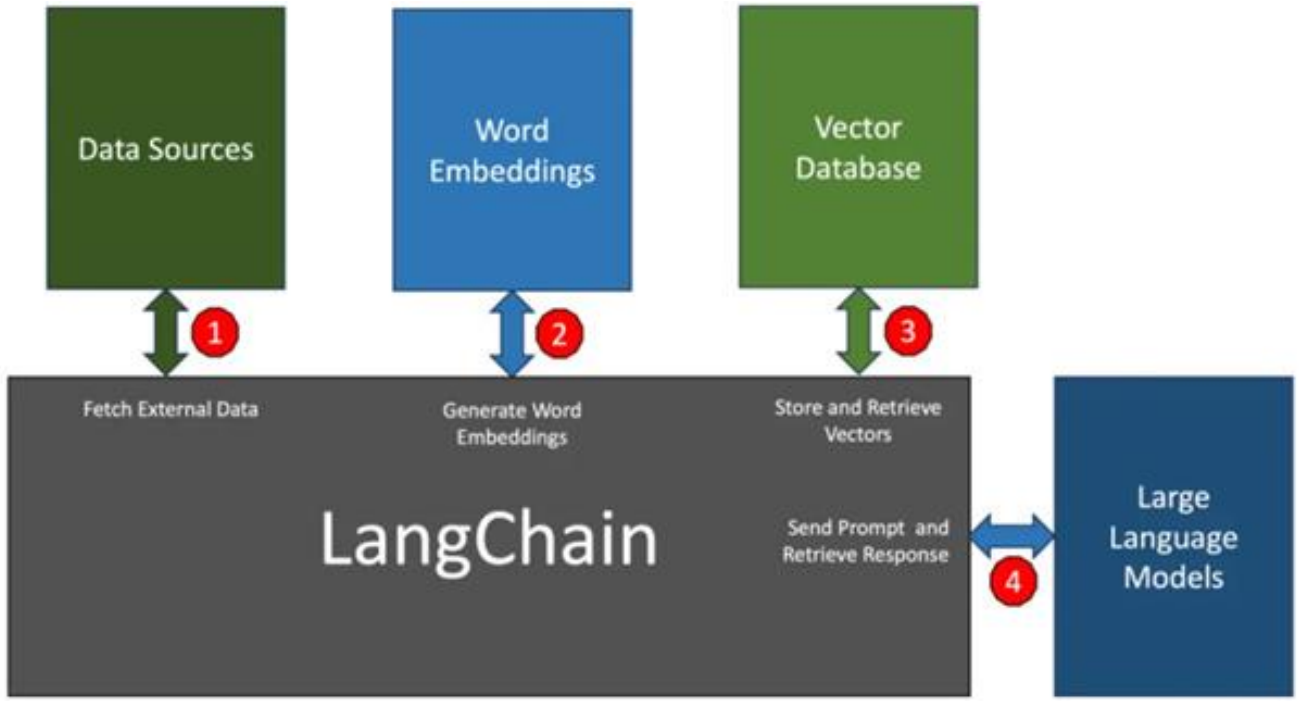
Langchain: LLM Framework

jh-cho

Langchain

- 랭체인(LangChain)은 다양한 **LLM** 애플리케이션을 쉽게 개발할 수 있도록 지원하는 **프레임워크**
  - 랭체인은 LLM과 애플리케이션의 통합을 간소화하도록 설계된 SDK
  - 랭체인은 간단하고 통합된 API를 노출하여 기본 LLM의 구현 세부 사항을 요약하는데, 이 API를 통해 개발자들은 코드를 크게 변경하지 않고 모델을 쉽게 교체하거나 대체할 수 있음

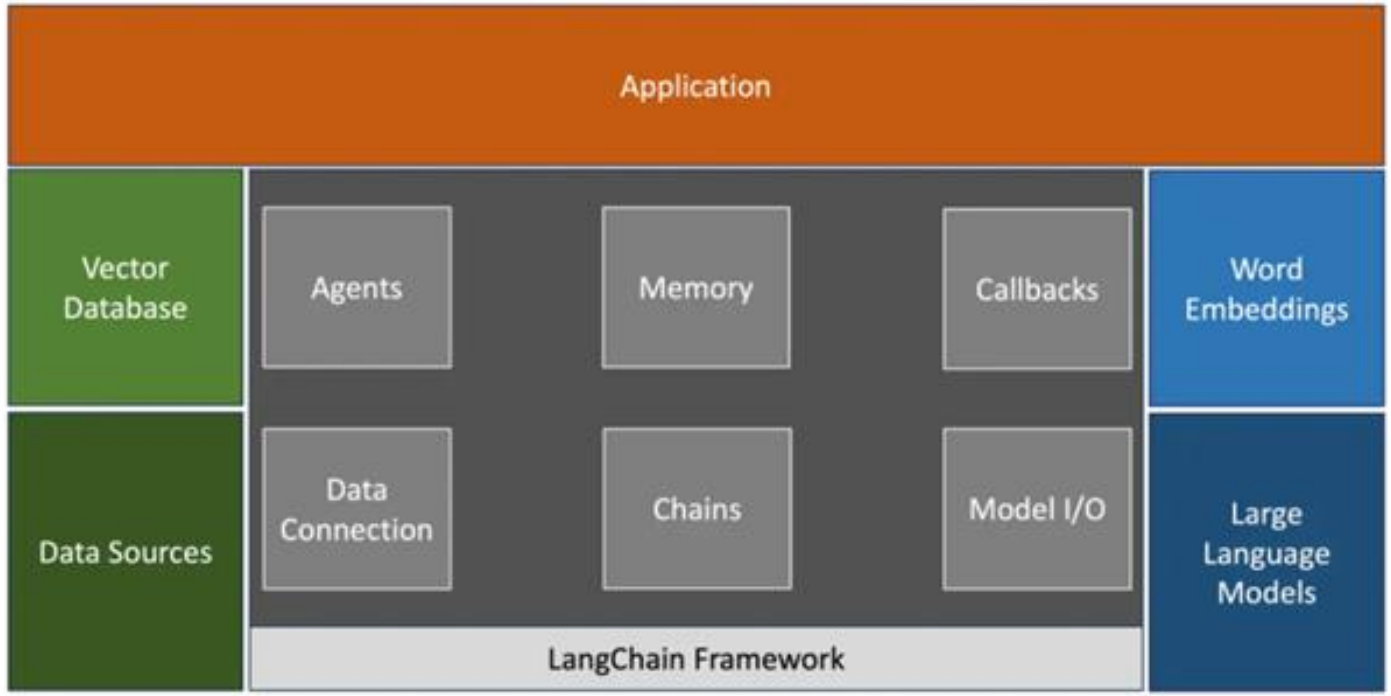


Langchain

- 데이터 소스
  - 애플리케이션이 LLM에 대한 컨텍스트를 구축하기 위해 **PDF, 웹 페이지, CSV, 관계형 데이터베이스**와 같은 **외부** 소스에서 데이터를 검색해야 하는 경우가 있다. 랭체인은 서로 다른 소스에서 데이터에 액세스하고 검색할 수 있는 모듈과 원활하게 통합된다.
- 워드 임베딩
  - 검색된 데이터는 **벡터로 변환**되어야 한다. 그러면 텍스트를 LLM과 관련된 **워드 임베딩 모델**에 전달하게 된다. 랭체인은 선택한 LLM을 기반으로 최적의 임베딩 모델을 선택한다.
- 벡터 데이터베이스
  - 생성된 임베딩은 **유사성 검색**을 위해 벡터 데이터베이스에 저장된다. 랭체인은 메모리 내 배열부터 파인콘(Pinecone)과 같은 호스팅 벡터 데이터베이스에 이르기까지 다양한 소스에서 벡터를 쉽게 저장하고 검색할 수 있도록 지원한다.
- LLM
  - 랭체인은 **오픈AI, 코히어(Cohere), AI21**에서 제공하는 주류 LLM과 **허깅페이스(Hugging Face)**에서 제공되는 오픈소스 LLM을 지원한다. 지원되는 모델과 API 엔드포인트 목록은 빠르게 증가하고 있다.

Langchain

랭체인의 주요 모듈



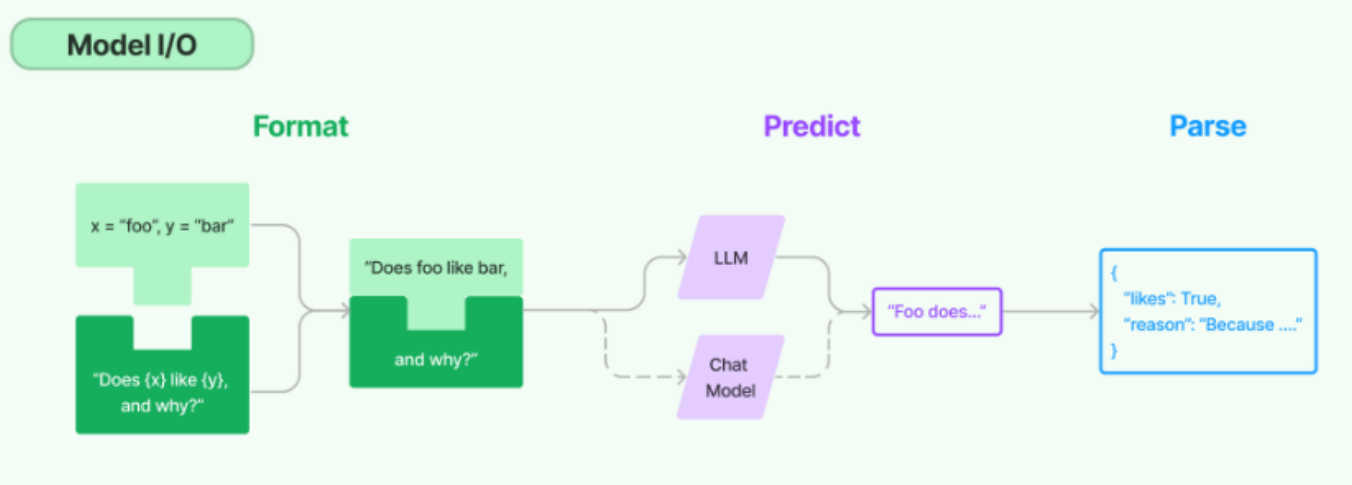
Langchain

랭체인의 주요 모듈

1. 모델 I/O

- 모델 I/O 모듈은 LLM과의 상호 작용을 다룬다. 이는 기본적으로 효과적인 **프롬프트를 생성**하고, **모델 API를 호출**하고, **결과 해석**을 돕는다. 이 모듈은 LLM 제공자가 노출하는 인증, API 매개변수, 엔드포인트를 요약한다. 이 모듈은 마지막으로, 모델에서 보낸 응답을 애플리케이션서 사용할 수 있는 원하는 형식으로 해석하는 작업을 돕는다.

- Prompt : 모델 입력을 템플릿화, 동적으로 선택 및 관리하기
- Language models : 언어 모델: 공통 인터페이스를 통해 언어 모델을 호출
- Output Parser : 모델 출력에서 정보 추출



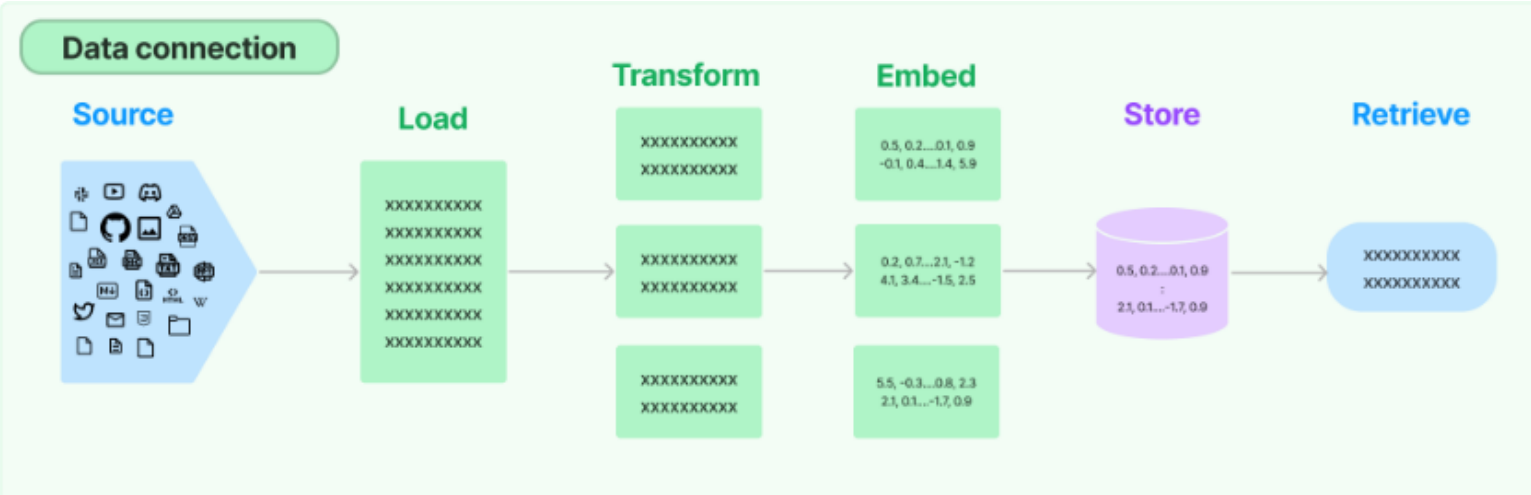
Langchain

랭체인의 주요 모듈

2. 데이터연결(Data Connection)

- 데이터 연결 모듈은 PDF 또는 엑셀 파일과 같은 **외부 문서를 로드**하고, 이를 처리하기 위해 일괄적으로 **단어 임베딩으로 변환**한 다음, 임베딩을 **벡터 데이터베이스에 저장**하고, 마지막으로 **쿼리를 통해 검색**한다.

- Document loaders: 다양한 소스에서 문서 불러오기
- Document transformers: 문서 분할, Q&A 형식으로 문서 변환, 중복 문서 삭제 등 다양한 작업 수행
- Text embedding models: 비정형 텍스트를 float 숫자 목록으로 변환
- Vector stores: embedding된 데이터 저장 및 검색
- Retrievers: 데이터 쿼리 처리



Langchain

랭체인의 주요 모듈

3. Chains

- 간단한 애플리케이션의 경우 LLM을 단독으로 사용할 수 있지만 더 복잡한 애플리케이션의 경우 서로 또는 다른 기능을 수행하는 컴포넌트와 함께 LLM을 연결해야 한다.
- LangChain은 이러한 컴포넌트를 "chain"으로 연결할 수 있는 인터페이스를 제공하고 컴포넌트에 대한 호출 시퀀스로 정의한다. 예를 들어, 체인은 문서를 요약한 다음 이에 대한 감정 분석을 수행하는 프롬프트가 포함될 수 있다.

4. Agents

- 에이전트는 체인이 사용할 도구를 선택하여 동작 하도록 지원한다.
- 에이전트의 핵심 아이디어는 LLM을 사용하여 수행할 작업 순서를 선택하는 것으로, 언어 모델이 추론 엔진으로 사용되어 **어떤 작업을 어떤 순서로 수행할지 결정**한다.
- 에이전트는 tool이라는 것을 이용하여 목표를 달성하려고 노력한다. 여러가지의 tool을 묶어 특정 능을 구성해 놓은 것을 **toolkit**이라고 한다.

Langchain

랭체인의 주요 모듈

5. Memory

- 체인 실행 사이에 이전 상황을 기억하여 애플리케이션 상태 유지
- 대부분의 LLM 애플리케이션에는 대화형 인터페이스가 있으며, 메모리는 이전 대화에 소개된 정보를 참조할 수 있도록 지원한다.

6. Callbacks

- LangChain은 LLM 애플리케이션의 다양한 단계에 연결할 수 있는 콜백 시스템을 제공한다. 이는 로깅, 모니터링, 스트리밍 및 기타 작업에 유용하며 API 전체에서 사용할 수 있는 콜백 인수를 사용하여 이러한 이벤트를 받을 수 있다.

Langchain

랭체인 파이썬 모듈, 예시

langchain 0.2.7 🔗 LangChain 0.2.7 (랭체인 다큐먼트)

- *langchain, langchain_core, langchina_community* 등의 모듈들과 각 모듈의 클래스, 각 클래스의 함수 소개)

How-to guides | 🔗 LangChain (랭체인 가이드)

1. 모델I/O

- 프롬프트:

```
from langchain_core.prompts import PromptTemplate
example_prompt = PromptTemplate.from_template("Question: {question}\n{answer}")
```

- LLM:

```
from langchain.globals import set_llm_cache
from langchain_openai import OpenAI

# To make the caching really obvious, lets use a slower model.
llm = OpenAI(model_name="gpt-3.5-turbo-instruct", n=2, best_of=2)
```

Langchain

랭체인 파이썬 모듈, 예시

2. 데이터연결

- Document Loaders (ex. csv)

```
from langchain_community.document_loaders.csv_loader import CSVLoader
file_path = ()
loader = CSVLoader(file_path=file_path)
data = loader.load()
```

- Text Splitters

```
from langchain_text_splitters import CharacterTextSplitter

text_splitter = CharacterTextSplitter(
        separator="\n\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len,
        is_separator_regex=False)
texts = text_splitter.create_documents([state_of_the_union])
```

Langchain

랭체인 파이썬 모듈, 예시

2. 데이터연결

- Embedding Models (ex openai)

```
from langchain_openai import OpenAIEmbeddings
embeddings_model = OpenAIEmbeddings()
embeddings = embeddings_model.embed_documents(
["Hi there!",
"Oh, hello!",
"What's your name?",
"My friends call me World",
"Hello World!"])
```

- Vector Stores (ex. Chroma, FAISS) / Retrivers

```
from langchain_chroma import Chroma
db = Chroma.from_documents(documents, embeddings)
```

```
from langchain_community.vectorstores import FAISS
vectorstore = FAISS.from_documents(texts, embeddings)
retriever = vectorstore.as_retriever()
docs = retriever.invoke("what did the president say about ketanji brown jackson?")
```

Langchain

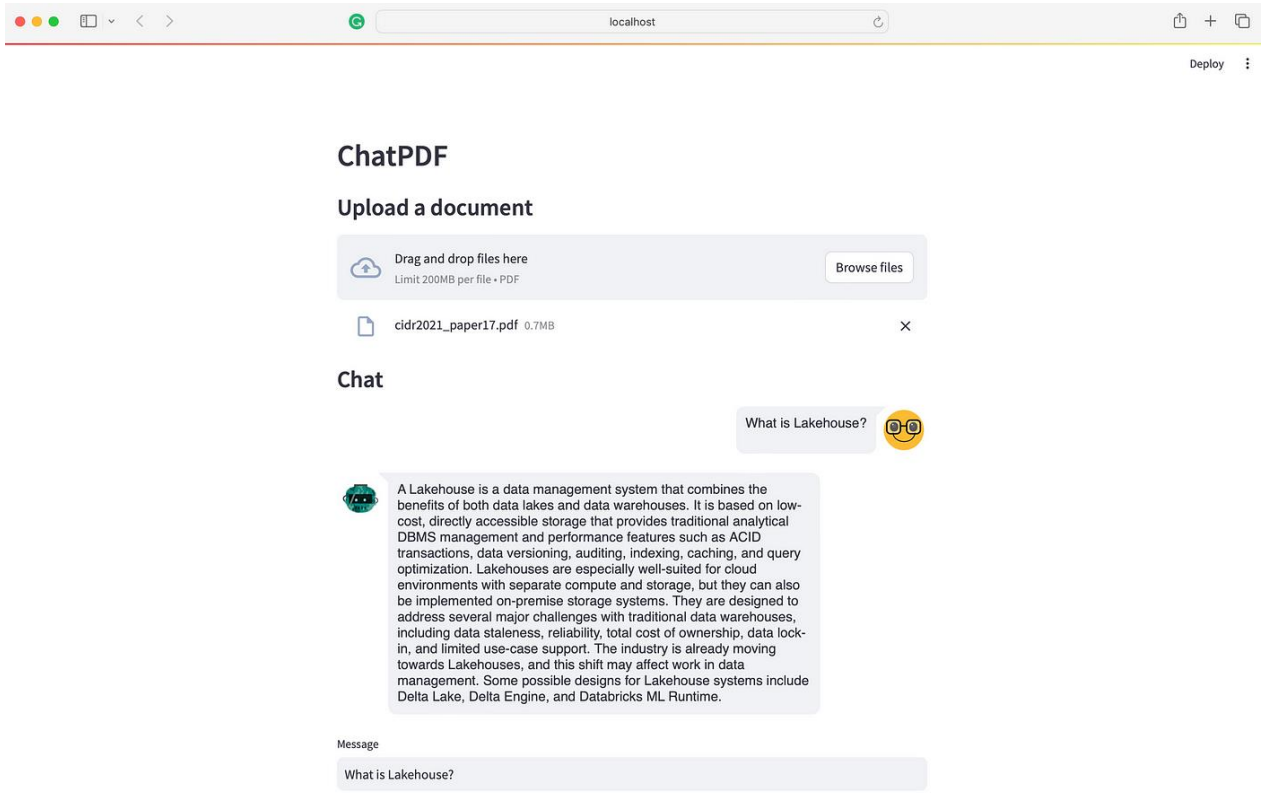
랭체인 튜토리얼

https://github.com/teddylee777/langchain-kr/tree/main (한국어 튜토리얼)

Streamlit

Streamlit은 데이터 사이언티스트, 엔지니어, 그리고 분석가들이 Python으로 데이터 앱을 빠르고 쉽게 만들 수 있도록 돕는 오픈 소스 앱 프레임워크이다.

이 툴은 복잡한 **프론트엔드** 개발 지식이 필요 없이, 데이터를 시각화하고, 데이터 기반 앱을 구축하며, 이를 공유할 수 있게 해준다. 사용자는 HTML, CSS, JavaScript에 대한 깊은 지식 없이도 멋진 **UI를 구성**할 수 있다.



Example

예시: 간단한 PDF 요약 서비스 예시

https://github.com/jo-cho/nlp_study/blob/main/2_NLP_EX/pdf_summarize.py

1. Document(pdf) load → PDF 파일 로딩
2. Document process – Text split → PDF에서 text 추출 및 chunk로 분할
3. Vectore store 저장 → chunk를 불러온 embedding model로 벡터스토어(FAISS)에 저장
4. 모델 설정 및 실행 → OpenAI 모델 혹은 허깅페이스의 오픈소스 모델을 불러옴. Generate & decode
5. 일련의 과정을 streamlit 세팅에 넣고 run하는 함수 (def main()) 생성
6. 위 각 과정을 함수로 생성한 뒤에 실행가능한 .py python파일 하나로 만듦.
7. Terminal(ex. Anaconda prompt)에서 “streamlit run filename.py” 코드 실행 → 웹페이지 생성됨