
- 출처: LangChain 공식 문서, 조코딩의 랭체인으로 AI 에이전트 서비스 만들기
- 깃허브 저장소 출처: https://github.com/sw-woo/hanbit-langchain

음식 리뷰 평가 AI 만들기

- **Open AI - gpt-4o-mini 로 시도**
- *출처: 위에 표기*

```
# 환경변수 처리 및 클라이언트 생성
from langsmith import Client
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_A
org = "설정됨" if os.getenv('LANGCHAIN_ORGANIZATION') else "설

if langchain_tracing_v2 == "true" and os.getenv('LANGCHAIN_
    print(f"✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='
    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 확
    if langchain_tracing_v2 != "true":
        print(f"  - LANGCHAIN_TRACING_V2가 'true'로 설정되어 있
    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.
```

```
import os
from dotenv import load_dotenv
import openai

from langchain_openai import ChatOpenAI

# .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.

프롬프트 템플릿 정의하기

- 리뷰와 점수 범위를 입력 받아 AI 모델이 평가를 생성할 수 있도록 작성

```
from langchain_core.prompts import PromptTemplate
from langchain_core.output_parsers import StrOutputParser

prompt_template = "이 음식 리뷰 '{review}'에 대해 '{rating1}'점부터 '{rating2}'점까지의 평가를 해주세요."

prompt = PromptTemplate(
    input_variables=[
        "review", "rating1", "rating2"], template=prompt_template
)


print(type(prompt_template))                    # <class 'str'>
print(type(prompt))                             # <class 'langchain_core.prompts.prompt.PromptTemplate'>
```

LLM 초기화

- **temperature = 0.7 → 창의성이 높도록 설정**

```
# LLM 생성

llm = ChatOpenAI(
    temperature=0.7,
    openai_api_key=api_key,
    model="gpt-4o-mini",
    )
```

Chain 생성

```
chain_gpt = prompt | llm | StrOutputParser()
```

사용자 리뷰에 대한 평가 요청해보기

```
try:
    response = chain_gpt.invoke({
        "review": "맛은 있엇지만 배달 포장이 부족하여서 아쉬웠습니다.",
        "rating1": "1",
```

```
        "rating2": "5"
    })
    print(f"평가 결과: {response}")

except Exception as e:
    print(f"Error: {e}")
```

- 셀 출력 (2.1s)

```
평가 결과: 이 리뷰는 음식의 맛에 대해서는 긍정적인 평가를 하고 있지만, 배달 포장에 대한 불만을 표현하고 있습니다. 맛이 좋았다는 점은 긍정적인 요소이

따라서, 평가를 한다면 '3점' 정도가 적절할 것 같습니다. 이는 음식의 맛이 좋았지만, 서비스나 포장 측면에서 아쉬움이 있었다는 점을 반영한 점수입니다.
```

---

gemini로 시도해보기

- **gemini-flash-lite 모델로 시도**

```
import os
from dotenv import load_dotenv

from datetime import datetime
from langchain_google_genai import ChatGoogleGenerativeAI

load_dotenv()

# Google API 키 설정
api_key = os.getenv("GOOGLE_API_KEY")

# 모델 초기화 (gemini-2.5-flash-lite)
try:
    gemini_lc = ChatGoogleGenerativeAI(
        model="gemini-2.5-flash-lite",
        temperature=0.7,
        max_output_tokens=4096,
    )
    print("✅ Google GenAI 모델 초기화 성공.")
# 호출 실패 시를 대비한 디버깅용
except Exception as e:
    print(f"❌ Google GenAI 모델 초기화 실패: {e}")
    print("  -> GOOGLE_API_KEY 환경 변수가 올바르게 설정되었는지 확인하세요.")
    print("---")
    exit() # 모델 초기화 실패 시 프로그램 종료
```

- 셀 출력 (0.8s)

```
✅ Google GenAI 모델 초기화 성공.
```

- **체인 재생성**

```
# llm 모델을 변경해 새로운 체인 생성하기

chain_gemini = prompt | gemini_lc | StrOutputParser()
```

- **사용자 리뷰에 대한 평가 요청**

```
try:
    response = chain_gemini.invoke({
        "review": "맛은 있엇지만 배달 포장이 부족하여서 아쉬웠습니다.",
        "rating1": "1",
        "rating2": "5"
    })
```

```
    print(f"평가 결과: {response}")
except Exception as e:
    print(f"Error: {e}")
```

- 셀 출력 (1.6s)

```
평가 결과: 이 음식 리뷰에 대한 평가는 다음과 같습니다.

**평점: 3점**

**이유:**

*   **긍정적인 부분 (맛):** "맛은 있었지만"이라는 표현은 음식 자체의 맛에 대해서는 긍정적인 평가를 내리고 있음을 보여줍니다. 이는 중요한 요소이며
*   **부정적인 부분 (포장):** "배달 포장이 부족하여서 아쉬웠습니다"라는 부분은 음식의 맛과는 별개로, 배달 과정에서의 경험이 좋지 않았음을 명확히

**종합적인 판단:**

맛은 좋았다는 점은 분명한 장점이지만, 포장 문제는 전체적인 경험을 해치기 때문에 최고 점수를 주기에는 어렵습니다. 반대로 맛이 좋았다는 점 때문에 최저
```

- **`gpt-4o-mini` vs `gemini-2.5-flash-lite`** 결과 비교

	gpt-4o-mini
응답 속도	2.1s
평가 결과	평가 결과: 이 리뷰는 음식의 맛에 대해서는 긍정적인 평가를 하고 있지만, 배달 포장에 대한 불만을 표현하고 있습니다. 맛이 좋았다는 점은 긍정적인 요소이지만, 포장 문제로 인해 전체적인 경험이 저하된 것으로 따라서, 평가를 한다면 '3점' 정도가 적절할 것 같습니다. 이는 음식의 맛이 좋았지만, 서비스나 포장 측면에서 아쉬움이 있었다는 점을 반영한 점수입니다.
평점	3점
이유 (긍정적)	이 리뷰는 음식의 맛에 대해서는 긍정적인 평가를 하고 있지만,
이유 (부정적)	배달 포장에 대한 불만을 표현하고 있습니다.
종합적인 판단	이는 음식의 맛이 좋았지만, 서비스나 포장 측면에서 아쉬움이 있었다는 점을 반영한 점수입니다.