기본 RAG 시스템 구축

202000826 김연범

1. 목표

대한민국 헌법 PDF 파일을 기반으로 Q&A가 가능한, 가장 기초적인 챗봇 구축

2. RAG 시스템 구조

a. 모델 결정 : OpenAI의 GPT를 API로 활용

a-1 근거 : Closed source LLM 활용시 품질이 안정적이고, 빠르게 구축 가능

a-2 근거 : 가장 기초적인 형태

b. Document Loader 선정 : PDF Loader 선정

b-1 근거 : PDF 파일 로드

c. 문서를 분할할 Text Splitter 선정 : RecursiveCharacterTextSplitter

c-1 근거 : 다른 요소에 비해 고려할 요소 적음

c-2 근거 : chunk_overlap 파라미터 설정만으로 충분한 성능 보장

d. 임베딩 모델 선정 : OpenAI의 text-embedding-3-small 모델

d-1 근거 : 한글 데이터셋으로 충분히 사전 학습 됨

d-2 근거 : 맥락을 충분히 담을만한 긴 길이의 임베딩이 가능(max_token=8191)

e. 벡터 스토어 결정 : Chroma

e-1 근거 : 순수 벡터 데이터베이스로 RAG를 구축하는 것이 간단, 쉬움

f. Retriever 결정 : 벡터 스토어 기반 검색기 활용

f-1 근거 : Chroma 위에 벡터 스토어 기반 검색기 활용 가능

g. LCEL 기반 Chain 하나로 묶어 구성

3. 세부사항

a. 헌법 PDF 파일 로드

a-1) PyPDFLoader를 통해 헌법 PDF 파일을 Document 객체로 불러옴

a-2) load_and_split() 함수를 통해 페이지 단위로 잘라 pages에 저장

b. 청크 분할

b-1) pages에 저장된 Document 객체 텍스트 길이가 1000자 이사인 경우, 이를 1000자 이

하의 청크로 분할하기 위해 RecursiveCharacterTextSplitter 선언
b-2) chunk_size=1000, chunk_overlap=100 설정(청크 간 앞 뒤로 100자씩 겹치도록)
b-3) docs 에 저장

c. 임베딩 벡터 변환 및 Retriever 생성
c-1) docs에 저장된 청크들을 임베딩 벡터 형태로 변환
c-2) as_retriever() 함수를 통해 Retriever 생성

d. RAG에 활용할 LLM 호출
d-1) GPT-4o-mini 모델 선언
d-2) 프롬프트 템플릿을 Langchain Hub에서 가져옴 -> RAG 전용 프롬프트로 가장 많이 사용되는 rlm/rag-prompt 호출

e. Chain 구축
e-1) LCEL을 활용해 Chain 구축 -> RAG 구성 요소들을 순서에 맞게 '|' 로 연결
e-2) get_graph().print_ascii() 함수를 통해 Chain 시각화

4. 코드 및 실행결과

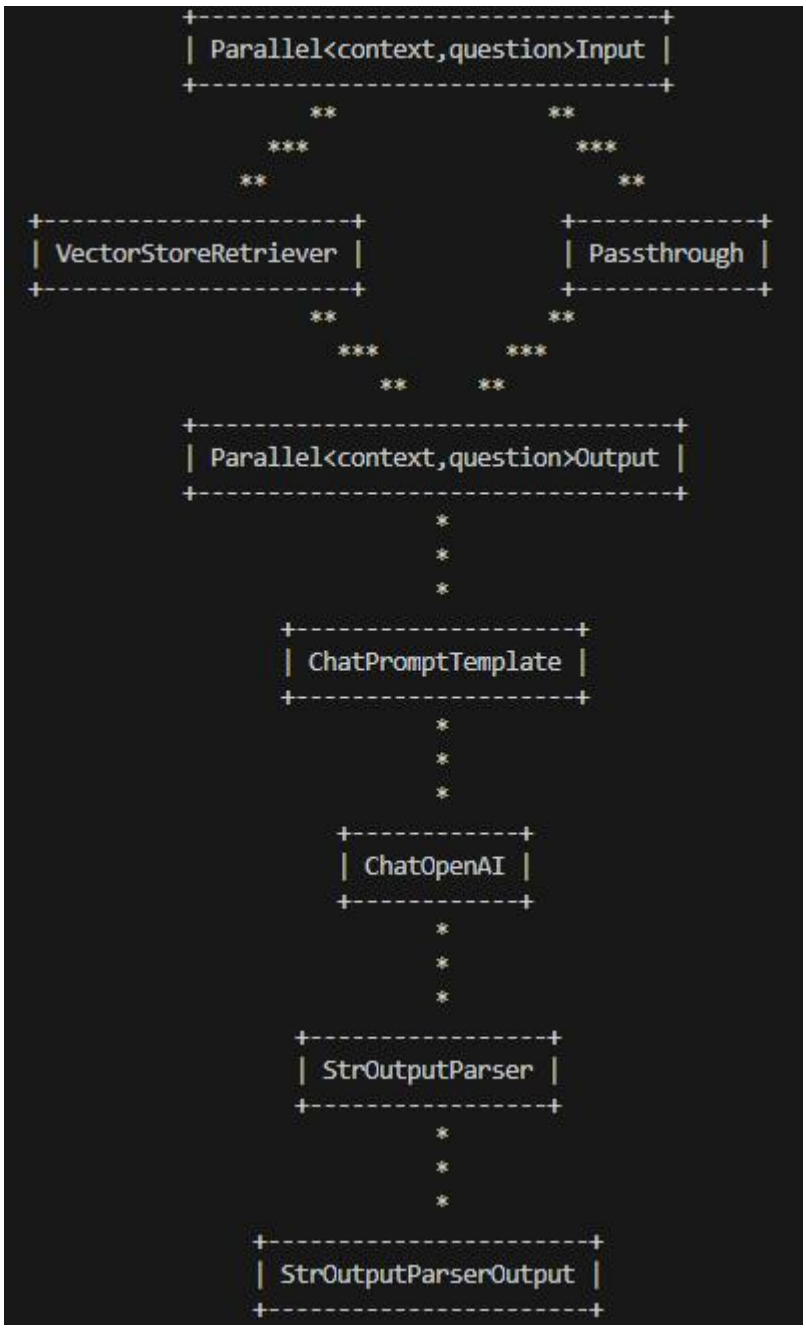
a. 코드

```python
from langchain import hub
from langchain.document_loaders import PyPDFLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_openai import OpenAIEmbeddings
from langchain_chroma import Chroma
from langchain_openai import ChatOpenAI
from langchain_core.runnables import RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser
import os

# OpenAI 의 API 키 등록
os.environ['OPENAI_API_KEY'] = "sk-proj-950hCG9R3OttgJ5609m7AAN_Le2ij20LHmv-Doh2cU49Al5zKO6xrsaBGA-HtBt2jmlbjxUapbT

# 헌법 PDF 파일 로드
loader = PyPDFLoader(r"헌법.pdf")
pages = loader.load_and_split()

# PDF 파일을 1000자 청크로 분할
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000, chunk_overlap=100
)
docs = text_splitter.split_documents(pages)

# ChromaDB에 청크들을 벡터 임베딩으로 저장(OpenAI 임베딩 모델 활용)
vectorstore = Chroma.from_documents(docs, OpenAIEmbeddings(
    model="text-embedding-3-small"
), persist_directory="RAG_EX/Chroma_DB")
retriever = vectorstore.as_retriever()

# GPT-4o-mini 모델 선언
llm = ChatOpenAI(model="gpt-4o-mini")

# Langchain Hub에서 RAG 프롬프트 호출
prompt = hub.pull('rlm/rag-prompt')

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

# 테스트
answer = rag_chain.invoke("국회의원의 의무는 뭐야?")
print(answer)

print(rag_chain.get_graph().print_ascii())
```

b. 실행결과

```
국회의원의 의무는 청렴을 유지하고 국가이익을 우선하여 직무를 수행하는 것입니다. 또한, 국회의원은 그
지위를 남용하여 재산상의 권리를 취득하거나 타인을 위해 도움을 줄 수 없습니다. 이러한 의무는 국회의원
으로서의 책임과 권리를 보장하기 위해 필요합니다.
```

c. Chain 시각화


Parallel<context,question>Input
VectorStoreRetriever
Passthrough
Parallel<context,question>Output
ChatPromptTemplate
ChatOpenAI
StrOutputParser
StrOutputParserOutput