



Search

Write



◆ Member-only story

# Creating a Retrieval-Augmented Generation (RAG) Model on Your Laptop for Free

Stan · [Follow](#)

7 min read · Feb 18, 2024



39



...

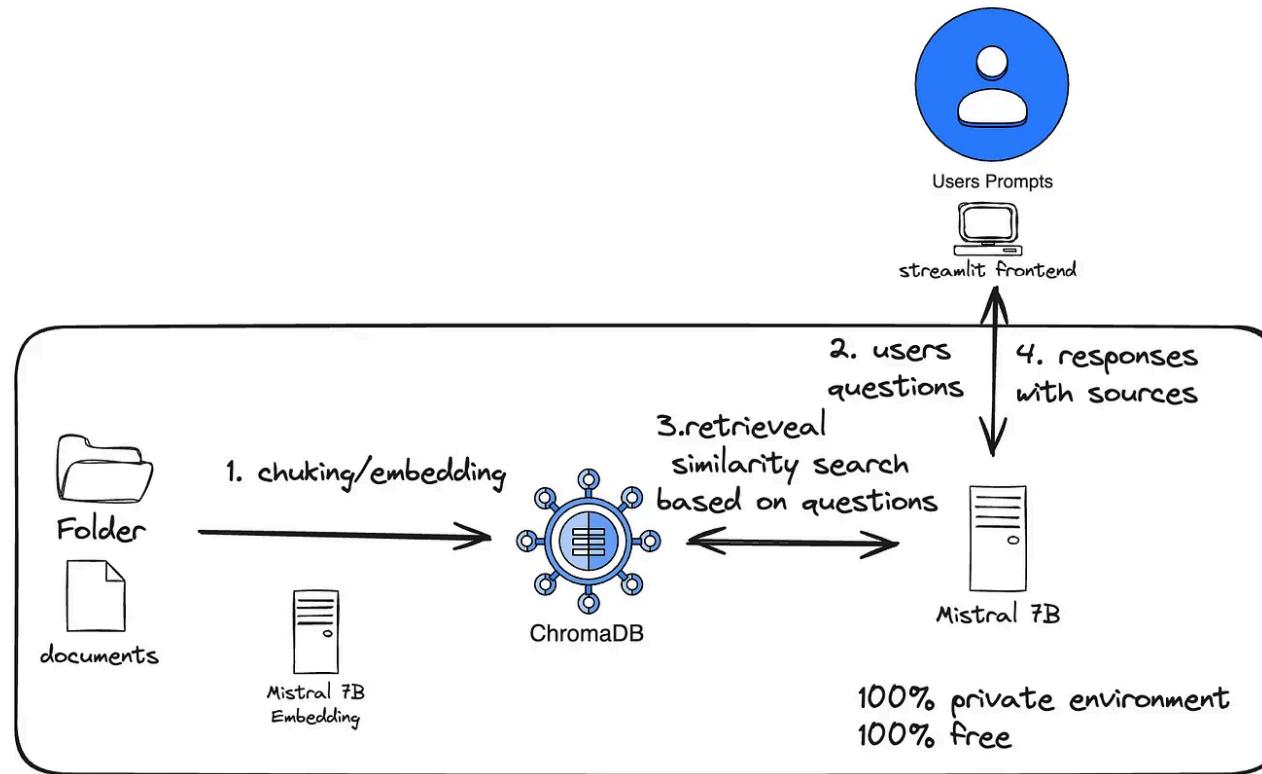
Construct a Robust Knowledge Base Using Mistral-7B, LangChain, ChromaDB, and Streamlit



A llama reads large documents to answer your questions; Mistral7B model running on the Ollama framework to comprehend the customer knowledge base. Image created by ChatGPT.

One of the major challenges in deploying large language models (LLMs) for industrial applications is the significant cost associated with token embeddings and API calls. An alternative to using API-driven LLMs is to download open-source models. However, these models tend to be exceedingly large for standard local machines, compelling users to depend on costly cloud computing services like AWS's EC2. Typically, G5 or higher-tier instances are necessary to host and run these large language models efficiently.

Recent advancements in Large Language Models (LLMs) have paved the way for more cost-effective utilization. In this article, we will guide you through the process of constructing a comprehensive knowledge base from start to finish. We'll leverage the capabilities of Mistral-7B and ChromaDB, orchestrated by LangChain, and encapsulate the functionality within a Streamlit-powered frontend interface.



architecture of Mistral7B based RAG system

## Architecture

The Retrieval-Augmented Generation (RAG) system developed here is powered by the Mistral 7B model and orchestrated using LangChain. This setup runs seamlessly on a MacBook Pro equipped with an M2 chip, 16GB of memory, and macOS Sonoma 14.2.1.

1. user datasets undergo a preprocessing step to create a vector database, for which we employ ChromaDB as our database host. This vector database is established once and subsequently available for quick loading. The embedding process utilizes the Mistral 7B embedding model, ensuring efficient and accurate data representation.

When users input questions, the system retrieves relevant information from the vector database, including sources as metadata. The Mistral model then synthesizes this retrieved information to formulate comprehensive answers, which are delivered through the Streamlit frontend, offering users insightful responses with contextual depth.

### **A Sneak Peek at the Streamlit Interface**

It's noteworthy to highlight that within this application, aptly named 'Chat with Your PDFs', the model demonstrates remarkable versatility. It is adept at processing a wide range of file formats, including texts, JSON files, Word documents, PowerPoint presentations, and more.

# Chat with your PDFs

Ask me a question!

who is albert einstein

Albert Einstein was not mentioned in the context provided. The context discussed objects in scenarios, technical growth, and teamwork for potential projects, as well as some academic papers related to artificial intelligence and computer vision.

## 2. Response

Sources: ./data/PDFs/Self-correcting LLM-controlled Diffusion Models.pdf  
./data/PDFs/How\_to\_build\_your\_carreer\_in\_AI.pdf ./data/PDFs/What Should Data Science Education Do with Large Language Models?.pdf ./data/PDFs/Self-correcting LLM-controlled Diffusion Models.pdf

## 3. source file names

what data scientist do with large language model?

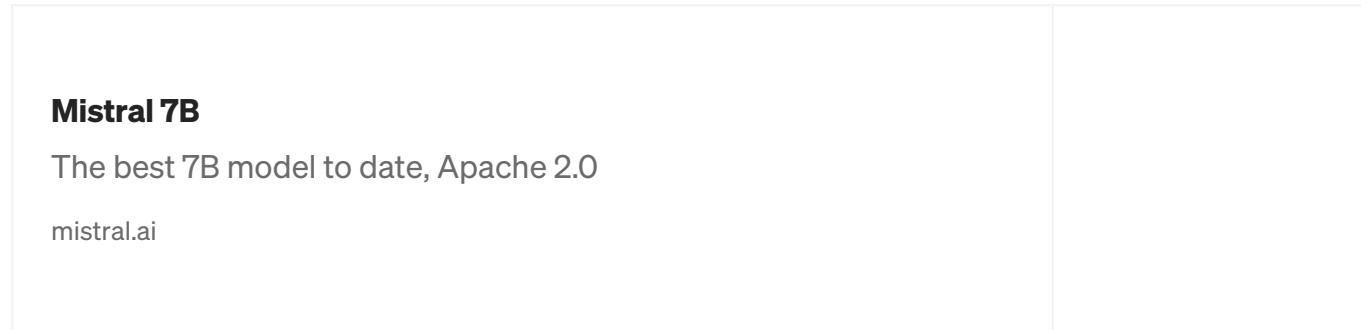
Based on the context provided, data scientists work with large language models (LLMs) such as ChatGPT by using them to streamline complex data science processes. Instead of focusing on hands-on coding and data wrangling, their responsibilities shift towards assessing and managing analyses performed by these automated AIs. LLMs can perform tasks like data cleaning, model building, interpretation, and report writing autonomously, allowing data scientists to focus on strategic planning, resource coordination, and overseeing the overall project life cycle. Additionally, LLMs can improve models by executing hyper-parameter searches and generating reports.

## 1. Type your Questions



## Mistral 7B Model Brief Introduction

To democratize large language models, Mistral AI has provided a 7B base model. The Mistral 7B model can be downloaded and run on a local machine or deployed in a cloud service such as AWS, GCP, or Azure. Mistral 7B is easy to fine-tune for any task. Its model, when fine-tuned for chat, outperforms the Llama 2 13B chat model.



Model	Modality	MMLU	HellaSwag	WinoGrande	PIQA	Arc-e	Arc-c	NQ	TriviaQA	HumanEval	MBPP	MATH	GSM8K
LLaMA 2 7B	Pretrained	44.4%	77.1%	69.5%	77.9%	68.7%	43.2%	24.7%	63.8%	11.6%	26.1%	3.9%	16.0%
LLaMA 2 13B	Pretrained	55.6%	80.7%	72.9%	80.8%	75.2%	48.8%	29.0%	69.6%	18.9%	35.4%	6.0%	34.3%
Code LLaMA 7B	Finetuned	36.9%	62.9%	62.3%	72.8%	59.4%	34.5%	11.0%	34.9%	31.1%	52.5%	5.2%	20.8%
Mistral 7B	Pretrained	60.1%	81.3%	75.3%	83.0%	80.0%	55.5%	28.8%	69.9%	30.5%	47.5%	13.1%	52.1%

Mistral7B shows strong performance in the benchmark tests

## Workflow Tutorial

Following the reference, we use Ollama to run Mistral7B on our local machine. Ollama is a lightweight, extensible framework designed for building and running language models on local machines. It offers a straightforward API for creating, running, and managing models, along with a library of pre-built models that can be easily integrated into various applications.

Download Ollama via the following link:

[https://ollama.com/download?source=post\\_page---2b4ba77358e0---](https://ollama.com/download?source=post_page---2b4ba77358e0---)

After downloading, use the following command to launch Mistral

```
ollama run mistral
```

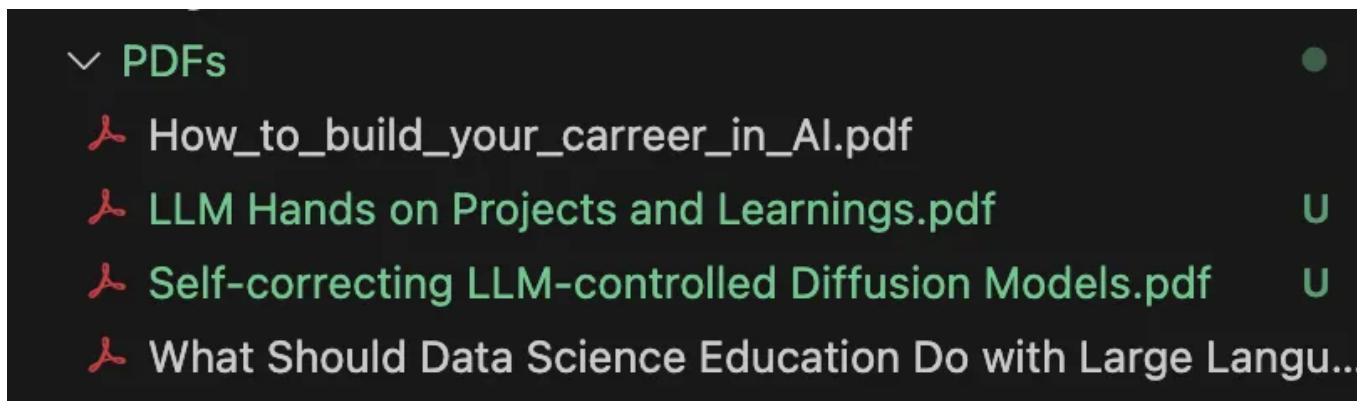
```
pulling manifest
pulling e8a35b5937a5... 100% [progress bar] 4.1 GB
pulling 43070e2d4e53... 100% [progress bar] 11 KB
pulling e6836092461f... 100% [progress bar] 42 B
pulling ed11eda7790d... 100% [progress bar] 30 B
pulling f9b1e3196ecf... 100% [progress bar] 483 B
verifying sha256 digest
writing manifest
removing any unused layers
success
```

the above command will download 4G midek it your local machine

The code for following project can be found in Github [link](#)

## Data

We use four small-sized PDF files (papers) for this test. However, I have also tested it on large PDF files, including books and documents that span hundreds of pages. There is no limitation on document sizes. Initial data processing and embedding can be time-consuming, with document processing taking about 5–6 minutes. For larger datasets, the process could be more time-consuming. However, once the vector database is created, loading and using it becomes efficient.



The following text preprocessing code loads the PDF files in batches, chunks the text, and combines the texts for embedding.

```
# Load environment variables
load_dotenv(find_dotenv())
OPENAI_API_KEY = os.getenv('OPENAI_API_KEY')

# PDF processing
pdf_folder_path = './data/PDFs'
if not os.path.exists(pdf_folder_path):
    raise FileNotFoundError(f"The directory {pdf_folder_path} does not exist.")

def load_and_chunk_pdfs(pdf_folder_path):
    loaders = [UnstructuredPDFLoader(os.path.join(pdf_folder_path, fn))
               for fn in os.listdir(pdf_folder_path) if fn.endswith('.pdf')]
    all_texts = []
    for loader in loaders:
        data = loader.load()
        text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
        texts = text_splitter.split_documents(data)
        all_texts.extend(texts)

    return all_texts
```

```
    all_texts.extend(texts)
    return all_texts
```

In this test, we evaluated both the OpenAI API and the Mistral7B model for embedding, which can be accomplished by:

```
OllamaEmbeddings(model="mistral")
```

ChromaDB is used to create a vector database. However, if the vector database (VDB) already exists, we simply load the VDB using the following code:

```
embeddings = OllamaEmbeddings(model="mistral")
vectorstore_directory = 'vdb_multipdfs'
if not os.path.exists(vectorstore_directory) or not os.listdir(vectorstore_directory):
    all_texts = load_and_chunk_pdfs(pdf_folder_path)
    # Embedding and storing
    vectorstore = Chroma.from_documents(documents=all_texts,
                                         embedding=embeddings,
                                         persist_directory=vectorstore_directory)
else:
    vectorstore = Chroma(persist_directory=vectorstore_directory,
```

Deploy the Mistral model on a local machine using the following url:  
base\_url="http://localhost:11434"

```
# LLM and QA Chain setup
from langchain.schema import LLMResult

llm = Ollama(base_url="http://localhost:11434",
              model="mistral",
              verbose=True,
              )
```

Use a prompt template to further guide the base model's responses. Set up `return_source_documents=True` to ensure that the metadata, including source file names, is returned.

and we can also improve model responses by using rules such as

- If you don't know the answer, just say I do not know
- based on your answer based on provided documents only.
- do NOT answer question out of context.

```
def generate_response(question):
    prompt_template = """Use the following pieces of context to answer the question
- If you don't know the answer, just say I do not know.
- based on your answer based on provided documents only.
- do NOT answer question out of context.

{context}

Question: {question}
"""
    PROMPT = PromptTemplate(
        template=prompt_template, input_variables=["context", "question"]
    )
    chain_type_kwargs = {"prompt": PROMPT}

    qa = RetrievalQA.from_chain_type(
        llm=llm,
        chain_type="stuff",
        retriever=vectorstore.as_retriever(),
        verbose=True,
        chain_type_kwargs=chain_type_kwargs
        ,return_source_documents=True,
    )
    res = qa({"query": question})
    return res
```

Wrap up the code within a streamlit app.

```
# Streamlit interaction
st.title("Chat with your PDFs")
st.session_state.setdefault('messages', [{"role": "assistant", "content": "Ask me a question about your PDFs!"}])

# Handle the chat input and response logic
if prompt := st.chat_input("Your question"):
    st.session_state.messages.append({"role": "user", "content": prompt})
    response = generate_response(prompt)
    assistant_response = response.get('result', "I'm sorry, I couldn't find an answer to your question")
    st.session_state.messages.append({"role": "assistant", "content": assistant_response})

    if 'source_documents' in response and response['source_documents']:
        # Initialize an empty list to hold the source metadata
        sources_metadata = []

        # Loop through each document in the source documents
        for doc in response['source_documents']:
            # Assuming doc is an instance of Document and has a method or property 'metadata'
            # And assuming metadata is an object or dictionary that includes a 'source' key
            try:
                # Attempt to access the source information from the metadata
                source_info = doc.metadata['source'] # Use this if metadata is a plain dictionary
                # If metadata is accessed through methods or properties, adjust accordingly
                sources_metadata.append(source_info)
            except AttributeError:
                # Handle cases where doc does not have a metadata attribute or is not a Document
                print(f"Document {doc} does not have the expected metadata structure")

        # Concatenate all source metadata into one string, separated by new line
        sources_concatenated = '\n'.join(sources_metadata)

        # Append the concatenated source metadata to the assistant's message
        st.session_state.messages.append({"role": "assistant", "content": f"Sources:{sources_concatenated}"})
```

```
# Display messages
for message in st.session_state.messages:
    with st.container():
        role = message["role"]
        #     st.success(message["content"])
        if role == "assistant":
            st.info(message["content"])
            # st.info(message["source"])
        else:
            st.success(message["content"])
```

Finally, run the streamlit app with following command

```
streamlit run Mistral_RAG_Streamlit_End2End.py
```

## Results

The model returns relevant answers based on user-uploaded documents. It is capable of answering questions and providing the source file names. The top four file names are reported.

## what data scientist do with large language model?

Based on the context provided, data scientists work with large language models (LLMs) such as ChatGPT by using them to streamline complex data science processes. Instead of focusing on hands-on coding and data wrangling, their responsibilities shift towards assessing and managing analyses performed by these automated AIs. LLMs can perform tasks like data cleaning, model building, interpretation, and report writing autonomously, allowing data scientists to focus on strategic planning, resource coordination, and overseeing the overall project life cycle. Additionally, LLMs can improve models by executing hyper-parameter searches and generating reports.

Sources: ./data/PDFs/Self-correcting LLM-controlled Diffusion Models.pdf ./data/PDFs/What Should Data Science Education Do with Large Language Models?.pdf ./data/PDFs/What Should Data Science Education Do with Large Language Models?.pdf ./data/PDFs/What Should Data Science Education Do with Large Language Models?.pdf

Another example of answer questions providing source file names

which paper talking about diffusion model? provide paper names

Based on the context provided, the following papers mention or discuss diffusion models:

[4] Diffedit: Diffusion-based semantic image editing with mask guidance - by Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. [5] Diffusion self-guidance for controllable image generation - by Dave Epstein, Allan Jabri, Ben Poole, Alexei A. Efros, and Aleksander Holynski. [6] Layoutgpt: Compositional visual planning and generation with large language models - by Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Ar-jun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang.

Therefore, the answer is: The papers [4], [5], and [6] mention or discuss diffusion models.

Sources: ./data/PDFs/Self-correcting LLM-controlled Diffusion Models.pdf ./data/PDFs/What Should Data Science Education Do with Large Language Models?.pdf ./data/PDFs/Self-correcting LLM-controlled Diffusion Models.pdf ./data/PDFs/Self-correcting LLM-controlled Diffusion Models.pdf

## 2. Hallucination Free

The system does not answer questions irrelevant to the document, a feature crucial for business owners.

# Chat with your PDFs

Ask me a question!

who is albert einstein

Albert Einstein was not mentioned in the context provided. The context discussed objects in scenarios, technical growth, and teamwork for potential projects, as well as some academic papers related to artificial intelligence and computer vision.

Sources: ./data/PDFs/Self-correcting LLM-controlled Diffusion Models.pdf  
./data/PDFs/How\_to\_build\_your\_carreer\_in\_AI.pdf ./data/PDFs/What Should Data Science Education Do with Large Language Models?.pdf ./data/PDFs/Self-correcting LLM-controlled Diffusion Models.pdf

since the provided document does not have information about albert Einstein, it returns negative information

In summary, by using the workflow outlined above, it's possible to construct an end-to-end knowledge base and Retrieval-Augmented Generation (RAG) system using a large language model on a standard laptop at no cost.

With ongoing research and efforts to democratize large language models, we're moving towards the capability to use these models on edge devices, such as mobile apps or smart watch or cars.

My next project will involve utilizing this tool to create a personal knowledge base, assisting a family member in managing their personal financial data, insurance, and medical bills to identify potential ‘fine print’ or red flag entries or fraudulent transactions.

If you found this article helpful, please follow me on Medium or give it a clap (5 times or more). If you have any questions or suggestions for improvement, I'd love to hear from you.

## References

**GitHub - mistralai/mistral-src: Reference implementation of Mistral AI 7B v0.1 model.**

Reference implementation of Mistral AI 7B v0.1 model. -  
mistralai/mistral-src

[github.com](https://github.com/mistralai/mistral-src)

## Talk to your files in a local RAG application using Mistral 7B, LangChain 🦜🔗 and Chroma DB (No...)

I will show how you can use the Mistral 7B model on your local machine to talk to your personal files in a Chroma...

medium.com

Rag

Llm

Streamlit

Mistral7b

Chromadb



**Written by Stan**

1.1K Followers

A director data scientist working in a tech start-up who is passionate about making a positive impact on people around him

Follow

## More from Stan



Stan

### TimeSeries Split with Sklearn Tips

3 min read · Nov 20, 2021

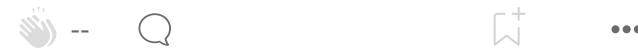


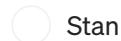
Stan

### Summarizing and Visualizing Your CSV Using Large Language Models

LIDA Automatic Generation of Visualizations and Infographics using Large Language...

★ · 6 min read · Jan 18, 2024





Stan

## Integrate Python Script for Data Processing, Machine learning...

Power BI, a leader in BI and data visualization, offers robust querying with Power Query....

★ · 4 min read · Jan 28, 2024



--



...



Stan

## Reading Multiple PDFs and Identifying Sources using...

Many use cases require people to investigate multiple PDF files, such as combing through...

★ · 7 min read · Jul 15, 2023



--



2



...

See all from Stan

## Recommended from Medium



Júlio Almeida in Python in Plain English

### Claude 3: The king of data extraction

Unleashing the Future of Data Extraction:  
Inside Claude 3's Revolutionary Leap



· 7 min read · Mar 27, 2024



--



4



...



(λx.x)eranga in Effectz.AI

### Build RAG Application Using a LLM Running on Local Computer with...

Privacy-preserving LLM without GPU

17 min read · Mar 25, 2024



--



3



...

## Lists

## Natural Language Processing

1355 stories · 837 saves

## Predictive Modeling w/ Python

20 stories · 1071 saves

## ChatGPT prompts

47 stories · 1381 saves

 Benedict Neo in bitgrit Data Science Publication

## Roadmap to Learn AI in 2024

A free curriculum for hackers and  
programmers to learn AI

11 min read · Mar 11, 2024



--



103



...

 Maya Akim

## How to Add ANY Model to Ollama

I have somewhat accidentally created a  
“smart” LLM with 7b parameters.

 · 5 min read · Mar 17, 2024



--



1



...

 Florian June in AI Advances

## Advanced RAG 09: Prompt Compression

Method Classification, Algorithm Principles and Code Explanation

◆ · 29 min read · 3 days ago



1



...

 Kong Nopwattanapong in Credera Engineering

## Build a simple RAG chatbot with LangChain

In this blog post, Engineer Kong Nopwattanapong explains how to create a...

12 min read · Feb 12, 2024



3



...

See more recommendations