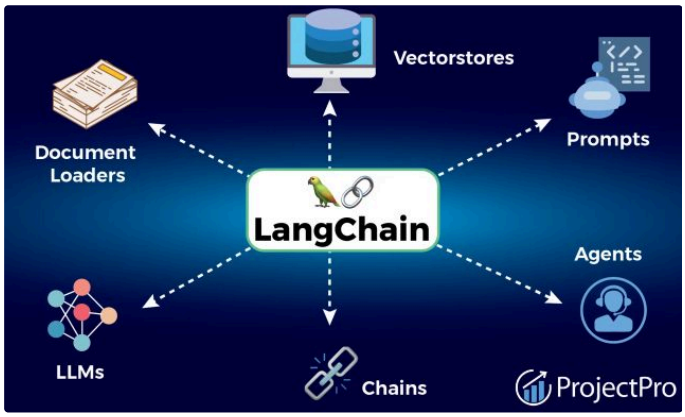
2024-07-23 [Langchain] Langchain의 개념과 작동원리

GitHub - langchain-ai/langchain: 🦜🔗 Build context-aware reasoning applications

What is Langchain?

LLM(Large Language Model) 기반의 어플리케이션을 만드는 데 필요한 여러 구성요소들을 통합해놓은 프레임워크입니다.



LLM(Large Language Model)

LM(Large Language Model)은 방대한 양의 텍스트 데이터를 학습하여 일반적인 언어 생성 및 자연어 처리 작업을 수행할 수 있는 컴퓨터 모델입니다. 이러한 모델은 통계적 관계를 학습하여 능력을 습득합니다. (입력 텍스트를 기반으로 다음 단어를 반복적으로 예측하여 텍스트를 생성하는 방식)

LLM은 주로 2017년에 도입된 트랜스포머(Transformer) 아키텍처를 활용하며, 이는 빠른 처리 속도와 대규모 텍스트 데이터의 효율적인 생성 및 처리를 가능하게 합니다.

대표적인 LLM

- OpenAI의 GPT 시리즈
- Google의 BERT
- Meta의 LLaMA
- Anthropic의 Claude

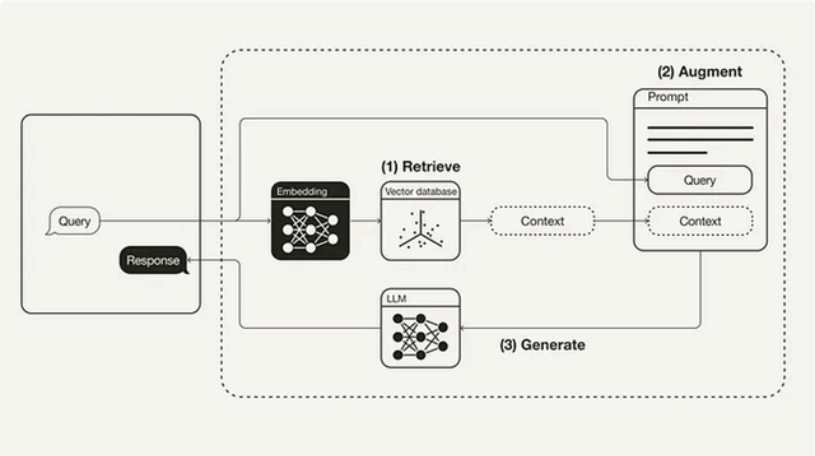
등등

RAG(Retrieval Augmented Generation)

RAG(Retrieval Augmented Generation)은 정보 검색과 텍스트 생성을 결합한 기술입니다.

기존에 사용자가 LLM에 query를 전달하면, LLM이 학습했던 데이터에 기반하여 답변이 생성되었습니다.

RAG를 활용하면 사용자는 자신이 가지고 있는 데이터를 기반으로 종합적인 prompt를 작성하여 적절한 답변의 생성이 이루어지게 할 수 있습니다.



Langchain의 기능들을 거시적인 관점에서 봤을 때, Langchain은 효과적으로, 효율적으로 RAG를 실시하기 위한 프레임워크라고 볼 수도 있을 것 같습니다.

Why Langchain?

Langchain을 통해 사용자는 LLM 기반 어플리케이션이 작동하는 각 프로세스를 단일 프레임워크 내에서 원하는 플랫폼을 활용하여 구축할 수 있습니다.

예) 고객 지원 챗봇 구축

- 고객 지원 관련 데이터(예: FAQ, 제품 매뉴얼)을 로드하고, OpenAI임베딩 모델을 활용하여 데이터를 벡터화합니다.
- 임베딩된 데이터는 검색 속도를 높이기 위해 Redis와 같은 캐시 시스템에 저장됩니다.
- 사용자가 질문을 입력하면, 챗봇은 해당 질문을 임베딩하여 데이터베이스에서 유사한 질문과 답변을 검색합니다.
- 검색된 정보를 바탕으로 사용자의 질문에 대한 답변을 생성합니다.

Structure of Langchain

langchain 0.2.10 — 🦜🔗 LangChain 0.2.10 (Langchain api reference)

Components | 🦜🔗 LangChain (Langchain v0.1 docs) - v

Introduction | 🦜🔗 LangChain (Langchain v0.2 docs)

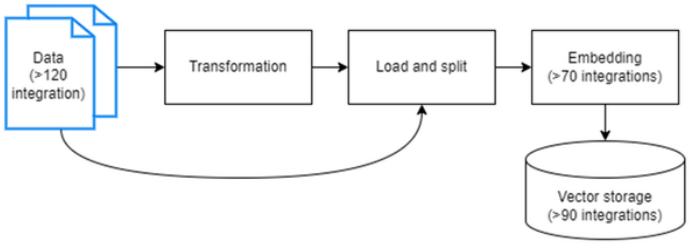
- `langchain-core`: 기본 추상화 및 LCEL을 위한 패키지
- `langchain-community`: 써드파티 플랫폼 통합을 위한 패키지
  - 일부 패키지는 **partner packages** 로 분류되어있습니다. EX) `langchain_openai` & `langchain_anthropic`.
- `langchain`: Chains, agents, and retrieval strategies that make up an application's cognitive architecture.

지속적으로 개선되고 있다고 하지만 Langchain library는 직관적으로 이해하기에는 상당히 복잡합니다. 예를 들어 langchain과 langchain_core의 경계가 명료하지 않으며, langchain에서 활용되는 플랫폼 중 일부는 파트너 패키지로 분류되어 langchain_openai, langchain_aws와 같이 활용되

기도 합니다.

따라서 RAG의 관점에서 보았을 때, Langchain의 구성요소들이 어떤 역할을 하는지 정리해보았습니다.

Retrieval



Module	Functions	Example
Document loader	각종 데이터 형태에 따라 데이터를 로드합니다.	Pdf, csv, txt, docx….
Document transformation	특정 데이터를 원하는 형식에 따라 변환합니다.	bs4를 통해 수집한 HTML에서 특정 태그를 제외하거나 특정 태그만을 추출할 수 있습니다.
Text splitter	데이터를 분리합니다. (Retrieval이 검색할 문서의 단위를 정하는 것과 같습니다.)	RecursiveCharacterTextSplitter는 글자 뭉치의 크기를 기준으로 데이터를 분리시켜줍니다.
Embedding models	주어진 데이터를 임베딩하여 벡터화하고, 이를 Cache합니다.	
Vector stores	임베딩된 데이터를 저장합니다.	
Retrieval	사용자로부터 query를 받아 이와 관계된 데이터를 탐색합니다.	

Augmentation

Module	Functions	Example
PromptTemplate	LLM에게 전달될 prompt를 구축하기 위한 틀을 제공합니다.	
Memory	LLM과 나누었던 대화를 저장하여 context에 이를 포함시키도록 합니다.	

```python
from langchain.prompts import ChatPromptTemplate

template = ChatPromptTemplate.from_messages(
    [
        ("system", "You are a geography expert. And you only reply in {language}."),
        ("ai", "Ciao, mi chiamo {name}!"),
        ("human",
         "What is the distance between {country_a} and {country_b}. Also, what is your name?"
        ),
    ]
)

prompt = template.format_messages(
    language="Greek",
    name="Socrates",
    country_a="Mexico",
    country_b="Thailand",
)
```

a Example of PromptTemplate

Generation

Module	Functions	Example
LLMs / Chatmodels	사용자로부터 prompt를 받아 그에 맞는 응답을 생성합니다.	GPT, llama, claude, mistral….
Agent	LLM이 자체적으로 해결하지 못하는 문제들을 여러 Tools을 활용하여 해결하도록 돕습니다.	‘Tavily search’라는 웹 검색 툴과 ‘Wikidata’라는 위키 검색 툴을 가지고 있는 에이전트는 사용자로부터 query를 받았을 때, 더 적합하다고 여겨지는 툴을 활용하여 내용을 검

LCEL(LangChain Expression Language)

Langchain은 이름에서 알 수 있듯이 여러 구성요소들을 쉽게 Chain할 수 있도록 합니다. 이를 위해 LCEL라는 문법을 활용하여 Chain이 이루어질 수 있도록 합니다.

이 때, 각 구성요소들을 엮어주는 역할을 하는 것이 pipe operator 또는 pipe character입니다.

- | : Pipe operator

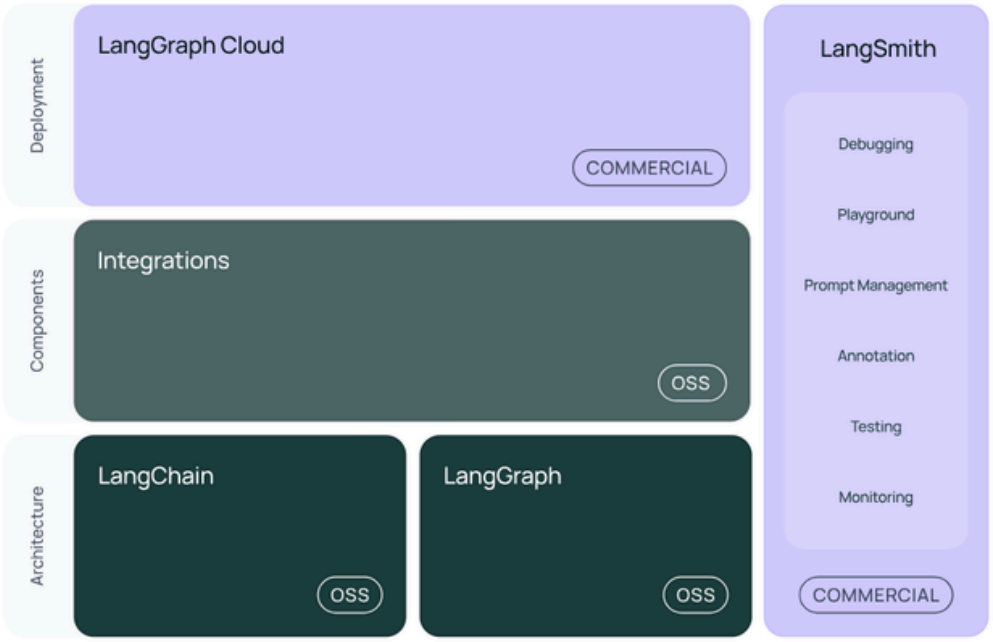
다음은 Pipe operator를 통해 연결 가능한 Component들입니다. 각 Component들은 Input Type에 해당하는 데이터를 받아서 Output Type으로 반환합니다.

Component	Input Type	Output Type
Prompt	Dictionary	PromptValue
ChatModel	Single string, list of chat messages or a PromptValue	ChatMessage
LLM	Single string, list of chat messages or a PromptValue	String
OutputParser	The output of an LLM or ChatModel	Depends on the parser
Retriever	Single string	List of Documents
Tool	Single string or dictionary, depending on the tool	Depends on the tool

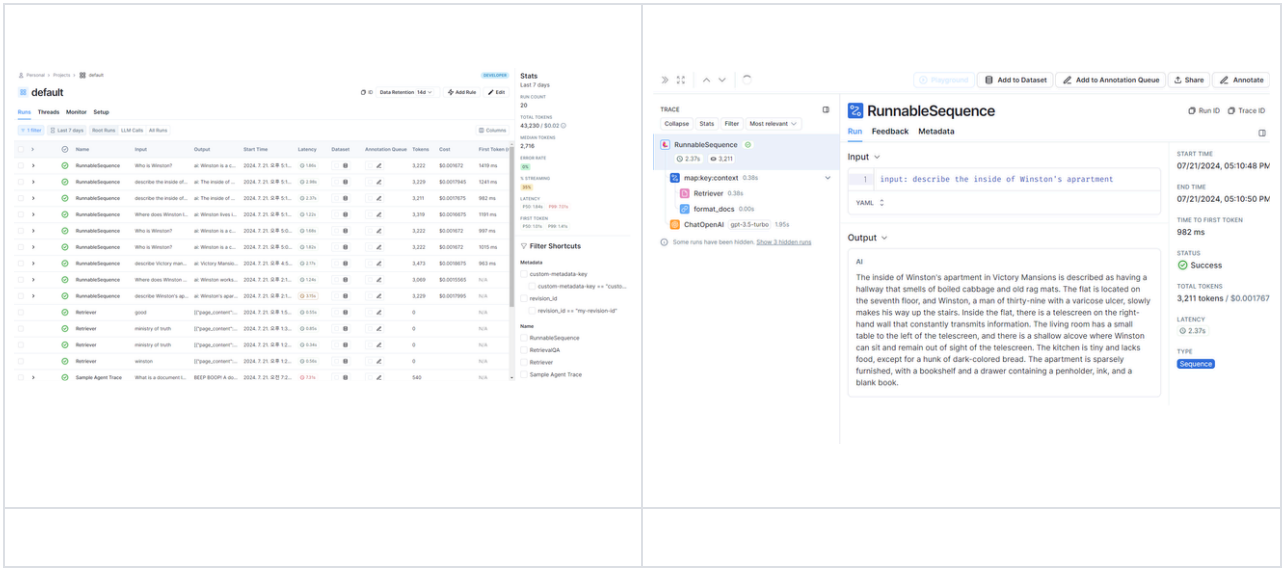
```python
import getpass
import os
from langchain_core.output_parsers import StrOutputParser
from langchain_core.prompts import ChatPromptTemplate
from langchain_openai import ChatOpenAI

openai_api_key = os.getenv("OPENAI_API_KEY")

model = ChatOpenAI(model="gpt-3.5-turbo-0125")
prompt = ChatPromptTemplate.from_template("tell me a joke about {topic}")

chain = prompt | model | StrOutputParser()

chain.invoke({
    "topic" : "student"
})
```

example of using pipe operator

Langchain ecosystem



- Langsmith : Langchain과 연결되어 Langchain의 작동 과정을 추적합니다.



- Langgraph : LangGraph는 여러 에이전트를 활용하는 어플리케이션을 만들 수 있도록 합니다. 각 에이전트는 특정 작업에 최적화되어 역할을 분담하며, 복잡한 상호작용을 효율적으로 관리합니다.