exercise6

September 9, 2024

1 Import Libraries and LLM setup

[37]:
```
import bs4
from langchain_openai import ChatOpenAI
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.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_community.document_loaders import PyPDFLoader
```

[38]:
```
llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0)
```

2 Define the text splitter for chunking

[39]:
```
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=600, chunk_overlap=100, add_start_index=True
)
```

3 Load and Split the Code of Conduct

[40]:
```
file_path = (
    "../code_of_conduct.pdf"
)
loader = PyPDFLoader(file_path)
code_of_conduct_pages = loader.load_and_split(text_splitter)
len(code_of_conduct_pages)
```

[40]:
```
39
```

4 Load and Split Freedom Of Remote Work

[41]:
```
loader = YoutubeLoader.from_youtube_url(
    "https://youtu.be/noU9iUMIbq0?list=PL1VW8Wpejk2y9zhwprLdxVS79n36dWc39",
    add_video_info=True,
    language=["en", "id"],
    translation="en",
)
freedom_of_remote_work_pages = loader.load_and_split(text_splitter)
len(freedom_of_remote_work_pages)
```

[41]: 9

5 Load and Split Cross Cultural Collaboration

[42]:
```
# strainer only keeps the content
bs4_strainer = bs4.SoupStrainer()
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},
)
look_at_the_byte_side_pages = loader.load_and_split(text_splitter)
len(look_at_the_byte_side_pages)
```

[42]: 46

6 Concat all the docs

[43]:
```
all_documents = look_at_the_byte_side_pages + freedom_of_remote_work_pages +␣
 ↪code_of_conduct_pages
```

7 Store in a vectore DB

[44]:
```
vectorstore = Chroma.from_documents(documents=all_documents,␣
 ↪embedding=OpenAIEmbeddings())
```

8 Define Retreiving Algorithm

[45]:
```
retriever = vectorstore.as_retriever(search_type="similarity",␣
 ↪search_kwargs={"k": 5})
```

9 System Prompt for the RAG

[46]:
```
system_prompt = (
    """
    You are an assistant specialized in answering questions based solely on the␣
 ↪provided context.
    Use only the given context to form your responses.
    If the information is not provided or unclear, state explicitly that you␣
 ↪don't know.
    Keep your answers brief, limited to a maximum of three sentences.
    {context}
    """
)

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

10 Define RAG Chain

[48]:
```
question_answer_chain = create_stuff_documents_chain(llm, prompt)
rag_chain = create_retrieval_chain(retriever, question_answer_chain)
```

11 Test Output

[49]:
```
response = rag_chain.invoke({"input": "What is Establishing clear communication␣
 ↪and setting expectations in Meeting Halfway"})
print(response["answer"])
```

```
Establishing clear communication and setting expectations in Meeting Halfway
involves ensuring that all team members understand each other's backgrounds,
preferences, and expectations. This can be achieved by creating a space where
team members can voice out their preferences and expectations, promoting
inclusion and equity within the team.
```