
- 출처: LangChain 공식 문서 또는 해당 교재명
- 원본 URL: https://smith.langchain.com/hub/teddynote/summary-stuff-documents

---

ConversationEntityMemory

- 대화에 등장하는 **특정 인물** 이나 **사물** 등의 **중요한 정보(엔티티) 를 따로 저장** 하는 메모리
- 즉, **엔티티 메모리 = 대화에서 특정 엔티티에 대한 주어진 사실 기억**
- **역할**:
  - 대화의 흐름과 관계없이, 특정 중요한 사실 들을 오랫동안 기억
  - **누 사람들의 이름이나 별명을 기억하는 수첩**
- **LLM 사용 → 엔티티에 대한 정보 추출 → 시간이 지남에 따라 해당 엔티티에 대한 지식 축적**

```
# 환경변수 처리 및 클라이언트 생성
from langsmith import Client
from langchain.prompts import PromptTemplate
from langchain.prompts import ChatPromptTemplate
from dotenv import load_dotenv

import os
import json

# 클라이언트 생성
api_key = os.getenv("LANGSMITH_API_KEY")
client = Client(api_key=api_key)
```

```
# LangSmith 추적 설정하기 (https:smith.langchin.com)
# LangSmith 추적을 위한 라이브러리 임포트
from langsmith import traceable

# LangSmith 환경 변수 확인

print("\n--- LangSmith 환경 변수 확인 ---")
langchain_tracing_v2 = os.getenv('LANGCHAIN_TRACING_V2')
langchain_project = os.getenv('LANGCHAIN_PROJECT')
langchain_api_key_status = "설정됨" if os.getenv('LANGCHAIN_API_
org = "설정됨" if os.getenv('LANGCHAIN_ORGANIZATION') else "설정

if langchain_tracing_v2 == "true" and os.getenv('LANGCHAIN_API
    print(f"✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='{la
    print(f"✅ LangSmith 프로젝트: '{langchain_project}'")
    print(f"✅ LangSmith API Key: {langchain_api_key_status}")
    print("  -> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요."
else:
    print("❌ LangSmith 추적이 완전히 활성화되지 않았습니다. 다음을 확인하
    if langchain_tracing_v2 != "true":
        print(f"  - LANGCHAIN_TRACING_V2가 'true'로 설정되어 있지
    if not os.getenv('LANGCHAIN_API_KEY'):
        print("  - LANGCHAIN_API_KEY가 설정되어 있지 않습니다.")
    if not langchain_project:
        print("  - LANGCHAIN_PROJECT가 설정되어 있지 않습니다.")
```

"@traceable" 주석은 허용되지 않습니다. 허용되는 값은 다음과 같습니다. [@param, @title, @markdown]

- 셀 출력

```
--- LangSmith 환경 변수 확인 ---
✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='true')
✅ LangSmith 프로젝트: 'LangChain-prantice'
✅ LangSmith API Key: 설정됨
-> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요.
```

- 엔티티 메모에 필요한 모듈 임포트

```
from langchain.chains import ConversationChain          # 대화 체인 구성하는 클래스, 대화의 흐름 관리

from langchain.memory import ConversationEntityMemory   # 대화에서 엔티티(개체)를 추적하고 기억하는 메모리 클래스
# 대화 중 언급된 개체들을 기억하고 추적하는 기능을 제공

from langchain.memory.prompt import ENTITY_MEMORY_CONVERSATION_TEMPLATE     # 엔티티 메모리를 사용하는 대화 템플릿
# 대화에서 엔티티를 추적하고 기억하는 데 사용되는 프롬프트 템플릿
```

- `Entity memory`를 효과적으로 사용하기 위해 제공되는 프롬프트 사용하기

```
# Entity Memory를 사용하는 프롬프트 내용 출력하기

print(ENTITY_MEMORY_CONVERSATION_TEMPLATE.template)
```

- 셀 출력

```
You are an assistant to a human, powered by a large language model trained by OpenAI.

You are designed to be able to assist with a wide range of tasks, from answering simple questions to providing in

You are constantly learning and improving, and your capabilities are constantly evolving. You are able to process

Overall, you are a powerful tool that can help with a wide range of tasks and provide valuable insights and infor

Context:
{entities}

Current conversation:
{history}
Last line:
Human: {input}
You:
```

- **LLM 생성하기**

```
import os
from dotenv import load_dotenv
import openai

# .env 파일에서 환경변수 불러오기
load_dotenv()

# 환경변수에서 API 키 가져오기
api_key = os.getenv("OPENAI_API_KEY")

# OpenAI API 키 설정
openai.api_key = api_key

# OpenAI를 불러오기
# ✅ 디버깅 함수: API 키가 잘 불러와졌는지 확인
def debug_api_key():
    if api_key is None:
        print("❌ API 키를 불러오지 못했습니다. .env 파일과 변수명을 확인하세요.")
    elif api_key.startswith("sk-") and len(api_key) > 20:
        print("✅ API 키를 성공적으로 불러왔습니다.")
    else:
        print("⚠️ API 키 형식이 올바르지 않은 것 같습니다. 값을 확인하세요.")

# 디버깅 함수 실행
debug_api_key()
```

- 셀 출력

  ✅ API 키를 성공적으로 불러왔습니다.

```
import os
from dotenv import load_dotenv
from openai import OpenAI

# .env에서 API 키 로드
load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")

# 클라이언트 객체 생성 (필수)
client = OpenAI(api_key=api_key)

# GPT 호출
response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "user", "content": "안녕! 오늘 날씨 어때?"}
    ]
)

# 응답 출력
print(response.choices[0].message.content)
```

- 셀 출력 (2.4s)

  안녕하세요! 제가 실시간 날씨 데이터를 제공할 수는 없지만, 현재 날씨를 확인하려면 기상청 웹사이트나 날씨 앱을 이용하시면 좋습니다. 오늘 날씨는 어떤가

```
import os
from dotenv import load_dotenv
from openai import OpenAI
from langchain.chat_models import ChatOpenAI

# .env에서 API 키 로드
load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")

# LLM 생성하기
llm = ChatOpenAI(
    temperature=0,
    openai_api_key=api_key,
    model="gpt-4o-mini",
    )

# ConversationChain 생성하기
conversation = ConversationChain(
    llm=llm,
    prompt=ENTITY_MEMORY_CONVERSATION_TEMPLATE,
    memory=ConversationEntityMemory(llm=llm),
)
```

- 대화 시작하기
- 입력한 대화를 바탕으로 **`ConversationEntityMemory`는 주요 `Entity` 정보를 별도로 저장**

```
conversation.predict(
    input="테디와 셜리는 한 회사에서 일하는 동료입니다."
    "테디는 개발자이고 셜리는 디자이너입니다. "
    "그들은최근 회사에서 일하는 것을 그만두고 자신들의 회사를 차릴 계획을 세우고 있습니다."
)
```

- 셀 출력 (25.6s)

'그렇군요! 테디와 셜리가 함께 회사를 차릴 계획이라니 흥미로운 이야기네요. 그들이 어떤 분야에서 회사를 운영할 계획인지, 또는 어떤 제품이나 서비스를 제

- Entity = **memory.entity_store.store 에서 확인 가능**

```
# entity memory 출력해보기

conversation.memory.entity_store.store
```

- 셀 출력

```
{'테디': '테디는 개발자로, 셜리와 함께 한 회사에서 일하며 최근 회사를 그만두고 자신들의 회사를 차릴 계획을 세우고 있습니다.',
'셜리': '셜리는 한 회사에서 일하는 디자이너이며, 최근에 동료 테디와 함께 회사를 차릴 계획을 세우고 있습니다.'}


print(type(conversation.memory.entity_store.store))             # <class 'dict'>
```

---

test

```
# 환경변수 처리 및 클라이언트 생성
from langsmith import Client
from langchain.prompts import PromptTemplate
from langchain.prompts import ChatPromptTemplate
from dotenv import load_dotenv
from langsmith import traceable

import os
import json

# 클라이언트 생성
api_key = os.getenv("LANGSMITH_API_KEY")
client = Client(api_key=api_key)


from langchain.chains import ConversationChain
from langchain.memory import ConversationEntityMemory
from langchain.memory.prompt import ENTITY_MEMORY_CONVERSATION_TEMPLATE


import os
import json
from dotenv import load_dotenv
from openai import OpenAI
from langchain_openai import ChatOpenAI

# .env에서 API 키 로드
load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")

# LLM 생성하기
llm = ChatOpenAI(
    temperature=0,
    openai_api_key=api_key,
    model="gpt-4o-mini",
    )

# ConversationChain 생성하기
conversation = ConversationChain(
    llm=llm,
    prompt=ENTITY_MEMORY_CONVERSATION_TEMPLATE,
    memory=ConversationEntityMemory(llm=llm),
)
```

- 각 **엔티티 메모리를 활용한 대화 확장**
    - 주제: **음악** 분야

- 각 예시에서 **`ConversationEntityMemory`** = 알렉스 (작곡가), 리사 (가수)의 **정보 를 저장 + 새로운 정보 를 대화에 추가 할 때마다 엔티티 갱신**
  - **새로운 아이디어, 계획이 들어오면 그에 맞는 엔티티 추가**
  - **테스트 및 피드백과 관련된 상황 발생 시 → 이를 반영하여 엔티티 정보를 수정함**

```
#_1 알렉스와 리사의 협업

conversation.predict(
    input="알렉스는 유명한 작곡가로, 리사는 그의 오랜 친구이자 가수입니다. 두 사람은 함께 새로운 앨범을 준비 중입니다."
)
```

- 셀 출력 (4.4s)

```
'그렇군요! 알렉스와 리사가 함께 앨범을 준비하고 있다니 정말 흥미로운 프로젝트일 것 같아요. 어떤 스타일의 음악을 작업하고 있는지, 또는 앨범의 주제에
```

```
# entity memory 출력해보기

conversation.memory.entity_store.store
```

- 셀 출력

```
{'알렉스': '알렉스는 유명한 작곡가로, 리사와 함께 새로운 앨범을 준비 중입니다.',
'리사': '리사는 알렉스의 오랜 친구이자 가수이며, 현재 알렉스와 함께 새로운 앨범을 준비 중입니다.'}
```

```
#_2 앨범의 스타일과 테마

conversation.predict(
    input="알렉스와 리사는 이번 앨범에서 감성적인 발라드와 강렬한 록 음악을 섞은 새로운 스타일을 시도하고 있습니다."
)
```

- 셀 출력 (5.9s)

```
'그렇군요! 감성적인 발라드와 강렬한 록 음악을 섞은 새로운 스타일이라니, 정말 흥미로운 조합이네요. 두 장르의 요소가 어떻게 어우러질지 궁금합니다. 앨범
```

```
# entity memory 출력해보기

conversation.memory.entity_store.store
```

- 셀 출력

```
{'알렉스': '알렉스는 유명한 작곡가로, 리사와 함께 감성적인 발라드와 강렬한 록 음악을 섞은 새로운 스타일의 앨범을 준비 중입니다.',
'리사': '리사는 알렉스의 오랜 친구이자 가수이며, 현재 알렉스와 함께 감성적인 발라드와 강렬한 록 음악을 섞은 새로운 스타일의 앨범을 준비 중입니다.'}
```

- 추가된 부분
  - 알렉스 + 리사: 감성적인 발라드와 함께 강렬한 록 음악을 섞은 새로운 스타일의

```
#_3 첫 번째 싱글 발표

conversation.predict(
    input="알렉스와 리사는 앨범 출시 전에 첫 번째 싱글을 발표했습니다. 첫 번째 싱글은 록 장르의 곡으로, 이미 팬들 사이에서 큰 화제가 되고 있습니다."
)
```

- 셀 출력 (28.2s)

```
'와, 첫 번째 싱글이 록 장르의 곡이라니 정말 기대되네요! 팬들 사이에서 큰 화제가 되고 있다는 것은 곡이 많은 사랑을 받고 있다는 뜻이겠죠. 이 곡의 제
```

```
# entity memory 출력해보기

conversation.memory.entity_store.store
```

```
{'알렉스': '알렉스는 유명한 작곡가로, 리사와 함께 감성적인 발라드와 강렬한 록 음악을 섞은 새로운 스타일의 앨범을 준비 중이며, 앨범 출시 전에 첫 번째 싱글을 발표했으며, 이 곡은 록 장르로 팬들 사이에서 큰 화제가 되고 있습니다.',
 '리사': '리사는 알렉스의 오랜 친구이자 가수이며, 현재 알렉스와 함께 감성적인 발라드와 강렬한 록 음악을 섞은 새로운 스타일의 앨범을 준비 중이며, 앨범 출시 전에 첫 번째 싱글을 발표했습니다. 첫 번째 싱글은 록 장르의 곡으로, 이미 팬들 사이에서 큰 화제가 되고 있습니다.'}
```

- 셀 출력

```
{'알렉스': '알렉스는 유명한 작곡가로, 리사와 함께 감성적인 발라드와 강렬한 록 음악을 섞은 새로운 스타일의 앨범을 준비 중이며, 앨범 출시 전에 첫 번
'리사': '리사는 알렉스의 오랜 친구이자 가수이며, 현재 알렉스와 함께 감성적인 발라드와 강렬한 록 음악을 섞은 새로운 스타일의 앨범을 준비 중이며, 앨범
```

- 추가된 부분
  - 알렉스 + 리사: **앨범 출시 전에 첫 번째 싱글을 발표, 록 장르의 곡, 이미 팬들 사이에서 큰 화제**

```
#_4 음악적 영감

conversation.predict(
    input="알렉스는 이번 앨범을 준비하면서 많은 클래식 록 밴드들의 영향을 받았다고 밝혔습니다. 리사는 그의 음악적 영감을 받아 가사 작업에 깊이 몰두하고 있습니다.
)
```

- 셀 출력 (28.6s)

```
'알렉스가 클래식 록 밴드들의 영향을 받았다는 것은 정말 흥미로운 사실이네요! 그런 영향이 그의 작곡 스타일에 어떻게 반영될지 기대됩니다. 리사가 가사 작
```

```
# entity memory 출력해보기

conversation.memory.entity_store.store
```

- 셀 출력

```
{'알렉스': '알렉스는 유명한 작곡가로, 리사와 함께 감성적인 발라드와 강렬한 록 음악을 섞은 새로운 스타일의 앨범을 준비 중이며, 앨범 출시 전에 첫 번
'리사': '리사는 알렉스의 오랜 친구이자 가수이며, 현재 알렉스와 함께 감성적인 발라드와 강렬한 록 음악을 섞은 새로운 스타일의 앨범을 준비 중이며, 앨범
```

- 추가된 부분
  - 알렉스: **이번 앨범을 준비하며 많은 클래식 록 밴드들의 영향을 받음**
  - 리사: **알렉스의 음악적 영감을 받아 가사 작업에 몰두함**

- *next: 대화 지식그래프 메모리(ConversationKGMemory)*
