IDG
STRATEGIC
MARKETING SERVICES

ITWORLD CIO

ITWORLD HOW TO

# "LLM 앱을 개발하는 빠르고 쉬운 방법"

## 개발자를 위한 랭체인 입문서

> "LLM 개발을 더 간편하게" 랭체인의 개념과 이해
> "다양한 모델을 체인으로 연결한다" 예제로 시작하는 랭체인
> 구글 PaLM 2 API와 랭체인을 함께 사용하는 방법
> 랭체인과 구글 PaLM 2를 사용한 Q&A 앱 만들기
> LLM용 추적 및 디버깅 도구 '랭스미스' 따라잡기

# "LLM 개발을 더 간편하게" 랭체인의 개념과 이해

**Janakiram MSV** | InfoWorld

대규모 언어 모델(Large Language Model, LLM)에 대한 최신 트렌드를 쫓는 소프트웨어 개발자라면 연일 쏟아지는 AI 관련 소식이 당황스러울 수 있다. 이는 마치 새로운 오픈소스 모델이 매일 출시되고 상용 서비스 업체가 중요한 신기능을 매일 발표하는 상황과 비슷하다.

어느덧 LLM은 최신 소프트웨어 스택의 한 요소로 빠르게 부상했다. 그러나 오픈AI와 같은 서비스 업체가 제공하는 모델 API를 사용하든, 오픈소스 모델을 앱에 포함하든, LLM 기반 애플리케이션을 구축하려면 프롬프트를 보내고 응답을 기다리는 것 이상의 작업이 필요하다. 매개 변수 조정부터 프롬프트 보강, 응답 조정에 이르기까지 개발자가 고려해야 할 요소는 매우 많다.

LLM은 상태를 저장하지 않으므로 이전 대화 내용을 기억하지 못한다. 기록을 유지하고 LLM에 컨텍스트를 제공하는 것은 개발자의 책임이다. 새로운 대화에서 이전에 사용한 컨텍스트를 다시 가져오기 위해 영구 데이터베이스에 저장해야 할 수도 있다. 이렇듯 LLM에 장단기 메모리를 추가하는 것은 개발자의 주요 임무다.

또 다른 문제는 LLM에 대한 일률적인 규칙이 없다는 점이다. 감정 분석, 분류, 질문 답변과 요약 등 서로 다른 시나리오에 특화된 다양한 모델을 사용해야 할 수도 있다. 여러 LLM을 처리하는 작업은 복잡하며, 상당한 노력을 필요로 한다.

## LLM 앱 구축을 위한 통합 API 계층

랭체인(LangChain)은 LLM과 애플리케이션의 통합을 간소화하도록 설계된 SDK로, 앞서 설명한 문제 대부분을 해결한다. 표준 SQL 구문에 집중할 수 있도록 백엔드 데이터베이스의 구현 세부 정보를 추상화하는 ODBC, JDBC 드라이버와 비슷하다고 할 수 있다. 랭체인은 간단하고 통합된 API를 노출해 기본 LLM의 구현 세부 사항을 추상화하는데, 이런 API를 통해 개발자는 코드를 크게 변경하지 않고도 모델을 쉽게 교체하거나 대체할 수 있다.

랭체인은 챗GPT와 거의 같은 시기에 등장했다. 개발자 해리슨 체이스는 2022년 10월 말, LLM 열풍이 일어나기 시작한 시점에 랭체인을 처음 선보였다. 그 이후로 커뮤니티 구성원들이 적극적으로 기여했으며, 이로 인해 랭체인은 LLM과 상호 작용하는 탁월한 도구로 부상했다.

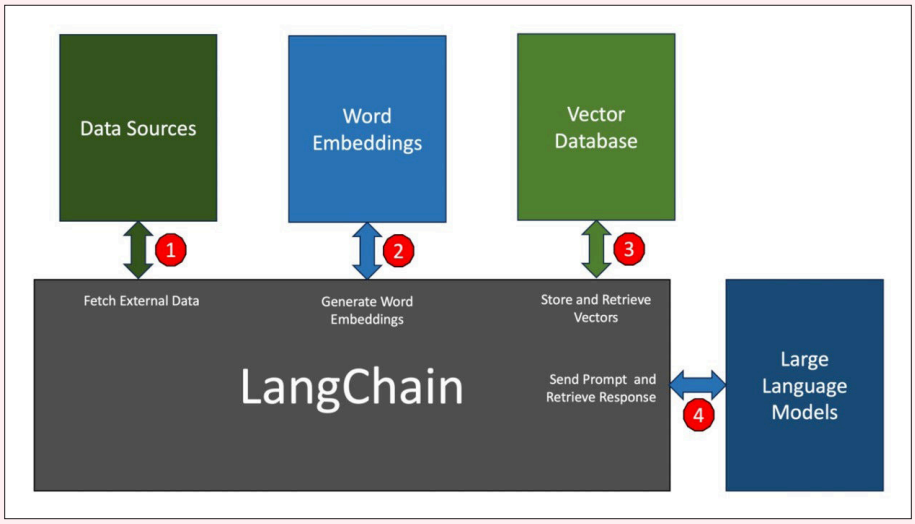



그림 1 | 랭체인의 핵심 모듈 4가지

랭체인은 외부 도구와 통합돼 환경을 조성하는 강력한 프레임워크다. 여기서는 랭체인을 구성하는 핵심 모듈과 각 모듈의 작동 방식을 알아본다.

## 랭체인의 핵심 모듈 4가지

- **데이터 소스** : 애플리케이션은 LLM에 대한 컨텍스트를 구축하기 위해 PDF, 웹 페이지, CSV, 관계형 데이터베이스와 같은 외부 소스에서 데이터를 검색해야 한다. 랭체인은 서로 다른 소스에서 데이터에 액세스하고 검색할 수 있는 모듈과 원활하게 통합된다.

- **단어 임베딩** : 일부 외부 소스에서 검색한 데이터는 벡터로 변환해야 한다. 이런 작업은 단어 임베딩 모델에 텍스트를 전달함으로써 이뤄진다. 예를 들어, 오픈AI의 GPT-3.5 모델에는 컨텍스트 전송에 사용해야 하는 단어 임베딩 모델이 포함되어 있다. 랭체인은 선택한 LLM을 기반으로 최적의 임베딩 모델을 선택하므로 개발자는 모델을 페어링할 때 추측하지 않아도 된다.

- **벡터 데이터베이스** : 생성된 임베딩은 유사성 검색을 위해 벡터 데이터베이스에 저장된다. 랭체인은 메모리 내 배열부터 파인콘(Pinecone)과 같은 호스팅된 벡터 데이터베이스에 이르기까지 다양한 소스에서 벡터를 쉽게 저장하고 검색할 수 있도록 지원한다.

- **LLM** : 랭체인은 오픈AI, 코히어(Cohere), AI21에서 제공하는 주류 LLM과 허깅페이스(Hugging Face)에서 제공하는 오픈소스 LLM을 지원한다. 지원되는 모델과 API 엔드포인트 목록은 빠르게 증가하고 있다.

## 모듈의 작동 방식

<그림 2>는 랭체인 프레임워크의 핵심을 나타낸다. 스택 상단의 애플리케이션은 파이썬 또는 자바스크립트 SDK를 통해 여러 랭체인 모듈 중 하나와 상호 작용한다. 각 모듈의 역할은 다음과 같다.

- **모델 I/O** : LLM과의 상호 작용을 처리한다. 기본적으로 효과적인 프롬프트 생성, 모델 API 호출, 출력 구문 분석에 도움이 된다. 생성형 AI의 핵심인 프롬프트 엔지니어링은 랭체인에서 잘 처리한다. 모델 I/O 모듈은 LLM 제공업체가 노출하는 인증, API 매개변수, 엔드포인트를 추상화한다. 마지막으로, 모델에서 보낸 응답을 애플리케이션에서 사용할 수 있는 형식으로 구문 분석할 수 있다.

- **데이터 연결** : 데이터 연결 모듈은 LLM 애플리케이션의 ETL 파이프라인에 해당한다. PDF 또는 엑셀 파일과 같은 외부 문서를 로드하고, 일괄적으로 단어 임베딩으로 변환한 다음, 임베딩을 벡터 데이터베이스에 저장하고, 마지막으로 쿼리를 통해 검색한다. 앞서 설명한 바와 같이 랭체인의 가장 중요한 구성 요소(component)다.

- **체인** : LLM과 상호 작용하는 것은 유닉스 파이프라인을 사용하는 것과 여러 면에서 유사하다. 한 모듈의 출력이 다른 모듈에 입력으로 전송된다. 때로는 원하는 결과를 얻을 때까지 응답을 명확히 하고 요약하기 위해 LLM에 의존해야 한다. 랭체인의 체인은 빌딩 블록과 LLM을 활용해 예상되는 응답을 얻을 수 있는 효율적인 파이프 라인을 구축하도록 설계됐다. 하나의 프롬프트와 LLM으로 구성된 간단한 체인뿐 아니라 재귀와 같이 LLM을 여러 번 호출해 결과를 얻는 매우 복잡한 체인을 구축하는 것도 가능하다. 문서를 요약한 다음 감정 분석을 요청하는 프롬프트는 복잡한 체인의 예다.

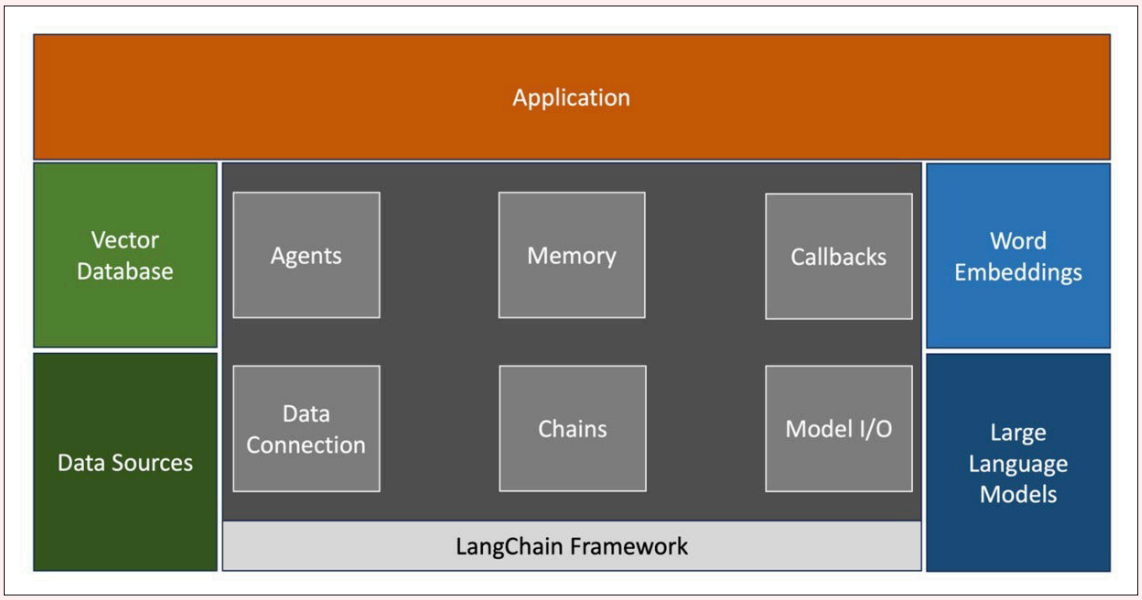



그림 2 | 랭체인 프레임워크

- **메모리** : 기본적으로 LLM은 상태를 저장하지 않지만 정확한 응답을 위해서는 컨텍스트가 필요하다. 랭체인의 메모리 모듈은 모델에 단기 및 장기 메모리를 쉽게 추가할 수 있도록 돕는다. 단기 메모리는 간단한 메커니즘을 통해 대화 기록을 유지한다. 메시지 기록은 레디스(Redis) 같은 외부 소스에 저장돼 장기 메모리를 유지할 수 있다.

- **콜백** : 랭체인은 LLM 애플리케이션의 다양한 단계에 연결할 수 있는 콜백 시스템을 제공한다. 로깅, 모니터링, 스트리밍 등의 작업에 유용하다. 파이프라인 내에서 특정 이벤트가 발생할 때 호출되는 사용자 지정 콜백 핸들러를 작성할 수 있다. 랭체인의 기본 콜백은 모든 단계의 출력을 콘솔에 간단히 내보내는 STDOUT을 가리킨다.

- **에이전트** : 에이전트는 랭체인에서 아주 강력한 모듈이다. LLM은 추론과 행동이 가능한데, 이를 ReAct 프롬프트 기법이라고 한다. 랭체인의 에이전트는 프롬프트를 행동 계획으로 추출하는 ReAct 프롬프트 제작을 단순화한다. 랭체인 에이전트의 기본 개념은 LLM을 사용해 일련의 작업을 선택하는 것으로, 일종의 동적 체인이다. 동작의 순서는 체인(코드)으로 하드코딩된다. 언어 모델은 에이전트 내에서 어떤 순서로 어떤 동작을 취할지 결정하는 추론 엔진으로 사용된다.

랭체인은 생성형 AI 기반 애플리케이션에서 중요한 구성 요소로 빠르게 자리 잡고 있다. 활발하게 발전하는 생태계 덕분에 다양한 빌딩 블록을 지원할 수 있다. 오픈소스 및 상용 LLM, 벡터 데이터베이스, 데이터 소스 및 임베딩을 지원하는 랭체인은 개발자에게 없어서는 안 될 필수 도구다.

# "다양한 모델을 체인으로 연결한다" 예제로 시작하는 랭체인

**Martin Heller** | InfoWorld

LLM을 효과적으로 사용할 수 있는 프롬프트 작성에는 어느 정도 기술이 필요하지만, LLM 사용법은 대체로 간단하다. 반면 언어 모델을 사용한 프로그래밍은 어려울 수 있다. 그럴 때는 랭체인을 사용하면 된다.

랭체인은 언어 모델 기반의 애플리케이션을 개발하는 프레임워크다. 랭체인을 사용해 챗봇 또는 개인 비서를 만들고, 문서 또는 구조화된 데이터에 대한 Q&A를 요약, 분석, 생성하고, 코드를 쓰거나 이해하고, API와 상호작용하고, 생성형 AI를 활용하는 여러 애플리케이션을 만들 수 있다. 현재 랭체인은 파이썬과 타입스크립트/자바스크립트 두 가지 버전이 있다.

언어 모델은 랭체인을 통해 데이터 소스와 연결되고 해당 환경과 상호작용한다. 랭체인의 주요 요소는 모듈식 추상화 및 추상화 구현의 모음으로 구성된다. 랭체인 기성형(off-the-shelf) 체인은 특정 상위 수준의 작업을 수행하기 위한 구성 요소(component)의 구조화된 어셈블리다. 구성 요소를 사용해 기존 체인을 맞춤설정하고 새 체인을 만들 수 있다.

언어 모델에는 LLM과 채팅 모델, 두 종류가 있다. LLM은 문자열을 입력으로 받고 문자열을 반환한다. 채팅 모델은 메시지 목록을 입력으로 받고 채팅 메시지를 반환한다. 채팅 메시지에는 내용과 역할, 두 가지 구성 요소가 포함된다. 역할은 내용의 출처를 사람, AI, 시스템, 함수 호출 또는 일반 입력 등으로 지정한다.

일반적으로 LLM은 입력에 프롬프트 템플릿을 사용한다. 프롬프트 템플릿을 사용해 LLM 또는 채팅 모델이 수행할 역할, 예를 들어 “영어를 프랑스어로 번역하는 유용한 비서” 등을 지정할 수 있다. 또한 프롬프트 템플릿을 이용하면 번역할 구문 목록과 같은 다양한 내용 인스턴스에 해당 템플릿을 적용할 수 있다.

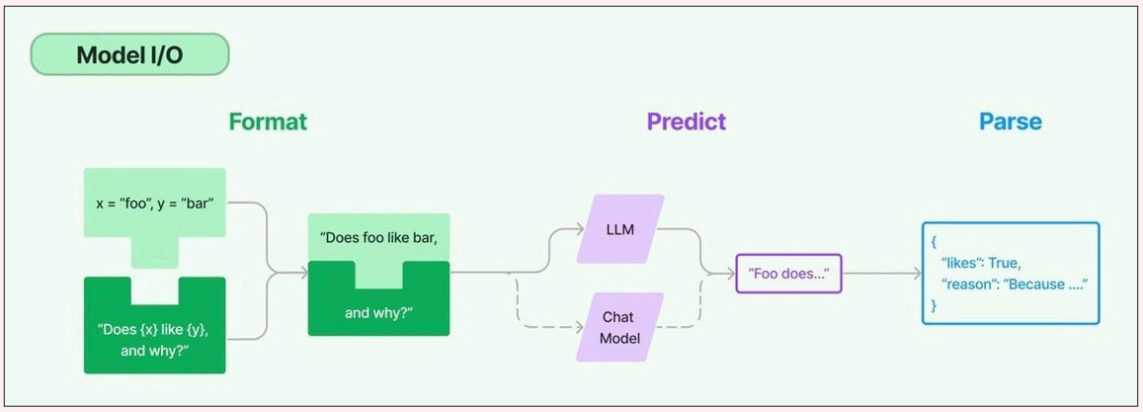
## 랭체인의 작동 방식

랭체인에는 6개의 모듈이 있다.

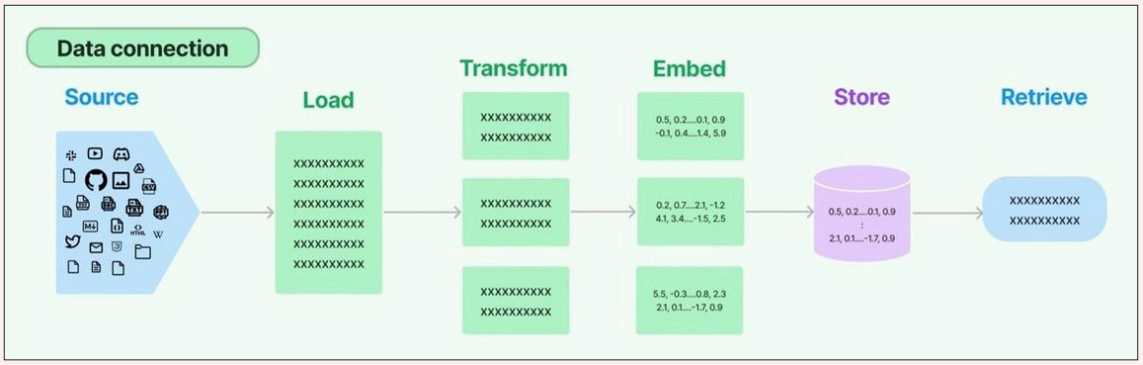
- **모델 I/O** : 언어 모델과의 인터페이스
- **데이터 연결** : 애플리케이션별 데이터와의 인터페이스
- **체인** : 호출 시퀀스 구축
- **에이전트** : 상위 지시문이 주어지면 체인이 사용할 툴을 선택할 수 있도록 함

• **메모리** : 체인 실행 간 애플리케이션 상태 유지
• **콜백** : 체인의 중간 단계를 기록 및 스트리밍

모델 I/O에서 프롬프트를 관리하고 공통 인터페이스를 통해 언어 모델을 호출하고 모델 출력에서 정보를 추출할 수 있다.



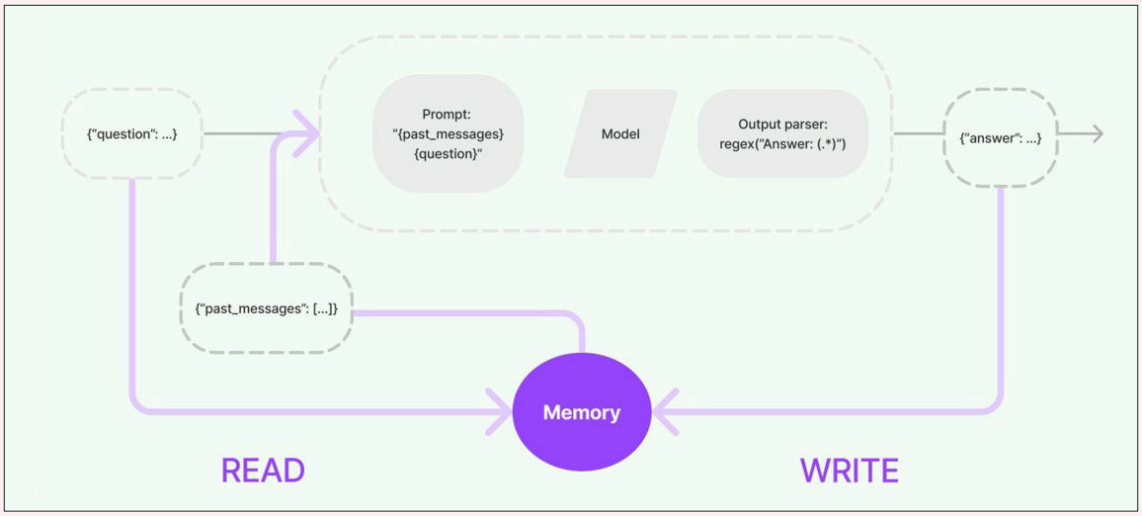

데이터 연결은 데이터를 로드, 변환, 저장 및 쿼리하기 위한 빌딩 블록을 제공한다.




복잡한 애플리케이션은 LLM을 상호, 또는 다른 구성요소와 체인으로 연결해야 한다. 랭체인은 이렇게 '체인으로 연결된' 애플리케이션을 위한 체인 인터페이스를 제공한다.

대화형 시스템은 어느 정도 기간의 과거 메시지에 직접 액세스할 수 있어야 한다. 랭체인에서는 이 기능을 메모리라고 한다.

시퀀스를 하드 코딩하는 체인과 달리, 에이전트는 언어 모델을 추론 엔진으로 사용해 어떤 작업을 어느 순서에 따라 수행할지를 결정한다.




콜백은 LLM 애플리케이션의 다양한 단계에 연결할 수 있게 해준다. 로깅, 모니터링, 스트리밍 등의 작업에 유용하다.

### 랭스미스를 사용한 디버깅

랭스미스(LangSmith)는 프로토타입에서 프로덕션으로 이전하기 위해 랭체인 언어 모델 애플리케이션 및 지능형 에이전트를 추적하고 평가하는 데 유용하다. 2024년 2월 초 기준 비공개 베타 단계다. 랭스미스 안내 자료와 문서는 베타 테스트에 참여하지 않아도 홈페이지에서 볼 수 있다.

### 랭체인 사용례

랭체인 사용례를 보면 문서에 대한 Q&A, API와의 통합, SQL, 챗봇, 코드 작성, 추출, 그래프 데이터 분석, 자가 검사, 요약, 태깅 등이 포함돼 있다. 일부 사용례에는 많은 예제가 있다. 예를 들어 Q&A에는 17개 정도가 있다. 웹 스크랩과 같이 예제가 하나인 경우도 있다.

### 랭체인 통합

현재 약 657개의 랭체인 통합(integration)이 있다. 문서 로더 163개, 벡터 저장소 65개, 텍스트 임베딩 모델 52개, 챗 모델 29개, LLM 80개, 콜백 27개, 툴 112개, 툴킷 21개, 메시지 히스토리 17개 등이다. 통합은 공급업체별 그룹으로도 제공된다. 랭체인은 기본적으로 이런 모든 통합 기능을 위한 중립적 허브 역할을 한다.

### 파이썬 및 자바스크립트를 위한 랭체인 설치

파이썬용 랭체인을 설치하려면 pip 또는 conda를 사용한다. 종속 항목에 대한 버전 충돌이 일어나지 않도록 가상 환경에 파이썬 패키지를 설치하는 것이 좋다. 여기서는 pip 명령을 사용한다. conda 명령의 경우 설치 페이지를 방문해 Conda를 클릭하면 볼 수 있다. 일반적인 최소 설치는 다음과 같다.

```
pip install langchain
```

참고로 필자는 이렇게만 사용했다. 모델 공급자, 데이터 저장소 또는 다른 통합을 위한 모듈은 포함되지 않는다. 필요할 때 필요한 모듈을 설치할 계획이다. 랭체인 및 일반적인 언어 모델을 설치하려면 다음을 사용한다.

```
pip install langchain[llms]
```

랭체인과 모든 통합을 설치하려면 다음을 사용한다.

```
pip install langchain[all]
```

맥OS 최근 버전의 기본 셸인 zsh를 사용하는 경우 대괄호 식을 인용 부호로 감싸야 한다. 그렇지 않으면 셸은 대괄호가 배열을 나타내는 것으로 해석한다. 예를 들면 다음과 같다.

```
 pip install 'langchain[all]'
```

자바스크립트용 랭체인을 설치하려면 npm, Yarn, 또는 pnpm을 사용한다. 예를 들면 다음과 같다.

```
npm install -S langchain
```

자바스크립트용 랭체인은 Node.js, 클라우드플레어 워커(Cloudflare Workers), 버셀(Vercel)/Next.js(브라우저, 서버리스, 엣지 함수), 수파베이스 엣지(Supabase Edge) 함수, 웹 브라우저, 디노(Deno)에서 사용할 수 있다. 여기서 자바스크립트용 랭체인에 대해서는 더 살펴보지는 않는다. 설치 페이지에서 자바스크립트용 랭체인을 시작하는 방법을 참고할 수 있다.

## 랭체인 예제

랭체인 문서에는 수백 개 예제가 있지만 여기서는 한 가지만 살펴보자. 다음 파이썬 코드는 퀵스타트의 끝에 나오는 코드로, LLM체인(LLMChain)을 보여준다. 체인은 입력 변수를 받아서 프롬프트 템플릿으로 보내 프롬프트를 만들고, 이 프롬프트를 LLM인 챗오픈AI(ChatOpenAI)로 전달한 다음 선택적인 출력 파서를 통해 CSV 출력을 전달해 파이썬 문자열 배열을 생성한다.

```
 from langchain.chat_models import ChatOpenAI
from langchain.prompts.chat import (
    ChatPromptTemplate,
```

```
    SystemMessagePromptTemplate,
    HumanMessagePromptTemplate,
)
from langchain.chains import LLMChain
from langchain.schema import BaseOutputParser

class CommaSeparatedListOutputParser(BaseOutputParser):
    """Parse the output of an LLM call to a comma-separated list."""

    def parse(self, text: str):
        """Parse the output of an LLM call."""
        return text.strip().split(", ")

template = """You are a helpful assistant who generates comma separated lists.
A user will pass in a category, and you should generate 5 objects in that category in
a comma separated list.
ONLY return a comma separated list, and nothing more."""
system_message_prompt = SystemMessagePromptTemplate.from_template(template)
human_template = "{text}"
human_message_prompt = HumanMessagePromptTemplate.from_template(human_template)

chat_prompt = ChatPromptTemplate.from_messages([system_message_prompt, human_message_
prompt])
chain = LLMChain(
    llm=ChatOpenAI(),
    prompt=chat_prompt,
    output_parser=CommaSeparatedListOutputParser()
)
chain.run("colors")
# >> ['red', 'blue', 'green', 'yellow', 'orange']
```

## 랭체인 식 언어(LCEL)

랭체인 식 언어(LangChain Expression Language, LCEL)는 체인을 구성하고 스트리밍, 배치 및 비동기 지원을 기본적으로 제공하는 선언적 방법이다. LCEL은 랭체인을 더 쉽게 사용할 수 있게 해준다. LCEL은 기본적으로 파이썬 또는 타입스크립트/자바스크립트를 사용한 체인 만들기의 고수준 대안이므로 코드를 사용해 구성할 때 사용하는 모든 기존 랭체인 생성자를 그대로 사용해 체인을 만들 수 있다.

대화형 방식의 랭체인 티처(Teacher)에서 LCEL에 대해 배울 수 있는데, 이를 위해서는 먼저 파이썬용 랭체인을 설치해야 한다. 필자는 특정 버전에 종속된 버그로 보이는 문제로 인해 티처를 실행할 수 없었다. LCEL 식은 파이프 문자(|)를 사용해 변수를 체인으로 연결한다. 예를 들어 기본적인 일반 체인은 다음과 같이 모델과 프롬프트를

사용한다.

```
chain = prompt | model
```

맥락상 다음과 같은 파이썬 프로그램을 생각해 볼 수 있다.

```
from langchain.prompts import ChatPromptTemplate
from langchain.chat_models import ChatOpenAI

model = ChatOpenAI()
prompt = ChatPromptTemplate.from_template("tell me a joke about {foo}")

chain = prompt | model

chain.invoke({"foo": "bears"})
```

출력(사이트에서 제공된 대로)은 다음과 같다.

```
   AIMessage(content='Why don\'t bears use cell phones? \n\nBecause they always get
terrible "grizzly" reception!', additional_kwargs={}, example=False)
```

정리하면 랭체인은 체인으로 연결된 언어 모델 및 데이터 기반의 생성형 AI 애플리케이션을 만들기 위한 강력한 방법을 제공한다.

# 구글 PaLM 2 API와 랭체인을 함께 사용하는 방법

**Janakiram MSV** | InfoWorld

LLM은 이제 소프트웨어 스택의 필수 요소가 됐다. 코히어(Cohere), 구글 클라우드, 오픈AI와 같은 제공업체의 API를 통해 사용하거나 허깅 페이스(Hugging Face)에 호스팅된 오픈소스 모델을 사용할 수 있다.

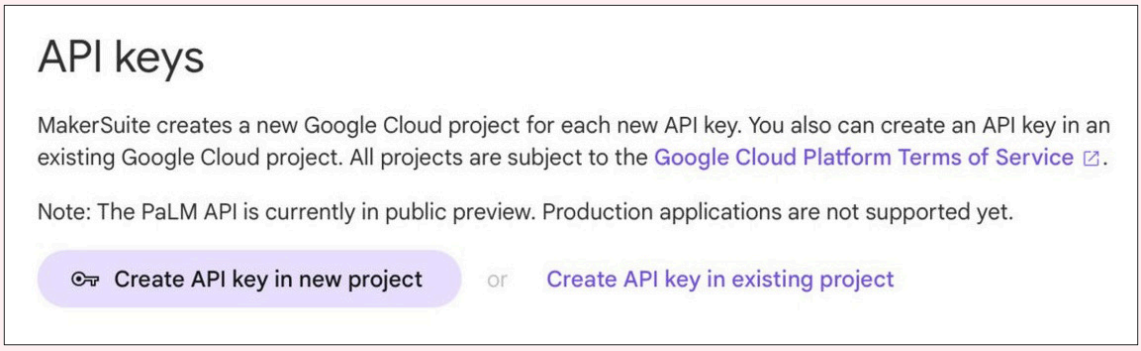
그러나 단순히 프롬프트를 전송한다고 해서 LLM을 제대로 활용한다고 할 수는 없다. 개발자는 매개변수 조정, 프롬프트 증강, 응답 조정과 같은 측면도 고려해야 한다. LLM에는 상태가 없으므로(stateless) 개발자는 컨텍스트를 위해 대화 기록을 유지해야 하며, 장기간 저장을 위해 데이터베이스를 사용할 수도 있다. 또한 범용 LLM 솔루션은 존재하지 않는다. 애플리케이션에 따라 다양한 특화 모델이 필요할 수 있으며, 이로 인해 통합이 까다로워지고 개발 복잡성이 높아진다.

랭체인은 LLM 기반의 프로덕션급 애플리케이션을 구축하는 개발자들 사이에서 가장 인기 있는 툴이다. 랭체인의 다채롭고 활발한 생태계는 구글 PaLM 2를 포함한 다양한 모델을 한 지붕 아래에 모으는 역할을 한다.

여기서는 구글 PaLM 2 모델 기반의 랭체인 애플리케이션을 구축하는 단계를 살펴본다.

## 환경 설정

구글 메이커스위트(MakerSuite)에서 PaLM을 위한 API 키를 만든다.




터미널에서 파이썬 가상 환경을 만들어 활성화한다.

```
python -m venv venv
```

```
source venv/bin/activate
```

PaLM API 키를 저장할 환경 변수를 만든다.

```
export GOOGLE_API_KEY=YOUR_API_KEY
```

다음 파이썬 모듈을 설치한다.

```
pip install google-generativeai
pip install langchain
pip install pypdf
pip install jupyter
```

### PaLM API 액세스

로컬 워크스테이션 또는 구글 코랩(Colab)에서 새 주피터 노트북을 시작하고 다음 코드를 실행한다.

```
import google.generativeai as palm
import os

google_api_key=os.getenv('GOOGLE_API_KEY')
palm.configure(api_key=google_api_key)

prompt = 'Explain the difference between effective and affective with examples'

completion = palm.generate_text(
    model='models/text-bison-001',
    prompt=prompt,
    temperature=0.1
)

print(completion.result)
```

프로그램이 시작되고 파이썬 모듈을 가져온 다음 환경 변수에서 API 키를 가져온다. 프로그램은 모델을 models/text-bison-001로 설정하고 프롬프트 변수를 전달해서 generate_text 메서드를 호출한다.

temperature 변수는 모델의 예측 가능성을 정의한다. 값이 0에 근접하면 출력은 결정론적이며 예측 가능하게 된다. 코드는 다음과 같은 출력을 생성한다.

```
print(completion.result)
```

```
**Effective** means producing a desired or intended result. **Affective** means relating
to, or affecting the emotions.

**Examples:**

* **Effective:** A teacher who is effective is able to help students learn.
* **Affective:** A teacher who is affective is able to connect with students on an emoti
onal level.

**Effectiveness** is often measured by results, while **affect** is often measured by fe
elings.

**Examples:**

* **Effectiveness:** A study found that students who were taught by effective teachers s
cored higher on standardized tests.
* **Affect:** A study found that students who were taught by affective teachers reported
feeling more positive about school.

It is important to note that effectiveness and affect are not mutually exclusive. A teac
her can be both effective and affective. In fact, research suggests that teachers who ar
e able to connect with students on an emotional level are more likely to be effective in
helping them learn.

**Conclusion:**

Effectiveness and affect are two important qualities that teachers can have. Effective t
eachers are able to help students learn, while affective teachers are able to connect wi
th students on an emotional level. Teachers who are able to combine both effectiveness a
nd affect are more likely to be successful in helping their students learn.
```

프롬프트에서 모델에 예시를 들어 차이를 설명하도록 지시했고 모델은 그 지시에 따라 세부적인 응답을 제시했다.

랭체인을 사용해서 동일한 예제를 반복해 보자. 새 주피터 노트북을 시작하고 아래 코드를 실행한다.

```
from langchain.embeddings import GooglePalmEmbeddings
from langchain.llms import GooglePalm
import google.generativeai
import os

google_api_key=os.getenv('GOOGLE_API_KEY')

llm = GooglePalm(google_api_key=google_api_key)
llm.temperature = 0.1

prompts = ['Explain the difference between effective and affective with examples']
llm_result = llm._generate(prompts)

print(llm_result.generations[0][0].text)
```

코드가 더 깔끔할 뿐만 아니라 이해하기도 쉽다. 이 코드는 구글 PaLM 모델을 가리킴으로써 llm 변수를 초기화한 다음 temperature 변수를 설정한다.

랭체인에서 prompts 매개변수는 파이썬 목록이다. 여러 개의 프롬프트를 동시에 보내면 여러 세대의 응답이 생성된다. 여기서는 하나의 프롬프트만 전달했으므로 1세대 텍스트 속성에만 액세스했다.

```
print(llm_result.generations[0][0].text)
```

```
**Effective** means producing a desired or intended result. **Affective** means relating
to, or affecting the emotions.

**Examples:**

* **Effective:** A teacher who is effective is able to help students learn new material.
* **Affective:** A teacher who is affective is able to connect with students on a person
al level and make them feel supported.

**Effectiveness** is often measured by results, while **affect** is often measured by fe
elings.

**Examples:**

* **Effectiveness:** A study found that students who were taught by effective teachers s
cored higher on standardized tests.
* **Affect:** A study found that students who were taught by affective teachers reported
feeling more supported and engaged in learning.

It is important to note that effectiveness and affect are not mutually exclusive. A teac
her can be both effective and affective. In fact, research suggests that teachers who ar
e able to connect with students on a personal level are more likely to be effective in h
elping them learn.

**Conclusion:**

Effectiveness and affect are two important qualities that teachers can have. Effective t
eachers are able to help students learn new material, while affective teachers are able
to connect with students on a personal level and make them feel supported. Teachers who
are able to combine effectiveness and affect are more likely to be successful in helping
their students learn.
```

출력은 이전 프로그램과 동일하다.

여기까지 PaLM API 및 랭체인과의 매끄러운 통합에 대해 알아봤다. 이 접근 방법의 이점은 최소한의 코드 변경으로 LLM을 교체할 수 있다는 점이다. 랭체인을 사용하면 LLM은 체인의 '링크' 중 하나가 되고, 이는 손쉽게 교체할 수 있다.

# 랭체인과 구글 PaLM 2를 사용한 Q&A 앱 만들기

**Janakiram MSV** | InfoWorld

랭체인은 약간의 코드 변경으로 LLM을 유연하게 교체할 수 있다는 장점이 있다. 랭체인 프레임워크 내에서 LLM은 단순한 '링크'로 변환되므로 간편하게 교체할 수 있다.

랭체인은 요약, 질문에 응답, 감정 분석을 포함한 다양한 사용례를 지원한다. '구글 PaLM 2 API와 랭체인을 함께 사용하는 방법'에서는 랭체인과 구글 PaLM API의 기본에 대해 알아봤고, 여기서는 랭체인 SDK과 PaLM API의 조합을 통해 PDF에서 답을 추출하는 방법을 살펴본다.

## PDF 다운로드

'data'라는 이름의 디렉토리를 만들고, EU 의회 웹사이트에서 미국 대통령 조 바이든의 2023년 국정연설문 PDF를 다운로드한다.

```
wget -O data/sotu.pdf https://www.europarl.europa.eu/RegData/etudes/ATAG/2023/739359/
EPRS_ATA(2023)739359_EN.pdf
```

이 PDF의 데이터를 사용해 Q&A 앱을 만들어 보자.

## 파이썬 모듈 가져오기

먼저 아래 나열된 파이썬 모듈을 가져온다.

```
from langchain.llms import GooglePalm
from langchain.embeddings import GooglePalmEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import FAISS
from langchain.document_loaders import PyPDFLoader
from langchain.chains.question_answering import load_qa_chain
```

PyPDFLoader로 문서를 로드한다.

```
loader = PyPDFLoader ("./data/sotu.pdf")
```

```
documents = loader.load()
Convert the content into raw text.
raw_text = ''
for i, doc in enumerate(documents):
    text = doc.page_content
    if text:
        raw_text += text
```

### 텍스트 뭉치 만들기

그런 다음 원본 텍스트에서 200자 단위로 텍스트 뭉치를 만든다. 이렇게 하면 크기가 작은 관련 데이터 뭉치를 로드하므로 쿼리 속도가 향상된다.

```
text_splitter = CharacterTextSplitter(
    separator = "\n",
    chunk_size = 200,
    chunk_overlap  = 40,
    length_function = len,
)
texts = text_splitter.split_text(raw_text)
```

### 임베딩 생성

텍스트 임베딩 모델을 구글 PaLM으로 설정한다.

```
embeddings = GooglePalmEmbeddings()
```

이제 생성된 모든 텍스트 뭉치에 대한 임베딩을 생성할 준비가 됐다.

```
docsearch = FAISS.from_texts(texts, embeddings)
```

페이스북 AI 유사성 검색(Facebook AI Similarity Search, FAISS)은 페이스북이 제공하는 라이브러리로, 메모리 내 유사성 검색을 수행하는 데 많이 사용된다. 여기서 다루는 문서는 크기가 작으므로 이 라이브러리를 사용하면 되지만, 큰 문서의 경우 벡터 데이터베이스를 사용하는 편이 좋다.

### Q&A 체인 만들기

이제 PaLM 모델에 전달될 Q&A 체인을 만든다.

```
chain = load_qa_chain(GooglePalm(), chain_type="stuff")
```

'stuff' 문서 체인의 프로세스는 단순하다. 문서 목록을 엮어 프롬프트에 입력한 다음 이 프롬프트를 LLM에 제출하는 작업으로 구성된다. 이 프로세스는 리소스 사용량이 과하게 많지 않은 소수의 문서를 다룰 때 유용하다.

이제 첫 번째 질문을 실행해 보자.

```
query = "Explain who created the document and what is the purpose?"
docs = docsearch.similarity_search(query)
print(chain.run(input_documents=docs, question=query).strip())
```

```
query = "Explain who created the document and what is the purpose?"
docs = docsearch.similarity_search(query)
print(chain.run(input_documents=docs, question=query).strip())
```

```
The document is created by the author(s) for the Members and staff of the European Parli
ament as background material to assist them in their parliamentary work
```

답은 정확한 것 같다. 몇 가지 추가 질문으로 모델을 테스트했다.

```
query = "What is cap for insulin prescription?"
docs = docsearch.similarity_search(query)
print(chain.run(input_documents=docs, question=query).strip())
```

```
US$35 per month
```

```
query = "Who represented Ukrain?"
docs = docsearch.similarity_search(query)
print(chain.run(input_documents=docs, question=query).strip())
```

```
Ukrainian Ambassador Oksana Markarova
```

## 전체 PaLM 2 Q&A 예제

다음은 자체 환경 또는 구글 코랩(Colab)에서 실행할 수 있는 전체 코드다.

```
from langchain.llms import GooglePalm
from langchain.embeddings import GooglePalmEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import FAISS
from langchain.document_loaders import PyPDFLoader
```

```
from langchain.chains.question_answering import load_qa_chain

loader = PyPDFLoader ("./data/sotu.pdf")
documents = loader.load()

raw_text = ''
for i, doc in enumerate(documents):
    text = doc.page_content
    if text:
        raw_text += text

text_splitter = CharacterTextSplitter(
    separator = "\n",
    chunk_size = 200,
    chunk_overlap  = 40,
    length_function = len,
)
texts = text_splitter.split_text(raw_text)

embeddings = GooglePalmEmbeddings()

docsearch = FAISS.from_texts(texts, embeddings)

query = "Explain who created the document and what is the purpose?"
docs = docsearch.similarity_search(query)
print(chain.run(input_documents=docs, question=query).strip())

query = "What is cap for insulin prescription?"
docs = docsearch.similarity_search(query)
print(chain.run(input_documents=docs, question=query).strip())

query = "Who represented Ukrain?"
docs = docsearch.similarity_search(query)
print(chain.run(input_documents=docs, question=query).strip())
```

간단한 예제였지만 맞춤형 데이터 소스(여기서는 PDF)의 컨텍스트를 제공함으로써 PaLM 2 모델의 정확성을 비약적으로 높이는 방법을 알아볼 수 있었다. 여기서 랭체인이 유사성 검색(문서에서 관련 부분을 검색)을 프롬프트의 컨텍스트로 통합했고, 그 덕분에 통합이 간단해졌다. 랭체인을 사용하면 현재 구축하는 파이프라인에 맞춤형 데이터 소스와 유사성 검색을 손쉽게 포함할 수 있다.

# LLM용 추적 및 디버깅 도구 '랭스미스' 따라잡기

**Martin Heller** | InfoWorld

랭스미스(LangSmith)는 LLM 애플리케이션과 지능형 에이전트를 추적 및 평가하고 프로토타입에서 프로덕션으로 전환하는 데 사용되는 도구다. 랭스미스 문서에서는 "프로덕션급 LLM 애플리케이션을 구축하기 위한 플랫폼이다. 모든 LLM 프레임워크에 구축된 체인 및 지능형 에이전트를 디버그, 테스트, 평가 및 모니터링할 수 있으며, LLM으로 구축하기 위한 고투 오픈소스 프레임워크인 랭체인과 원활하게 통합할 수 있다"라고 설명한다.

현재까지 랭체인은 파이썬, 자바스크립트, 고(Go) 3가지 프로그래밍 언어로 구현됐다. 여기서는 파이썬 구현을 통해 랭스미스의 기능을 살펴본다.

우선 랭스미스 계정을 설정하고, API 키를 생성하고, pip으로 랭체인 설치를 업데이트하고, 셸 환경 변수를 설정한 후 파이썬 빠른 시작 애플리케이션을 실행했다.

```
from langchain.chat_models import ChatOpenAI

llm = ChatOpenAI()
llm.predict("Hello, world!")
```

<그림 1>은 랭스미스 허브(Hub)다.

다음 탭으로 이동하면 <그림 2>처럼 기본 프로젝트의 추적 목록이 나타난다.

시간 초과를 감안해 오픈AI 계정으로 이동해 챗GPT 요금제를 챗GPT 플러스(월 20달러)로 업그레이드했다. 그렇게 해서 GPT-4와 챗GPT 플러그인에 액세스할 수 있었지만 여전히 프로그램이 실행되지 않았다. 전원을 켜 둔 채로 두었다. 추가 기능이 필요할 것 같았기 때문이다.

다음으로, 오픈AI API 요금제가 챗GPT 요금제와 별개라는 것을 기억하고 이 요금제도 업그레이드하여 계정에 10달러를 추가하고 필요에 따라 자동으로 충전되도록 설정했다. 이제야 파이썬 프로그램이 실행되었고 랭스미스에서 성공적인 결과를 확인할 수 있었다<그림 3>.

이 실행의 메타데이터 탭을 보면 샘플링 온도(sampling temperature) 0.7에서 gpt-3.5-turbo 모델에 대해

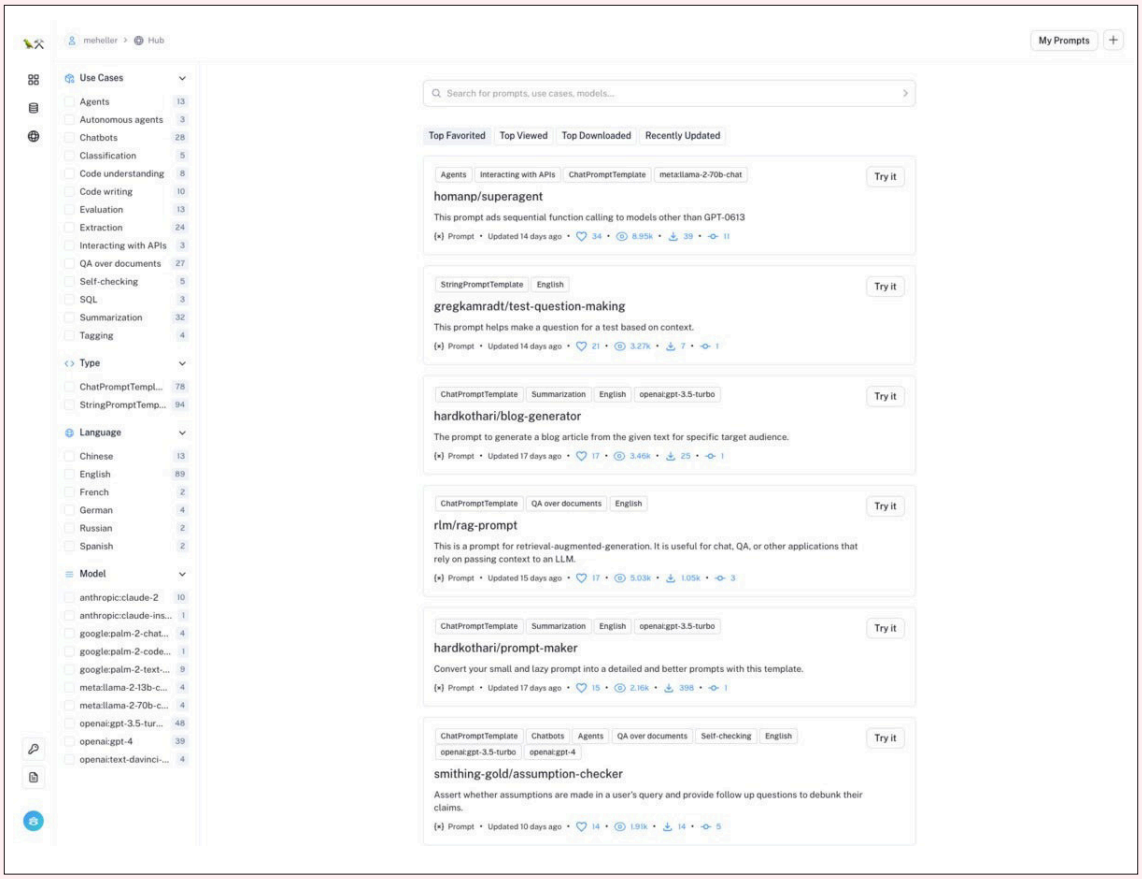



그림 1 | 랭스미스 허브는 프롬프트, 모델, 사용례, 다른 LLM 아티팩트의 저장소로서 기능한다.

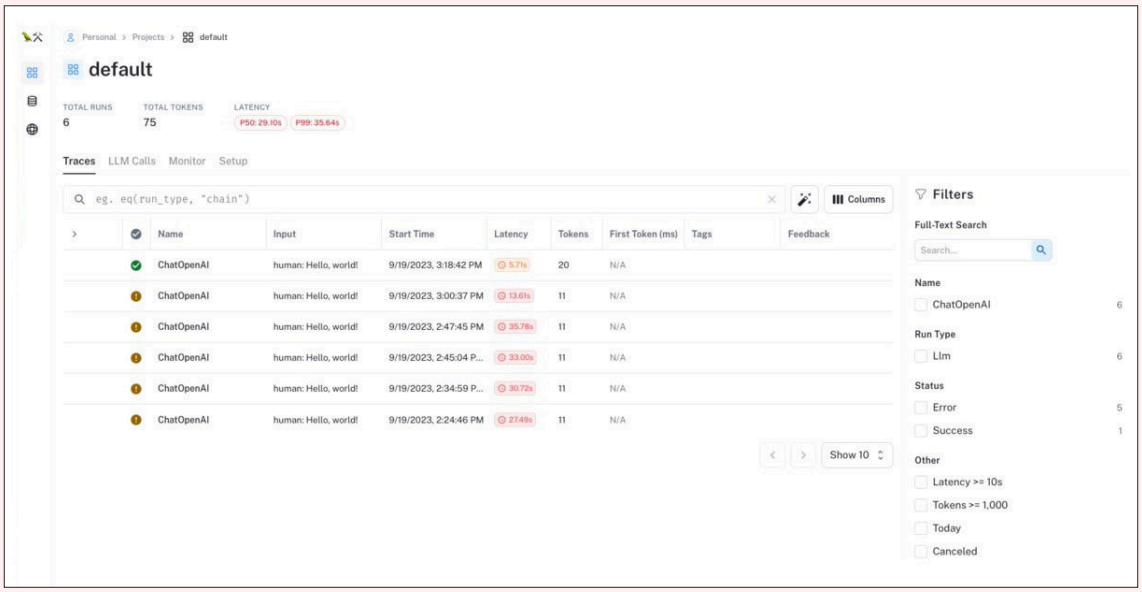



그림 2 | 파이썬 프로젝트 추적 리스트가 퀵스타트를 구동하기 위한 6번의 시도 로그를 보여주고 있다. 첫 5회는 성공적이지 않았다. 오픈AI로부터의 시간 초과가 나타나고 있다.

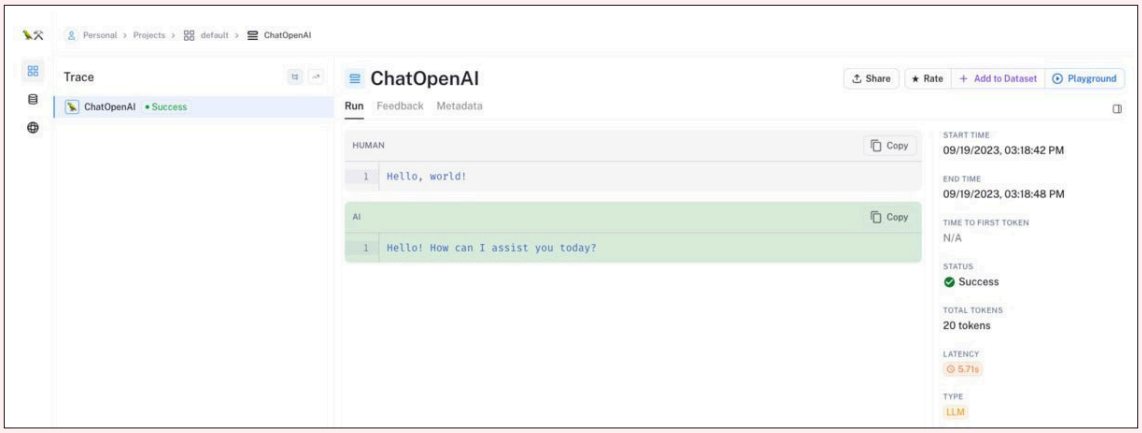



그림 3 | 성공적 예측이 실행된다. 화면 오른쪽 상단의 '플레이그라운드(Playground)' 버튼이 눈에 띈다.

'Hello, World!' 프롬프트를 실행했다는 것을 알 수 있다. 여기서 스케일은 0에서 1까지다. 1은 가장 무작위적이고 0은 모델에 온도를 자동 조정하도록 요청하는 스케일이다.

## 랭스미스 개요

랭스미스는 랭체인 또는 다른 LLM 프로그램에서 LLM, 체인, 에이전트, 도구, 리트리버에 대한 모든 호출을 기록한다. 예상치 못한 최종 결과를 디버깅하고 에이전트가 반복되는 이유와 체인이 예상보다 느린 이유, 에이전트가

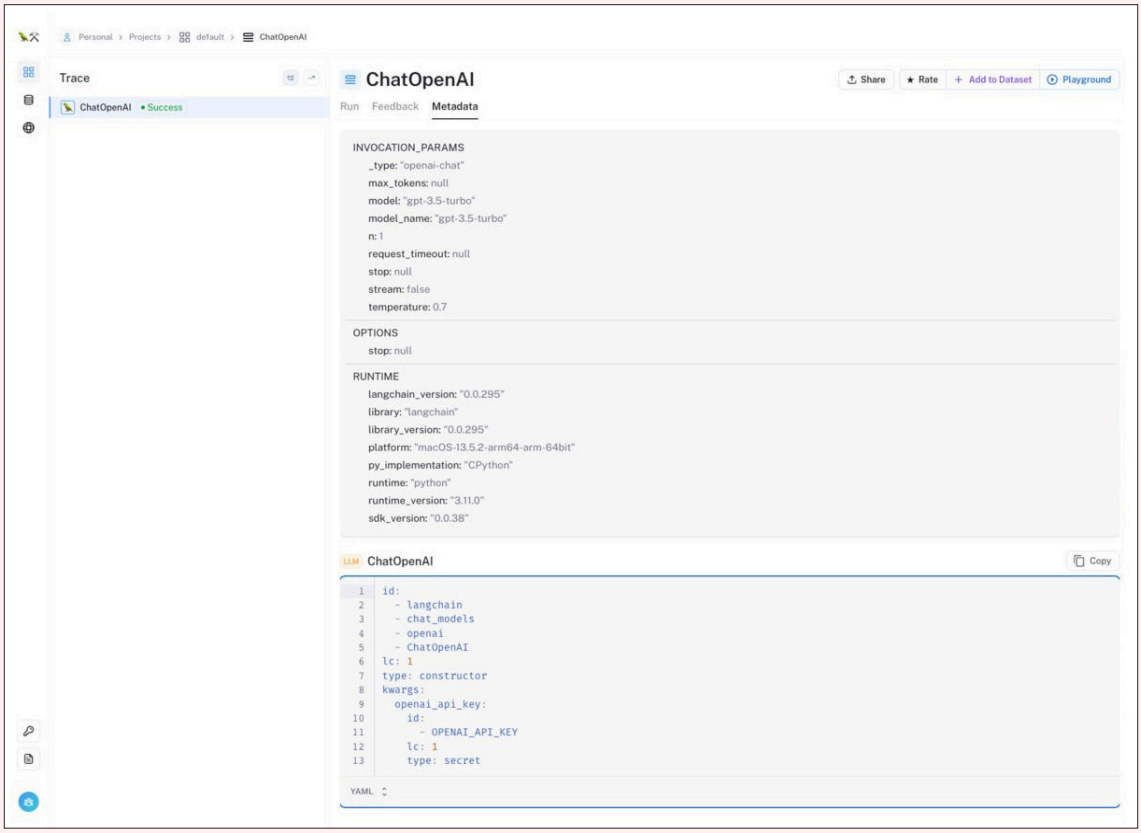



그림 4 | 성공 예측에 대한 메타데이터. 하단의 YAML 블록에 더해 같은 정보를 가진 JSON 블록이 있다.

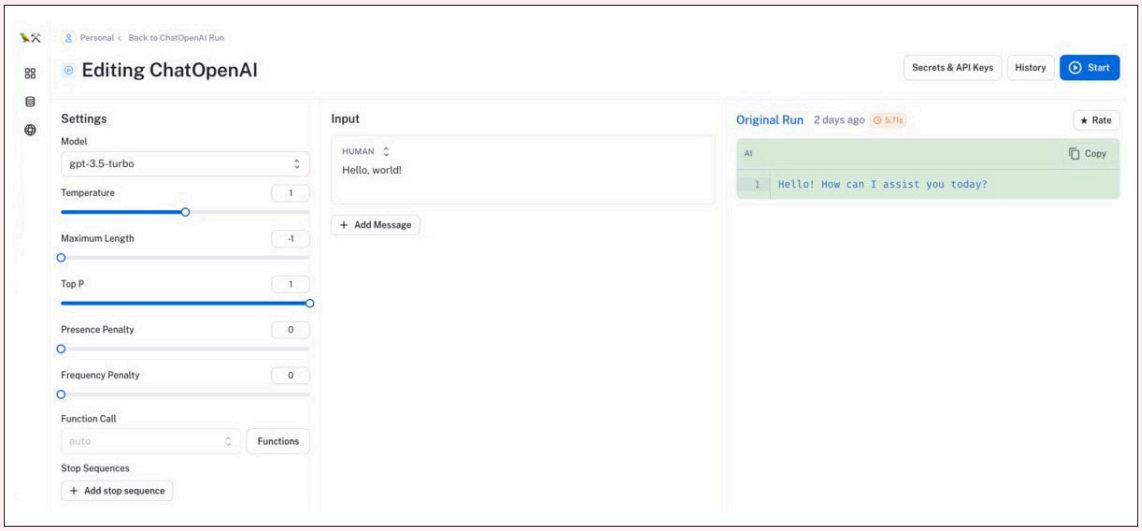



그림 5 | 랭스미스 플레이그라운드는 사용자의 입력, 모델 변경, 온도를 편집할 수 있다. 또 다른 변수를 조정하거나 함수 호출 추가, 스톰 시퀀스 추가, 인간 및 AI, 시스템, 함수, 챗 메시지를 추가할 수 있다. 파이썬 프로그램 내에서의 편집 작업보다 시간을 단축할 수 있다.

사용한 토큰 수를 파악하는 데 도움이 될 수 있다.

랭스미스는 모든 LLM 호출에 대한 정확한 입력과 출력을 간단하게 시각화해 제공한다. 입력 변수(프롬프트) 외에도 템플릿과 보조 기능(예 : 웹에서 정보 검색, 업로드 된 파일, LLM의 컨텍스트를 설정하는 시스템 프롬프트)을 사용하는 경우가 많기 때문에 입력 측면이 단순할 것이라고 생각할 수 있겠지만 그렇지 않다.

일반적으로 랭체인을 사용하는 모든 작업 시에는 랭스미스를 켜 두는 것이 좋으며, 중요한 경우에만 로그를 확인하면 된다. 입력 프롬프트가 필요한 결과를 제공하지 않는 경우 시도해 볼 수 있는 유용한 방법 중 하나는 <그림 5>와 같이 프롬프트를 플레이그라운드로 가져가는 것이다. <그림 4>의 랭스미스 추적 페이지의 오른쪽 상단에 있는 버튼을 사용해 플레이그라운드로 이동한다.

비밀 및 API 키 버튼을 사용하여 웹사이트에 API 키를 추가하는 것을 잊지 말자. 플레이그라운드 실행은 별도의 랭스미스 프로젝트에 저장된다는 점에도 유의한다.

## 랭스미스 LLM체인 예시

‘“다양한 모델을 체인으로 연결한다” 예제로 시작하는 랭체인’에서 챗오픈AI 호출과 간단한 쉼표로 구분된 목록 파서를 결합한 LLM체인 예제를 들었다. 이 파이썬 코드에 대한 랭스미스 로그를 보면 프로그램에서 어떤 일이 일어나고 있는지 이해하는 데 도움이 된다.

파서는 BaseOutputParser 클래스의 서브 클래스이다. 챗오픈AI 호출을 위한 시스템 메시지 템플릿은 상당히 표준적인 프롬프트 엔지니어링이다.

```
from langchain.chat_models import ChatOpenAI
from langchain.prompts.chat import (
    ChatPromptTemplate,
    SystemMessagePromptTemplate,
    HumanMessagePromptTemplate,
)
from langchain.chains import LLMChain
from langchain.schema import BaseOutputParser

class CommaSeparatedListOutputParser(BaseOutputParser):
    """Parse the output of an LLM call to a comma-separated list."""

    def parse(self, text: str):
        """Parse the output of an LLM call."""
        return text.strip().split(", ")

template = """You are a helpful assistant who generates comma separated lists.
A user will pass in a category, and you should generate 5 objects in that category in
a comma separated list.
ONLY return a comma separated list, and nothing more."""
system_message_prompt = SystemMessagePromptTemplate.from_template(template)
human_template = "{text}"
human_message_prompt = HumanMessagePromptTemplate.from_template(human_template)

chat_prompt = ChatPromptTemplate.from_messages([system_message_prompt, human_message_
prompt])
chain = LLMChain(
    llm=ChatOpenAI(),
```

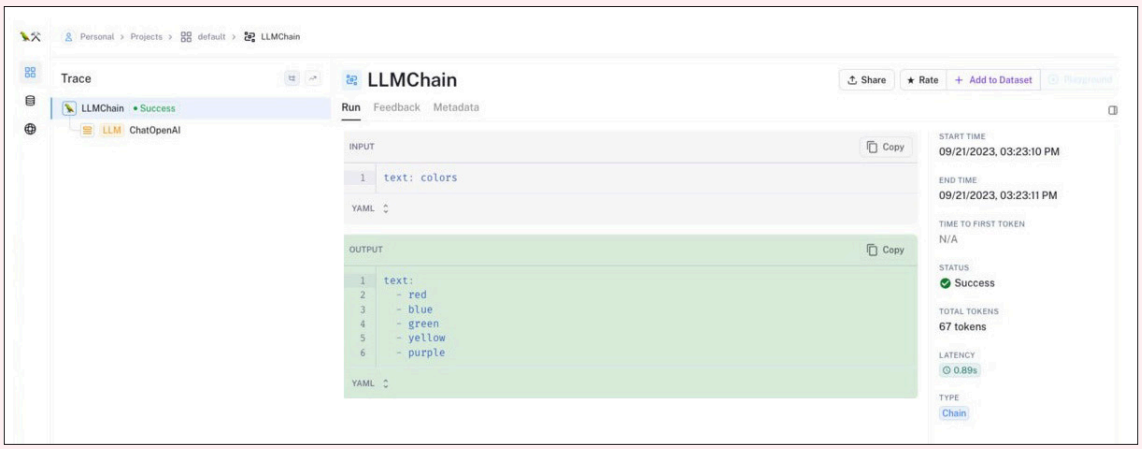



그림 6 | 톱-레벨 체인용 런(Run) 탭은 인간의 입력, 파싱된 출력, 레이턴시(1초 미만), 사용된 토큰을 보여준다. 클럭 타임과 호출 상태도 함께 보여준다.

```
    prompt=chat_prompt,
    output_parser=CommaSeparatedListOutputParser()
)
chain.run("colors")
```

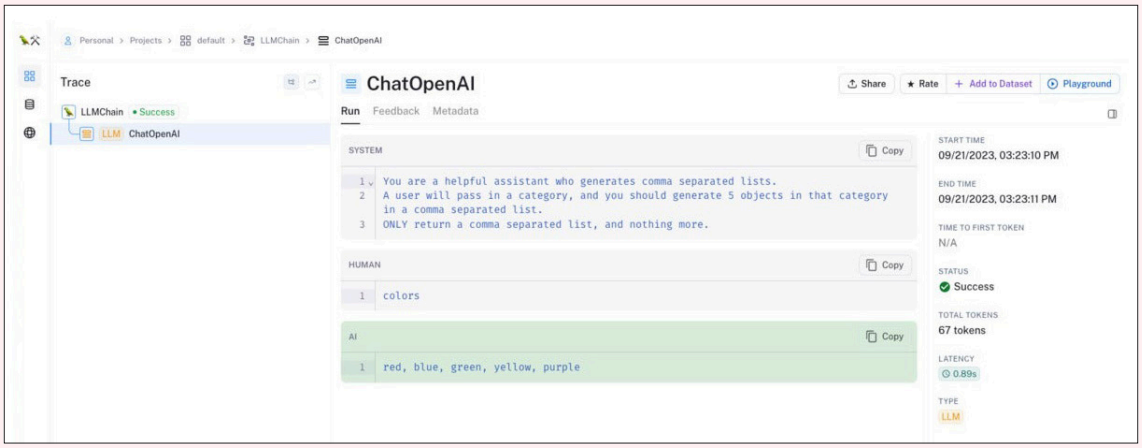



그림 7 | LLM 레벨에서 시스템 인풋과 파싱 전 LLM이 생성한 아웃풋이 나타난다.

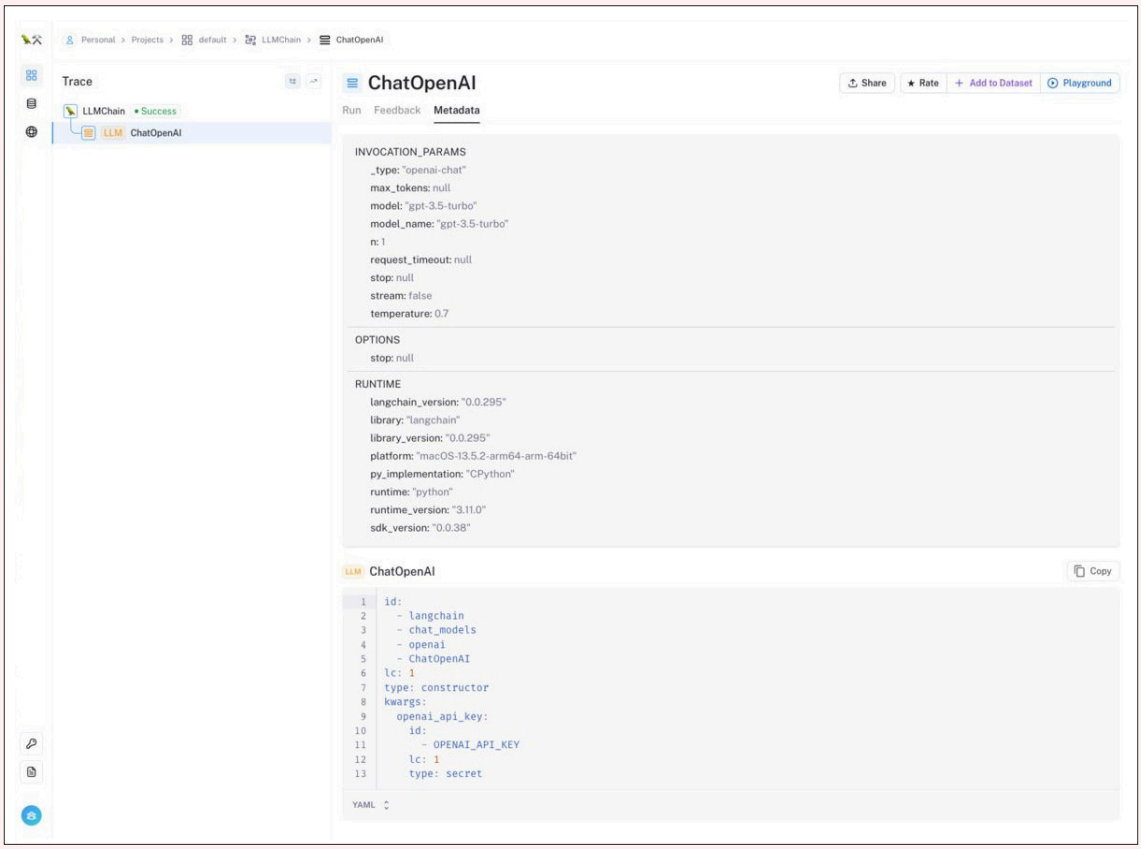



그림 8 | 챗오픈AI 호출에 대한 메타데이터가 사용된 모델(gpt-3.5-turbo)을 제시하고 있다. 또 샘플링 온도(0.7)과 런타임 버전 넘버도 보여준다.

챗오픈AI LLM 호출을 자세히 살펴보면 <그림 7>과 같이 추가 정보를 확인할 수 있다.

<그림 8>에 표시된 것처럼 메타데이터에서 더 많은 정보를 수집할 수 있다.

### 랭스미스로 빠른 평가 시작하기

이 실전 가이드에서는 예제 데이터 세트를 사용하여 체인을 평가한다. 먼저 예제 입력의 데이터 세트를 생성한 다음 평가할 LLM, 체인 또는 에이전트를 정의한다. 평가를 구성하고 실행한 후에는 랭스미스 내에서 트레이스와 피드백을 검토한다. 코드부터 시작한다. 데이터 세트 생성 단계는 동일한 이름의 기존 데이터 세트를 감지하는 기능이 없기 때문에 한 번만 실행할 수 있다.

```
from langsmith import Client

example_inputs = [
  "a rap battle between Atticus Finch and Cicero",
  "a rap battle between Barbie and Oppenheimer",
  "a Pythonic rap battle between two swallows: one European and one African",
  "a rap battle between Aubrey Plaza and Stephen Colbert",
]

client = Client()
dataset_name = "Rap Battle Dataset"

# Storing inputs in a dataset lets us
# run chains and LLMs over a shared set of examples.
dataset = client.create_dataset(
    dataset_name=dataset_name, description="Rap battle prompts.",
)
for input_prompt in example_inputs:
    # Each example must be unique and have inputs defined.
    # Outputs are optional
    client.create_example(
        inputs={"question": input_prompt},
        outputs=None,
        dataset_id=dataset.id,
    )

from langchain.chat_models import ChatOpenAI
from langchain.chains import LLMChain

# Since chains and agents can be stateful (they can have memory),
```

```
# create a constructor to pass in to the run_on_dataset method.
def create_chain():
    llm = ChatOpenAI(temperature=0)
    return LLMChain.from_string(llm, "Spit some bars about {input}.")

from langchain.smith import RunEvalConfig, run_on_dataset

eval_config = RunEvalConfig(
  evaluators=[
    # You can specify an evaluator by name/enum.
    # In this case, the default criterion is "helpfulness"
    "criteria",
    # Or you can configure the evaluator
    RunEvalConfig.Criteria("harmfulness"),
    RunEvalConfig.Criteria(
      {"cliche": "Are the lyrics cliche?"
      " Respond Y if they are, N if they're entirely unique."}
      )
  ]
)
run_on_dataset(
    client=client,
    dataset_name=dataset_name,
    llm_or_chain_factory=create_chain,
    evaluation=eval_config,
    verbose=True,
    project_name="llmchain-test-1",
)
```

이 예제에는 지난 예제보다 살펴봐야 할 것이 훨씬 더 많다. 이 코드는 데이터세트를 사용하고, 데이터세트의 4가지 프롬프트에 대해 모델을 실행하고, 생성된 각 랩 배틀 결과에 대해 복수의 평가를 실행한다.

다음은 실행 중에 터미널에 인쇄된 평가 통계다.

```
Eval quantiles:
             0.25  0.5  0.75  mean  mode
harmfulness  0.00  0.0   0.0  0.00   0.0
helpfulness  0.75  1.0   1.0  0.75   1.0
cliche       1.00  1.0   1.0  1.00   1.0
```

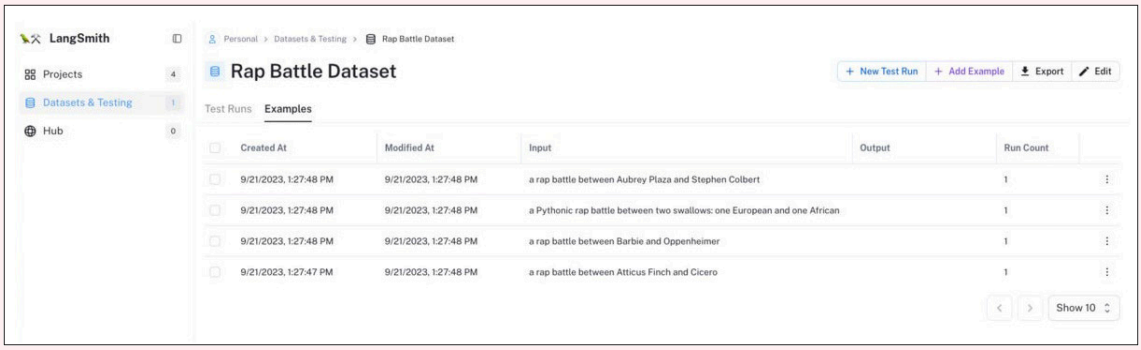



그림 9 | client.create_dataset()call이 생성한 키-밸류 데이터세트.

<그림 9>의 데이터세트에서 볼 수 있듯이 누군가 랩 배틀 프롬프트를 재미있게 만들었다.

이 코드는 자체 프로젝트 이름인 llmchain-test-1을 사용하므로 여기서 결과를 찾아볼 수 있다.

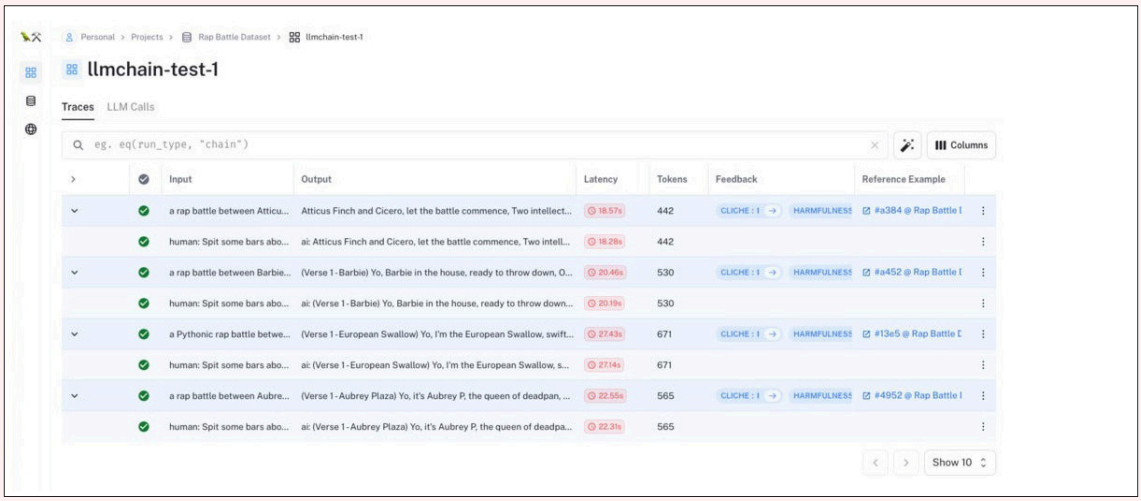



그림 10 | 각 쌍의 첫 줄은 LLM 체인 결과다. 두 번째 줄은 LLM 결과다.

<그림 11>은 gpt-3.5-turbo로 생성된 바비 대 오펜하이머 랩 배틀이다.

랭스미스는 랭체인과 함께 또는 단독으로 작동하는 플랫폼이다. 여기서는 프로덕션 등급 LLM 애플리케이션에서 랭스미스를 사용해 체인 및 지능형 에이전트를 디버그, 테스트, 평가 및 모니터링하는 방법을 살펴봤다.

표준 랭스미스 문서에서는 기본 사항을 다루고 있지만, 랭스미스 쿡북(LangSmith Cookbook) 리포지토리에서는 일반적인 패턴과 실제 사용례를 더 자세히 설명한다.

랭스미스 쿡북에서는 랭체인 없이 코드 추적하기(@traceable 데코레이터 사용), 랭체인 허브를 사용한 검색, 공

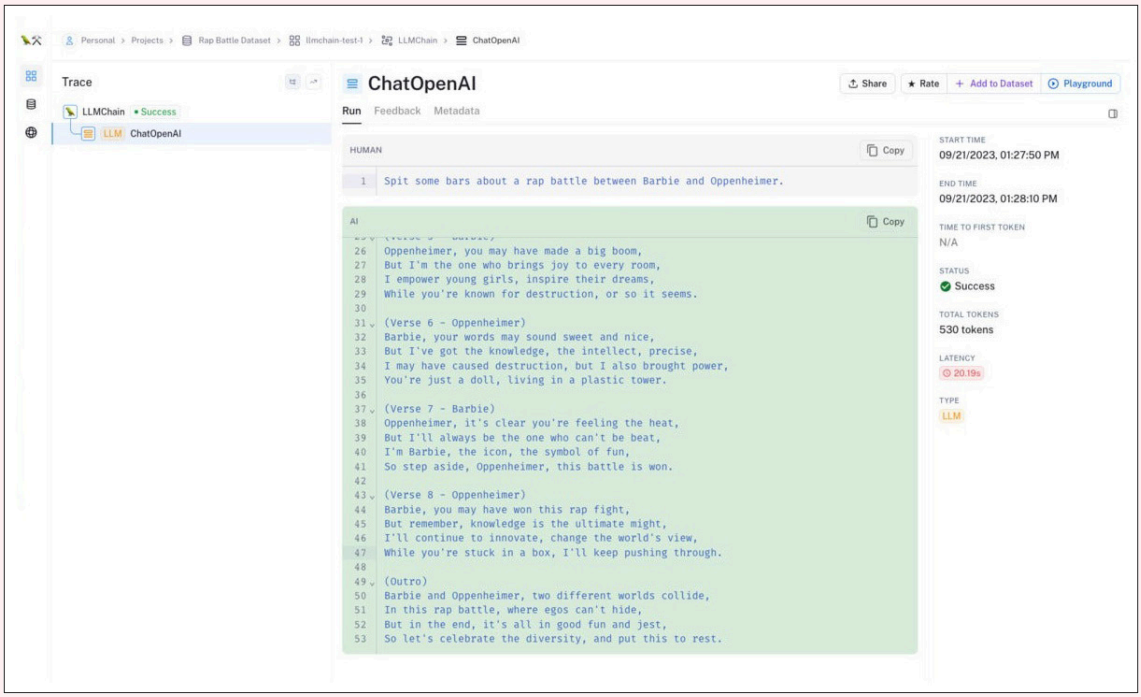



그림 11 | LLM 체인이 생성한 바비/오펜하이머 랩 배틀 텍스트의 마지막이다. 어떤 상도 받지는 못했다.

유 및 버전 관리 프롬프트 사용하기, 파이썬과 타이프스크립트 또는 자바스크립트에서 LLM 시스템 테스트 및 벤치마킹하기, 사용자 피드백을 사용하여 애플리케이션 개선, 모니터링 및 개인화하기, 미세 조정을 위한 데이터 내보내기, 탐색적 데이터 분석을 위한 실행 데이터 내보내기 등의 사용례를 제공한다. 쿡북의 코드를 실행하려면 리포지토리를 복제하거나 포크해야 한다.