```
!pip install -qU \langchain==0.0.292 \openai==0.28.0 \datasets==2.10.1 \pinecone-client==2.2.4 \tiktoken==0.5.1
# Set OpenAI API key
import os
os.environ["OPENAI_API_KEY"] = "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"
# Initialize OpenAI Chat model
from langchain.chat_models import ChatOpenAI
chat = ChatOpenAI(openai_api_key=os.environ["OPENAI_API_KEY"], model='gpt-3.5-turbo')
# Document path
document_path = "/content/PARIS1.txt"
# Read document content
with open(document_path, "r", encoding="utf-8") as file:
    document_content = file.read()
# Building the Knowledge Base
import pinecone
# Get API key from app.pinecone.io and environment from console
pinecone.init(api_key='c86b72b1-4266-4b63-9b7e-8a7735663780', environment='gcp-starter')
# Initialize the index
import time
index_name = 'llama-2-rag'
if index_name not in pinecone.list_indexes():
    pinecone.create_index(index_name, dimension=300, metric='cosine')
    # Wait for index to finish initialization
    while not pinecone.describe_index(index_name).status['ready']:
        time.sleep(1)
index = pinecone.Index(index_name)
# Create Vector embeddings
from langchain.embeddings.openai import OpenAIEmbeddings
embed_model = OpenAIEmbeddings(model="text-embedding-ada-002")
# Embed the document content
embed_document = embed_model.embed_documents([document_content])[0]
# Generate metadata for the document
document_metadata = {'text': document_content}
# Upsert the document into Pinecone
document_id = "custom-document-id"
index.upsert(vectors=[(document_id, embed_document, document_metadata)])
# Vector database
from langchain.vectorstores import Pinecone
text_field = "text"  # the metadata field that contains our text
# Initialize the vector store object
vectorstore = Pinecone(index, embed_model.embed_query, text_field)
# Query the index and see if we have any relevant information given our question
query = "What is so special about paris?"
vectorstore.similarity_search(query, k=2)
# Connect the output from our vector store to our chatbot
def augment_prompt(query: str):
    # Get top 3 results from the knowledge base
    results = vectorstore.similarity_search(query, k=2)
    # Get the text from the results
    source_knowledge = "\n".join([x.page_content for x in results])
    # Feed into an augmented prompt
    augmented_prompt = f"""Using the contexts below, answer the query.
    Contexts:
    {source_knowledge}
    Query: {query}"""
    return augmented_prompt
print(augment_prompt(query))
# Chat interaction
from langchain.schema import SystemMessage, HumanMessage, AIMessage
messages = [SystemMessage(content="You are a helpful assistant."),
            HumanMessage(content="Hi AI, how are you today?"),
            AIMessage(content="I'm great thank you. How can I help you?"),
            HumanMessage(content="I'd like to understand string theory.")]
# Create a new user prompt with augmented knowledge
prompt = HumanMessage(content=augment_prompt(query))
messages.append(prompt)
# Use the chat model to get a response
res = chat(messages)
print(res.content)
# Example with RAG
prompt_with_rag = HumanMessage(content=augment_prompt("What is the history of paris"))
res_with_rag = chat(messages + [prompt_with_rag])
print(res_with_rag.content)
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 14.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.5/76.5 kB 6.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 469.0/469.0 kB 11.3 MB/s eta 0:00:00
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179.4/179.4 kB 6.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 18.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.2/48.2 kB 2.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 kB 8.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 11.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.5/62.5 kB 3.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 300.4/300.4 kB 21.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.4/49.4 kB 5.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.3/134.3 kB 12.0 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the sourc
llmx 0.0.15a0 requires cohere, which is not installed.
/usr/local/lib/python3.10/dist-packages/pinecone/index.py:4: TqdmExperimentalWarning: Using `tqdm.autonotebook.tqdm` in notebook mode
  from tqdm.autonotebook import tqdm
/usr/local/lib/python3.10/dist-packages/langchain/vectorstores/pinecone.py:59: UserWarning: Passing in `embedding` as a Callable is de
  warnings.warn(
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 4.0 seconds as
Using the contexts below, answer the query.
    Contexts:
    PARIS



Introduction:
Paris, the capital city of France, is a global icon synonymous with romance, art, and sophistication. Often referred to as the "City c


Historical Background:
The history of Paris can be traced back to ancient times when it was a Roman settlement known as Lutetia. Over the centuries, Paris ha


Architectural Splendors:
Paris is a city that seamlessly blends the medieval with the modern, evident in its diverse architectural styles. The iconic Eiffel To


Cultural Heritage:
Paris has long been a global center for arts and culture. The city's bohemian neighborhoods, such as Montmartre, were once the gatheri




Culinary Excellence:
French cuisine is renowned worldwide, and Paris is at the heart of this gastronomic tradition. The city boasts an array of bistros, ca


Fashion Capital:
Paris stands tall as a global fashion capital, hosting prestigious events like Paris Fashion Week. The city's couture houses, includir


Quality of Life:
Parisians enjoy a high quality of life, with a focus on leisure, education, and a well-developed public transportation system. The cit
```