메타 데이터를 활용한 LangGraph

강사 장철원

Section 0

목차

❑ chromaDB를 활용한 topic 적용

메타 데이터를 활용한 LangGraph

Section 1. 메타 데이터를 포함한 데이터 활용하기

Section 1-1. 메타 데이터 포함 csv -> chromaDB

Section

메타데이터 포함 csv -> chromaDB

데이터 확인

[1]:
```
import pandas as pd
from langchain_chroma import Chroma
from langchain.schema import Document
from langchain_huggingface import HuggingFaceEmbeddings
```

[2]:
```
# 1. 데이터 로딩
df = pd.read_csv("./data/data_topic.csv", encoding="utf8")
df.head(10)
```

[2]:

	text	topic
0	Etching 공정 전에는 반드시 세정 공정이 완료되어야 합니다.	공정운영
1	PM 장비의 필터는 주 1회 정기적으로 교체해야 합니다.	유지보수
2	웨이퍼 투입 전 챔버 내부의 온도 안정화가 필요합니다.	공정운영
3	불량률이 2%를 초과할 경우 원인 분석 보고서를 제출해야 합니다.	이상대응
4	클린룸 입장 전에는 반드시 정전기 방지복을 착용해야 합니다.	안전규정
5	포토 공정 시 PR 코팅 두께는 1.5μm 이상 유지해야 합니다.	공정운영
6	검사 장비는 매일 초기화 후 기능 점검을 실시합니다.	유지보수
7	주간 생산 계획은 매주 월요일 오전 9시에 확정됩니다.	공정운영
8	X선 검사 장비는 비정상 신호 발생 시 즉시 사용 중지합니다.	유지보수
9	로더 설비의 진공 펌프는 월 1회 이상 윤활 상태를 점검합니다.	유지보수

Section

메타데이터포함csv -> chromaDB

저장

```
texts = df["text"].tolist()
topics = df["topic"].tolist()
```

```
# 2. LangChain 문서 객체 생성 (메타데이터 포함)
docs = []
for i in range(len(texts)):
    text = texts[i]
    topic = topics[i]
    doc = Document(page_content=text, metadata={"topic": topic})
    docs.append(doc)
```

```
docs[0:3]
```

```
[Document(metadata={'topic': '공정운영'}, page_content='Etching 공정 전에는 반드시 세정 공정이 완료되어야 합니다.'),
 Document(metadata={'topic': '유지보수'}, page_content='PM 장비의 필터는 주 1회 정기적으로 교체해야 합니다.'),
 Document(metadata={'topic': '공정운영'}, page_content='웨이퍼 투입 전 챔버 내부의 온도 안정화가 필요합니다.')]
```

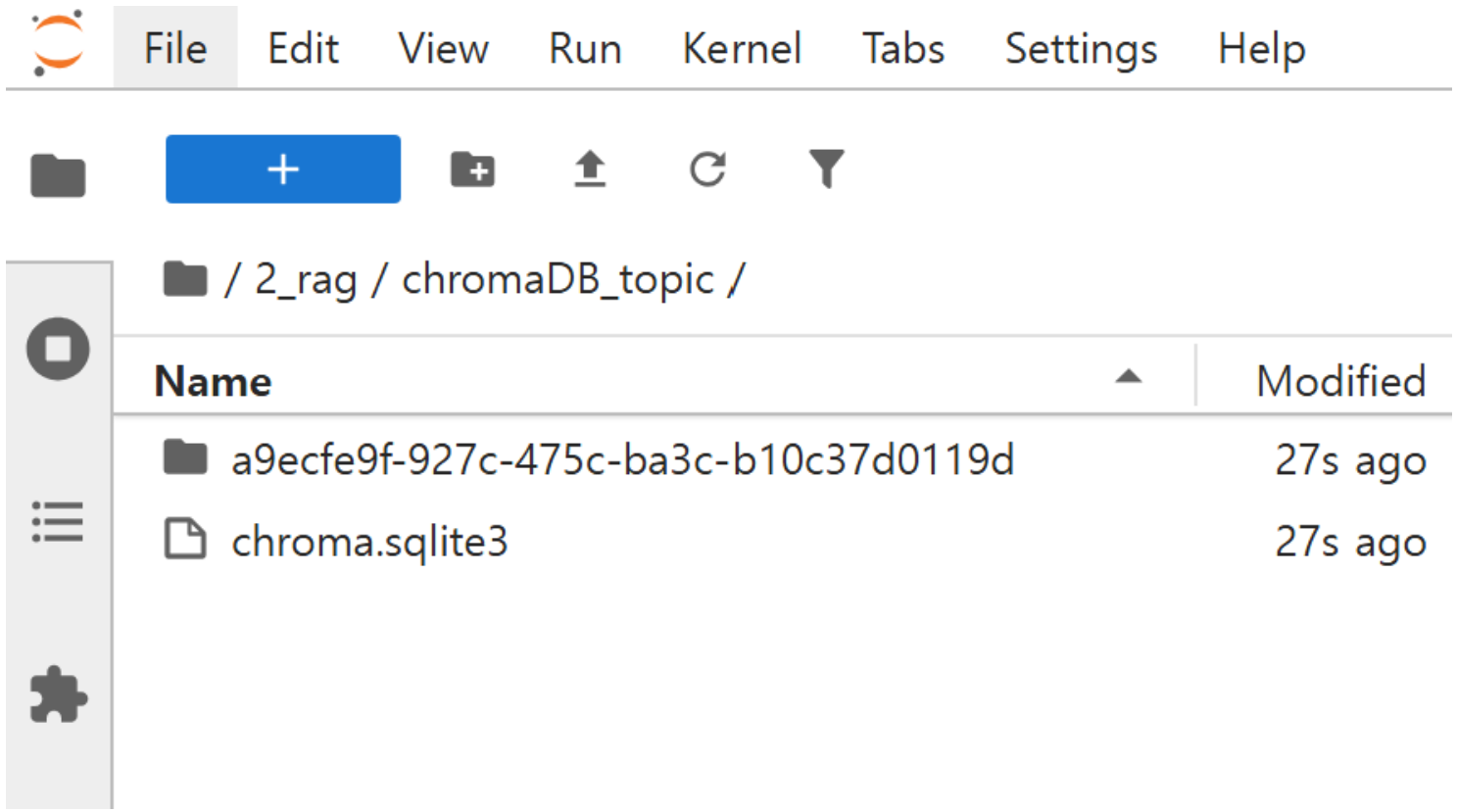
```
# 3. 임베딩 모델 정의
embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")
```

```
# 4. ChromaDB에 저장
vectorstore = Chroma.from_documents(
    documents=docs,
    embedding=embedding_model,
    persist_directory="./chromaDB_topic"
)
```

Section

메타데이터 포함 csv -> chromaDB

저장 확인



메타 데이터를 활용한 LangGraph

Section 1. 메타 데이터를 포함한 데이터 활용하기

Section 1-1. 메타 데이터 포함 chromaDB -> LangGraph

Section

메타데이터 포함 chromaDB -> LangGraph

라이브러리 & 임베딩 모델

[1]:
```python
import uuid
from langchain_chroma import Chroma
from langgraph.graph import StateGraph, END
from langchain_huggingface import HuggingFaceEmbeddings, HuggingFacePipeline
from transformers import AutoTokenizer, AutoModelForQuestionAnswering, pipeline
from langchain_core.chat_history import InMemoryChatMessageHistory
from langchain_core.messages import get_buffer_string
from langchain_core.runnables import RunnableConfig
from langchain_core.chat_history import BaseChatMessageHistory
```

[2]:
```python
# 1. 임베딩 모델
embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")
```

Section

메타데이터포함chromaDB -> LangGraph

QA 모델 & 파이프라인

[3]:
```
# 2. QA 모델 & 파이프라인
model_id = "monologg/koelectra-base-v3-finetuned-korquad"
qa_tokenizer = AutoTokenizer.from_pretrained(model_id)
qa_model = AutoModelForQuestionAnswering.from_pretrained(model_id)

text_gen = pipeline(
    "question-answering",
    model=qa_model,
    tokenizer=qa_tokenizer,
    device=-1,
    max_length=512,
    do_sample=False,
    temperature=0.1,
    truncation=True
)
llm = HuggingFacePipeline(pipeline=text_gen)
```

```
Device set to use cpu
```

Section

메타데이터 포함 chromaDB -> LangGraph

히스토리 관리

[4]:
```python
# 3. 히스토리 관리
chats_by_session_id = {}
def get_chat_history(session_id: str) -> BaseChatMessageHistory:
    if session_id not in chats_by_session_id:
        chats_by_session_id[session_id] = InMemoryChatMessageHistory()
    return chats_by_session_id[session_id]
```

Section

메타데이터포함chromaDB -> LangGraph

노드 정의(1)

```python
# 4. 노드 정의
def ask_question(state, config: RunnableConfig):
    session_id = config["configurable"]["session_id"]
    chat_history = get_chat_history(session_id)

    # 질문과 토픽 입력 받기
    question = input("질문을 입력해주세요: ").strip()
    topic = input("관련 토픽을 입력해주세요 (공정운영, 유지보수, 이상대응, 전산관리, 안전규정): ").strip()

    # Topic 기반 Retriever 설정
    retriever = Chroma(
        persist_directory="./chromaDB_topic",
        embedding_function=embedding_model
    ).as_retriever(search_kwargs={"k": 3, "filter": {"topic": topic}})

    docs = retriever.invoke(question)
    top_docs = docs[:3]
    best_contexts = [doc.page_content for doc in top_docs]
    combined_context = "\n".join(best_contexts)

    history_text = get_buffer_string(chat_history.messages)
    full_context = f"{history_text}\n\n{combined_context}" if history_text else combined_context

    result = text_gen(question=question, context=full_context)

    chat_history.add_user_message(question)
    chat_history.add_ai_message(result["answer"])

    return {
        "question": question,
        "answer": result["answer"],
        "context": best_contexts
    }
```

Section

메타데이터 포함 chromaDB -> LangGraph

노드 정의(2)

```python
def get_answer(state):
    print(f"\n질문: {state['question']}")
    print(f"답변: {state['answer']}")
    return state

def ask_reference(state):
    user_input = input("\n참고 문서를 보시겠습니까? (예/아니오): ").strip()
    state["show_reference"] = user_input.startswith("예")
    return state

def get_reference(state):
    if state.get("show_reference"):
        print("\n참고 문서:\n")
        for i, ctx in enumerate(state["context"], 1):
            print(f"[{i}] {ctx}\n")
    return state

def ask_continue(state):
    user_input = input("\n계속하시겠습니까? (예/아니오): ").strip()
    state["continue"] = user_input.startswith("예")
    return state

def should_show_reference(state):
    return "get_reference" if state.get("show_reference") else "ask_continue"

def should_continue(state):
    return "ask_question" if state.get("continue") else END
```

Section

메타데이터 포함 chromaDB -> LangGraph

LangGraph 구성

[6]:
```python
# 5. LangGraph 구성
graph = StateGraph(dict)
graph.add_node("ask_question", ask_question)
graph.add_node("get_answer", get_answer)
graph.add_node("ask_reference", ask_reference)
graph.add_node("get_reference", get_reference)
graph.add_node("ask_continue", ask_continue)

graph.set_entry_point("ask_question")
graph.add_edge("ask_question", "get_answer")
graph.add_edge("get_answer", "ask_reference")
graph.add_conditional_edges("ask_reference", should_show_reference, {
    "get_reference": "get_reference",
    "ask_continue": "ask_continue"
})
graph.add_edge("get_reference", "ask_continue")
graph.add_conditional_edges("ask_continue", should_continue, {
    "ask_question": "ask_question",
    END: END
})
```

[6]:
```
<langgraph.graph.state.StateGraph at 0x779e082d5550>
```

Section

메타데이터 포함 chromaDB -> LangGraph

실행

```
# 6. 실행
session_id = str(uuid.uuid4())
config = {"configurable": {"session_id": session_id}}
app = graph.compile()
app.invoke({}, config=config)
```

```
질문을 입력해주세요: MES 공정 로그 보관 기간은?
관련 토픽을 입력해주세요 (공정운영, 유지보수, 이상대응, 전산관리, 안전규정): 전산관리

질문: MES 공정 로그 보관 기간은?
답변: 6개월간

참고 문서를 보시겠습니까? (예/아니오): 예

참고 문서:

[1] MES 시스템에 기록된 공정 로그는 6개월간 보관됩니다.

[2] 소자 특성 분석 결과는 전자문서 시스템에 등록합니다.

[3] 사내 기술 포럼 자료는 R&D 인트라넷에 주기적으로 업데이트됩니다.


계속하시겠습니까? (예/아니오): 아니오
{'question': 'MES 공정 로그 보관 기간은?',
 'answer': '6개월간',
 'context': ['MES 시스템에 기록된 공정 로그는 6개월간 보관됩니다.',
  '소자 특성 분석 결과는 전자문서 시스템에 등록합니다.',
  '사내 기술 포럼 자료는 R&D 인트라넷에 주기적으로 업데이트됩니다.'],
 'show_reference': True,
 'continue': False}
```

감사합니다.

Q & A