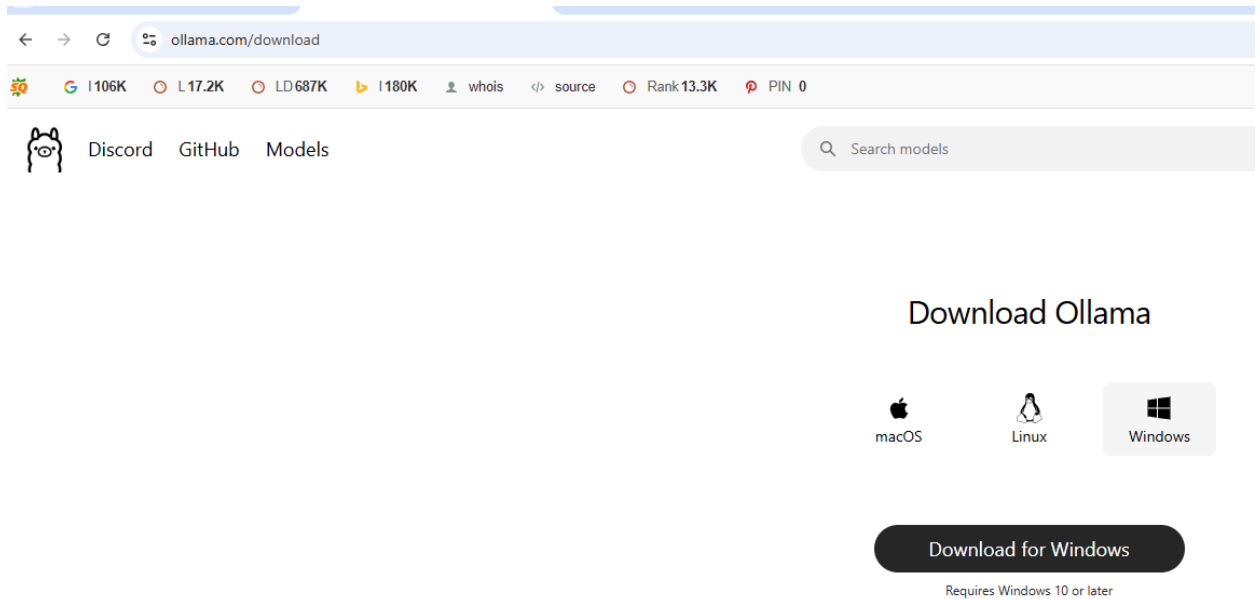


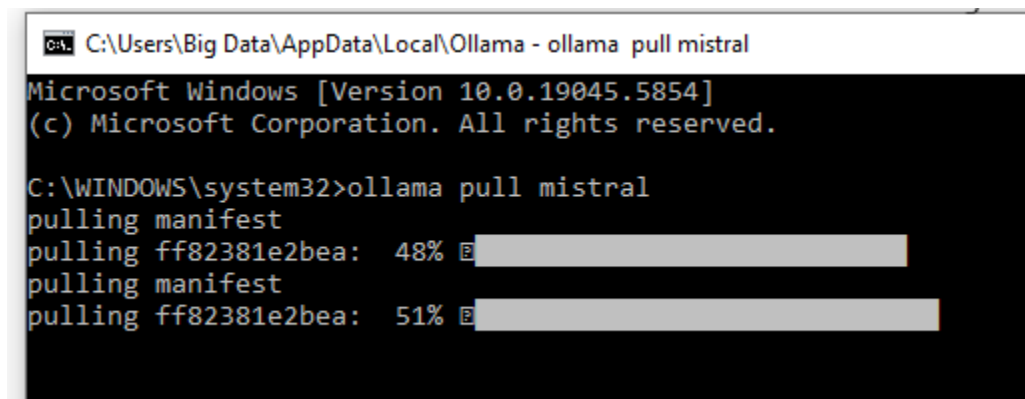
Run Mistral with Ollama (Easier for Local Devs)

1. Install Ollama: <https://ollama.com>



2. Run:

```
ollama pull mistral
```



```
C:\Users\Big Data\AppData\Local\Ollama
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ollama pull mistral
pulling manifest
pulling ff82381e2bea: 48% ▢ 2.0 GB/4.1 GB 11 MB/s 3m
pulling manifest
pulling ff82381e2bea: 100% ▢ 4.1 GB
pulling 43070e2d4e53: 100% ▢ 11 KB
pulling 491dfa501e59: 100% ▢ 801 B
pulling ed11eda7790d: 100% ▢ 30 B
pulling 42347cd80dc8: 100% ▢ 485 B
verifying sha256 digest
writing manifest
success

C:\WINDOWS\system32>
```

3. Use with LangChain:

Python:-

```
from langchain_community.llms import Ollama

llm = Ollama(model="mistral")
```

Here's a **complete Python script** for a simple PDF Q&A agent using **Mistral 7B Instruct** via Hugging Face Transformers, plus SentenceTransformers for embeddings and FAISS for vector search.

What this script does:

- Loads a PDF and extracts text.
- Splits text into chunks.
- Embeds chunks with SentenceTransformers.
- Builds a FAISS vector store.
- Runs queries using Mistral LLM locally.
- Returns answers based on relevant PDF context.

Prerequisites:

```
pip install fitz PyMuPDF transformers sentence-transformers faiss-cpu
langchain
```

Save following python code as `pdf_qa_mistral.py`:-

```
import fitz # PyMuPDF
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline
from sentence_transformers import SentenceTransformer
import faiss
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.vectorstores import FAISS

# ----- Step 1: Extract text from PDF -----
def extract_text_from_pdf(pdf_path):
    doc = fitz.open(pdf_path)
    text = ""
    for page in doc:
        text += page.get_text()
    return text

# ----- Step 2: Split text into chunks -----
def chunk_text(text, chunk_size=1000, chunk_overlap=200):
    splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size,
chunk_overlap=chunk_overlap)
    chunks = splitter.split_text(text)
    return chunks

# ----- Step 3: Embed text chunks -----
def embed_chunks(chunks):
    embedder = SentenceTransformer("all-MiniLM-L6-v2")
    embeddings = embedder.encode(chunks)
    vectorstore = FAISS.from_texts(chunks, embeddings)
    return vectorstore

# ----- Step 4: Load Mistral model -----
def load_mistral():
    model_id = "mistralai/Mistral-7B-Instruct-v0.1"
    tokenizer = AutoTokenizer.from_pretrained(model_id)
    model = AutoModelForCausalLM.from_pretrained(model_id, device_map="auto",
torch_dtype="auto")
    generator = pipeline("text-generation", model=model, tokenizer=tokenizer)
    return generator

# ----- Step 5: Generate answer using context -----
def answer_question(generator, question, relevant_chunks):
    context = "\n\n".join(relevant_chunks)
    prompt = f"Answer the question based on the following
context:\n{context}\n\nQuestion: {question}\nAnswer:"
```

```

    output = generator(prompt, max_new_tokens=200,
temperature=0.3)[0]["generated_text"]
    # Extract answer after "Answer:"
    answer = output.split("Answer:")[1].strip()
    return answer

# ----- Main Q&A Loop -----
def main(pdf_path):
    print("Extracting text from PDF...")
    text = extract_text_from_pdf(pdf_path)

    print("Chunking text...")
    chunks = chunk_text(text)

    print("Embedding chunks...")
    vectorstore = embed_chunks(chunks)

    print("Loading Mistral model...")
    generator = load_mistral()

    print("Ready! Ask questions about your PDF (type 'exit' to quit).")
    while True:
        query = input("Question: ").strip()
        if query.lower() in ['exit', 'quit']:
            break

        # Retrieve top 3 relevant chunks
        docs_and_scores = vectorstore.similarity_search_with_score(query, k=3)
        relevant_chunks = [doc.page_content for doc, _ in docs_and_scores]

        answer = answer_question(generator, query, relevant_chunks)
        print("\nAnswer:", answer)
        print("-" * 40)

if __name__ == "__main__":
    pdf_file = input("Enter path to your PDF file: ").strip()
    main(pdf_file)

```

1. Run:

```
python pdf_qa_mistral.py
```

3. Enter your PDF file path when prompted.
4. Ask questions about the PDF content!
5. Type `exit` to quit.

Here's a Streamlit web app version of your Mistral-powered PDF Q&A agent.

📄 What You'll Get

A local web app where users can:

Upload a PDF file

Ask questions about the PDF content

Get answers powered by Mistral 7B Instruct

📄 Prerequisites

`pip install streamlit PyMuPDF transformers sentence-transformers faiss-cpu langchain`

Optional (if you want GPU support):

`pip install torch accelerate bitsandbytes`

Save This As `pdf_qa_app.py` python code:-

```
import streamlit as st
```

```

import fitz # PyMuPDF

from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline

from sentence_transformers import SentenceTransformer

from langchain.text_splitter import RecursiveCharacterTextSplitter

from langchain.vectorstores import FAISS


# --- Extract text from PDF ---

def extract_text_from_pdf(pdf_file):

    doc = fitz.open(stream=pdf_file.read(), filetype="pdf")

    text = ""

    for page in doc:

        text += page.get_text()

    return text


# --- Chunk PDF text ---

def chunk_text(text, chunk_size=1000, chunk_overlap=200):

    splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size, chunk_overlap=chunk_overlap)

    return splitter.split_text(text)


# --- Embed chunks and build FAISS index ---

@st.cache_resource

def embed_chunks(chunks):

    embedder = SentenceTransformer("all-MiniLM-L6-v2")

    vectorstore = FAISS.from_texts(chunks, embedder)

    return vectorstore

```

```

# --- Load Mistral 7B model locally ---

@st.cache_resource

def load_mistral_model():

    model_id = "mistralai/Mistral-7B-Instruct-v0.1"

    tokenizer = AutoTokenizer.from_pretrained(model_id)

    model = AutoModelForCausalLM.from_pretrained(model_id, device_map="auto",
torch_dtype="auto")

    return pipeline("text-generation", model=model, tokenizer=tokenizer)


# --- Ask question using Mistral ---

def get_answer(generator, question, context_chunks):

    context = "\n\n".join(context_chunks)

    prompt = f"Answer the question based on the following context:\n\n{context}\n\nQuestion:
{question}\nAnswer:"

    output = generator(prompt, max_new_tokens=300, temperature=0.3)[0]["generated_text"]

    return output.split("Answer:")[1].strip()


# --- Streamlit App ---

st.title("📖 Ask My PDF (Mistral Edition)")

uploaded_pdf = st.file_uploader("Upload a PDF file", type="pdf")

if uploaded_pdf:

    with st.spinner("Reading PDF and preparing data..."):

        raw_text = extract_text_from_pdf(uploaded_pdf)

```

```
chunks = chunk_text(raw_text)

vectorstore = embed_chunks(chunks)

generator = load_mistral_model()

st.success("Ready to ask questions!")


question = st.text_input("Ask a question about the PDF:")


if question:

    docs_with_scores = vectorstore.similarity_search_with_score(question, k=3)

    relevant_chunks = [doc.page_content for doc, _ in docs_with_scores]

    with st.spinner("Thinking..."):

        answer = get_answer(generator, question, relevant_chunks)

    st.markdown("**Answer:**")

    st.write(answer)
```

How to Run It

```
streamlit run pdf_qa_app.py
```

Then open your browser to the local Streamlit URL (usually `http://localhost:8501`).