

## Install Dependencies

```
!pip install langchain==0.1.13 langchain-community langchain-google-genai sentence-transformers chromadb pypdf reportlab
```

```
Requirement already satisfied: langchain==0.1.13 in /usr/local/lib/python3.11/dist-packages (0.1.13)
Requirement already satisfied: langchain-community in /usr/local/lib/python3.11/dist-packages (0.0.38)
Requirement already satisfied: langchain-google-genai in /usr/local/lib/python3.11/dist-packages (1.0.4)
Requirement already satisfied: sentence-transformers in /usr/local/lib/python3.11/dist-packages (4.1.0)
Requirement already satisfied: chromadb in /usr/local/lib/python3.11/dist-packages (1.0.12)
Requirement already satisfied: pypdf in /usr/local/lib/python3.11/dist-packages (5.6.0)
Requirement already satisfied: reportlab in /usr/local/lib/python3.11/dist-packages (4.4.1)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (6.0.2)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (2.0.41)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (3.11.15)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (0.6.7)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (1.33)
Requirement already satisfied: langchain-core<0.2.0,>=0.1.33 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (0.1.33)
Requirement already satisfied: langchain-text-splitters<0.1,>=0.0.1 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (0.0.1)
Requirement already satisfied: langsmith<0.2.0,>=0.1.17 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (0.1.17)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (1.26.4)
Requirement already satisfied: pydantic<3,>=1 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (2.11.5)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (2.32.3)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in /usr/local/lib/python3.11/dist-packages (from langchain==0.1.13) (8.5.0)
Requirement already satisfied: google-generativeai<0.6.0,>=0.5.2 in /usr/local/lib/python3.11/dist-packages (from langchain-google-genai) (0.5.2)
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (4.41.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (4.67.1)
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (2.6.0+cu124)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.6.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.15.3)
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (0.20.0)
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (11.2.1)
Requirement already satisfied: typing_extensions>=4.5.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (4.5.0)
Requirement already satisfied: build>=1.0.3 in /usr/local/lib/python3.11/dist-packages (from chromadb) (1.2.2.post1)
Requirement already satisfied: fastapi>=0.115.9 in /usr/local/lib/python3.11/dist-packages (from chromadb) (0.115.9)
Requirement already satisfied: uvicorn>=0.18.3 in /usr/local/lib/python3.11/dist-packages (from uvicorn[standard]>=0.18.3->chromadb) (0.18.3)
Requirement already satisfied: posthog>=2.4.0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (4.8.0)
Requirement already satisfied: onnxruntime>=1.14.1 in /usr/local/lib/python3.11/dist-packages (from chromadb) (1.22.0)
Requirement already satisfied: opentelemetry-api>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (1.27.0)
Requirement already satisfied: opentelemetry-exporter-otlp-proto-grpc>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (1.27.0)
Requirement already satisfied: opentelemetry-instrumentation-fastapi>=0.41b0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (0.41b0)
Requirement already satisfied: opentelemetry-sdk>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (1.27.0)
Requirement already satisfied: tokenizers>=0.13.2 in /usr/local/lib/python3.11/dist-packages (from chromadb) (0.21.1)
Requirement already satisfied: pypika>=0.48.9 in /usr/local/lib/python3.11/dist-packages (from chromadb) (0.48.9)
Requirement already satisfied: overrides>=7.3.1 in /usr/local/lib/python3.11/dist-packages (from chromadb) (7.7.0)
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.11/dist-packages (from chromadb) (6.5.2)
Requirement already satisfied: grpcio>=1.58.0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (1.72.1)
Requirement already satisfied: bcrypt>=4.0.1 in /usr/local/lib/python3.11/dist-packages (from chromadb) (4.3.0)
Requirement already satisfied: typer>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (0.16.0)
Requirement already satisfied: kubernetes>=28.1.0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (33.1.0)
Requirement already satisfied: mmh3>=4.0.1 in /usr/local/lib/python3.11/dist-packages (from chromadb) (5.1.0)
Requirement already satisfied: orjson>=3.9.12 in /usr/local/lib/python3.11/dist-packages (from chromadb) (3.10.18)
Requirement already satisfied: httpx>=0.27.0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (0.28.1)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (13.9.4)
Requirement already satisfied: jsonschema>=4.19.0 in /usr/local/lib/python3.11/dist-packages (from chromadb) (4.24.0)
Requirement already satisfied: starlette<0.46.0,>=0.40.0 in /usr/local/lib/python3.11/dist-packages (from fastapi==0.115.9->chromadb) (0.40.0)
Requirement already satisfied: chardet in /usr/local/lib/python3.11/dist-packages (from reportlab) (5.2.0)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (2.3.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.1.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (25.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.1.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (6.0.0)
```

## Upload PDFs from Local

```
from google.colab import files
import shutil
import os

# Remove existing folder if exists
if os.path.exists("/content/papers"):
    shutil.rmtree("/content/papers")

# Create target directory
os.makedirs("/content/papers", exist_ok=True)

# Upload files
uploaded = files.upload()

# Move only valid non-empty PDFs into the folder
for filename in uploaded.keys():
    if filename.lower().endswith('.pdf') and os.path.getsize(filename) > 0:
        shutil.move(filename, f"/content/papers/{filename}")
    else:
```

```
print(f"Skipped invalid or empty file: {filename}")
```

```
print("Files uploaded successfully.")
```



Choose Files 3 files

- **FewShot.pdf**(application/pdf) - 6768044 bytes, last modified: 6/13/2025 - 100% done
- **RAG.pdf**(application/pdf) - 885323 bytes, last modified: 6/13/2025 - 100% done
- **Transformer.pdf**(application/pdf) - 2215244 bytes, last modified: 6/13/2025 - 100% done

Saving FewShot.pdf to FewShot.pdf

Saving RAG.pdf to RAG.pdf

Saving Transformer.pdf to Transformer.pdf

Files uploaded successfully

## Load and Process PDF Documents

```
from langchain_community.document_loaders import PyPDFDirectoryLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
import glob
from pypdf.errors import PdfReadError

# Helper: remove truly empty PDFs before loading
for pdf_file in glob.glob("/content/papers/*.pdf"):
    if os.path.getsize(pdf_file) == 0:
        print(f"Removing empty file: {pdf_file}")
        os.remove(pdf_file)

# Load PDFs
loader = PyPDFDirectoryLoader("/content/papers")
try:
    docs = loader.load()
except PdfReadError as e:
    print(f"Error reading a PDF: {e}")

# Add metadata
for i, doc in enumerate(docs):
    source = doc.metadata.get('source', f'doc_{i}.pdf')
    doc.metadata['filename'] = source.split('/')[-1]
    doc.metadata['page'] = doc.metadata.get('page', i + 1)

# Split into text chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size=300, chunk_overlap=50)
chunks = text_splitter.split_documents(docs)

print(f"Loaded {len(docs)} pages and split into {len(chunks)} chunks.")
print("Final list of files to process:")
print(os.listdir("/content/papers"))

Loaded 109 pages and split into 1422 chunks.
Final list of files to process:
['FewShot.pdf', 'Transformer.pdf', 'RAG.pdf']
```

## Generate Embeddings and Build Vectorstore

```
from langchain_community.embeddings import SentenceTransformerEmbeddings
from langchain.vectorstores import Chroma

# Load embedding model
embeddings = SentenceTransformerEmbeddings(model_name="all-MiniLM-L6-v2")

# Create vectorstore from chunks
vectorstore = Chroma.from_documents(chunks, embeddings)

# Create retriever
retriever = vectorstore.as_retriever(search_kwargs={'k': 5})
```

## Set Up Gemini API & LLM

```
import os

# Set your Google API key
os.environ["GOOGLE_API_KEY"] = "AIzaSyDnxnyeCE5v3TsVZBJro14Q7XJ5KB0VzZ4"

from langchain_google_genai import ChatGoogleGenerativeAI

# Initialize LLM
llm = ChatGoogleGenerativeAI(model="gemini-1.5-flash", temperature=0.5)
```

## Define Prompt and Context Formatter

```

from langchain.prompts import ChatPromptTemplate

# Prepare context from retrieved docs
def prepare_context_with_sources(documents):
    context_blocks = []
    source_citations = set()

    for doc in documents:
        filename = doc.metadata.get("filename", "unknown_file")
        page = doc.metadata.get("page", "N/A")
        content = doc.page_content.strip().replace("\n", " ")

        context_blocks.append(f"[{filename}, Page {page}]: {content}")
        source_citations.add((filename, page))

    return "\n\n".join(context_blocks), source_citations

# Define prompt template
template = """
<context>
{context}
</context>

You are an AI assistant answering questions based on academic papers.
Answer the following question truthfully and clearly using only the above context.
Do not hallucinate or make up information.

```

```

Question: {query}
"""

```

```

prompt = ChatPromptTemplate.from_template(template)

```

## Define RAG Function & History Tracker

```

# Initialize history list
qa_history = []

# Function to run RAG + source citations
def rag_with_sources(query):
    docs = retriever.get_relevant_documents(query)
    context, sources = prepare_context_with_sources(docs)

    inputs = {"context": context, "query": query}
    answer = llm.invoke(prompt.format_prompt(**inputs).to_messages())

    formatted_sources = [f"{file}, Page {page}" for file, page in sources]
    qa_entry = {
        "question": query,
        "answer": answer.content.strip(),
        "sources": formatted_sources
    }
    qa_history.append(qa_entry)

    return qa_entry

```

## Sample questions test

```


sample_questions = [
    "What are the main components of a RAG model, and how do they interact?",
    "What are the two sub-layers in each encoder layer of the Transformer model?",
    "Explain how positional encoding is implemented in Transformers and why it is necessary.",
    "Describe the concept of multi-head attention in the Transformer architecture. Why is it beneficial?",
    "What is few-shot learning, and how does GPT-3 implement it during inference?"
]

for q in sample_questions:
    result = rag_with_sources(q)
    print(f"\nQ: {result['question']}\nA: {result['answer']}\nSources: {' '.join(result['sources'])}\n")

# Interactive loop
import sys
while True:
    user_input = input("Ask a question (or type 'exit'): ")
    if user_input.lower() == "exit":
        print("Exiting Q&A.")
        break
    if user_input.strip() == "":
        continue
    result = rag_with_sources(user_input)

```

```
print(f"\nQ: {result['question']}\nA: {result['answer']}\nSources: {'', '.join(result['sources'])}\n")
```

 /usr/local/lib/python3.11/dist-packages/langchain\_core/\_api/deprecation.py:119: LangChainDeprecationWarning: The method `BaseRetriever.warn\_deprecated`

Q: What are the main components of a RAG model, and how do they interact?

A: Based on the provided text, RAG models use an input sequence (x) to retrieve text documents (z). These retrieved documents (z) are then used to generate the final output (y).  
Sources: RAG.pdf, Page 9, RAG.pdf, Page 1, RAG.pdf, Page 2

Q: What are the two sub-layers in each encoder layer of the Transformer model?

A: The first sub-layer is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network.  
Sources: Transformer.pdf, Page 7, Transformer.pdf, Page 2

Q: Explain how positional encoding is implemented in Transformers and why it is necessary.

A: In Transformers, positional encodings are added to the input embeddings at the bottom of both the encoder and decoder stacks. This allows the model to process sequences of varying lengths.  
Sources: Transformer.pdf, Page 5, Transformer.pdf, Page 1, Transformer.pdf, Page 2

Q: Describe the concept of multi-head attention in the Transformer architecture. Why is it beneficial?

A: Based on the provided text, multi-head attention in the Transformer architecture allows the model to jointly attend to information from different parts of the input sequence.  
Sources: Transformer.pdf, Page 4, Transformer.pdf, Page 1, Transformer.pdf, Page 2, Transformer.pdf, Page 9

Q: What is few-shot learning, and how does GPT-3 implement it during inference?

A: Based on the provided text, few-shot learning is a method where a language model, like GPT-3, is given a small number of examples to learn from.  
Sources: FewShot.pdf, Page 4, FewShot.pdf, Page 33, FewShot.pdf, Page 11, FewShot.pdf, Page 12

Ask a question (or type 'exit'): What is Few-Shot Learning?

Q: What is Few-Shot Learning?

A: Based on the provided text, few-shot learning is a method that improves dramatically with model size and uses a small amount of data.  
Sources: FewShot.pdf, Page 4, FewShot.pdf, Page 5, FewShot.pdf, Page 17, FewShot.pdf, Page 33

Ask a question (or type 'exit'): exit

Exiting Q&A.