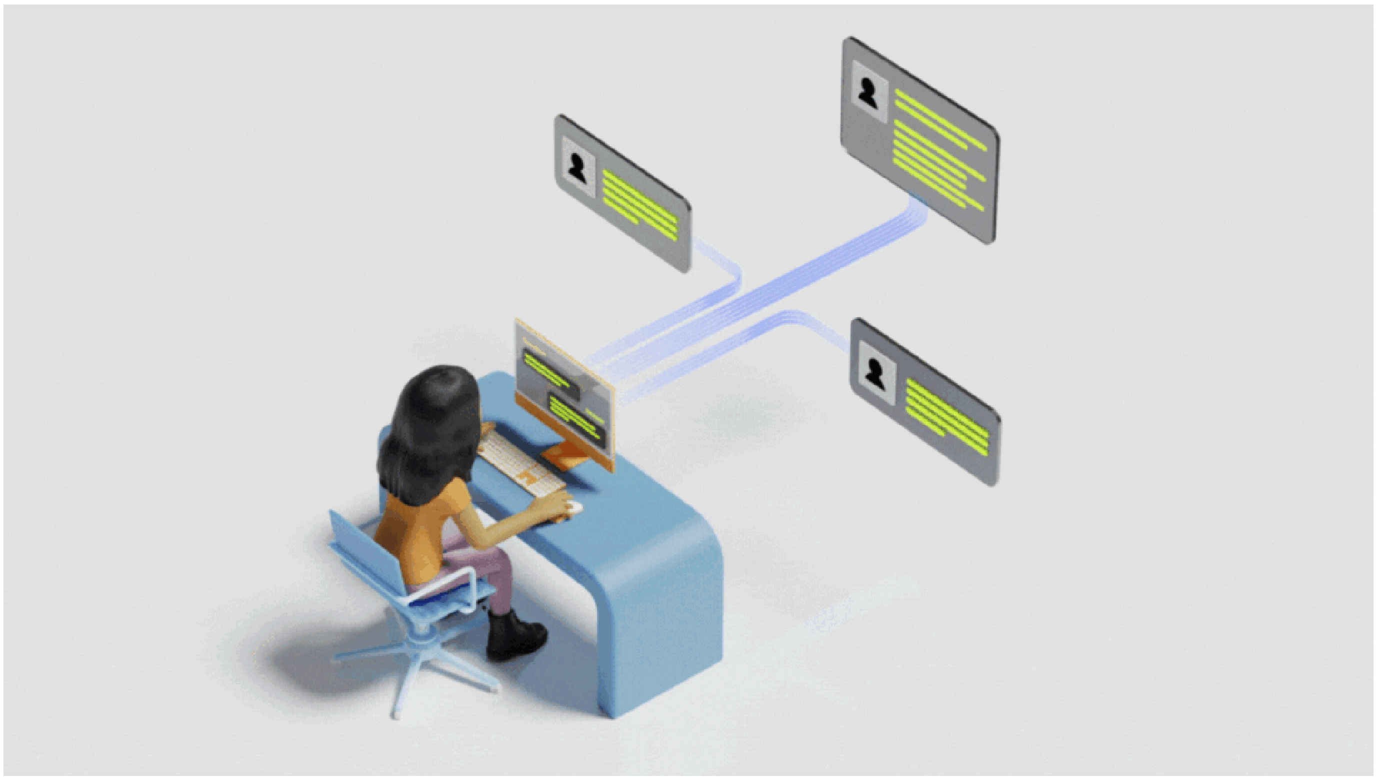
Generative AI English

RAG 101: Retrieval-Augmented Generation Questions Answered

Dec 18, 2023 +30 Like Discuss (2)

By Hayden Wolff, Daniel Rohrer and Jiwei Liu



Data scientists, AI engineers, MLOps engineers, and IT infrastructure professionals must consider a variety of factors when designing and deploying a RAG pipeline: from core components like LLM to evaluation approaches.

Augmented Generation Pipelines. All these stages provide opportunities to make design decisions according to your needs.

Learn how RAG can supercharge your projects at NVIDIA GTC 2024 – the #1 AI conference.

Here's a list of top questions and answers.

When should you fine-tune the LLM vs. using RAG?

In the world of LLMs, choosing between fine-tuning, Parameter-Efficient Fine-Tuning (PEFT), prompt engineering, and retrieval-augmented generation (RAG) depends on the specific needs and constraints of your application.

- *Fine-tuning* customizes a pretrained LLM for a specific domain by updating most or all of its parameters with a domain-specific dataset. This approach is resource-intensive but yields high accuracy for specialized use cases.
- *PEFT* modifies a pretrained LLM with fewer parameter updates, focusing on a subset of the model. It strikes a balance between accuracy and resource usage, offering improvements over prompt engineering with manageable data and computational demands.
- *Prompt engineering* manipulates the input to an LLM to steer its output, without altering the model's parameters. It's the least resource-intensive method, suitable for applications with limited data and computational resources.
- RAG enhances LLM prompts with information from external databases, effectively a sophisticated form of prompt engineering.

It's not about using one technique or another. In fact, these techniques can be used in tandem. For example, PEFT might be integrated into a RAG system for further refinement of the LLM or embedding model. The best approach depends on the application's specific requirements, balancing accuracy, resource availability, and computational constraints.

For more information about customization techniques that you can use to improve domain-specific accuracy, see Selecting Large Language Model Customization Techniques.

When building an application with LLMs, start by implementing RAG to enhance the model's responses with external information. This approach quickly improves relevance and depth.

targeted improvements through model customization with efficient development and continuous enhancement strategies.

How to increase RAG accuracy without fine-tuning?

This question deserves not just its own post but several posts. In short, obtaining accuracy in enterprise solutions that leverage RAG is crucial, and fine-tuning is just one technique that may (or may not) improve accuracy in a RAG system.

First and foremost, find a way to measure your RAG accuracy. If you don't know where you're beginning, you won't know how to improve. There are several frameworks for evaluating RAG systems, such as Ragas, ARES, and Bench.

After you have done some evaluation for accuracy, there are numerous places to look to improve the accuracy that does not require fine-tuning.

Although it may sound trivial, first check to make sure that your data is being parsed and loaded correctly in the first place. For example, if documents contain tables or even images, certain data loaders may miss information in documents.

After data is ingested, it is *chunked*. This is the process of splitting text into segments. A chunk can be a fixed character length, but there are various chunking methods, such as sentence splitting and recursive chunking. How text is chunked determines how it is stored in an embedding vector for retrieval.

On top of this, there are many indexing and associated retrieval patterns. For example, several indexes can be constructed for various kinds of user questions and a user query can be routed according to an LLM to the appropriate index.

There are also a variety of retrieval strategies. The most rudimentary strategy is using cosine similarity with an index, but BM25, custom retrievers, or knowledge graphs can also improve the retrieval.

down more complex questions. Even just changing the LLM's system prompt can drastically change accuracy.

At the end of the day, it's important to take time to experiment and measure the changes in accuracy that various approaches provide.

Remember, models like the LLM or embedding model are merely a part of a RAG system. There are many ways to improve RAG systems to achieve high accuracy without doing any fine-tuning.

How to connect LLMs to data sources?

There are a variety of frameworks for connecting LLMs to your data sources, such as LangChain and LlamaIndex. These frameworks provide a variety of features, like evaluation libraries, document loaders, and query methods. New solutions are also coming out all the time. We recommend reading about various frameworks and picking the software and components of the software that make the most sense for your application.

Can RAG cite references for the data that it retrieves?

Yes. In fact, it improves the user experience if you can cite references for retrieved data. In the AI chatbot RAG workflow example found in the /NVIDIA/GenerativeAIExamples GitHub repo, we show how to link back to source documents.

What type of data is needed for RAG? How to secure data?

Right now, textual data is well supported for RAG. Support in RAG systems for other forms of data like images and tables is improving as more research into multi-modal use cases progresses. You may have to write additional tools for data preprocessing depending on your data and where it's located. There are a variety of data loaders available from LlamaHub and

Securing data, particularly for an enterprise, is paramount. For example, some indexed data may be intended for only a particular set of users. Role-based access control (RBAC), which restricts access to a system depending on roles, can provide data access control. For example, a user session token can be used in the request to the vector database so that information that's out of scope for that user's permissions is not returned.

A lot of the terms for securing a model in the environment are the same as you might use for securing a database or other critical asset. Think about how your system will log activities—the prompt inputs, outputs, and error results—that are the results of production pipelines. These may provide a rich set of data for product training and improvement, but also a source of data leaks like PII that must be carefully managed just as you are managing the model pipelines themselves.

AI models have many common patterns to cloud deployments. You should take every advantage of tools like RBAC, rate limiting, and other controls common in those environments to make your AI deployments more robust. Models are just one element of these powerful pipelines. For more information, see Best Practices for Securing LLM Enabled Applications

One aspect important in any LLM deployment is the nature of interaction with your end users. So much of RAG pipelines are centered on the natural language inputs and outputs. Consider ways to ensure that the experience meets consistent expectations through input/output moderation.

People can ask questions in many different ways. You can give your LLM a helping hand through tools like NeMo Guardrails, which can provide secondary checks on inputs and outputs to ensure that your system runs in tip-top shape, addresses questions it was built for, and helpfully guides users elsewhere for questions that the LLM application isn't built to handle.

How to accelerate a RAG pipeline?

RAG systems consist of many components, so there are ample opportunities to accelerate a RAG pipeline:

- Data preprocessing
- Indexing and retrieval

Data preprocessing

Deduplication is the process of identifying and removing duplicate data. In the context of RAG data preprocessing, deduplication can be used to reduce the number of identical documents that must be indexed for retrieval.

NVIDIA NeMo Data Curator uses NVIDIA GPUs to accelerate deduplication by performing min hashing, Jaccard similarity computing, and connected component analysis in parallel. This can significantly reduce the amount of time it takes to deduplicate a large dataset.

Another opportunity is chunking. Dividing a large text corpus into smaller, more manageable chunks must be done because the downstream embedding model can only encode sentences below the maximum length. Popular embedding models such as OpenAI can encode up to 1536 tokens. If the text has more tokens, it is simply truncated.

NVIDIA cuDF can be used to accelerate chunking by performing parallel data frame operations on the GPU. This can significantly reduce the amount of time required to chunk a large corpus.

Lastly, you can accelerate a tokenizer on the GPU. Tokenizers are responsible for converting text into integers as tokens, which are then used by the embedding model. The process of tokenizing text can be computationally expensive, especially for large datasets.

Indexing and retrieval

The generation of embeddings is frequently a recurring process since RAG is well-suited for knowledge bases that are frequently updated. Retrieval is done at inference time, so low latency is a requirement. These processes can be accelerated by NVIDIA NeMo Retriever. NeMo Retriever aims to provide state-of-the-art, commercially ready models and microservices, optimized for the lowest latency and highest throughput.

LLM inference

At a minimum, an LLM is used for the generation of a fully formed response. LLMs can also be used for tasks such as query decomposition and routing.

With several calls to an LLM, low latency for the LLM is crucial. NVIDIA NeMo includes TensorRT-LLM for model deployment, which optimizes the LLM to achieve both ground-breaking inference

NVIDIA Triton Inference Server also enables the optimized LLM to be deployed for high performance, cost-effective, and low-latency inference.

What are solutions to improve latency for a chatbot?

Beyond using the suggestions to accelerate your RAG pipeline like NeMo Retriever and NeMo inference container with Triton Inference Server and TensorRT-LLM, it is important to consider using streaming to improve the perceived latency of the chatbot. As responses can be long, a streaming UI displaying parts of the response as they become available can mitigate perceived latency.

It may be worthwhile to consider using a smaller LLM that is fine-tuned for your use case. In general, smaller LLMs have much lower latency than larger LLMs.

Some fine-tuned 7B models have demonstrated out-performing the accuracy of GPT-4 on specific tasks, for example, SQL generation. For example, ChipNeMo, a custom LLM built for internal use at NVIDIA to help engineers generate and optimize software for chip design, uses a 13B fine-tuned model instead of a 70B-parameter model. TensorRT-LLM delivers model optimizations such as flashing, FlashAttention, PagedAttention, Distillation, and Quantization for running smaller fine-tuned models locally, which can be used to decrease the memory used by the LLM.

Latency for LLM responses is a function of the time to first token (TTFT) and the time per output token (TPOT).

Both TTFT and TPOT will be lower for a smaller LLM.

Get started building RAG in your enterprise

By using RAG, you can provide up-to-date and proprietary information with ease to LLMs and build a system that increases user trust, improves user experiences, and reduces hallucinations.

Explore the NVIDIA AI chatbot RAG workflow to get started building a chatbot that can accurately answer domain-specific questions in natural language using up-to-date information.

For those building enterprise RAG applications, NVIDIA NeMo is an end-to-end platform for developing custom generative AI, anywhere. Deliver enterprise-ready models with precise data curation, cutting-edge customization, RAG, and accelerated performance.

Related resources

- **GTC session:** Advanced RAG Pipelines: Engineer Scalable Retrieval Systems for Enterprise AI
- **NGC Containers:** rag-application-query-decomposition-agent
- **Webinar:** Building Intelligent AI Chatbots Using RAG
- **Webinar:** Achieve World-Class Text Retrieval Accuracy for Production-Ready Generative AI

Discuss (2) +30 Like

Tags

Conversational AI | Generative AI | Retail / Consumer Packaged Goods | NeMo Guardrails | NeMo Retriever | TensorRT | Triton Inference Server | Intermediate Technical | Deep Dive | LangChain | LLMs

About the Authors

About Hayden Wolff

Hayden Wolff (he/they) is a technical marketing engineer for Enterprise Products at NVIDIA. He works on integrating generative AI and LLMs into real-world applications. Prior to NVIDIA, he graduated from Tufts University with a major in computer science and a minor in science, technology, and society.

View all posts by Hayden Wolff >

About Daniel Rohrer

Daniel is VP of Software Product Security at NVIDIA, having started as an intern over 22 years ago. Over his career, he has held a variety of technical and leadership roles, ranging from 3D software, to managing 3D and kernel development, to creating the NVIDIA GPU software chip development organization. Daniel has taken his integrated knowledge of everything NVIDIA to hone security practices through the delivery of advanced technical solutions, reliable processes, and strategic investments to deliver the most trustworthy solutions. He has a Master's in Computer Graphics from the University of North Carolina, Chapel Hill.

View all posts by Daniel Rohrer >

infrastructure, including the RAPIDS data science framework.

View all posts by Jiwei Liu >

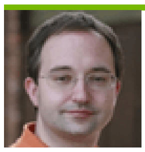
Comments

Notable Replies

December 18, 2023

hwolff

Hopefully this clears up some common questions for folks! If you have any questions or comments, let us know.



January 4, 2024

lorenz6

Hi!
Nice short overview of the topic!
Just one question/comment. Are you sure about this paragraph: “Another opportunity is chunking. Dividing a large text corpus into smaller, more manageable chunks must be done because the downstream embedding model can only encode sentences below the maximum length. Popular embedding models such as OpenAI can encode up to 1536 tokens. If the text has more tokens, it is simply truncated.”?
It reads a bit like tokens and embeddings have been mixed. Especially since, for example, a maximum of 8191 tokens are possible with “text-embedding-ada-002”. The known 1536 refers to the number of dimensions in the returned vector.

Continue the discussion at forums.developer.nvidia.com

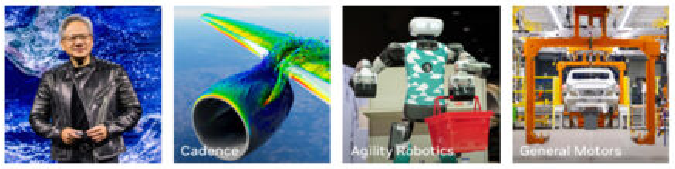
Participants

Related posts

Build an Agentic RAG Pipeline with Llama 3.1 and NVIDIA NeMo Retriever NIMs

Evaluating Retriever for Enterprise-Grade RAG

New Self-Paced Course: Augment Your LLM Using Retrieval-Augmented Generation


Cadence
Agility Robotics
General Motors