Streamlit으로 RAG 헌법 Q&A 챗봇 구현하기

202000826 김연범

1. 목표

파이썬에서 제공하는 Streamlit 프레임워크를 활용해 RAG 헌법 Q&A 챗봇을 구현해보자.

2. Streamlit 개념

스트림릿(Streamlit)은 데이터 과학자, AI/ML 엔지니어가 간단한 코드만으로 인터랙티브한 데이터 앱을 제공할 수 있도록 돕는 파이썬 프레임워크로, 최근 LLM을 활용한 챗봇 수요가 크게 늘면서 챗봇 기능이 탑재되었다.

※ Streamlit은 사용자가 상호작용을 시도할 때마다 전체 스크립트를 다시 실행하기 때문에, Streamlit에서 사용자와의 상호작용 간 데이터 유지를 위한 상태 관리 객체 session_state을 꼭 선언해야 한다.

3. 세부 사항

※ @st.cache_resource : Streamlit으로 구현된 웹앱이 한번 구동할 때, 생성된 데이터를 캐싱하는 기능. 이는 Streamlit의 기본 특성에 기인해 동일 함수를 재실행 할 때, 캐싱된 데이터를 그대로 가져와 실행 시간을 크게 줄여주는 역할을 함.

a) 필요 라이브러리 호출 및 OpenAI API key 설정

b) PDF 문서 로드 및 벡터화 함수

b-1) load_and_split_pdf(file_path) : file_path로 지정된 파일 경로상의 PDF 파일을 PyPDFLoader로 로드하고, 이를 페이지별로 분할하여 리턴함.

b-2) create_vector_store(_docs) : 주어진 Document 객체를 Recursive CharacterTextSplitter로 분할 후 이를 Chroma 벡터 DB에 저장함.
※ 임베딩 벡터로 변환할 모델로 OpenAI의 text-embedding-3-small 모델을 사용

b-3) get_vector_store(_docs) : 만약 벡터 DB가 이미 존재하는 경우, 이를 로드함.

b-4) format_docs(docs) : Document 객체들의 page_content를 추출하고 이를 하나로 결합해줌.

c) RAG 체인 구성

c-1) load_and_split() 함수 실행 : 먼저 파일을 불러올 경로를 설정하고, 이 경로를 매개변수로 지정해 PDF 파일을 로드함

c-2) get_vector_store() 함수 실행 : 만들어진 Document 객체를 매개변수로 넣어 벡터 임베딩으로 변환 후, Chroma DB에 저장.
이렇게 만들어진 벡터 DB를 기반으로 Retriever를 선언하기 위해 as_retriever() 함수를 실행.

c-3) RAG 프롬프트 선언 : RAG 프롬프트를 선언하고, 이것을 ChatPromptTemplate의 시스템 프롬프트로 지정.
HumanMessage에는 {input}을 지정하여 입력되는 값을 Human Message에 매핑.

c-4) LLM 지정 및 RAG 체인 결합 : LLM으로 OpenAI의 GPT-4o를 활용, rag_chain으로 앞서 저장한 요소들을 하나로 결합.

d) Streamlit UI 구성

d-1) 제목은 “헌법 Q&A 챗봇”으로 설정

d-2) 처음 Streamlit 실행 시, 챗봇이 “헌법에 대해 무엇이든 물어보세요!”라는 안냇말 생성

d-3) chat_input에 입력되는 사용자 프롬프트를 prompt_message로 받고, chat_message를 활용해 채팅 버블을 구현

d-4) message session_state에 사용자의 프롬프트를 저장하여 이어지는 사용자 질문과 AI 응답들이 모두 쌓이도록 함.

4. 코드 및 실행화면

a) 코드

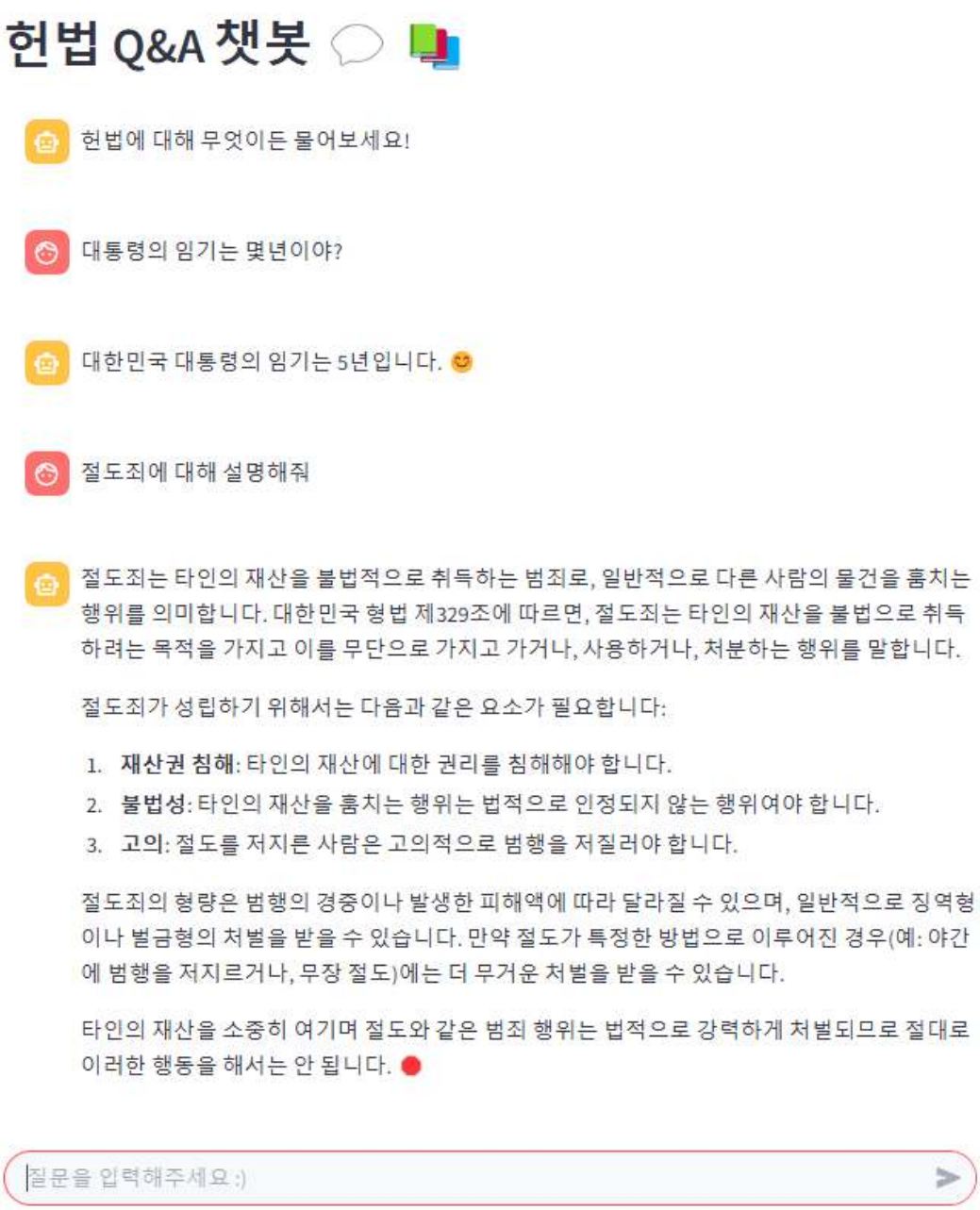
a-1) 오픈 AI API 키 설정 및 PDF 문서 로드, 벡터화 함수 선언

```python
import os
import streamlit as st
from langchain.document_loaders import PyPDFLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_openai import OpenAIEmbeddings, ChatOpenAI
from langchain_chroma import Chroma
from langchain.prompts import ChatPromptTemplate
from langchain_core.runnables import RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser

#오픈AI API 키 설정
os.environ["OPENAI_API_KEY"] = "sk-proj-950hCG9R3OttgJ5609m7AAN_Le2ij20LHmv-Doh2cU49Al

@st.cache_resource
def load_and_split_pdf(file_path):
    loader = PyPDFLoader(file_path)
    return loader.load_and_split()

#텍스트 청크들을 Chroma 안에 임베딩 벡터로 저장
@st.cache_resource
def create_vector_store(_docs):
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
    split_docs = text_splitter.split_documents(_docs)
    persist_directory = "./chroma_db"
    vectorstore = Chroma.from_documents(
        split_docs,
        OpenAIEmbeddings(model='text-embedding-3-small'),
        persist_directory=persist_directory
    )
    return vectorstore

#만약 기존에 저장해둔 ChromaDB가 있는 경우, 이를 로드
@st.cache_resource
def get_vectorstore(_docs):
    persist_directory = "./chroma_db"
    if os.path.exists(persist_directory):
        return Chroma(
            persist_directory=persist_directory,
            embedding_function=OpenAIEmbeddings(model='text-embedding-3-small')
        )
    else:
        return create_vector_store(_docs)

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

a-2) RAG 체인 구성 및 UI

```python
# Initialize the LangChain components
@st.cache_resource
def chaining():
    file_path = r"헌법.pdf"
    pages = load_and_split_pdf(file_path)
    vectorstore = get_vectorstore(pages)
    retriever = vectorstore.as_retriever()

    # Define the answer question prompt
    qa_system_prompt = """
    You are an assistant for question-answering tasks. \
    Use the following pieces of retrieved context to answer the question. \
    If you don't know the answer, just say that you don't know. \
    Keep the answer perfect. please use imogi with the answer.
    Please answer in Korean and use respectful language.\
    {context}
    """

    qa_prompt = ChatPromptTemplate.from_messages(
        [
            ("system", qa_system_prompt),
            ("human", "{input}"),
        ]
    )

    llm = ChatOpenAI(model="gpt-4o-mini")
    rag_chain = (
        {"context": retriever | format_docs, "input": RunnablePassthrough()}
        | qa_prompt
        | llm
        | StrOutputParser()
    )
    return rag_chain

# Streamlit UI
st.header("헌법 Q&A 챗봇 💬 📚")
rag_chain = chaining()


if "messages" not in st.session_state:
    st.session_state["messages"] = [{"role": "assistant", "content": "헌법에 대해 무엇이든 물어보세요!"}]

for msg in st.session_state.messages:
    st.chat_message(msg['role']).write(msg['content'])

if prompt_message := st.chat_input("질문을 입력해주세요 :)"):
    st.chat_message("human").write(prompt_message)
    st.session_state.messages.append({"role": "user", "content": prompt_message})
    with st.chat_message("ai"):
        with st.spinner("Thinking..."):
            response = rag_chain.invoke(prompt_message)
            st.session_state.messages.append({"role": "assistant", "content": response})
            st.write(response)
```

b) 실행화면


헌법 Q&A 챗봇
헌법에 대해 무엇이든 물어보세요!
대통령의 임기는 몇년이야?
대한민국 대통령의 임기는 5년입니다.
절도죄에 대해 설명해줘
절도죄는 타인의 재산을 불법적으로 취득하는 범죄로, 일반적으로 다른 사람의 물건을 훔치는 행위를 의미합니다. 대한민국 형법 제329조에 따르면, 절도죄는 타인의 재산을 불법으로 취득하려는 목적을 가지고 이를 무단으로 가지고 가거나, 사용하거나, 처분하는 행위를 말합니다.
절도죄가 성립하기 위해서는 다음과 같은 요소가 필요합니다:
1. 재산권 침해: 타인의 재산에 대한 권리를 침해해야 합니다.
2. 불법성: 타인의 재산을 훔치는 행위는 법적으로 인정되지 않는 행위여야 합니다.
3. 고의: 절도를 저지른 사람은 고의적으로 범행을 저질러야 합니다.
절도죄의 형량은 범행의 경중이나 발생한 피해액에 따라 달라질 수 있으며, 일반적으로 징역형이나 벌금형의 처벌을 받을 수 있습니다. 만약 절도가 특정한 방법으로 이루어진 경우(예: 야간에 범행을 저지르거나, 무장 절도)에는 더 무거운 처벌을 받을 수 있습니다.
타인의 재산을 소중히 여기며 절도와 같은 범죄 행위는 법적으로 강력하게 처벌되므로 절대로 이러한 행동을 해서는 안 됩니다.
질문을 입력해주세요 :)