuracy and relevance are critical, real-time performance also matters. Latency refers to the system's response time—the speed at which the system retrieves documents and generates answers. Low latency is particularly important in applications where timely responses are vital, such as customer support or live Q&A systems.

Coverage

Coverage measures how well the RAG system handles a wide variety of queries. In domain-specific applications, ensuring that the system can handle the full scope of potential user queries is key to providing comprehensive support.

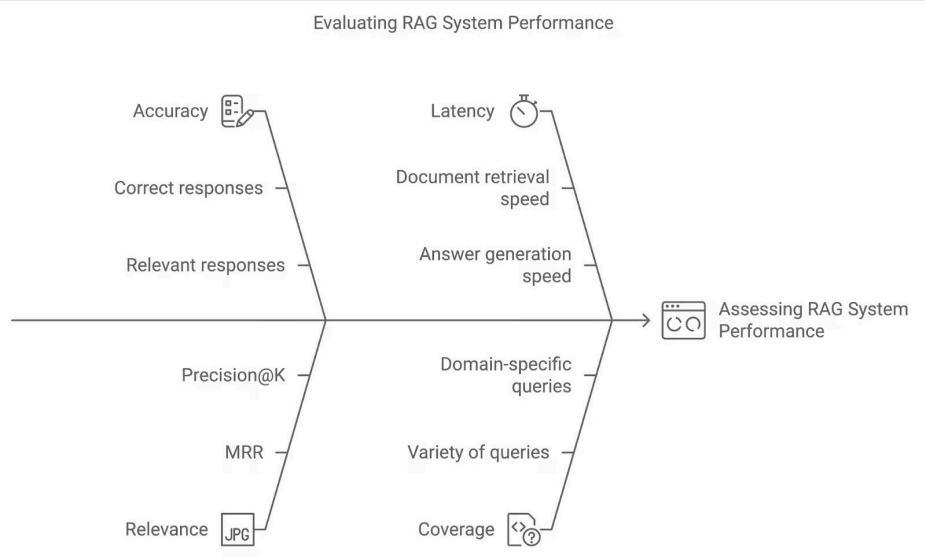


Diagram generated with napkin.ai

Use Cases of Advanced RAG Techniques

Advanced RAG techniques open up a wide range of possibilities across different industries and applications.

Complex question-answering systems

One of the most impactful use cases of advanced RAG techniques is in complex question-answering (QA) systems. These systems require more than just simple document retrieval—they must understand the context, break down multi-step queries, and provide comprehensive answers based on retrieved documents.

Domain-specific knowledge retrieval

In industries where domain-specific knowledge is important, advanced RAG systems can be built to retrieve and generate highly specialized content. Some noteworthy applications include:

- **Healthcare**: Medical professionals rely on up-to-date research, clinical guidelines, and patient records when making decisions. Advanced RAG systems can retrieve medical papers, summarize patient histories, and generate treatment options. Filtering and reranking are particularly important in ensuring the retrieved content is recent, accurate, and highly relevant to the patient's condition.
- **Financial services**: In the financial sector, RAG systems can retrieve market reports, regulatory filings, and economic forecasts, helping analysts generate accurate, data-driven insights. Query expansion and dense retrieval can ensure that analysts receive the most relevant and comprehensive data available.

Personalized recommendations

Personalized recommendation systems are another key use case for advanced RAG techniques. By combining user preferences, behavior, and external data sources, RAG systems can generate personalized recommendations for products, services, or content, including:

- **E-commerce**: RAG systems can recommend products by retrieving product descriptions, customer reviews, and user profiles to generate personalized suggestions. Hybrid search (combining keyword-based and vector-based retrieval) and reranking are essential for improving the relevance of these recommendations.
- **Content platforms**: On streaming or news platforms, RAG systems can recommend content based on user preferences and recent trends. Query expansion and context distillation can help these systems provide more nuanced recommendations, tailoring suggestions based on past behavior and current interests.

The next generation of RAG systems will integrate more diverse data sources, improving reasoning capabilities, and addressing current limitations like ambiguity and complex query handling.

A key development area is the integration of various data sources, moving beyond the reliance on single datasets. Future systems will combine information from diverse sources such as databases, APIs, and real-time feeds, enabling more comprehensive and multidimensional answers to complex queries.

Handling ambiguous or incomplete queries is another challenge that future RAG systems will address. By combining probabilistic reasoning with better contextual understanding, these systems will manage uncertainty more effectively.

Additionally, multi-step reasoning will become more integral to how RAG systems process complex queries, breaking them down into smaller components and synthesizing the results across multiple documents or steps. This will be especially beneficial in fields like legal research, scientific discovery, and customer support, where queries often require connecting diverse pieces of information.

As personalization and context awareness continue to improve, future RAG systems will tailor their responses based on user history, preferences, and past interactions. Real-time adaptation to new information will allow for more dynamic and productive conversations.

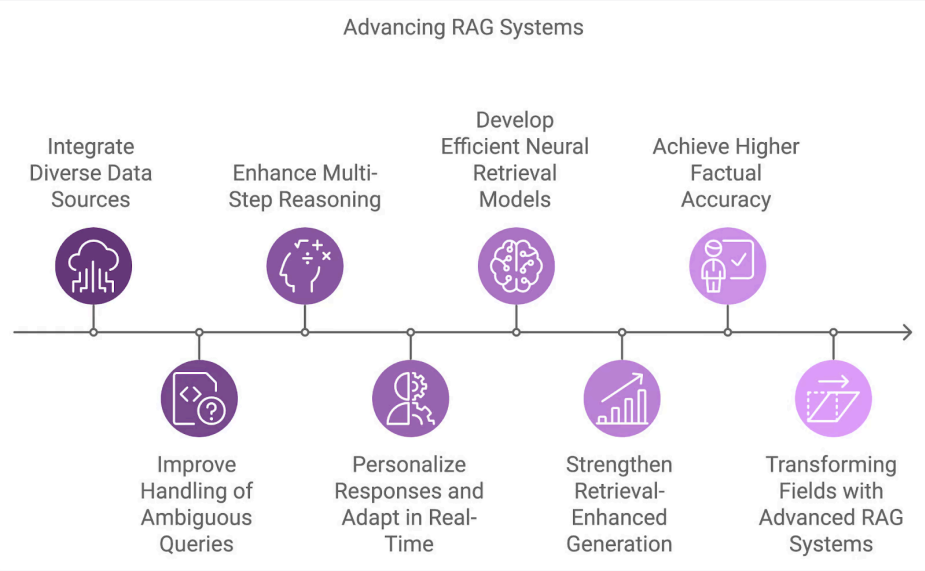


Diagram generated with napkin.ai

Current dense retrieval models are highly effective, but research is ongoing to develop even more efficient and accurate neural retrieval models. These models aim to better capture semantic similarities across a wider range of query-document pairs while improving efficiency in large-scale retrieval tasks.

Papers like Karpukhin et al. (2020) introduced Dense Passage Retrieval (DPR) as a core method for open-domain question answering, while more recent studies like Izacard et al. (2022) focus on few-shot learning to adapt RAG systems for domain-specific tasks.

Another emerging research area focuses on improving the connection between retrieval and generation through retrieval-enhanced generation models. These models aim to seamlessly integrate retrieved documents into the generation process, allowing the language model to condition its output more directly on the retrieved content.

This can reduce hallucinations and improve the factual accuracy of generated responses, making the system more reliable. Notable works include Huang et al. (2023) with the RAVEN model, which improves in-context learning using retrieval-augmented encoder-decoder models.

Conclusion

Looking forward, the evolution of RAG systems will be driven by innovations like cross-lingual capabilities, personalized generation, and handling more diverse data sources.

If you want to keep learning and get more hands-on with RAG systems, I recommend these tutorials:

- Corrective RAG (CRAG) Implementation With LangGraph
- RAG With Llama 3.1 8B, Ollama, and Langchain
- Using a Knowledge Graph to Implement a RAG Application

AUTHOR
Stanislav Karzhev

Recent MSc graduate who specialises in Artificial Intelligence

TOPICS

Artificial Intelligence Large Language Models

Learn AI with these courses!

COURSE

Retrieval Augmented Generation (RAG) with LangChain

3 hr 2.9K

Learn cutting-edge methods for integrating external data with LLMs using Retrieval Augmented Generation (RAG) with LangChain.

See Details → Start Course

See More →

Related

BLOG

What is Retrieval Augmented Generation (RAG)?

TUTORIAL

Boost LLM Accuracy with

See More →

Grow your data skills with DataCamp for Mobile

Make progress on the go with our mobile courses and daily 5-minute coding challenges.

LEARN

Learn Python

Learn AI

Learn Power BI

Learn Data Engineering

Assessments

Career Tracks

Skill Tracks

Courses

Data Science Roadmap

DATA COURSES

Python Courses

R Courses

SQL Courses

Power BI Courses

Tableau Courses

Alteryx Courses

Azure Courses

AWS Courses

Google Sheets Courses

Excel Courses

AI Courses

Data Analysis Courses

Data Visualization Courses