Open source LLM으로 RAG 구축하기

202000826 김연범

1. 목표

Open source LLM으로 RAG를 구축해 우리나라 헌법에 관련된 Q&A가 가능한 챗봇을 구현해보자.

2. Open source의 필요성

Closed source를 사용할 때 기업에서 RAG를 구축하는 경우, Q&A를 진행할 때마다 컨텍스트로 기밀 정보가 LLM 제공 업체의 API를 기반으로 전달될 때 유출 가능성이 있기 때문이다. 즉, RAG 시스템에 축적되는 정보가 기밀일수록 RAG를 구축할 때에는 Open source LLM을 활용하는 경우가 많다.

이번 구현 과정에서 Open source LLM은 허깅페이스 플랫폼의 EEVE 모델(GGUFver.)을, Ollama라는 도구를 통해 이를 실행해 보도록 하겠다.

3. EEVE 모델 및 Ollama

a) EEVE 모델은 업스테이지에서 개발한 LLM인 SOLAR에 한글 단어를 추가해 DPO로 강화학습한 언어 모델로, 일반 사용자가 그대로 사용하기에는 너무 높은 GPU 사양이 요구되므로 기반 LLM을 양자화한 GGUF 버전을 활용해야 한다.

※ GGUF란 딥러닝 모델을 경량화하는 방법론 중 하나로, 모델 로드 및 추론 시 필요한 GPU RAM 사양을 크게 낮춰준다. 우리는 이번 구현 과정에서 Q5_K_M.gguf 파일을 사용하겠다.

=> https://huggingface.co/teddylee777/EEVE-Korean-Instruct-10.8B-v1.0-gguf

b) Ollama는 대규모 언어 모델을 로컬 컴퓨터에서 설정하거나 실행할 수 있도록 해주는 고급 AI 도구, 내지는 프레임워크로, 언어 모델을 생성, 실행, 관리할 수 있는 간단한 API와 사전 구축된 모델 라이브러리를 제공한다. Ollama는 Modelfile을 통해 모델 가중치, 구성, 데이터를 패키징하고 GPU 사용 등의 설정을 최적화 할 수 있다.

※ 이번 구현 과정에서는 다음과 같은 Modelfile을 작성했다.

```
FROM EEVE-Korean-Instruct-10.8B-v1.0-Q5_K_M.gguf

TEMPLATE """{{- if.System }}
<s>{{ .System }}</s>
{{- end }}
<s>Human:
{{ .Prompt }}</s>
<s>Assistant:
"""

SYSTEM """A chat between a curious user and an artificial intelligence assistant.
The assistant gives helpful, detailed, and polite answers to the user's
questions."""

PARAMETER temperature 0
PARAMETER num_predict 3000
PARAMETER num_ctx 4096
PARAMETER stop <s>
PARAMETER stop </s>
```

4. 모델 세부사항

a) Ollama를 활용해 EEVE Open source LLM 모델 로드

b) 헌법 PDF 파일을 PyPDFLoader로 로드 및 load_and_split()로 페이지별 분할 수행

c) 청크 사이즈를 500으로 지정한 RecursiveCharacterTextSplitter로 텍스트를 재귀적으로 분할 및 Document 객체 생성

d) HuggingfaceEmbedding을 사용해 임베딩하여 Chroma 벡터 DB에 담고 이것들을 Retriever로 활용
※ RAG 프롬프트 템플릿은 랭체인 Hub에서 가져와 prompt에 저장
※ hub.pull("rlm/rag-prompt") => https://smith.langchain.com/hub/rlm/rag-prompt

e) format_docs 함수를 통해 사용자 질문과 유사한 청크들을 하나의 텍스트로 묶는 작업을 수행

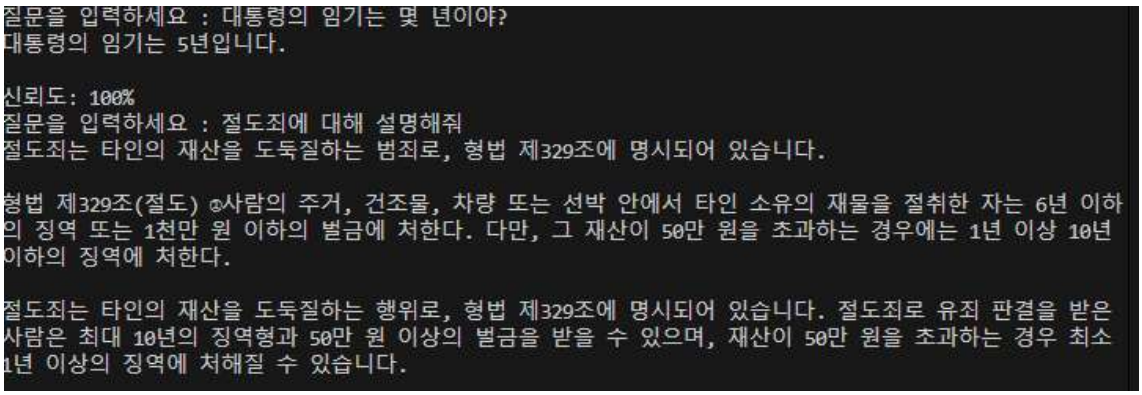
f) 위의 요소들을 rag_chain에 묶음
: RAG의 근거 문서로 주어질 문서들을 Retriever 검색을 통해 찾아냄
-> 하나의 format_docs로 텍스트화 + 사용자의 질문은 RunnablePassthrough()로 그대로 통과
-> Retriever로 검색한 context와 사용자의 질문을 합쳐 prompt에 결합
-> 결합된 prompt가 LLM으로 전달
-> StrOutputParser를 통해 문자열 결과 값을 반환

5. 코드 및 실행 결과

a) 코드

```python
from langchain_ollama import ChatOllama
from langchain.document_loaders import PyPDFLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_chroma import Chroma
from langchain import hub
from langchain_core.runnables import RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser

llm = ChatOllama(model="EEVE-Korean-10.8B:latest")

loader = PyPDFLoader(r"헌법.pdf")
pages = loader.load_and_split()

text_splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=0)
docs = text_splitter.split_documents(pages)

model_name = "jhgan/ko-sbert-nli"
model_kwargs = {'device': 'cpu'}
encode_kwargs = {'normalize_embeddings': True}

embedding = HuggingFaceEmbeddings(
    model_name = model_name,
    model_kwargs = model_kwargs,
    encode_kwargs = encode_kwargs
)

vectorstore = Chroma.from_documents(docs, embedding=embedding, persist_directory="./Chroma_DB")
retriever = vectorstore.as_retriever()

prompt = hub.pull("rlm/rag-prompt")

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

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

while(True):
    ques = str(input('질문을 입력하세요 : '))
    if ques == "break": break

    for chunk in rag_chain.stream(ques):
        print(chunk, end="", flush=True)


    print()
```

b) 실행 결과


질문을 입력하세요 : 대통령의 임기는 몇 년이야?
대통령의 임기는 5년입니다.
신뢰도: 100%
질문을 입력하세요 : 절도죄에 대해 설명해줘
절도죄는 타인의 재산을 도둑질하는 범죄로, 형법 제329조에 명시되어 있습니다.
형법 제329조(절도) ②사람의 주거, 건조물, 차량 또는 선박 안에서 타인 소유의 재물을 절취한 자는 6년 이하의 징역 또는 1천만 원 이하의 벌금에 처한다. 다만, 그 재산이 50만 원을 초과하는 경우에는 1년 이상 10년 이하의 징역에 처한다.
절도죄는 타인의 재산을 도둑질하는 행위로, 형법 제329조에 명시되어 있습니다. 절도죄로 유죄 판결을 받은 사람은 최대 10년의 징역형과 50만 원 이상의 벌금을 받을 수 있으며, 재산이 50만 원을 초과하는 경우 최소 1년 이상의 징역에 처해질 수 있습니다.