CHAPTER 06. 메모리

1. VectorStoreRetrieverMemory

대화 내용을 벡터스토어 라는 데이터베이스에 저장하고 나중에 조회를 해볼 수 있는 기능.

벡터 스토어에 메모리를 저장하고 호출될 때마다 가장 '눈에 띄는' 상위 K개의 문서를 쿼리 합니다.
시간 순서를 고려하는 것이 아니라 과거의 대화 내용을 검색을 통해서 필요한 내용을 가져온다.

예제 - VectorStoreRetrieverMemory

```
from dotenv import load_dotenv
import os

load_dotenv(verbose=True)
key = os.getenv('OPENAI_API_KEY')
```

```
import faiss
from langchain_openai import OpenAIEmbeddings
from langchain.docstore import InMemoryDocstore
from langchain.vectorstores import FAISS
```

```
embeddings_model = OpenAIEmbeddings()     # 임베딩 모델을 정의합니다.
```

```
# Vector Store 를 초기화 합니다.
embedding_size = 1536
index = faiss.IndexFlatL2(embedding_size)
vectorstore = FAISS(embeddings_model, index, InMemoryDocstore({}), {})
```

```
from langchain.memory import VectorStoreRetrieverMemory
```

```
# 벡터 조회가 여전히 의미적으로 관련성 있는 정보를 반환한다는 것을 보여주기 위해서입니다.
retriever = vectorstore.as_retriever(search_kwargs={"k": 1})
```

```
memory = VectorStoreRetrieverMemory(retriever=retriever)

# 임의의 대화를 저장합니다.
memory.save_context(
    inputs={
        "human": "안녕하세요, 오늘 면접에 참석해주셔서 감사합니다. 자기소개 부탁드립니다."
    },
    outputs={
        "ai": "안녕하세요. 저는 컴퓨터 과학을 전공한 신입 개발자입니다. 대학에서는 주로 자바와 파이썬을 사용했으며, 최근에는 웹 개발 프로젝트에 참여하여 실제 사용자를 위한 서비스를 개발하는 경험을 했습니다."
    },
)
memory.save_context(
    inputs={"human": "프로젝트에서 어떤 역할을 맡았나요?"},
    outputs={
        "ai": "제가 맡은 역할은 백엔드 개발자였습니다. 사용자 데이터 처리와 서버 로직 개발을 담당했으며, RESTful API를 구현하여 프론트엔드와의 통신을 담당했습니다. 또한, 데이터베이스 설계에도 참여했습니다."
    },
)
memory.save_context(
    inputs={
        "human": "팀 프로젝트에서 어려움을 겪었던 경험이 있다면 어떻게 해결했나요?"
    },
    outputs={
        "ai": "프로젝트 초기에 의사소통 문제로 몇 가지 어려움이 있었습니다. 이를 해결하기 위해 저희 팀은 정기적인 미팅을 갖고 각자의 진행 상황을 공유했습니다. 또한, 문제가 발생했을 때는 적극적으로 의견을 나누고, 합리적인 해결책을 찾기 위해 노력했습니다."
    },
)
memory.save_context(
    inputs={"human": "개발자로서 자신의 강점은 무엇이라고 생각하나요?"},
    outputs={
        "ai": "제 강점은 빠른 학습 능력과 문제 해결 능력입니다. 새로운 기술이나 도구를 빠르게 습득할 수 있으며, 복잡한 문제에 직면했을 때 창의적인 해결책을 제시할 수 있습니다. 또한, 팀워크를 중시하며 동료들과 협력하는 것을 중요하게 생각합니다."
    },
)
```

```
# 다음의 질문을 했을 때 Vector Store 로 부터 1개(k=1 이기 때문)의 가장 관련성 높은 대화를 반환합니다.
# 질문: "면접자 전공은 무엇인가요?"

# 메모리에 질문을 통해 가장 연관성 높은 1개 대화를 추출합니다.
print(memory.load_memory_variables({"prompt": "면접자 전공은 무엇인가요?"})["history"])
```

```
# 이번에는 다른 질문을 통해 가장 연관성 높은 1개 대화를 추출합니다.
# 질문: "면접자가 프로젝트에서 맡은 역할은 무엇인가요?"

print(
    memory.load_memory_variables(
        {"human": "면접자가 프로젝트에서 맡은 역할은 무엇인가요?"}
    )["history"]
)
```

결과

```
human: 안녕하세요, 오늘 면접에 참석해주셔서 감사합니다. 자기소개 부탁드립니다.
ai: 안녕하세요. 저는 컴퓨터 과학을 전공한 신입 개발자입니다. 대학에서는 주로 자바와 파이썬을 사용했으며, 최근에는 웹 개발 프로젝트에 참여하여 실제 사용자를 위한 서비스를 개발하는 경험을 했습니다.

human: 프로젝트에서 어떤 역할을 맡았나요?
ai: 제가 맡은 역할은 백엔드 개발자였습니다. 사용자 데이터 처리와 서버 로직 개발을 담당했으며, RESTful API를 구현하여 프론트엔드와의 통신을 담당했습니다. 또한, 데이터베이스 설계에도 참여했습니다.
```

```
pip install faiss-cpu
```

- 출처 : 랭체인LangChain 노트 (https://wikidocs.net/233343)