LangChain을 이용한
LLM 어플리케이션 만들기

LangChain

LLM 어플리케이션의 쉬운 개발

- 여러 LLM을 통합할 수 있는 추상화 레이어
- Open/Closed LLM 모두, 동일한 인터페이스에서 사용

LLM Orchestration 프레임워크

- 모델, 프롬프트 템플릿, 출력 파서, 데이터로더,
  메모리, 외부 툴, 에이전트 등의 기능 지원

→ 복잡한 프로그래밍 없이도, LLM 기반 작업을 쉽게 구현하고 연결

대표적 LangChain 모듈

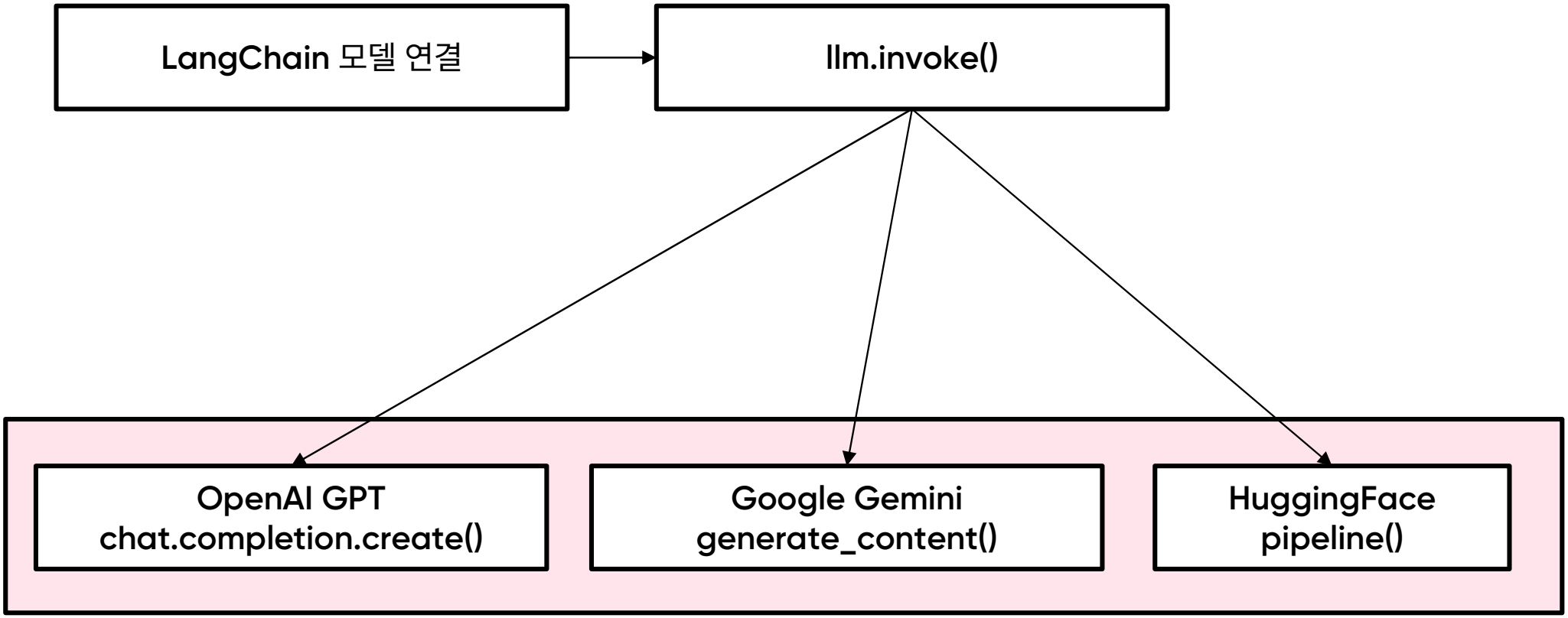
Prompt Template

- 전체 프롬프트의 템플릿을 사전에 구성하고, 입력 변수만 입력하여 체인 실행
- Ex) {topic}에 대해 설명해줘!

Output Parser

- LLM의 출력 형태를 변환 (string, json, pydantic class, ...)
- Ex) json 파싱 후 데이터베이스 저장, 전처리 결과 반환

예시) 모델 추론 실행하기


LangChain 모델 연결
llm.invoke()
OpenAI GPT
chat.completion.create()
Google Gemini
generate_content()
HuggingFace
pipeline()

예시) LangChain ChatHuggingFace

오픈 모델들의 다양한 템플릿 연동

Hugging Face

- Llama 3.3-Instruct

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are a helpful assistant.
<|eot_id|><|start_header_id|>user<|end_header_id|>
Hello! <|eot_id|>
```

Meta

- Qwen 2.5-Instruct

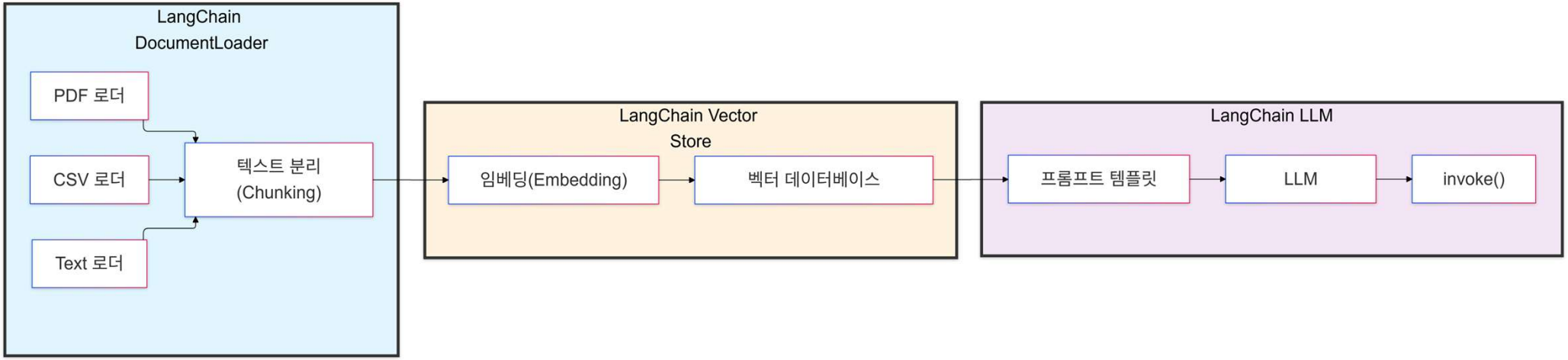
```
<|im_start|>system
You are a helpful assistant.
<|im_end|>
<|im_start|>user
Hello!
<|im_end|>
```

Qwen2.5

LangChain의 높은 추상화: 다양한 모듈과의 연동

예시) 검색 증강 생성(RAG)

- 서브모듈을 직접 만들지 않고도 다양한 기능 구현 가능



LangChain Expression Language (LCEL)

랭체인의 구성 문법

- 파이프(|)를 통해 모듈을 체인으로 연결
- Chain = Prompt | LLM | Parser() → 입력 후 출력 결과 변환하여 전달
- Chain.invoke()를 통해 한 번에 실행 가능

복잡한 체인 구성

- LLM을 여러 번 실행: Prompt | LLM | Parser() | Prompt | LLM | ...
- 외부 함수와의 연결: Prompt | LLM | Parser() | Function | ...

이렇게 편한 LangChain, 개선점은?

중간 상태의 명시적 관리가 어려움

- 파이프 구조는 직관적이나, 이전 단계를 보존하지 않음
- 조건부 로직이나, 중간 값 변화 등을 처리하기 까다로움

디버깅과 모니터링의 어려움

- 중간에 오류가 생기면 이를 추적하기 어려움
- 모니터링/테스팅을 위한 기능 부족

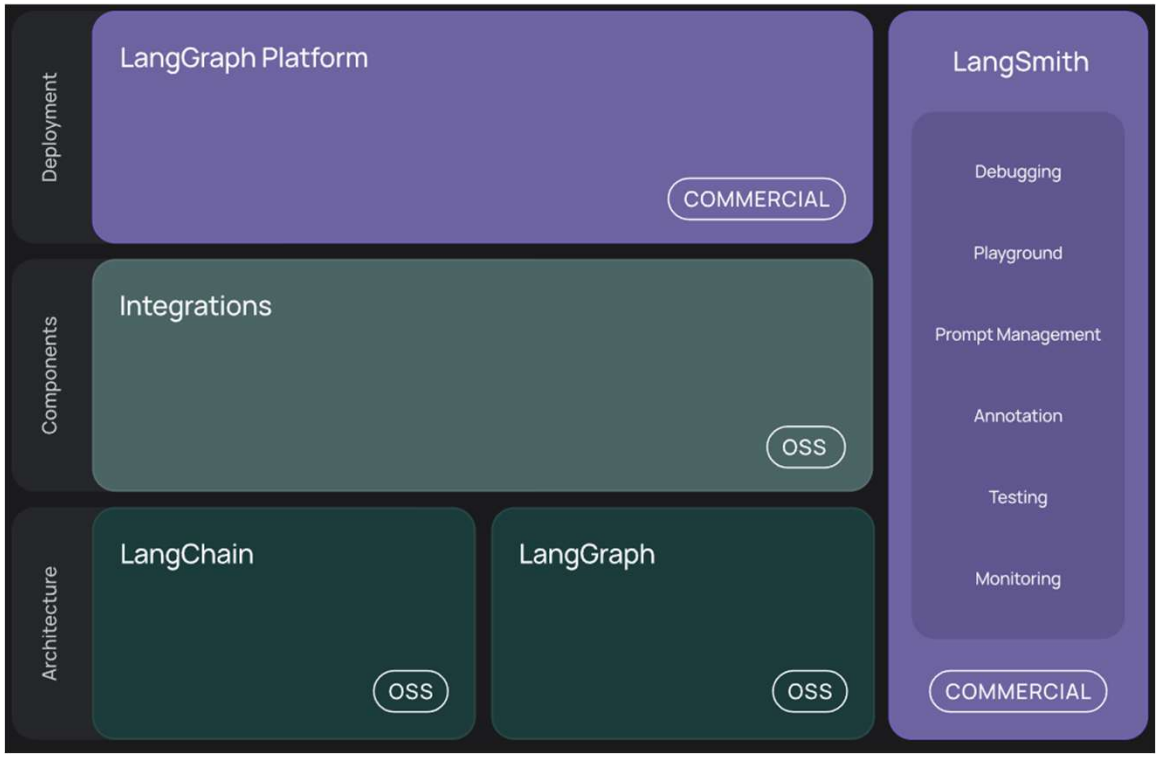
Beyond LangChain

LangGraph (Agentic)

- 상태(State) 기반의 그래프 구조
- 복잡한 Agent와 Workflow를 쉽게 구현

LangSmith (LLMOps)

- 랭체인의 중간 실행 결과를 트래킹
- 평가 및 모니터링 도구를 지원



Source: https://python.langchain.com/docs/introduction/

[실습] LangChain을 이용한 LLM 어플리케이션 만들기

랭체인의 기본 모듈 이해하기

- LLM, Prompt Template, Chain

LangChain Expression Language 이해하기

- 파이프(|) 기반의 LCEL 체인 구성하기
- RunnablePassthrough, RunnableParallel과 같은 특수 체인 이해하기