What is Retrieval-Augmented Generation (RAG)?

Soumyadarshan Dash

Last Updated : 04 Apr, 2025

78

Retrieval-Augmented Generation (RAG) is a smart AI technique that combines two powerful tools: information retrieval and text generation. Imagine a system that can search for relevant facts or data (like a librarian) and then use that information to create clear, accurate, and detailed answers (like a writer). RAG is used in chatbots, virtual assistants, and other AI tools to provide better, more informed responses. It's like having a super-smart helper that knows how to find and use the right information. In this article, you will get to know all about Retrieval-Augmented Generation, its uses, applications, and how it will shape the future of RAG and LLMs.

This article was published as a part of the Data Science Blogathon

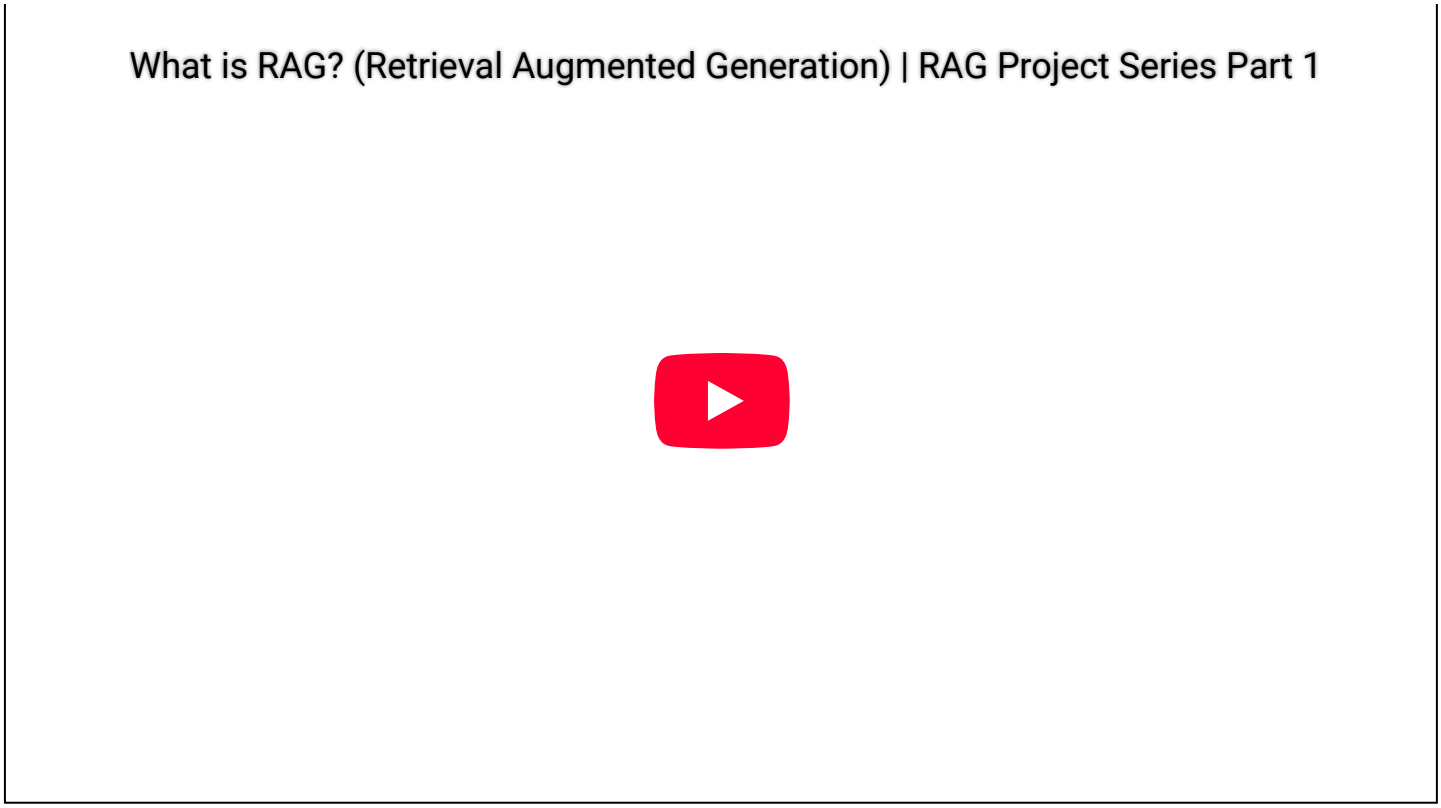
Table of contents

What is RAG?

Retrieval-Augmented Generation, or RAG, represents a cutting-edge approach to **artificial intelligence** (AI) and **natural language processing** (NLP). At its core, RAG LLM is an innovative framework that combines the strengths of retrieval-based and generative models, revolutionizing how AI systems understand and generate human-like text.

Why Use RAG?

The development of RAG (Retrieval-Augmented Generation) is a direct response to the limitations of large language models (LLMs) like GPT. While LLMs have



The Fusion of Retrieval-Based and Generative Models

RAG is fundamentally a hybrid model that seamlessly integrates two critical components. Retrieval-based methods involve accessing and extracting information from external knowledge sources such as databases, articles, or websites.

On the other hand, generative models excel in generating coherent and contextually relevant text. What distinguishes RAG Model is its ability to harmonize these two components, creating a symbiotic relationship that allows it to comprehend user queries deeply and produce responses that are not just accurate but also contextually rich.

To grasp the essence of RAG LLM, it's essential to deconstruct its operational mechanics. RAG operates through a series of well-defined steps:

- Begin by receiving and processing user input.
- Analyze the user input to understand its meaning and intent.
- Utilize retrieval-based methods to access external knowledge sources. This enriches the understanding of the user's query.
- Use the retrieved external knowledge to enhance comprehension.
- Employ generative capabilities to craft responses. Ensure responses are factually accurate, contextually relevant, and coherent.
- Combine all the information gathered to produce responses that are meaningful and human-like.
- Ensure that the transformation of user queries into responses is done effectively.

The Role of Language Models and User Input

Central to understanding RAG is appreciating the role of **Large Language Models (LLMs)** in AI systems. LLMs like GPT are the backbone of many NLP applications, including chatbots and virtual assistants. They excel in processing user input and generating text, but their accuracy and contextual awareness are paramount for successful interactions. RAG strives to enhance these essential aspects by integrating retrieval and generation

RAG's distinguishing feature is its ability to integrate external knowledge sources seamlessly. By drawing from vast information repositories, RAG augments its understanding, enabling it to provide well-informed and contextually nuanced responses. Incorporating external knowledge elevates the quality of interactions and ensures that users receive relevant and accurate information.

Generating Contextual Responses

Ultimately, RAG's hallmark is its ability to generate contextual responses. Moreover, it considers the broader context of user queries, leverages external knowledge, and produces responses demonstrating a deep understanding of the user's needs. Consequently, these context-aware responses are a significant advancement, as they facilitate more natural and human-like interactions, making AI systems powered by RAG highly effective in various domains.

Retrieval Augmented Generation (RAG) is a transformative concept in AI and NLP. Additionally, by harmonizing retrieval and generation components, RAG addresses the limitations of existing language models and paves the way for more intelligent and context-aware AI interactions. Furthermore, its ability to seamlessly integrate external knowledge sources and generate responses that align with user intent positions RAG as a game-changer in developing AI systems that can truly understand and communicate with users in a human-like manner.

Range of Data Sources to Empower RAG Models

APIs and Real-time Databases

APIs (Application Programming Interfaces) and real-time databases are dynamic sources that provide up-to-the-minute information to RAG-driven models. They also allow models to access the latest data as it becomes available.

Document Repositories

Document repositories serve as valuable knowledge stores, offering structured and unstructured information. Additionally, they are fundamental in expanding the knowledge base that RAG models can draw upon.

Webpages and Scraping

Web scraping is a method for extracting information from web pages. It enables RAG LLM models to access dynamic web content, making it a crucial source for real-time data retrieval.

Databases and Structured Information

Databases provide structured data that can be queried and extracted. Additionally, RAG models can utilize databases to retrieve specific information, enhancing their responses' accuracy.

Benefits of Retrieval-Augmented Generation (RAG)

RAG addresses the information capacity limitation of traditional Language Models (LLMs). Traditional LLMs have a limited memory called “Parametric memory.” RAG introduces a “Non-Parametric memory” by tapping into external knowledge sources. This significantly expands the knowledge base of LLMs, enabling them to provide more comprehensive and accurate responses.

Improved Contextualization

RAG enhances the contextual understanding of LLMs by retrieving and integrating relevant contextual documents. This empowers the model to generate responses that align seamlessly with the specific context of the user’s input, resulting in accurate and contextually appropriate outputs.

Updatable Memory

A standout advantage of RAG is its ability to accommodate real-time updates and fresh sources without extensive model retraining. Moreover, this keeps the external knowledge base current and ensures that LLM-generated responses are always based on the latest and most relevant information.

Source Citations

RAG-equipped models can provide sources for their responses, thereby enhancing transparency and credibility. Moreover, users can access the sources that inform the LLM’s responses, promoting transparency and trust in AI-

Studies have shown that RAG models exhibit fewer hallucinations and higher response accuracy. They are also less likely to leak sensitive information. Reduced hallucinations and increased accuracy make RAG models more reliable in generating content.

These benefits collectively make Retrieval Augmented Generation (RAG) a transformative framework in Natural Language Processing. Consequently, it overcomes the limitations of traditional language models and enhances the capabilities of AI-powered applications.

Also Read: [Traditional RAG to Graph RAG: The Evolution of Knowledge Retrieval Systems in Artificial Intelligence]

Diverse Approaches in RAG

RAG Model offers a spectrum of approaches for the retrieval mechanism, catering to various needs and scenarios:

- **Simple:** Retrieve relevant documents and seamlessly incorporate them into the generation process, ensuring comprehensive responses.
- **Map Reduce:** Combine responses generated individually for each document to craft the final response, synthesizing insights from multiple sources.
- **Map Refine:** Iteratively refine responses using initial and subsequent documents, enhancing response quality through continuous improvement.

- **Contextual Compression:** Extract pertinent snippets from documents, generating concise and informative responses and minimizing information overload.
- **Summary-Based Index:** Leverage document summaries, index document snippets, and generate responses using relevant summaries and snippets, ensuring concise yet informative answers.
- **Forward-Looking Active Retrieval Augmented Generation (FLARE):** Predict forthcoming sentences by initially retrieving relevant documents and iteratively refining responses. Flare ensures a dynamic and contextually aligned generation process.

These diverse approaches empower RAG to adapt to various use cases and retrieval scenarios, allowing for tailored solutions that maximize the relevance, accuracy, and efficiency of AI-generated responses.

Ethical Considerations in RAG

RAG introduces ethical considerations that demand careful attention:

- **Ensuring Fair and Responsible Use:** Ethical deployment of RAG involves using the technology responsibly and refraining from any misuse or harmful applications. Moreover, developers and users must adhere to ethical guidelines to maintain the integrity of AI-generated content.
- **Addressing Privacy Concerns:** RAG's reliance on external data sources may

- **Mitigating Biases in External Data Sources:** External data sources can inherit biases in their content or collection methods. Moreover, developers must implement mechanisms to identify and rectify biases, ensuring AI-generated responses remain unbiased and fair. This process involves constant monitoring and refinement of data sources and training processes.

Want to master RAG? Here's a detailed learning path to become a RAG specialist in 2025!

Applications of Retrieval Augmented Generation (RAG)

RAG finds versatile applications across various domains, enhancing AI capabilities in different contexts:

- **Chatbots and AI Assistants:** RAG-powered systems excel in question-answering scenarios, providing context-aware and detailed answers from extensive knowledge bases. These systems enable more informative and engaging interactions with users.
- **Education Tools:** RAG can significantly improve educational tools by offering students access to answers, explanations, and additional context based on textbooks and reference materials. This facilitates more effective learning and comprehension.
- **Legal Research and Document Review:** Legal professionals can leverage RAG models to streamline document review processes and conduct efficient legal

provide access to the latest medical literature and clinical guidelines, thereby aiding in accurate diagnosis and treatment recommendations.

- **Language Translation with Context:** RAG enhances language translation tasks by considering the context in knowledge bases. This approach results in more accurate translations, accounting for specific terminology and domain knowledge, which is particularly valuable in technical or specialized fields.

These applications highlight how RAG's integration of external knowledge sources empowers AI systems to excel in various domains, providing context-aware, accurate, and valuable insights and responses.

The Future of RAGs and LLMs

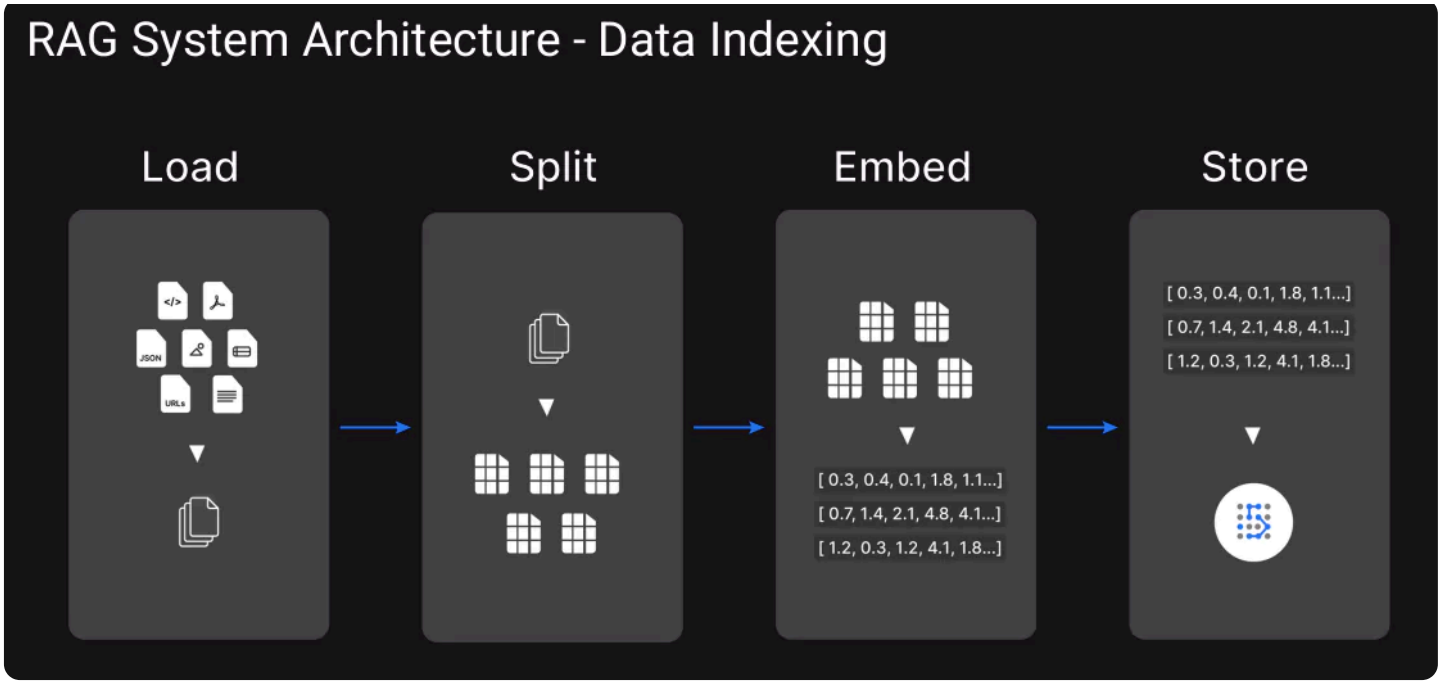
The evolution of Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs) is poised for exciting developments:

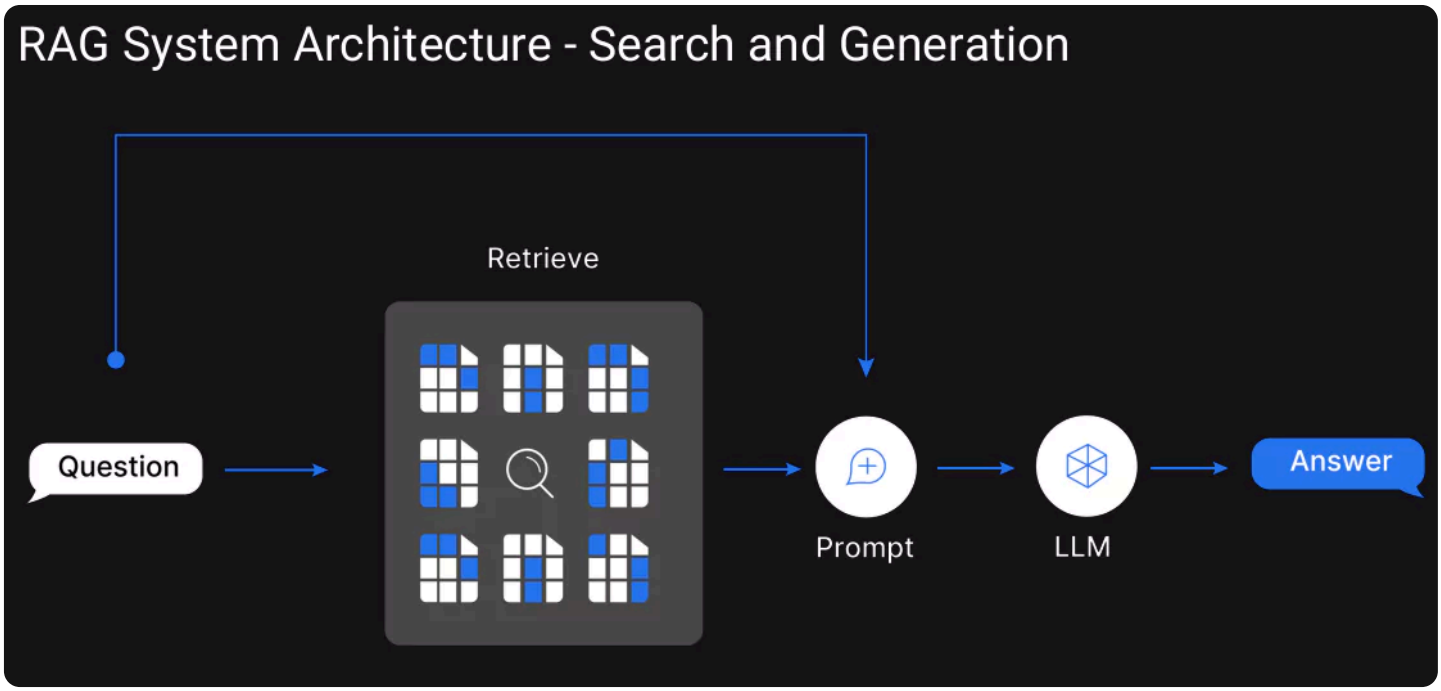
- **Advancements in Retrieval Mechanisms:** The future of RAG will witness refinements in retrieval mechanisms. Furthermore, these enhancements will focus on improving the precision and efficiency of document retrieval, ensuring that LLMs access the most relevant information quickly. Moreover, advanced algorithms and AI techniques will play a pivotal role in this evolution.
- **Integration with Multimodal AI:** The synergy between RAG and multimodal AI, which combines text with other data types like images and videos, holds

- **RAG in Industry-Specific Applications:** As RAG matures, it will find its way into industry-specific applications. Healthcare, law, finance, and education sectors will harness RAG-powered LLMs for specialized tasks. For example, in healthcare, RAG models will aid in diagnosing medical conditions by instantly retrieving the latest clinical guidelines and research papers, ensuring doctors have access to the most current information.
- **Ongoing Research and Innovation in RAG:** The future of RAG is marked by relentless research and innovation. Furthermore, AI researchers will continue to push the boundaries of what RAG can achieve, exploring novel architectures, training methodologies, and applications. Consequently, this ongoing pursuit of excellence will result in more accurate, efficient, and versatile RAG models.
- **LLMs with Enhanced Retrieval Capabilities:** LLMs will evolve to possess enhanced retrieval capabilities as a core feature. Furthermore, they will seamlessly integrate retrieval and generation components, making them more efficient at accessing external knowledge sources. Consequently, this integration will lead to LLMs that are proficient in understanding context and excel in providing context-aware responses.

How Does Retrieval-Augmented Generation Work?

The following diagrams illustrate the LangChain workflow for RAG.





These images depict the architecture of a Retrieval-Augmented Generation (RAG) system. The various components are as follows:

1. **Load:** Raw data from various formats (JSON, PDFs, URLs) is gathered and

3. **Embed:** The data chunks are transformed into numerical embeddings that capture their semantic meaning.
4. **Store:** The embeddings are saved in a vector database for fast retrieval during future queries.
5. **Question:** A user query or question is provided as input to the system.
6. **Retrieve:** The system retrieves relevant documents as context from the vector database based on the question.
7. **Prompt:** The retrieved information is sent along with the prompt that guides the large language model (LLM).
8. **LLM:** The LLM uses context and prompts to generate a coherent and contextually relevant answer.
9. **Answer:** The final answer addresses the user's initial query based on the retrieved information.

Also Read: 8 Types of Chunking for RAG Systems

Utilizing LangChain for Enhanced RAG

Installation of LangChain and OpenAI Libraries

This line of code installs the **LangChain** and OpenAI libraries. LangChain is critical for handling text data and embedding, while OpenAI provides access to state-of-the-art Large Language Models (LLMs). This installation step is

It is best practice to store the API keys in the .env file and load them using the below code:

```
from dotenv import load_dotenv
load_dotenv('/.env')
```

Copy Code

Also Read: A New Era of Text Generation: RAG, LangChain, and Vector Databases

Web Data Loading for the RAG Knowledge Base

- The code utilizes LangChain's "WebBaseLoader."
- Three web pages are specified for data retrieval: YOLO-NAS object detection, DeciCoder's code generation efficiency, and a Deep Learning Daily newsletter.
- This step is essential for building the knowledge base used in RAG, enabling contextually relevant and accurate information retrieval and integration into language model responses.

```
from langchain_community.document_loaders import WebBaseLoader

yolo_nas_loader = WebBaseLoader("https://deci.ai/blog/yolo-nas-object-detection-f

decicoder_loader = WebBaseLoader("https://deci.ai/blog/decicoder-efficient-and-ac

yolo_newsletter_loader = WebBaseLoader("https://deeplearningdaily.substack.com/p/
```

Copy Code

Split the Data into Chunks

```
text_splitter = CharacterTextSplitter(
        separator="\n\n",
        chunk_size=500,
        chunk_overlap=0,
        is_separator_regex=False,)
```

Let us apply the text_splitter for the data as below to split the data into chunks.

```
yolo_nas_chunks = text_splitter.split_documents(yolo_nas_loader)
decicoder_chunks = text_splitter.split_documents(decicoder_loader)
yolo_newsletter_chunks = text_splitter.split_documents(yolo_newsletter_loader)
```

Embedding and Vector Store Setup

- The code sets up embeddings for the RAG process.
- It uses “OpenAIEmbeddings” to create an embedding model.
- A “CacheBackedEmbeddings” object is initialized, allowing embeddings to be stored and retrieved efficiently using a local file store.
- A “FAISS” vector store is created from the preprocessed chunks of web data (yolo_nas_chunks, decicoder_chunks, and yolo_newsletter_chunks), enabling fast and accurate similarity-based retrieval.
- Finally, a retriever is instantiated from the vector store, facilitating efficient document retrieval during the RAG process.

```
from langchain_openai import OpenAIEmbeddings
from langchain.embeddings.cache import CacheBackedEmbeddings
from langchain_community.vectorstores import FAISS
```

```
embedder = CacheBackedEmbeddings.from_bytes_store(
    core_embeddings_model,
    store,
    namespace = core_embeddings_model.model
)

# store embeddings in vector store
vectorstore = FAISS.from_documents(yolo_nas_chunks, embedder)

vectorstore.add_documents(decicoder_chunks)

vectorstore.add_documents(yolo_newsletter_chunks)

# instantiate a retriever
retriever = vectorstore.as_retriever()
```

Checkout: Understanding Word Embeddings

Establishing the Retrieval System

- The code configures the retrieval system for Retrieval Augmented Generation (RAG) using LangChain Expression Language Chains.
- We will initialize ChatPromptTemplate using a prompt that is sent to the LLM.
- It uses “ChatOpenAI” from the LangChain library to set up a chat-based Large Language Model (LLM).
- The “rag_chain_from_docs” chain is created, incorporating the context, prompt and LLM.
- The “rag_chain_with_source” chain is created, using retriever and rag_chain_from_docs.

Copy Code

```
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.runnables import RunnablePassthrough, RunnableParallel

# this formats the docs returned by the retriever
def format_docs(docs):
        return "\n\n".join(doc.page_content for doc in docs)

# prompt to send to the LLM
prompt = """You are an assistant for question-answering tasks.
        Use the following pieces of retrieved context to answer the question.
        If you don't know the answer, just say that you don't know.
        Use three sentences maximum and keep the answer concise.

        Question: {question}

        Context: {context}

        Answer:
        """

prompt_template = ChatPromptTemplate.from_template(prompt)

llm = ChatOpenAI(model='gpt-4o-mini', streaming=True)


# This code defines a chain where input documents are first formatted, then passe
rag_chain_from_docs = (
        RunnablePassthrough.assign(context=(lambda x: format_docs(x["context"])))
        | prompt_template
        | llm
        )
# This code creates a parallel process: one retrieves the context (using a retrie
rag_chain_with_source = RunnableParallel(
        {"context": retriever, "question": RunnablePassthrough()}
).assign(answer=rag_chain_from_docs)
```

handler.

Issue Queries to the RAG System

It sends various user queries to the RAG system, prompting it to retrieve contextually relevant information.

Retrieves Responses

After processing the queries, the RAG system generates and returns contextually rich and accurate responses. The responses are printed on the console.

Copy Code

```
# This is to generate response with RAG system
response = rag_chain_with_source.invoke(
        "What does Neural Architecture Search have to do with how Deci creates it

print(response['answer'].content)
print(response['context'])
```

```
response = rag_chain_with_source.invoke(
    "What does Neural Architecture Search have to do with how Deci creates its models?")

print(response['answer'].content)
print(response['context'])

Neural Architecture Search (NAS) is integral to Deci's model creation, specifically through their AutoNAC engine, w
hich intelligently explores a vast architecture search space to identify optimal designs. In developing YOLO-NAS, D
eci utilized NAS to inspire the architecture and efficiently generate versions of the model that deliver high perfo
rmance. This approach allows for the creation of quantization-friendly architectures that enhance both efficiency a
nd accuracy.
[Document(metadata={'source': 'https://deci.ai/blog/yolo-nas-object-detection-foundation-model/', 'title': '- Archi
ves Page 1 | NVIDIA Blog', 'language': 'en-US'}, page_content='Deep Learning\nMost Popular\n\nLightweight Champ: NV
IDIA Releases Small Language Model With State-of-the-Art Accuracy \n\nDevelopers of generative AI typically face a
tradeoff between model size and accuracy. But a new language model…\t\t\tRead Article\t\t \n \n\nMost Popular \n\nL
ightweight Champ: NVIDIA Releases Small Language Model With State-of-the-Art Accuracy \n\nSLMming Down Latency: How
NVIDIA's First On-Device Small Language Model Makes Digital Humans More Lifelike'), Document(metadata={'source': "h
ttps://deci.ai/blog/decicoder-efficient-and-accurate-code-generation-llm/#:~:text=DeciCoder's%20unmatched%20through
```

Copy Code

```
response = rag_chain_with_source.invoke("What is DeciCoder")
print(response['answer'].content)
print(response['context'])
```

```
response = rag_chain_with_source.invoke("What is DeciCoder")

print(response['answer'].content)
print(response['context'])
```

```
DeciCoder does not appear to be directly mentioned in the provided context, which focuses on the YOLO-NAS architect
ure and its developments in object detection. Therefore, I don't know what DeciCoder is.
[Document(metadata={'source': 'https://deeplearningdaily.substack.com/p/unleashing-the-power-of-yolo-nas', 'title':
'Unleashing the Power of YOLO-NAS: A New Era in Object Detection and Computer Vision', 'description': 'The Future o
f Computer Vision is Here', 'language': 'en'}, page_content='Unleashing the Power of YOLO-NAS: A New Era in Object
Detection and Computer Vision'), Document(metadata={'source': 'https://deci.ai/blog/yolo-nas-object-detection-found
ation-model/', 'title': '- Archives Page 1 | NVIDIA Blog', 'language': 'en-US'}, page_content='How Snowflake Is Unl
ocking the Value of Data With Large Language Models \n\nApplications Now Open for $60,000 NVIDIA Graduate Fellowshi
p Awards \n\nBringing together the world's brightest minds and the latest accelerated computing technology leads to
powerful breakthroughs that help tackle some of the biggest research problems. To foster such innovation, the…\t\t
\tRead Article\t\t \n \n\nNVIDIA Researchers Harness Real-Time Gen AI to Build Immersive Desert World'), Document(m
etadata={'source': "https://deci.ai/blog/decicoder-efficient-and-accurate-code-generation-llm/#:~:text=DeciCoder's%
20unmatched%20throughput%20and%20low,re%20obsessed%20with%20AI%20efficiency.", 'title': '- Archives Page 1 | NVIDIA
Blog', 'language': 'en-US'}, page_content='How Snowflake Is Unlocking the Value of Data With Large Language Models
\n\nApplications Now Open for $60,000 NVIDIA Graduate Fellowship Awards \n\nBringing together the world's brightest
minds and the latest accelerated computing technology leads to powerful breakthroughs that help tackle some of the
biggest research problems. To foster such innovation, the…\t\t\tRead Article\t\t \n \n\nNVIDIA Researchers Harness
Real-Time Gen AI to Build Immersive Desert World'), Document(metadata={'source': 'https://deeplearningdaily.substac
k.com/p/unleashing-the-power-of-yolo-nas', 'title': 'Unleashing the Power of YOLO-NAS: A New Era in Object Detectio
n and Computer Vision', 'description': 'The Future of Computer Vision is Here', 'language': 'en'}, page_content="Su
```

Copy Code

```
response = rag_chain_with_source.invoke(
        "Write a blog about Deci and how it used NAS to generate YOLO-NAS and Dec
print(response['answer'].content)
print(response['context'])
```

```
response = rag_chain_with_source.invoke(
    "Write a blog about Deci and how it used NAS to generate YOLO-NAS and DeciCoder")
```

```
print(response['answer'])
print(response['context'])
```

```
content="Deci utilized Neural Architecture Search (NAS) through its AutoNAC engine to create YOLO-NAS, a novel obje
ct detection architecture that enhances performance in a competitive field. By intelligently exploring a vast archi
tecture space, Deci developed quantization-friendly versions of YOLO-NAS, which outperform existing models like YOL
Ov7 and YOLOv8. Additionally, DeciCoder is designed to streamline the training process and improve model efficienc
y, further demonstrating the capabilities of Deci's innovative technologies." response_metadata={'finish_reason':
'stop', 'model_name': 'gpt-4o-mini-2024-07-18', 'system_fingerprint': 'fp_48196bc67a'} id='run-198580d1-8a7b-4280-9
7cc-01ae818a5f03-0'
[Document(metadata={'source': 'https://deeplearningdaily.substack.com/p/unleashing-the-power-of-yolo-nas', 'title':
'Unleashing the Power of YOLO-NAS: A New Era in Object Detection and Computer Vision', 'description': 'The Future o
f Computer Vision is Here', 'language': 'en'}, page_content='Unleashing the Power of YOLO-NAS: A New Era in Object
Detection and Computer Vision'), Document(metadata={'source': 'https://deeplearningdaily.substack.com/p/unleashing-
the-power-of-yolo-nas', 'title': 'Unleashing the Power of YOLO-NAS: A New Era in Object Detection and Computer Visi
on', 'description': 'The Future of Computer Vision is Here', 'language': 'en'}, page_content="SubscribeSign inShare
this postUnleashing the Power of YOLO-NAS: A New Era in Object Detection and Computer Visiondeeplearningdaily.subst
ack.comCopy linkFacebookEmailNoteOtherUnleashing the Power of YOLO-NAS: A New Era in Object Detection and Computer
VisionThe Future of Computer Vision is HereDeep Learning Daily CommunityMay 05, 20236Share this postUnleashing the
Power of YOLO-NAS: A New Era in Object Detection and Computer Visiondeeplearningdaily.substack.comCopy linkFacebook
EmailNoteOtherShareWhat does it take to make a mark in the fiercely competitive world of object detection?Â\xa0In t
his newsletter edition, I want to take you on a behind-the-scenes journey of how YOLO-NAS, a novel, groundbreaking
object detection architecture that sets a new standard for State-of-the-Art, came into being.Igniting AmbitionOur r
```

This code exemplifies how RAG and LangChain can enhance information retrieval and generation in AI applications.

Explore these articles to know more about RAG and its applications:

- Build a RAG Pipeline With the LLamaIndex
- Enhancing RAG with Retrieval Augmented Fine-tuning
- A Comprehensive Guide to Building Agentic RAG Systems with LangGraph

Conclusion

Retrieval-augmented generation (RAG) represents a transformative leap in artificial intelligence. It seamlessly integrates large language models (LLMs) with external knowledge sources, addressing the limitations of LLMs' parametric

memory ensures responses are current without extensive model retraining. RAG also offers source citations, bolstering transparency and reducing data leakage. In summary, RAG empowers AI to provide more accurate, context-aware, and reliable information, promising a brighter future for AI applications across industries.

Key Takeaways

- Retrieval Augmented Generation (RAG) is a groundbreaking framework that enhances Large Language Models (LLMs) by integrating external knowledge sources.
- RAG overcomes the limitations of LLMs' parametric memory, enabling them to access real-time data, improving contextualization, and providing up-to-date responses.
- With RAG, AI-generated content becomes more accurate, context-aware, and transparent, as it can cite sources and reduce data leakage.
- RAG's updatable memory eliminates frequent model retraining, making it a cost-effective solution for various applications.
- This technology promises to revolutionize AI across industries, providing users with more reliable and relevant information.

Also Read:

- How to Measure Performance of RAG Systems

Hello there! I'm Soumyadarshan Dash, a passionate and enthusiastic person when it comes to data science and machine learning. I'm constantly exploring new topics and techniques in this field, always striving to expand my knowledge and skills. In fact, upskilling myself is not just a hobby, but a way of life for me.

Advanced | Artificial Intelligence | Generative AI | LLMs | RAG

Free Courses

Generative AI - A Way of Life

Explore Generative AI for beginners: create text and images, use top AI tools, learn practical skills, and ethics.

Getting Started with Large Language Models

Building LLM Applications using Prompt Engineering

This free course guides you on building LLM apps, mastering prompt engineering, and developing chatbots with enterprise data.

Improving Real World RAG Systems: Key Challenges & Practical Solutions

Explore practical solutions, advanced retrieval strategies, and agentic RAG systems to improve context, relevance, and accuracy in AI-driven applications.

Microsoft Excel: Formulas & Functions

Master MS Excel for data analysis with key formulas, functions, and LookUp tools in this comprehensive course.

RECOMMENDED ARTICLES

RAG's Innovative Approach to Unifying Ret...

Ask your Documents with Langchain and Deep Lake!

How to Become a RAG Specialist in 2025?

Unveiling Retrieval Augmented Generation (RAG)|...

Improving AI Hallucinations: How RAG Enhances A...

Improving Real-World RAG Systems: Key Challenge...

Top 20+ RAG Interview Questions

Top 6 Books on Retrieval Augmented Generation (...

- Reach a Global Audience
- Share Your Expertise with the World
- Build Your Brand & Audience

- Join a Thriving AI Community
- Level Up Your AI Game
- Expand Your Influence in Genrative AI

Flagship Programs

GenAI Pinnacle Program | GenAI Pinnacle Plus Program | AI/ML BlackBelt Program | Agentic AI Pioneer Program

Boosting | Loan Prediction | Time Series Forecasting | Tableau | Business Analytics | Vibe Coding in Windsurf | Model Deployment using FastAPI | Building Data Analyst AI Agent | Getting started with OpenAI o3-mini | Introduction to Transformers and Attention Mechanisms

Popular Categories

AI Agents | Generative AI | Prompt Engineering | Generative AI Application | News | Technical Guides | AI Tools | Interview Preparation | Research Papers | Success Stories | Quiz | Use Cases | Listicles

Generative AI Tools and Techniques

GANs | VAEs | Transformers | StyleGAN | Pix2Pix | Autoencoders | GPT | BERT | Word2Vec | LSTM | Attention Mechanisms | Diffusion Models | LLMs | SLMs | Encoder Decoder Models | Prompt Engineering | LangChain | LlamaIndex | RAG | Fine-tuning | LangChain AI Agent | Multimodal Models | RNNs | DCGAN | ProGAN | Text-to-Image Models | DDPM | Document Question Answering | Imagen | T5 (Text-to-Text Transfer Transformer) | Seq2seq Models | WaveNet | Attention Is All You Need (Transformer Architecture) | WindSurf | Cursor

Popular GenAI Models

Llama 4 | Llama 3.1 | GPT 4.5 | GPT 4.1 | GPT 4o | o3-mini | Sora | DeepSeek R1 | DeepSeek V3 | Janus Pro | Veo 2 | Gemini 2.5 Pro | Gemini 2.0 | Gemma 3 | Claude Sonnet 3.7 | Claude 3.5 Sonnet | Phi 4 | Phi 3.5 | Mistral Small 3.1 | Mistral NeMo | Mistral-7b | Bedrock | Vertex AI | Qwen QwQ 32B | Qwen 2 | Qwen 2.5 VL | Qwen Chat | Grok 3

AI Development Frameworks

n8n | LangChain | Agent SDK | A2A by Google | SmolAgents | LangGraph | CrewAI | Agno | LangFlow | AutoGen | LlamaIndex | Swarm | AutoGPT

Exploration | Big Data | Common Machine Learning Algorithms | Machine Learning | Google Data Science Agent