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

LCEL (LangChain Expression Language) 기반 숙박 시설 리뷰 평가 AI 만들기

- *출처: 위에 표기*

- **`LCEL` = 랭체인에서 제공하는 간결 하고 선언적 인 체인 구성 언어**
  - 기존: `LLM`, 프롬프트, 문서 검색기, 출력 파서 등을 **일일이 생성하고 메서드를 호출**
  - **`LCEL`**
    - 연산자 기반의 직관적 인 방식으로 `LLM` 워크플로우 구성 가능
    - = LLM 애플리케이션의 실행 흐름을 코드로 표현하는 언어이자 문법

- **`RunnablePassthrough.assign()`**
  - LangChain에서 **다양한 컴포넌트(예: LLM, 프롬프트, 파서 등)를 연결하고 실행하기 위한 기본 인터페이스**
  - 여러 컴포넌트를 연결 → 복잡한 데이터 처리 **파이프라인 구축** 가능
    - 예시: prompt → LLM에 전달 → LLM의 출력을 파서로 전달
  - `invoke`, `batch`, `stream` 등의 메서드를 통해 실행
  - 각 Runnabel은 입력 데이터를 받아 처리하고, 그 결과를 다음 Runnable에 전달하는 역할

```
# 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_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.
```

```
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.
```

- 환경 설정 및 LLM 초기화 공통

```
# 필요한 모듈 임포트

import os
from dotenv import load_dotenv

from langchain_core.prompts import PromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_core.runnables import RunnablePassthrough
from langchain_openai import ChatOpenAI                                    # gpt
from langchain_google_genai import ChatGoogleGenerativeAI                  # gemini
```

- **`temperature=0.7` → 다양한 대답이 나올 수 있도록 설정**

```
# gpt
load_dotenv()
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")

# ChatOpenAI 모델 초기화
# Temperature 설정 (모델의 응답 다양성 조절)
openai_llm = ChatOpenAI(
    model="gpt-4o-mini",
    api_key=OPENAI_API_KEY,
    temperature=0.7)
```

```
# gemini
load_dotenv()
api_key = os.getenv("GOOGLE_API_KEY")

# LLM 설정
gemini_lc = ChatGoogleGenerativeAI(
        model="gemini-2.5-flash-lite",
        temperature=0.7,
        api_key=api_key,
        max_output_tokens=4096,
    )
```

☆ 프롬프트 템플릿 정의하기

- **step_1. 리뷰 번역 = Prompt1** : 영어로 작성된 리뷰를 한글로 번역
- **step_2. 리뷰 요약 = Prompt2** : 번역된 리뷰를 한 문장으로 요약
- **step_3. 긍정/부정 점수 평가 = Prompt3** : 번역된 리뷰를 바탕으로 0점에서 10점 사이의 점수 매기기
- **step_4. 언어 감지 = Prompt4** : 원래 리뷰가 작성된 언어 감지
- **step_5. 공손한 답변 생성 = Prompt5** : 요약된 리뷰와 감지된 언어 정보를 기반으로 공손한 답변 생성하기
- **step_6. 답변 번역 = Prompt6** : 생성된 답변 한국어로 번역하기

```
# --- 프롬프트 템플릿 정의 ---

# 프롬프트 1: 리뷰 번역
prompt1 = PromptTemplate(
    input_variables=['review'],
    template="다음 숙박 시설 리뷰를 한글로 번역하세요.\n\n{review}"
)

# 프롬프트 2: 번역된 리뷰 요약
prompt2 = PromptTemplate.from_template(
    "다음 숙박 시설 리뷰를 한 문장으로 요약하세요.\n\n{translation}"
)

# 프롬프트 3: 번역된 리뷰 감성 점수 평가
prompt3 = PromptTemplate.from_template(
    "다음 숙박 시설 리뷰를 읽고 0점부터 10점 사이에서 부정/긍정 점수를 매기세요. 숫자만 대답하세요.\n\n{translation}"
)

# 프롬프트 4: 원본 리뷰 언어 식별
prompt4 = PromptTemplate.from_template(
    "다음 숙박 시설 리뷰에 사용된 언어가 무엇인가요? 언어 이름만 답하세요.\n\n{review}"
)

# 프롬프트 5: 요약에 대한 공손한 답변 생성 (원본 언어 사용)
prompt5 = PromptTemplate.from_template(
    "다음 숙박 시설 리뷰 요약에 대해 공손한 답변을 작성하세요.\n답변 언어:{language}\n리뷰 요약:{summary}"
)

# 프롬프트 6: 생성된 답변을 한국어로 번역
prompt6 = PromptTemplate.from_template(
    "다음 생성된 답변을 한국어로 번역해주세요. \n 리뷰 번역 {reply1}"
)


print(type(prompt1))                # <class 'langchain_core.prompts.prompt.PromptTemplate'>
```

☆ LLM Chain

- **프롬프트 템플릿 | LLM | 결과 파서 (StrOutputParser)**
- 여섯 단계를 체인으로 묶어서 처리 **⟶ 유지보수성, 확장성, 성능 측면에서 가장 효율적인 구조**
    - 각 단계의 출력을 그대로 다음 단계의 입력으로 넘김 ⟶ 중간 데이터 처리 코드 제거 가능

- 단계별 로그와 예외를 분리 = **디버깅 용이**
- 프롬프트, 모델, 파서 를 **모듈처럼 재사용하거나 교체 쉬움**
- 감성 점수, 언어 감지 같은 병렬 브랜치 추가하거나 다국어 답변을 동시에 생성하는 확장도 간단히 수행 가능

```
# --- LCEL을 사용한 체인 구성 요소 정의 ---
# gemini ver.

# 단계 1: 리뷰 번역 체인
translate_chain_component = prompt1 | gemini_lc | StrOutputParser()

# 단계 2: 번역된 리뷰 요약 체인
summarize_chain_component = prompt2 | gemini_lc | StrOutputParser()

# 단계 3: 감성 점수 평가 체인
sentiment_score_chain_component = prompt3 | gemini_lc | StrOutputParser()

# 단계 4: 언어 식별 체인
language_chain_component = prompt4 | gemini_lc | StrOutputParser()

# 단계 5: 첫 번째 답변 생성 체인
reply1_chain_component = prompt5 | gemini_lc | StrOutputParser()

# 단계 6: 두 번째 답변 (한국어 번역) 생성 체인
reply2_chain_component = prompt6 | gemini_lc | StrOutputParser()
```

- **RunnablePassthrough.assign → 리뷰 분석, 응답 생성 과정을 여러 단계로 나눠 연결**
  - **첫번째 assign**
    - 입력값: `{'review':'원본 리뷰 텍스트'}`
    - `review` → `translate_chain_component` → **한글 번역 결과(translation)를 생성**
    - 결과: 딕셔너리에 `{'translation':'번역된 텍스트'}` 가 추가됨

- **두번째 assign**
  - 입력값: 딕셔너리 내 기존 값 + `translation`
    - `review` → `language_chain_component` → **언어 판별(language)**
    - `translation` → `summarize_chain_component` → **요약 (summary)**
    - `translation` → `sentiment_score_chain_component` → **감성 점수 (sentiment_score)**
  - 결과: 딕셔너리에 `summary`, `sentiment_score`, `language` 추가됨

- **세번째 assign**
  - 입력값:
  -

- **네번째 assign**

```
# --- LCEL을 사용하여 전체 체인 결합 ---
# RunnablePassthrough.assign을 사용하여 각 단계의 출력을 다음 단계의 입력으로 전달하고,
# 중간 결과들을 딕셔너리에 누적

combined_lcel_chain = (
    RunnablePassthrough.assign(
        # 입력: {'review': '원본 리뷰 텍스트'}
        # translate_chain_component는 'review' 키를 사용하여 호출
        # 출력: {'review': '...', 'translation': '번역된 텍스트'}
        translation=lambda x: translate_chain_component.invoke({"review": x["review"]})
    )
    | RunnablePassthrough.assign(
        # 입력: {'review': '...', 'translation': '번역된 텍스트'}
        # summarize_chain_component와 sentiment_score_chain_component는 'translation' 키 사용
        # language_chain_component는 원본 'review' 키를 사용
        # 출력: {'review': '...', 'translation': '...', 'summary': '요약된 텍스트', 'sentiment_score': '감성 점수', 'language': '언어
```

```
        summary=lambda x: summarize_chain_component.invoke({"translation": x["translation"]}),
        sentiment_score=lambda x: sentiment_score_chain_component.invoke({"translation": x["translation"]}),
        language=lambda x: language_chain_component.invoke({"review": x["review"]})
    )
    | RunnablePassthrough.assign(
        reply1=lambda x: reply1_chain_component.invoke({"language": x["language"], "summary": x["summary"]})
    )
    | RunnablePassthrough.assign(
        reply2=lambda x: reply2_chain_component.invoke({"reply1": x["reply1"]})
    )
)


# 숙박 시설 리뷰 입력
review_text = """
The hotel was clean and the staff were very helpful.
The location was convenient, close to many attractions.
However, the room was a bit small and the breakfast options were limited.
Overall, a decent stay but there is room for improvement.
"""

# 체인 실행 및 결과 출력
try:
    # .invoke() 메서드에 초기 입력을 딕셔너리 형태로 전달합니다.
    result = combined_lcel_chain.invoke(input={'review': review_text})

    # 결과 딕셔너리에서 각 키를 사용하여 값을 출력합니다.
    print(f'translation 결과: {result.get("translation", "N/A")} \n')
    print(f'summary 결과: {result.get("summary", "N/A")} \n')
    print(f'sentiment_score 결과: {result.get("sentiment_score", "N/A")} \n')
    print(f'language 결과: {result.get("language", "N/A")} \n')
    print(f'reply1 결과: {result.get("reply1", "N/A")} \n')
    print(f'reply2 결과: {result.get("reply2", "N/A")} \n')
except Exception as e:
    print(f"Error: {e}")
```

- 교재 속 답변 (gpt-3.5-turbo)

  translation 결과: 호텔을 깨끗하고 직원들도 매우 친절했습니다.

  위치는 편리했고 많은 관광 명소에 가까웠습니다.
  그러나 방은 조금 작았고 아침식사 옵션도 제한적이었습니다.
  전반적으로 만족스러운 숙박이었지만 개선할 부분이 있습니다.

  summary 결과: 깨끗하고 친절한 호텔, 편리한 위치와 제한적인 아침식사 옵션, 만족스러운 숙박 경험.

  sentiment_score 결과: 8

  language 결과: 영어

  reply 결과: Thank you for taking the time to share your feedback with us. We are delighted to hear that you enjoye

  reply2 결과: 귀중한 의견을 공유해주셔서 감사합니다. 깨끗하고 친절한 호텔에서 즐거운 숙박을 즐겼다는 소식을 들어 기쁩니다. 편리한 위치가 전체 경험을

- gemini-2.5-flash-lite 답변 (4.6s)

  translation 결과: 다음은 숙박 시설 리뷰의 한국어 번역입니다.

  호텔은 깨끗했고 직원들은 매우 친절했습니다.
  여러 관광지와 가까워 위치가 편리했습니다.
  하지만 방이 조금 작았고 아침 식사 메뉴가 제한적이었습니다.
  전반적으로 괜찮은 숙박이었지만 개선의 여지가 있습니다.

  summary 결과: 이 호텔은 깨끗하고 친절한 직원, 편리한 위치가 장점이지만, 방이 작고 아침 식사 메뉴가 제한적이라는 점은 개선이 필요합니다.

  sentiment_score 결과: 7

  language 결과: English

reply1 결과: Sure, here's a polite response to the review summary:

Dear Guest,

Thank you for taking the time to share your feedback with us. We are delighted to hear that you found our hotel t

We also appreciate you bringing to our attention the areas where we can improve. We understand your feedback rega

We hope to have the opportunity to welcome you back to our hotel in the future, and we are confident that we can

Sincerely,
[Your Hotel Name]

reply2 결과: ## 리뷰 번역

안녕하세요 고객님,

저희 호텔에 대한 소중한 의견을 공유해주셔서 진심으로 감사드립니다. 저희 호텔이 깨끗하고, 직원들이 친절하며, 위치가 편리하다는 점에 만족하셨다니 매우

또한, 개선할 부분에 대해 알려주신 점도 감사드립니다. 객실 크기와 조식 메뉴의 제한점에 대한 고객님의 의견을 충분히 이해하고 있으며, 저희는 고객 경험

앞으로도 저희 호텔에 다시 방문해주실 기회가 있기를 바라며, 더욱 즐거운 숙박 경험을 제공해 드릴 수 있다고 확신합니다.

진심으로,
[호텔 이름]

- 결과 비교

translation 결과	호텔을 깨끗하고 직원들도 매우 친절했습니다. 위치는 편리했고 많은 관광 명소에 가까웠습니다. 그러나 방은 조금 작았고 아침식사 옵션도 제한적이었습니다. 전반적으로 만족스러운 숙박이었지만 개선
summary 결과	깨끗하고 친절한 호텔, 편리한 위치와 제한적인 아침식사 옵션, 만족스러운 숙박 경험.
sentiment_score 결과	8
language 결과	영어
reply 결과	Thank you for taking the time to share your feedback with us. We are delighted to hear that you enjoyed your stay at our clean and friendly hotel, and that our convenient l
reply2 결과	귀중한 의견을 공유해주셔서 감사합니다. 깨끗하고 친절한 호텔에서 즐거운 숙박을 즐겼다는 소식을 들어 기쁩니다. 편리한 위치가 전체 경험을 더 풍부하게 해 준 것으로 알고 있습니다. 아침식사 옵션이
