Memory 기능이 탑재된 RAG 시스템 구축

202000826 김연범

1. 목표

대한민국 헌법 PDF 파일을 기반으로 Q&A가 가능한, Memory 기능이 탑재된 챗봇 구현

2. Memory 기능이 탑재된 RAG

a. 사용자의 질문이 들어오면 그동안 적재한 채팅 히스토리와 통합한 후, 기존 RAG와 동일한 과정을 거쳐 답변을 생성하는 RAG 시스템

b. Retriever를 통해 통합된 질문과 유사한 문장 검색, 이를 Q&A 프롬프트 내에 맥락으로 주입해 LLM에게 답변할 수 있도록 힌트를 제공

c. LLM은 채팅 히스토리와 사용자의 질문, 그리고 답변을 위한 힌트 컨텍스트를 모두 갖춰 보다 정확한 답변을 할 수 있음

3. 세부사항

a. Retriever 생성까지 기존 모델(일반 RAG 모델)과 동일

b. 채팅 히스토리-사용자 질문 통합 작업 지시 프롬프트 작성

b-1) contextualize_q_system_prompt 변수에 채팅 히스토리-사용자 질문 통합 작업 지시 프롬프트 저장

b-2) ChatPromptTemplate를 통해 시스템 프롬프트에 위의 프롬프트 주입

c. 시스템 프롬프트 작성

c-1) RAG에서 컨텍스트와 사용자 질문을 함께 다룰 수 있도록 하는 시스템 프롬프트를 qa_system_prompt에 저장

c-2) 역시 ChatPromptTemplate를 통해 시스템 프롬프트에 위의 프롬프트 주입

d. 채팅 세션별 기록 자동 저장 RAG 체인 구축

d-1) 채팅의 세션별 서로 다른 채팅 히스토리를 store 딕셔너리에 저장

d-2) 사용자 정의 get_session_histroy 함수를 통해 주어진 session_id에 매칭되는 채팅 히스토리를 불러와 RAG 체인에 결합 -> RunnableWithMessageHistory 모듈을 통해 RAG 체인과 get_session_history 함수, 입력 값 키, 채팅 히스토리, 결과 값 키를 모두 결합해 conversational_rag_chain을 완성

4. 코드 및 실행결과

a. 코드 (전 RAG와 동일 부분 생략)

a-1) 코드 1

```python
from langchain.chains.combine_documents import create_stuff_documents_chain
from langchain.chains.history_aware_retriever import create_history_aware_retriever
from langchain.chains.retrieval import create_retrieval_chain
from langchain.prompts import ChatPromptTemplate, MessagesPlaceholder

# Define the contextualize question prompt
contextualize_q_system_prompt = """Given a chat history and the latest user
question \
which might reference context in the chat histroy, formulate a standalone question \
which can be understood without the chat history. Do NOT answer the question, \
just reformulate it if needed and otherwise return it as is."""

contextualize_q_prompt = ChatPromptTemplate.from_messages(
    [
        ("system", contextualize_q_system_prompt),
        MessagesPlaceholder("chat_history"),
        ("human", "{input}"),
    ]
)

history_aware_retriever = create_history_aware_retriever(llm, retriever, contextualize_q_prompt)

from langchain.chains.combine_documents import create_stuff_documents_chain

# 시스템 프롬프트 : RAG에서 컨텍스트와 사용자 질문을 함께 다룰 수 있도록 만들어줌
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. \
Use three sentences maximum and keep the answer concise. \

{context}"""

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

question_answer_chain = create_stuff_documents_chain(llm, qa_prompt)

rag_chain = create_retrieval_chain(history_aware_retriever, question_answer_chain)
```

a-2) 코드 2

```python
from langchain_community.chat_message_histories import ChatMessageHistory
from langchain_core.chat_history import BaseChatMessageHistory
from langchain_core.runnables.history import RunnableWithMessageHistory

# 채팅 세션별 기록 저장을 위한 Dictionary 선언
store = {}

# 주어진 session_id 값에 매칭되는 채팅 히스토리를 가져오는 함수 선언
def get_session_history(session_id: str) -> BaseChatMessageHistory:
    if session_id not in store:
        store[session_id] = ChatMessageHistory()
    return store[session_id]


# RunnableWithMessageHistory 모듈로 rag_chain에 채팅 기록 세션별로 자동 저장 기능 추가
conversational_rag_chain = RunnableWithMessageHistory(
    rag_chain,
    get_session_history=get_session_history,
    input_messages_key="input",
    history_messages_key="chat_history",
    output_messages_key="answer"
)

result = conversational_rag_chain.invoke(
    {"input": "대통령의 임기는 몇 년이야?"},
    config={
        "configurable": {"session_id" : "1234"}
    },
)['answer']

result2 = conversational_rag_chain.invoke(
    {"input": "국회의원은?"},
    config={
        "configurable": {"session_id": "1234"}
    }
)['answer']

print(result)
print("\n\n")
print(result2)
```

b. 실행결과