



LangChain Chatbot

2024.05.30(목)

김동억

목차

- 개요
- 구성 요소
- 기본 챗봇 구축
- 대화 기록 관리
- Prompt templates
- 대화 내역 관리
- Streaming
- 다음 단계



개요

- LLM에 기반한 챗봇 설계 및 구현
 - LLM만을 사용
 - 다루지 않는 내용(Conversational RAG, Agents)

구성 요소

챗 모델(Chat Models)

챗봇 인터페이스는 메시지를 중심으로 작동하며, 텍스트 LLM보다는 챗 모델에 적합합니다.

프롬프트 템플릿(Prompt Templates)

기본 메시지, 사용자 입력, 대화 기록 및 추가적으로 검색된 컨텍스트를 조합하여 **프롬프트를 구성하는 과정을 단순화**합니다.

대화 기록(Chat History)

챗봇이 **과거 상호작용**을 "기억"하고 후속 질문에 답변할 때 이를 고려할 수 있도록 합니다.

LangSmith를 사용한 디버깅 및 추적

애플리케이션을 디버깅하고 추적하는 방법을 다룹니다.



기본 챗봇 구축

```
from langchain_core.messages import HumanMessage  
from langchain_openai import ChatOpenAI
```

```
model = ChatOpenAI(model="gpt-3.5-turbo")
```

```
model.invoke([HumanMessage(content= " Hi! I ' m Bob " )])  
model.invoke([HumanMessage(content="What's my name?")])
```



기본 챗봇 구축

```
from langchain_core.messages import AIMessage
```

```
model.invoke(  
    [  
        HumanMessage(content="Hi! I'm Bob"),  
        AIMessage(content="Hello Bob! How can I assist you today?"),  
        HumanMessage(content="What's my name?"),  
    ]  
)
```



대화 기록 관리

```
from langchain_community.chat_message_histories import ChatMessageHistory
```

```
from langchain_core.chat_history import BaseChatMessageHistory
```

```
from langchain_core.runnables.history import RunnableWithMessageHistory
```

```
store = {}
```

```
def get_session_history(session_id: str) -> BaseChatMessageHistory:
```

```
    if session_id not in store:
```

```
        store[session_id] = ChatMessageHistory()
```

```
    return store[session_id]
```

```
with_message_history = RunnableWithMessageHistory(model, get_session_history)
```



대화 기록 관리

```
config = {"configurable": {"session_id": "abc2"}}
```

```
response = with_message_history.invoke(  
    [HumanMessage(content="Hi! I'm Bob")],  
    config=config,  
)
```

```
response = with_message_history.invoke(  
    [HumanMessage(content="What's my name?")],  
    config=config,  
)
```

```
response.content
```




Prompt templates

```
from langchain_core.prompts import ChatPromptTemplate, MessagesPlaceholder
```

```
prompt = ChatPromptTemplate.from_messages(  
    [  
        (  
            "system",  
            "You are a helpful assistant. Answer all questions to the best of your ability.",  
        ),  
        MessagesPlaceholder(variable_name="messages"),  
    ]  
)  
chain = prompt | model
```



Prompt templates

```
response = chain.invoke({"messages": [HumanMessage(content="hi! I'm bob")]}))
```

```
response.content
```



Prompt templates

```
with_message_history = RunnableWithMessageHistory(chain, get_session_history)
config = {"configurable": {"session_id": "abc5"}}
response = with_message_history.invoke(
    [HumanMessage(content="Hi! I'm Jim")],
    config=config,
)
response = with_message_history.invoke(
    [HumanMessage(content="What's my name?")],
    config=config,
)

response.content
```



RunnableWithMessageHistory

- API
 - https://api.python.langchain.com/en/latest/runnables/langchain_core.runnables.history RunnableWithMessageHistory.html
- LangChain Expression Language(LCEL)
 - <https://python.langchain.com/v0.2/docs/concepts/#langchain-expression-language-lcel>
 - Runnable interface(<https://python.langchain.com/v0.2/docs/concepts/#runnable-interface>)



Prompt templates

```
prompt = ChatPromptTemplate.from_messages(  
    [  
        (  
            "system",  
            "You are a helpful assistant. Answer all questions to the best of your ability in {language}.",  
        ),  
        MessagesPlaceholder(variable_name="messages"),  
    ]  
)
```

```
chain = prompt | model
```



Prompt templates

```
response = chain.invoke(  
    {"messages": [HumanMessage(content="hi! I'm bob")], "language": "Spanish"}  
)
```

```
response.content
```



Prompt templates

```
with_message_history = RunnableWithMessageHistory(  
    chain,  
    get_session_history,  
    input_messages_key="messages",  
)  
config = {"configurable": {"session_id": "abc11"}}  
response = with_message_history.invoke(  
    {"messages": [HumanMessage(content="hi! I'm todd")], "language": "Spanish"},  
    config=config,  
)  
  
response.content
```



Prompt templates

```
response = with_message_history.invoke(  
    {"messages": [HumanMessage(content="whats my name?")], "language": "Spanish"},  
    config=config,  
)
```

```
response.content
```




대화 내역 관리

```
from langchain_core.runnables import RunnablePassthrough
```

```
def filter_messages(messages, k=10):  
    return messages[-k:]
```

```
chain = (  
    RunnablePassthrough.assign(messages=lambda x: filter_messages(x["messages"]))  
    | prompt  
    | model  
)
```



대화 내역 관리

```
messages = [  
    HumanMessage(content="hi! I'm bob"),  
    AIMessage(content="hi!"),  
    HumanMessage(content="I like vanilla ice cream"),  
    AIMessage(content="nice"),  
    HumanMessage(content="whats 2 + 2"),  
    AIMessage(content="4"),  
    HumanMessage(content="thanks"),  
    AIMessage(content="no problem!"),  
    HumanMessage(content="having fun?"),  
    AIMessage(content="yes!"),  
]  
  
response = chain.invoke(  
    {  
        "messages": messages + [HumanMessage(content="what's my name?")],  
        "language": "English",  
    },  
)  
  
response = chain.invoke(  
    {  
        "messages": messages + [HumanMessage(content="what's my fav ice cream")],  
        "language": "English",  
    },  
)
```



대화 내역 관리

```
with_message_history = RunnableWithMessageHistory(  
    chain,  
    get_session_history,  
    input_messages_key="messages",  
)
```

```
config = {"configurable": {"session_id": "abc20"}}
```



대화 내역 관리

```
response = with_message_history.invoke(  
    {  
        "messages": messages + [HumanMessage(content="whats my name?")],  
        "language": "English",  
    },  
    config=config,  
)  
response = with_message_history.invoke(  
    {  
        "messages": [HumanMessage(content="whats my favorite ice cream?")],  
        "language": "English",  
    },  
    config=config,  
)
```



Streaming

```
config = {"configurable": {"session_id": "abc15"}}
for r in with_message_history.stream(
    {
        "messages": [HumanMessage(content="hi! I'm todd. tell me a joke")],
        "language": "English",
    },
    config=config,
):
    print(r.content, end="|")
```



Next Steps

- Conversational RAG
 - https://python.langchain.com/v0.2/docs/tutorials/qa_chat_history/
- Agents
 - <https://python.langchain.com/v0.2/docs/tutorials/agents/>
- Streaming
 - <https://python.langchain.com/v0.2/docs/how to/streaming/>
- How to add message history
 - <https://python.langchain.com/v0.2/docs/how to/message history/>



QnA

