assignment6

September 9, 2024

1 Getting started with Generative-AI

Submitted By: Jenish Twayana

Submitted Date: 9th September, 2024

1.1 Exercise 6: RAG Based Exercise

Objective:

Objective 1:
Using the Langchain document loader and splitter functionality. Load these following documents in your memory.
 Freedom of Remote Working (YouTube Video)
 Leapfrogs Code of Conduct (Confluence)
 Cross-cultural collaboration (Look into the Byte Side)

Objective 2:
Chunk/split these data using any of the document splitter. Which splitting would be best for each case and why?

Objective 3:
Create embeddings out of the previously created chunks of data.

Objective 4:
Store these chunks of data into the Chroma vector database.

Objective 5:
Ask relevant questions using similarity search and retrieve the relevant chunks from the vector store.

Objective 6:
Invoke the model with relevant chunks and query from the user to get a summarized answer to your questions. Use Prompt Template.

[26]:
```
import os
```

1.1.1 Set up the environment variables for LangChain Tracing

It logs the tracing data in the LangSmith web interface.

[27]:
```
os.environ["LANGCHAIN_TRACING_V2"]="true" # enables the tracing
os.environ["LANGCHAIN_ENDPOINT"]="https://api.smith.langchain.com"
os.environ["LANGCHAIN_API_KEY"]=os.getenv("LANGCHAIN_API_KEY")
os.environ["LANGCHAIN_PROJECT"]="assignment-6" #project name in the LangSmith␣
 ↪platform
os.environ['USER_AGENT'] = 'myagent'
```

1.1.2 Import the necessary packages and libraries

[28]:
```
from langchain_openai import ChatOpenAI
import bs4
from langchain_community.document_loaders import WebBaseLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_chroma import Chroma
from langchain_openai import OpenAIEmbeddings
from langchain_core.output_parsers import StrOutputParser
from langchain_core.runnables import RunnablePassthrough
from langchain.chains import create_retrieval_chain
from langchain.chains.combine_documents import create_stuff_documents_chain
from langchain_core.prompts import ChatPromptTemplate
from langchain_community.document_loaders import YoutubeLoader
from langchain.document_loaders import PyPDFLoader
```

1.1.3 Initialise the Chat Model

[29]:
```
llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0)
```

1.1.4 Document Loader for Web Pages

[30]:
```
bs4_strainer = bs4.SoupStrainer()
```

[31]:
```
docs = []
```

[32]:
```
loader = WebBaseLoader(
    web_paths=("https://www.linkedin.com/pulse/
 ↪cross-cultural-collaboration-modern-workplace-lftechnology-vgp7f?
 ↪trk=news-guest_share-article",),
    bs_kwargs={"parse_only": bs4_strainer},
)
docs.append(loader.load())
```

1.1.5 Document Loader for PDF files

[33]:
```
# load documents
file = 'code_of_conduct_leapfrog_confluence.pdf'
loader = PyPDFLoader(file)
docs.append(loader.load())
```

1.1.6 Document Loader for Youtube Videos

[34]:
```
loader = YoutubeLoader.from_youtube_url(
    "https://youtu.be/noU9iUMIbq0?list=PL1VW8Wpejk2y9zhwprLdxVS79n36dWc39",
    add_video_info=True,
    language=["en", "id"],
    translation="en",
)
docs.append(loader.load())
```

1.1.7 Document Splitter

[35]:
```
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000, chunk_overlap=200, add_start_index=True
)
```

[36]:
```
all_splits = []
for doc in docs:
    all_splits += text_splitter.split_documents(doc)

len(all_splits)
```

[36]:
```
64
```

[37]:
```
len(all_splits[0].page_content)
all_splits[0].metadata
```

[37]:
```
{'source': 'https://www.linkedin.com/pulse/cross-cultural-collaboration-modern-
workplace-lftechnology-vgp7f?trk=news-guest_share-article',
 'title': 'Cross-cultural collaboration in the modern workplace',
 'description': 'Written by: Kritika Rajbhandari, Account Manager There is no
stopping the world even during a pandemic. Advancements in technology and easy
internet access have enabled remote working more than ever.',
 'language': 'en',
 'start_index': 17}
```

1.1.8 Store the splitted documents into the Vector Store

[38]:
```
vectorstore = Chroma.from_documents(documents=all_splits,␣
↪embedding=OpenAIEmbeddings())
```

1.1.9 Retriver to retrieve the document chunks from the vector store

[39]:
```
retriever = vectorstore.as_retriever(search_type="similarity",␣
↪search_kwargs={"k": 4})
```

1.1.10 System prompt for RAG

[40]:
```
system_prompt = (
    "You are an assistant for question-answering tasks. "
    "Use the following pieces of retrieved context to answer "
    "the question."
    "You can only answer questions about the context."
    "The answer must be referenced from the context only."
    "If you don't know the answer or the answer cannot be found based"
    "on the context then, say that you don't know."
    "Use three sentences maximum and keep the "
    "answer concise."
    "\n\n"
    "{context}"
)

prompt = ChatPromptTemplate.from_messages(
    [
        ("system", system_prompt),
        ("human", "{input}"),
    ]
)
```

1.1.11 Function to format the retrieved docs

[41]:
```
def format_docs(docs):
    return "\n\n".join(doc.page_content for doc in docs)
```

[44]:
```
rag_chain = (
    {"context": retriever | format_docs, "input": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)

for chunk in rag_chain.stream('Who is Kritika?'):
    print(chunk, end="", flush=True)
```

```
Kritika is a person mentioned in the context as someone to reach out to via
LinkedIn for further discussion or thoughts on working in cross-cultural teams.
```

1.1.12 Built In Chain for the implementation of above LCEL

[48]:
```python
#built in chain for the implementation of above LCEL
question_answer_chain = create_stuff_documents_chain(llm, prompt)
rag_chain = create_retrieval_chain(retriever, question_answer_chain)

response = rag_chain.invoke({"input": "What is the scope of code of conducts at␣
 ↪leapfrog?"})
print(response["answer"])
```

```
The scope of the code of conduct at Leapfrog covers the workplace conduct of
full-time, probationary, and fixed-term employees, as well as any third party
working on behalf of Leapfrog Technology Nepal. It aims to outline the values
desired to be fostered within employees, serve as a benchmark for measuring
conduct and performance, and guide day-to-day decisions of team members as
formal representatives of the company. The document emphasizes that everything
done, including daily conduct, is centered around the company's mission, vision,
and values.
```