What is retrieval-augmented generation?

Retrieval-augmented generation (RAG) is a popular generative AI framework that enhances the capabilities of large language models (LLMs) by incorporating relevant, up-to-date information during the generation process. This approach allows LLMs to supplement their pre-trained knowledge with current, domain-specific data. RAG is a cost-effective solution to customize an LLM for specific use cases without the expensive and time-consuming process of fine-tuning or retraining the entire model.

Retrieval-augmented generation for more intelligent AI

Retrieval-augmented generation allows organizations to leverage general-purpose large language models for specialized applications without needing expensive, custom-trained models. RAG directly addresses the fundamental limitations of these models by augmenting queries with current, domain-specific information to enhance generation capabilities. This enables organizations to incorporate real-time information, proprietary datasets, and specialized documentation that is not part of the original model training. By transparently

providing evidence with responses, RAG improves trust and reduces the risk of hallucinations.

What are large language models?

LLMs are a form of artificial intelligence designed to understand and produce human-like text. As an advanced application of Natural Language Processing (NLP), LLMs can learn patterns, structures, and grammar from massive amounts of training data, enabling them to generate coherent responses to user prompts. The strength of LLMs lies in their ability to perform a wide range of language generation tasks without the need for task-specific training. This makes them versatile tools for applications such as chatbots, translation, content creation, and summarization.

The limitations of large language models

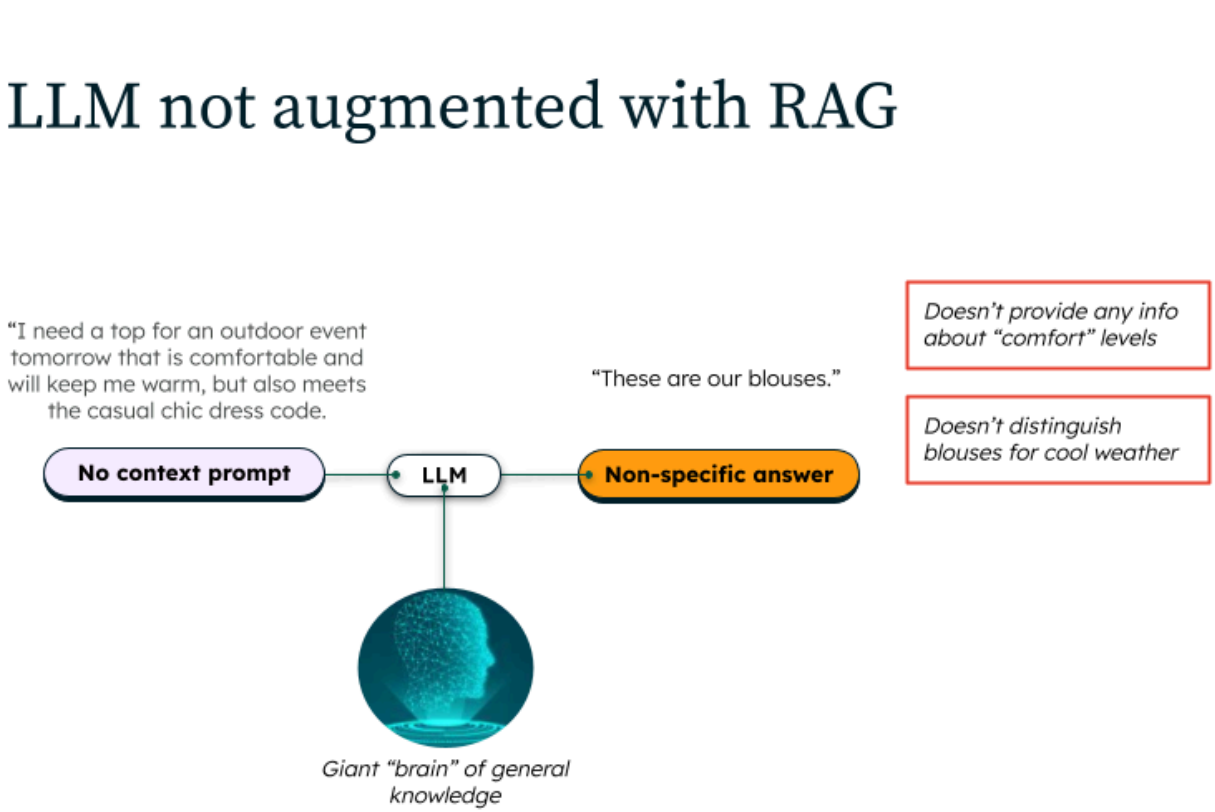
A large language model is a complex neural network that learns from analyzing massive training datasets. These models require substantial computational resources, making them extremely expensive and time-consuming to develop. Moreover, the specialized infrastructure required for hosting and maintaining custom LLMs creates significant financial barriers, limiting their accessibility to only well-resourced organizations with considerable technological investments.

LLMs are great at answering questions about historical content, but their knowledge is constrained by the limitations of their training data. This renders them less effective for queries requiring up-to-date knowledge, as they cannot answer queries about recent events without model retraining.

Similarly, LLMs cannot natively answer questions about internal company documentation or other domain-specific datasets unique to a particular organization. This limitation creates significant challenges for businesses seeking

to leverage AI technologies that require deep, specialized knowledge specific to their needs.

These limitations highlight another challenge of LLMs: hallucinations. Without verifiable information, language models can generate confident, plausible-sounding, entirely fabricated responses. This tendency to produce convincing but false information creates significant risks for applications requiring accuracy and reliability.



Benefits of retrieval-augmented generation

RAG has become popular because of its relatively simple architecture coupled with significant performance improvements.

Cost-effective

RAG allows organizations to use general-purpose pre-trained models for specialized applications without the expense of developing custom-trained models. Effective retrieval lowers API costs by ensuring that only necessary information is included to optimize for LLMs that charge per token.

Domain customization

RAG enables organizations to tailor pre-trained models to specific domains by integrating specialized knowledge libraries. This allows models to generate answers about proprietary and industry-specific documentation without customized model training. Fine-tuning can provide similar benefits but requires significantly more time, cost, and maintenance.

Real-time insights

RAG enables large language models to access and generate responses using current information by dynamically retrieving up-to-date data from external sources. This overcomes the knowledge limitations of static training datasets, allowing models to provide insights into recent events and emerging trends.

Transparency

RAG improves AI response reliability by providing source citations and evidence for generated content. By linking each response to specific sources in the knowledge base, RAG allows users to verify the origin and accuracy of information, reducing the risk of hallucinations and building trust in AI-generated outputs.

Adaptability

One key advantage of RAG is its ability to adapt easily to new state-of-the-art models. As advancements in language models or retrieval techniques emerge, organizations can swap in newer models or modify retrieval strategies without

overhauling the entire system. This flexibility ensures that a RAG system can stay up-to-date with cutting-edge technology.

How does retrieval-augmented generation work?

RAG consists of three distinct phases: ingestion, retrieval, and generation.

Data ingestion

During ingestion, organizations prepare their knowledge base for retrieval. Source data is collected from various repositories such as internal documentation, databases, or external resources. These documents are then cleaned, formatted, and split into smaller, manageable chunks. Each chunk is converted into a vector representation using an embedding model, which captures the semantic meaning of the text. These vectors are stored in vector databases that enable efficient semantic search and retrieval.

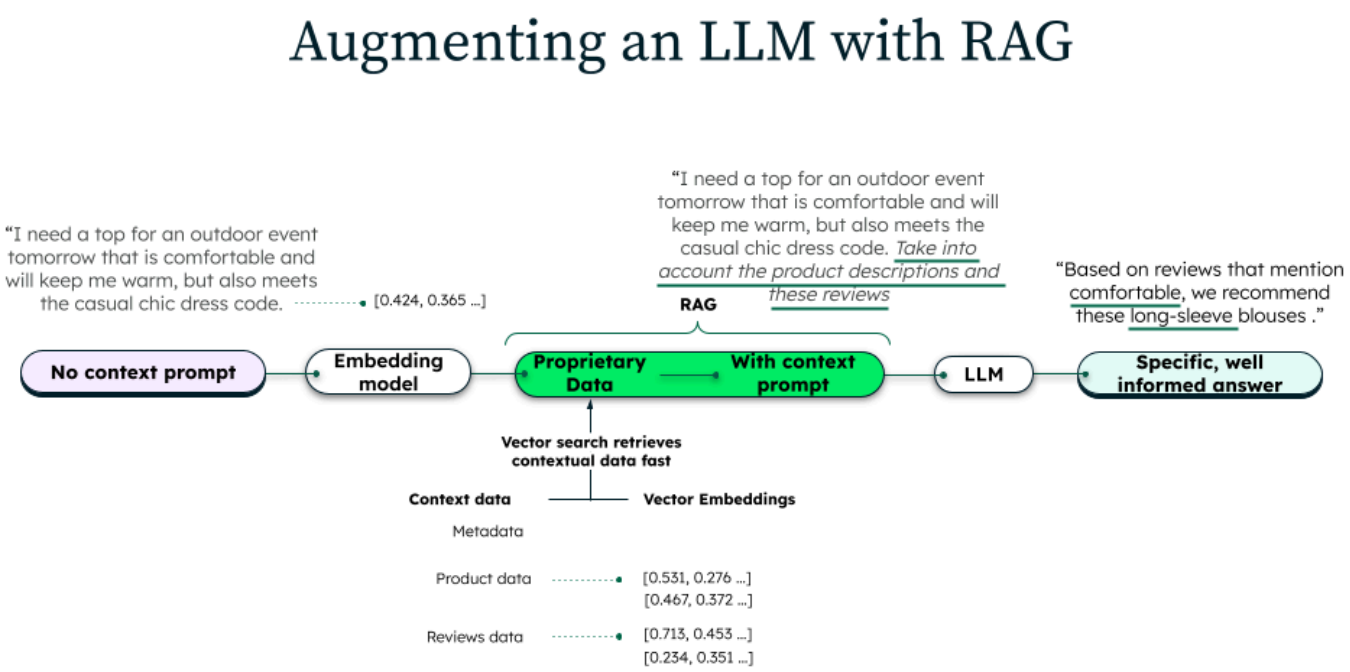
Information retrieval

When a user submits a query, the system retrieves relevant context before generation. The query is transformed into a vector representation using the same embedding model used during ingestion. Vector search will seach the database for the most semantically similar document chunks to the query. Additional filtering, ranking, or re-weighting techniques may be applied to ensure that only the most relevant information is retrieved, improving the accuracy of the final response.

Generation

Once the relevant context is retrieved, an augmented prompt is constructed using the original prompt, the same retrieved passages, and specific instructions. The LLM processes this prompt to generate a response that synthesizes its pre-trained knowledge with the retrieved content. This approach ensures that the

response is informed by external data sources and aligned with the user's intent, leading to a more accurate answer.



Industry use cases for retrieval-augmented generation

RAG is already being used across industries to unlock the transformative potential of large language models and AI.

- **Manufacturing:** Augment LLMs with equipment manuals and maintenance logs to provide real-time operational guidance. RAG enables technicians to quickly access precise information about machinery, reducing downtime and improving equipment performance.
- **Customer support:** Leverage internal documentation, product guides, and support history to diagnose issues. RAG helps support teams instantly retrieve helpful content, reducing response times and improving first-contact resolution rates to efficiently respond to customer queries.

- **Healthcare:** Synthesize medical research, clinical guidelines, and patient records to support diagnostic decisions and treatment recommendations. RAG allows healthcare professionals to access current medical knowledge while providing transparent, evidence-based insights.
- **Financial services:** Integrate regulatory documents, market reports, and compliance guidelines to support investment research, risk assessment, and regulatory compliance. RAG enables financial analysts to quickly retrieve and analyze complex, up-to-date financial information.
- **Software engineering:** Review documentation and code snippets to assist engineers while they write code. RAG can also help with debugging by suggesting potential fixes based on similar past issues, improving productivity and quality.

Key concepts for retrieval-augmented generation

Chunking

Chunking is a component of the data ingestion process that enhances system accuracy while reducing costs. It involves breaking large pieces of content into smaller, manageable segments to prepare them for retrieval. The goal is to create meaningful and fully contextualized chunks, ensuring they retain enough information to be useful while minimizing redundancy.

Effective chunking balances granularity and completeness, allowing the system to retrieve relevant information without overwhelming the LLM with unnecessary details. Well-structured chunks improve retrieval precision, reduce token usage, and lead to more accurate and cost-efficient responses.

Embedding models

Embedding models convert data into numerical representations called vectors that capture semantic meaning. This enables the system to understand

relationships between words, phrases, and documents, enhancing the accuracy of retrieving relevant information.

During ingestion, the embedding model processes each chunk of data, transforming it into a vector before storing it in a vector database. When a user submits a query, it is converted into a vector using the same embedding model.

models work well for broad applications, while domain-specific models are tailored to industries like legal, medical, or finance, improving retrieval accuracy in specialized fields. Multimodal models go beyond text processing to handle images, audio, and other data types, enabling more advanced retrieval capabilities. Some models can create a numerical representation of text that can be compared directly against an image or video for truly advanced multimodal retrieval.

Semantic search

Semantic search improves information retrieval by focusing on the meaning behind a user's query, significantly improving keyword search. Using embeddings, both queries and documents are converted into vectors that capture semantic meaning. When a user submits a query, the vector database searches to find the most relevant documents, even if the exact query terms aren't directly present in the content.

This approach enables better contextual understanding, ensuring more accurate and relevant results. By recognizing synonyms, related concepts, and word variations, semantic search enhances the user experience and reduces ambiguity, providing results that better match the user's intent.

Reranking

Reranking is a technique used to improve the relevance of search results after an initial retrieval phase. Once a set of documents is retrieved, a reranking model reorders them based on their relevance to the user's query. This model can leverage additional features such as document quality, contextual relevance, or machine learning-based scoring to refine the results.

Reranking helps prioritize the most useful and contextually appropriate information, improving accuracy and user satisfaction. It is especially useful when the initial retrieval phase may return a broad range of results, allowing the system to fine-tune the selection and present the most relevant answers.

Prompt engineering

Prompt engineering involves carefully crafting the input given to an LLM to guide its output in the desired direction. By structuring prompts effectively, you can ensure that the model generates more accurate, relevant, and appropriate responses. This process involves including clear instructions, relevant context, and sometimes examples to help the model understand the task.

In retrieval-augmented generation, prompt engineering plays a key role in combining retrieved documents with the original user query to produce coherent and precise responses. Well-engineered prompts reduce ambiguity, minimize irrelevant information, and ensure the model aligns with the user’s intent, leading to higher-quality outputs.

Optimizing your retrieval-augmented generation application

RAG solutions can be optimized to deliver higher accuracy and an overall improved experience for end users.

Optimizing information retrieval

Information retrieval for RAG can be improved through several strategies. First, review chunking techniques to ensure documents are split into meaningful, contextually relevant segments. Next, choose the right embedding model to capture the semantic meaning of your content. Domain-specific models may offer better results for certain use cases. While semantic search is most commonly used, consider whether keyword search or a hybrid approach can improve retrieval.

Additionally, apply reranking methods after initial retrieval to refine the accuracy of results. It's also crucial to adjust the number of documents retrieved: too many can introduce noise, while too few may miss important context. Finding the right balance helps improve retrieval performance and relevance.

Optimizing answer generation

Improving language generation in RAG can be achieved through several key approaches. First, focus on prompt engineering to structure queries and context in a way that guides the language model to generate more accurate and relevant responses. Clear instructions, context, and examples help reduce ambiguity and improve output quality. Next, evaluate different models or domain-specific LLMs to ensure the generated responses align with the nuances of your specific use case, improving relevance and accuracy. Additionally, tunable model parameters like temperature should be considered to control the creativity of model responses.

Optimizing for production scale

Ensure your RAG system is ready for production by choosing best-in-class vendors for your key application components.

For your vector database, opt for a platform that provides highly efficient search and indexing capabilities, particularly one that supports scalable, fast Approximate Nearest Neighbor (ANN) search. Advanced vector databases may

also support metadata filtering, which can improve accuracy and speed by narrowing down search results based on additional contextual information. This will enable your system to retrieve relevant documents quickly, even as the dataset grows.

When choosing an embedding model, it's important to balance the high dimensionality of vectors with efficient storage and retrieval. While higher-dimensional embeddings capture richer semantic relationships, they come with increased computational cost, storage requirements, and slower retrieval times.

Additionally, when selecting a Large Language Model (LLM) for the generation component, ensure it aligns with the specific needs of your use case. LLMs should be capable of accurately interpreting the retrieved information and generating coherent, contextually relevant responses. The choice of LLM also impacts the overall cost and performance of the system—larger models may deliver better accuracy but at the cost of higher latency and computational demands. It's crucial to assess your response time, output quality, and infrastructure requirements to select an LLM that strikes the right balance between performance and efficiency.

Challenges of retrieval-augmented generation

One of the key challenges of RAG is the difficulty in centralizing and organizing content for effective retrieval. RAG systems require access to vast amounts of data across diverse domains, but organizing this content in a way that allows the model to efficiently retrieve the most relevant and up-to-date information is a complex task. Data can be spread across different platforms, formats, and databases, making it difficult to ensure comprehensive coverage and accuracy. Additionally, ensuring consistency across multiple sources is crucial. The retrieved information may be contradictory, outdated, or incomplete, which can muddle the knowledge base and undermine the quality and reliability of generated

responses. These challenges highlight the need for more sophisticated indexing and retrieval systems to enable RAG models to pull in the best possible content and generate relevant, accurate outputs.

Another significant challenge is RAG's current limitation to answering questions rather than performing more complex tasks. While RAG systems excel at generating responses based on retrieved information, they struggle with executing actions beyond answering queries or generating content. This constraint arises because RAG is primarily designed to pull in relevant data from external sources and provide outputs based on that data rather than interact with or manipulate real-world environments. As a result, while RAG models can assist in information retrieval and content generation, their ability to carry out tasks like problem-solving or decision-making remains underdeveloped, limiting their potential for more dynamic applications.

Creating memory-enhanced interactive retrieval-augment generation

Enhancing RAG with memory expands its ability to create a more interactive experience by remembering key details and context from past interactions. Traditional RAG systems typically respond to queries without retaining information across multiple exchanges, leading to a disjointed experience. By integrating memory mechanisms, RAG systems could store relevant facts, preferences, or insights from current and previous conversations, allowing them to recall this information as needed. This enables the system to offer more personalized, context-aware responses and create a more seamless experience. Over time, the system builds a deeper understanding of the user’s needs, adapting its responses to be more relevant and engaging, making the experience feel like an ongoing conversation rather than a series of isolated queries.

The future of retrieval-augmented generation and generative AI

New techniques within RAG will continue to emerge, enhancing its ability to retrieve and generate information in more efficient, adaptive, and intelligent ways. One key area of growth is the development of advanced retrieval mechanisms, enabling RAG systems to dynamically access a broader range of sources, including specialized databases, unstructured content, and real-time information. These improvements will make RAG systems more contextually aware, allowing them to generate highly relevant and accurate outputs across various domains.

At the same time, integrating new generative AI agentic capabilities will empower AI systems to perform problem-solving, data analysis, and decision-making tasks. These agentic systems will not only retrieve and generate responses but also take actions based on the information they gather, making them more interactive, self-sufficient, and intelligent. As a result, RAG will become central to applications like automated research, personalized recommendations, and interactive virtual assistants, driving a new era of responsive and proactive AI.

Fine-tuning vs. retrieval-augmented generation

Fine-tuning is a process where a language model is modified through additional training on new content, essentially teaching the model new knowledge or behaviors that become permanently embedded in its parametric memory. This approach requires significant computational resources and expertise, has limited capacity for new information due to model size constraints, and any changes made are permanent and can't be easily updated. A fine-tuned model can provide domain-specific results but has significant training time requirements and cost implications, making it difficult to keep up-to-date.

Retrieval-augmented generation (RAG) dynamically retrieves content not part of the training data before the language generation occurs. This allows RAG models to incorporate new data without altering the model's underlying parameters, making it more flexible and scalable with the need for knowledge-intensive tasks like fine-tuning.

Build RAG applications with MongoDB Atlas and Voyage AI

MongoDB Atlas is a robust general-purpose database that supports vectors and vector search, making it an ideal choice for building production-grade RAG applications.

Voyage AI provides powerful embedding models and rerankers to create highly accurate information retrieval.

Take your projects to the next level–simplify your development process and unlock new value while benefitting from seamless integration with leading AI partners, major cloud providers, LLM model providers, and system integrators.

Resources

Explore MongoDB Atlas - the vector database with built-in search, vector capabilities, and more. **Register for free** now.

To learn more about Voyage AI, you can learn more at **this blog**.

Get strategic advice and implementation support for search and the rest of the **AI stack**, visit our **MongoDB AI Applications Program** for more details.

Visit our **AI Learning Hub** to learn more about MongoDB's AI solutions.

FAQs

What are hallucinations?

Hallucinations are fabricated statements generated by artificial intelligence that sound believable but are completely invented. They occur because AI models generate text based on likely word sequences patterns, without the ability to distinguish between true

What is a RAG model?

What is the difference between a foundation model and a large language model?

Get Started With MongoDB Atlas

Try Free

English

About

Careers

Investor Relations

Legal

Privacy Policy

GitHub

Security Information

Trust Center

Connect with Us

Support

Contact Us

Customer Portal

Atlas Status

Customer Support

Manage Cookies

Deployment Options

MongoDB Atlas

Data Basics

Vector Databases

Enterprise Advanced

Community Edition

NoSQL Databases

Document Databases

RAG Database

ACID Transactions

MERN Stack

MEAN Stack