✔ Hecho

Apertura: martes, 9 de abril de 2024, 08:23
Cierre: viernes, 19 de abril de 2024, 23:59

Using Lang CHain, Pinecone and OpenAI, develop the following challenges:

1. Using Python, write a program to send prompts to Chatgpt and retrieve responses.

https://python.langchain.com/docs/integrations/llms/openai

first program:

```
from langchain.chains import LLMChain
#from langchain.llms import OpenAI
from langchain_community.llms import OpenAI
from langchain.prompts import PromptTemplate

import os

os.environ["OPENAI_API_KEY"] = "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"



template = """Question: {question}

Answer: Let's think step by step."""

prompt = PromptTemplate(template=template, input_variables=["question"])

llm = OpenAI()

llm_chain = LLMChain(prompt=prompt, llm=llm)

question = "What is at the core of Popper's theory of science?"

response = llm_chain.run(question)
print(response)
```

2. Write a simple RAG using an in-memory vector database.

https://python.langchain.com/docs/use_cases/question_answering/

```
import bs4
from langchain import hub
from langchain_community.chat_models import ChatOpenAI
from langchain_community.document_loaders import WebBaseLoader
from langchain_community.embeddings import OpenAIEmbeddings
from langchain.schema import StrOutputParser
from langchain.schema.runnable import RunnablePassthrough
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.vectorstores import Chroma
import os

os.environ["OPENAI_API_KEY"] = "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"


loader = WebBaseLoader(
    web_paths=("https://lilianweng.github.io/posts/2023-06-23-agent/",),
    bs_kwargs=dict(
        parse_only=bs4.SoupStrainer(
            class_=("post-content", "post-title", "post-header")
        )
    ),
)
docs = loader.load()

text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
splits = text_splitter.split_documents(docs)
print(splits[0])
print(splits[1])

vectorstore = Chroma.from_documents(documents=splits, embedding=OpenAIEmbeddings())
retriever = vectorstore.as_retriever()

prompt = hub.pull("rlm/rag-prompt")
llm = ChatOpenAI(model_name="gpt-3.5-turbo", temperature=0)


def format_docs(docs):
    return "\n\n".join(doc.page_content for doc in docs)


rag_chain = (
    {"context": retriever | format_docs, "question": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)

response = rag_chain.invoke("What is Task Decomposition?")

print(response)
```

3. Write a RAG using Pinecone.

https://python.langchain.com/docs/integrations/vectorstores/pinecone

```
from langchain_community.document_loaders import TextLoader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_pinecone import PineconeVectorStore
from pinecone import Pinecone, PodSpec
import os

os.environ["OPENAI_API_KEY"] = "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"
os.environ["PINECONE_API_KEY"] = "eb0f1c59-78f7-4e47-9017-87941c145474"
os.environ["PINECONE_ENV"] = "gcp-starter"

def loadText():
    loader = TextLoader("Conocimiento.txt")
    documents = loader.load()
    #text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)

    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size = 1000,
        chunk_overlap  = 200,
        length_function = len,
        is_separator_regex = False,
    )


    docs = text_splitter.split_documents(documents)

    embeddings = OpenAIEmbeddings()

    import pinecone


    index_name = "langchain-demo"
    pc = Pinecone(api_key='eb0f1c59-78f7-4e47-9017-87941c145474')

    print(pc.list_indexes())

    # First, check if our index already exists. If it doesn't, we create it
    if len(pc.list_indexes())==0:
        # we create a new index
        #pc.create_index(name=index_name, metric="cosine", dimension=1536)
        pc.create_index(
            name=index_name,
            dimension=1536,
            metric="cosine",
            spec=PodSpec(
                environment=os.getenv("PINECONE_ENV"),
                pod_type="p1.x1",
                pods=1
            )
        )

    # The OpenAI embedding model `text-embedding-ada-002 uses 1536 dimensions`
    docsearch = PineconeVectorStore.from_documents(docs, embeddings, index_name=index_name)

def search():
    embeddings = OpenAIEmbeddings()

    index_name = "langchain-demo"
    # if you already have an index, you can load it like this
    docsearch = PineconeVectorStore.from_existing_index(index_name, embeddings)

    query = "What is a distributed pointcut"
    docs = docsearch.similarity_search(query)

    print(docs[0].page_content)

#loadText()
search()
```

4. (Optional) write a RAG for code.

https://python.langchain.com/docs/use_cases/question_answering/code_understanding

Deliverables:

1. Repository on Github with code and a README describing the experiments and showing the results.

OPENAI KEY: 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

requirements.txt:

```
jupyterlab
openai
tiktoken
langchain
openai
chromadb
langchainhub
bs4
pinecone-client
langchain-pinecone
langchain-community
```

Editar entrega | Borrar entrega

ESTADO DE LA ENTREGA

Estado de la entrega	Enviado para calificar
Estado de la calificación	Sin calificar
Tiempo restante	La tarea fue enviada 1 día 23 horas antes
Última modificación	jueves, 18 de abril de 2024, 00:35
Texto en línea	https://github.com/ELS4NTA/AREP-LAB-09
Comentarios de la entrega	Comentarios (0)

ENLACES INSTITUCIONALES

Biblioteca

Investigación e innovación

Enlace - Académico

ENLACES DE INTERÉS

Ministerio de Educación Nacional

Colombia Aprende

Red Latinoamericana de Portales Educativos

Red Universitarias Metropolitana de Bogotá

CONTACTE CON NOSOTROS

AK.45 No.205-59 (Autopista Norte).

Teléfono: +57(1) 668 3600

Email: contactocc@escuelaing.edu.co