Langchain

전자정보공학부(컴퓨터공학)
석사 1학년
백승민

LangChain?

- OpenAI의 GPT와 같은 LLM은 혁신적인 기술
- 하지만 이러한 LLM을 사용하기 위해선 적합한 파라미터와 프롬프트 등 사용자가 고려해야 할 사항이 많음.
- LangChain은 이러한 고려사항을 좀 더 완화할 수 있는 프레임워크로 등장함.

LangChain?

- LangChain은 LLM모델부터 Retriever, Document Loader등 많은 기능을 포함하여 여러 라이브러리에 대한 접근성이 높음.

- Api 문서를 참조한다면 수많은 라이브러리를 LangChain이라는 프레임워크 하나로 손쉽게 사용이 가능함.

Why should we use LangChain?

- 이용할 수 있는 라이브러리의 수도 많을 뿐더러 api 문서에 전부 정리가 되어있어 입문자가 접근하기에 편함.

- 단순히 모듈을 가져다 쓰기 때문에 간결하고 직관적인 코드의 구성이 가능함.

Why should we use LangChain?

- 최근 채용 공고에서 우대사항으로 LangChain 이용가능자를 명시 해두는 기업이 늘고 있음.

- LangChain을 잘 활용한다면, 취업시장에서도 유리할 것으로 전망.

LangChain

- 기본적으로 OpenAI, HuggingFace의 LLM모델 등을 가져와서 사용할 수 있음.

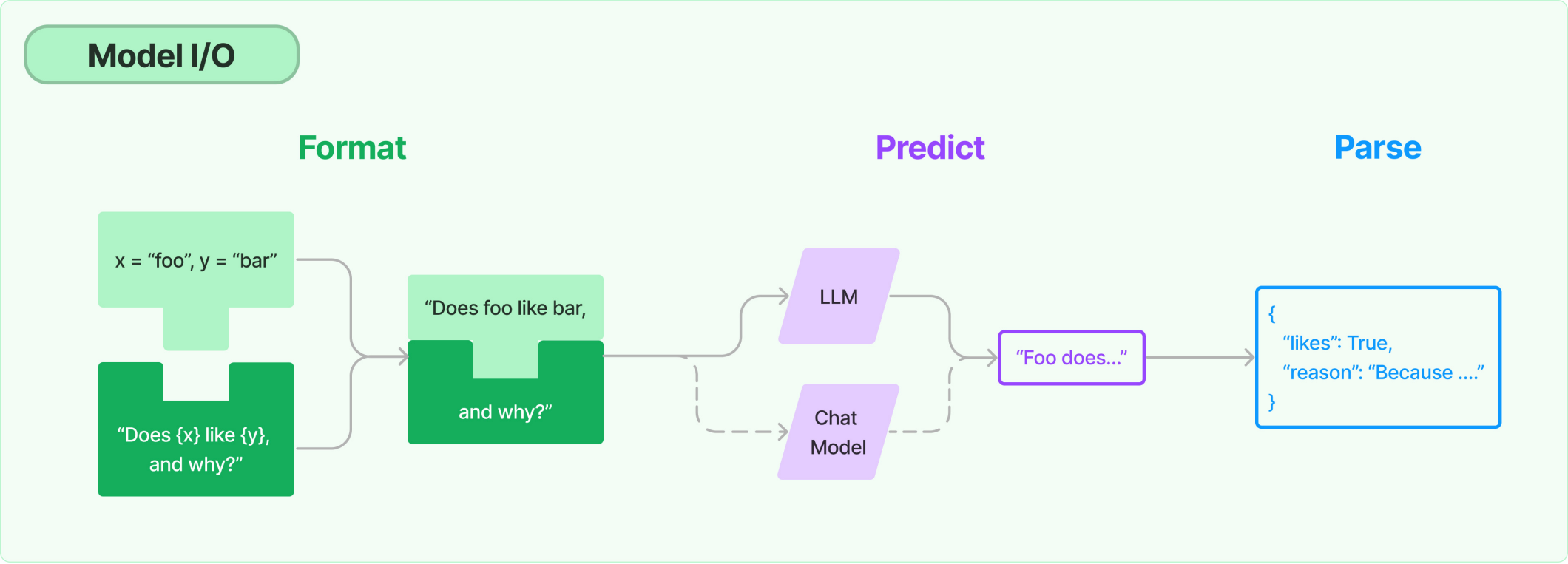
- BM25와 같은 Retriever 모델 또한 활용이 가능함.

LangChain

- LangChain에서는 크게 다음과 같은 모듈들을 제시하고 있음.
  - Model Input/Output 관리
  - Data-Connection
  - Chain
  - Agent
  - Memory
  - Evaluation(Beta)

- Evaluation
  - 다른 LLM을 검증하기 위한 모듈

Model I/O


Model I/O
Format
Predict
Parse
x = “foo”, y = “bar”
“Does {x} like {y}, and why?”
“Does foo like bar, and why?”
LLM
Chat Model
“Foo does...”
{
“likes”: True,
“reason”: “Because ....”
}

Prompt template

- LLM에 직접 Input으로 들어갈 Prompt의 템플릿, 즉 틀을 지정

- 변수의 경우 python의 f-string과 같이 중괄호를 통해 전달함

```
prompt_template = PromptTemplate.from_template(
    "Tell me a {adjective} joke about {content}."
)
prompt_template.format(adjective="funny", content="chickens")
```

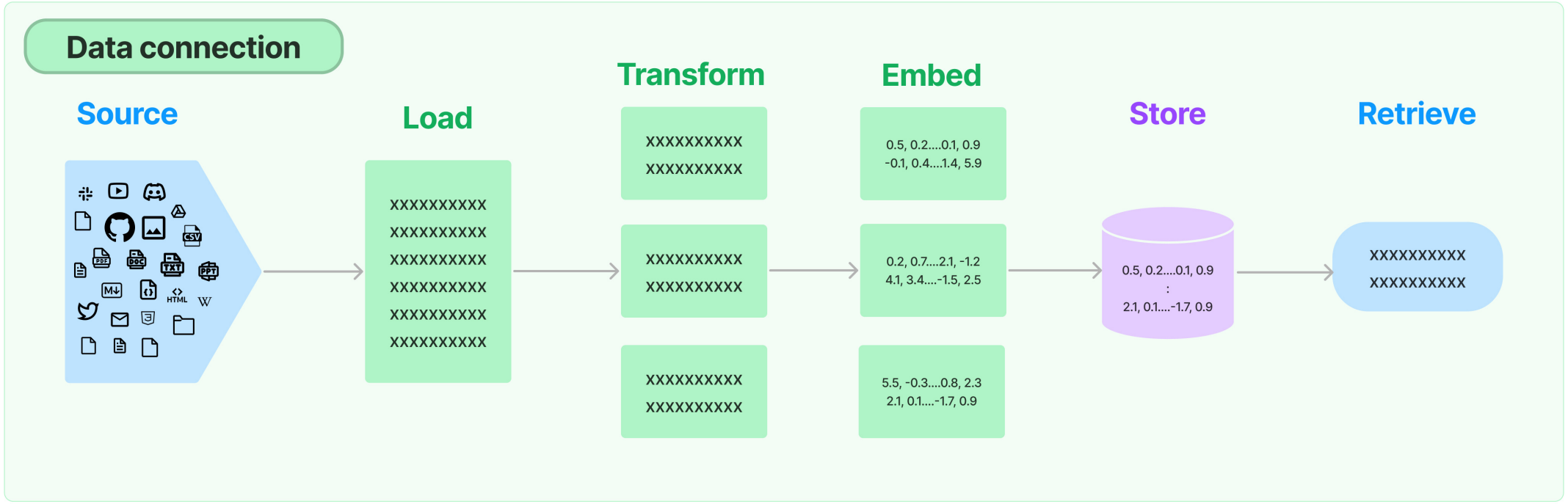
- 몇 가지의 예시를 주고 프롬프트를 제시하는 Few-shot prompt 또한 구현이 되어있음
  - 이를 일부 예시만 선택하는 Example Selector 또한 존재

Data Connection

- LLM의 지식은 충분히 많지만, task에 따라 추가적인 데이터로 학습이 필요할 때가 있음.

- LangChain은 Data Connection 모듈을 통해 이러한 외부 데이터를 관리할 수 있도록 함.

Data Connection



Chain

- LLM이나 Retriever, parser 등을 명시적으로 연결하여 실행하는 모듈

- LangChain에서 제시하는 가장 기본적인 Chain은 PromptTemplate과 LLM을 연결하는 Chain

- Chain을 여러 개 중첩하여 복합형 체인이 구성 가능함.

Chain

```
llm = OpenAI(temperature=0.9)

prompt = PromptTemplate(
    input_variables=["company", "product"],
    template="What is a good name for {company} that makes {product}?",
)
chain = LLMChain(llm=llm, prompt=prompt)
print(chain.run({
    'company': "ABC Startup",
    'product': "colorful socks"
    }))
```

Agent

- LLM을 기반으로 한, task 수행을 능동적으로 선택하는 모듈
- Agent input으로 주어진 Tool들을 활용하여 다음 행동을 설계하고 실행함.
- 설계-실행-분석을 반복하여 진행하다가, 주어진 Task가 완료 가능하면 종료

Agent

```
llm = ChatOpenAI(temperature=0, model="gpt-3.5-turbo-0613")
search = SerpAPIWrapper()
llm_math_chain = LLMMathChain.from_llm(llm=llm, verbose=True)
db = SQLDatabase.from_uri("sqlite:///../../../../../notebooks/Chinook.db")
db_chain = SQLDatabaseChain.from_llm(llm, db, verbose=True)
tools = [
    Tool(
        name = "Search",
        func=search.run,
        description="useful for when you need to answer questions about current events. You should ask targeted questions"
    ),
    Tool(
        name="Calculator",
        func=llm_math_chain.run,
        description="useful for when you need to answer questions about math"
    ),
    Tool(
        name="FooBar-DB",
        func=db_chain.run,
        description="useful for when you need to answer questions about FooBar. Input should be in the form of a question containing full context"
    )
]
```

Agent

```
agent = initialize_agent(tools, llm, agent=AgentType.OPENAI_FUNCTIONS, verbose=True)
agent.run("Who is Leo DiCaprio's girlfriend? What is her current age raised to the 0.43 power?")
```

```
> Entering new  chain...

    Invoking: `Search` with `{'query': 'Leo DiCaprio girlfriend'}`


    Amidst his casual romance with Gigi, Leo allegedly entered
     a relationship with 19-year old model, Eden Polani, in February 2023.
    Invoking: `Calculator` with `{'expression': '19^0.43'}`


> Entering new  chain...
    19^0.43```text
    19**0.43
    ```
    ...numexpr.evaluate("19**0.43")...

    Answer: 3.547023357958959
> Finished chain.
    Answer: 3.547023357958959Leo DiCaprio's girlfriend is reportedly Eden Polani.
    Her current age raised to the power of 0.43 is approximately 3.55.

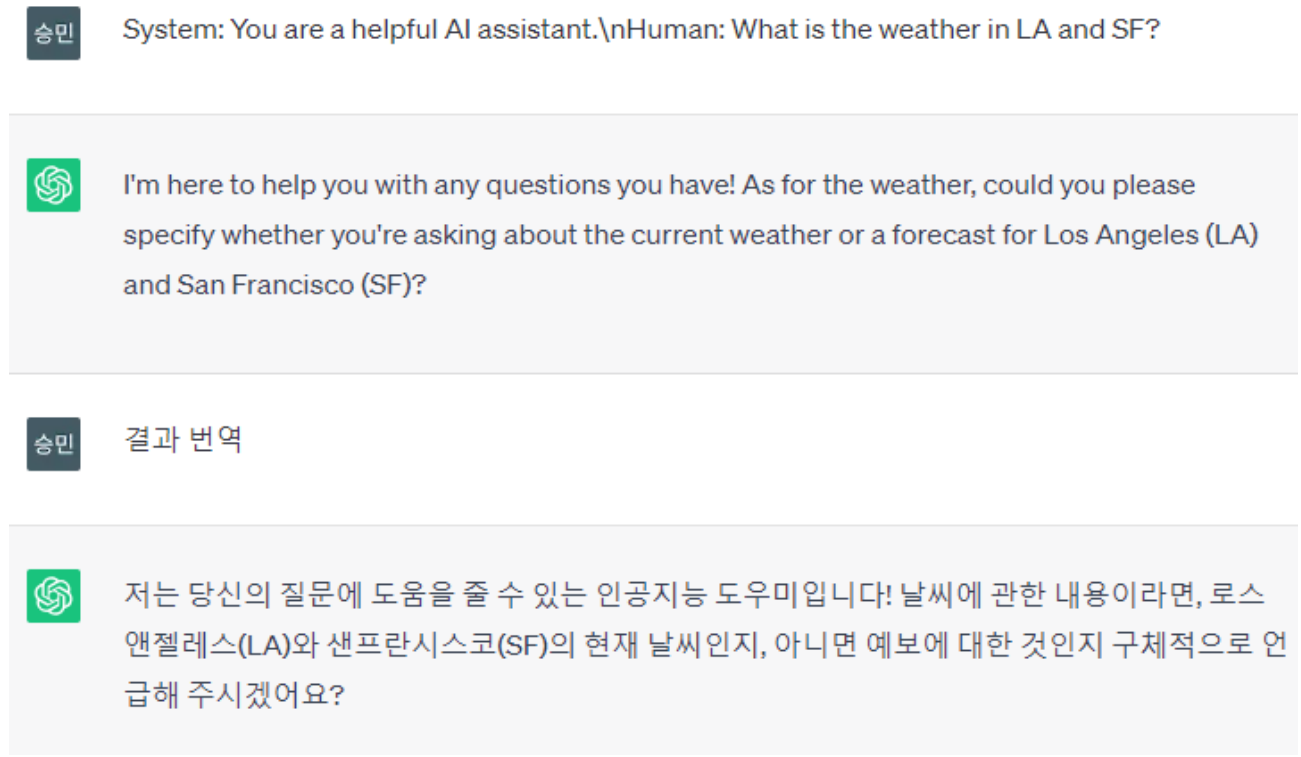
> Finished chain.

    "Leo DiCaprio's girlfriend is reportedly Eden Polani. Her current age raised to the power of 0.43 is approximately 3.55."
```
```

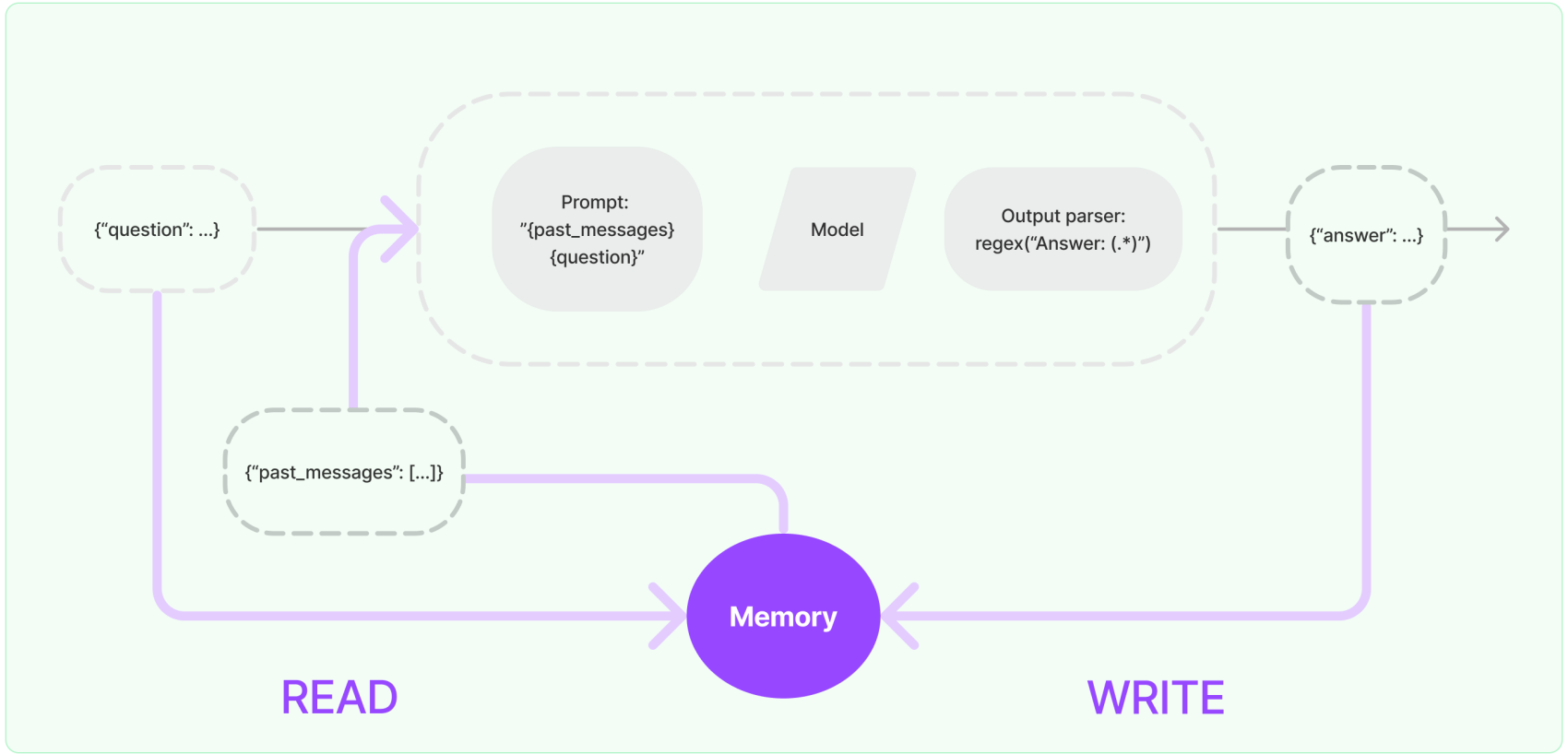
Memory

- Chat GPT와 같은 대화형 인터페이스는 현재 input 뿐만 아니라 이전의 context 또한 고려하여 output을 생성해야 함.

- Memory 모듈은 이러한 대화형 인터페이스를 구현하기 위해 제공된 모듈



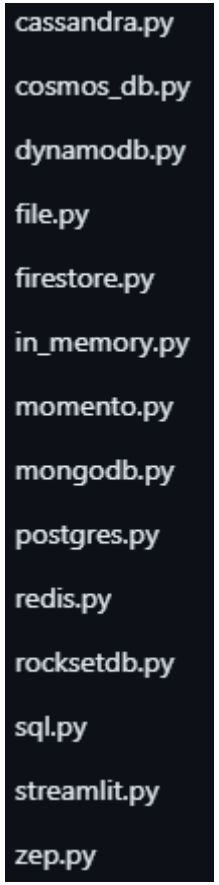
Memory


{“question”: ...}
Prompt:
“{past_messages}
{question}”
Model
Output parser:
regex(“Answer: (.*)”)
{“answer”: ...}
{“past_messages”: [...]}
Memory
READ
WRITE

Memory

- Memory가 기억을 불러오는 형태는 여러가지로 존재함.
  - 단순히 메시지 자체를 기록
  - 일정 구간을 기록
  - 일정한 구간을 요약한 내용
  - Vector store를 사용해 적재

- Memory는 단순히 str이나 list로 저장하는 방법 외에도, 다양한 DB를 사용해 저장하는 방법이 존재함.



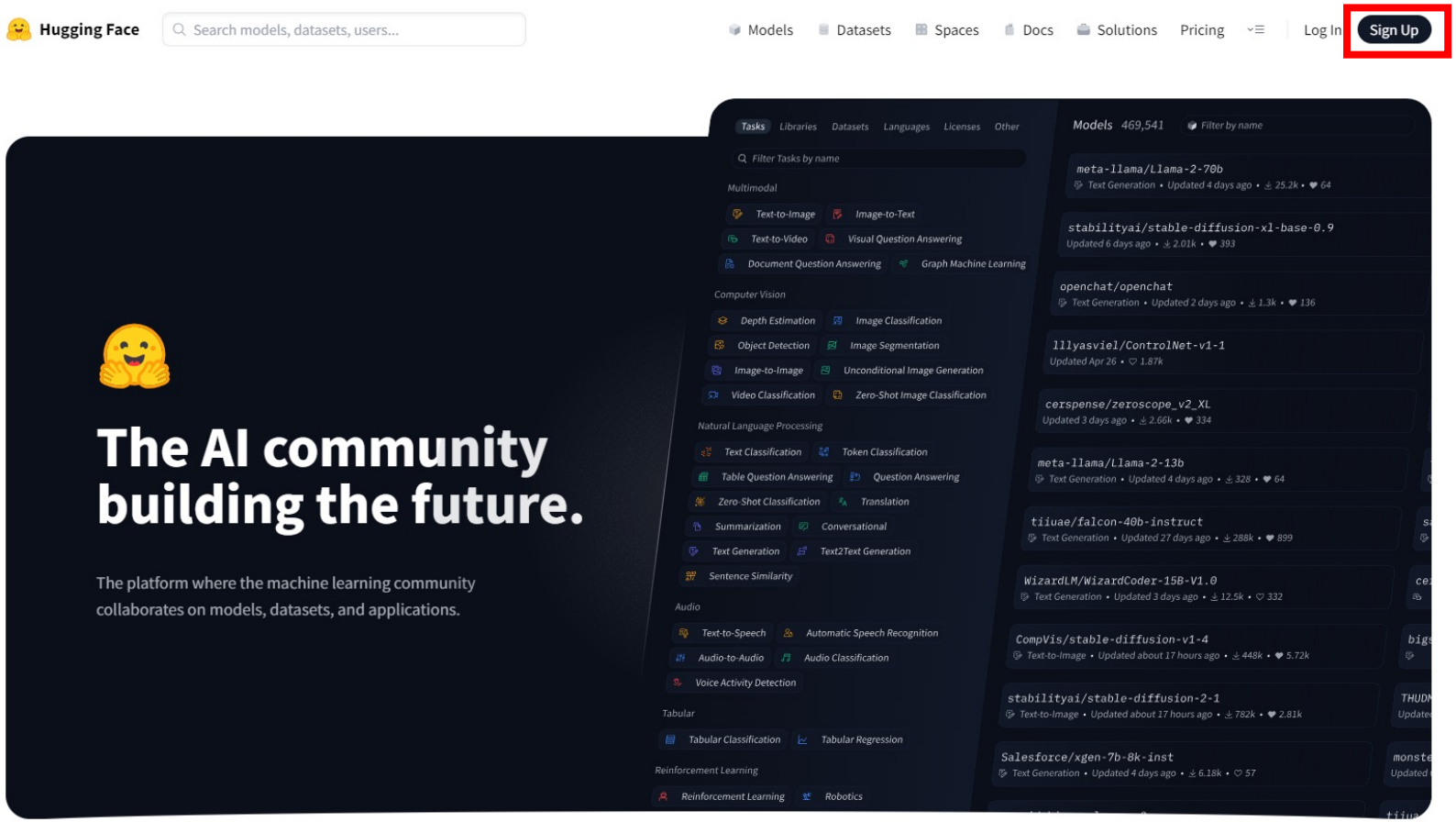
Practice

- LLM Chain을 이용하여 HuggingFace의 모델을 사용하는 QA봇

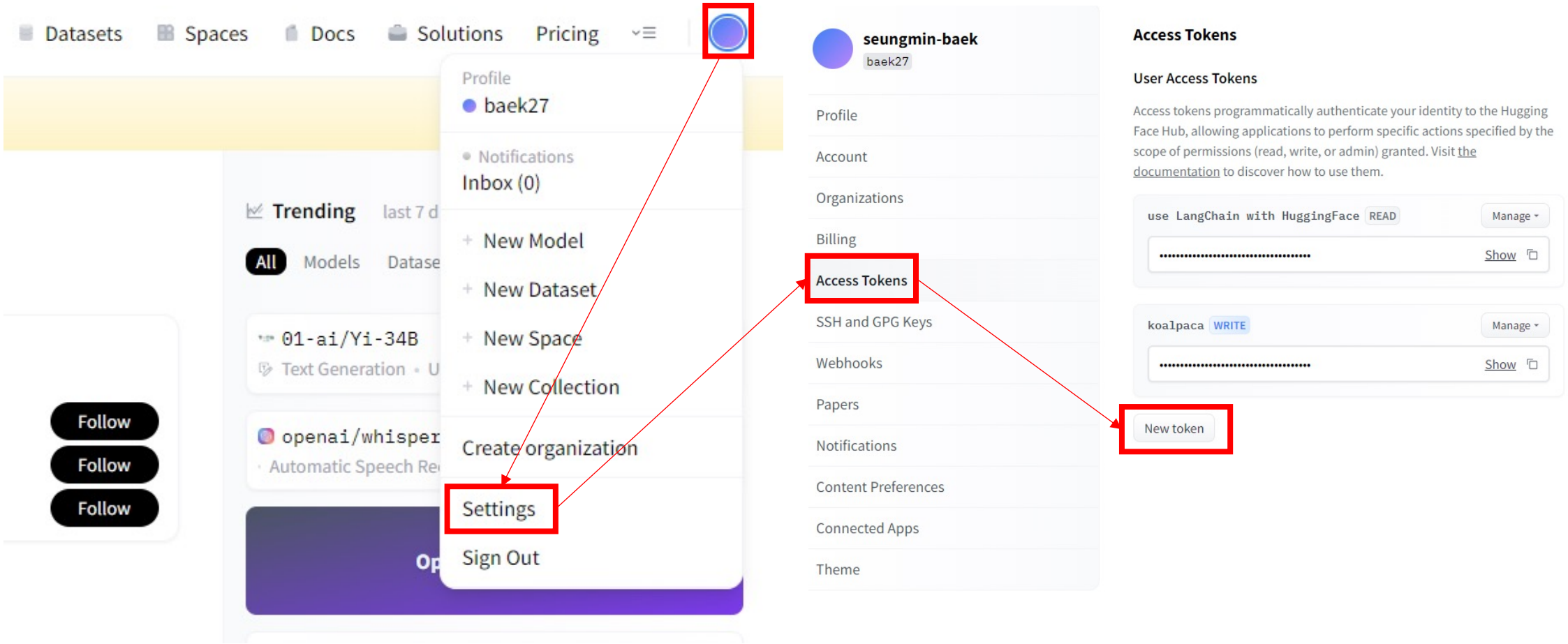
- 내가 가진 문서들을 추가로 받아들여 답변을 하는 QA봇

Practice

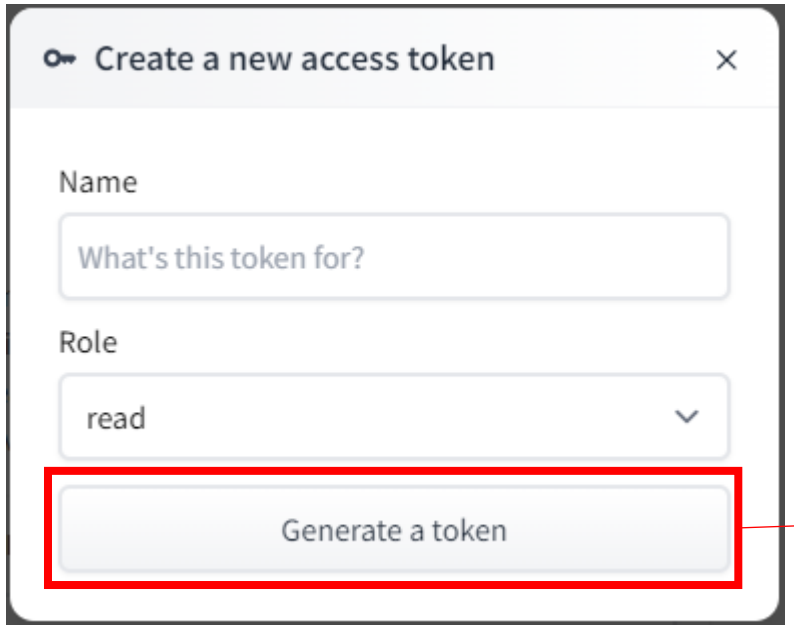
- HuggingFace 가입 및 token 발급

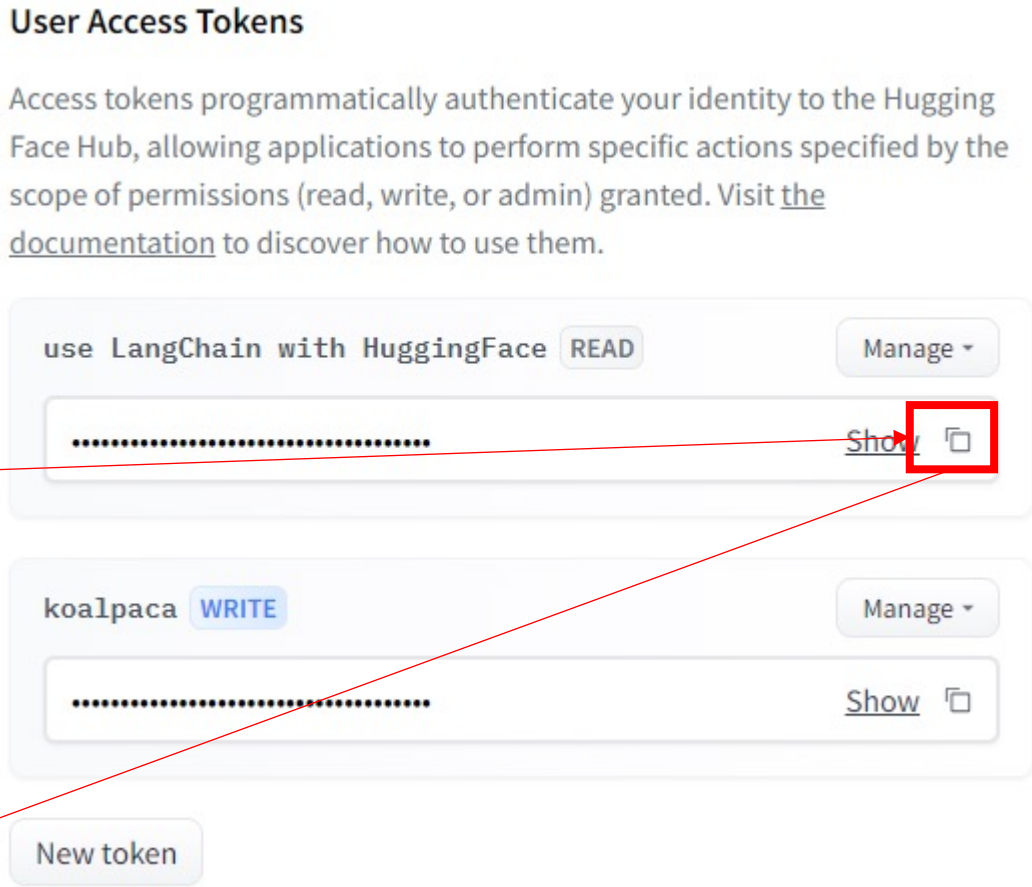


Practice


Datasets
Spaces
Docs
Solutions
Pricing
Profile
baek27
Notifications
Inbox (0)
New Model
New Dataset
New Space
New Collection
Create organization
Settings
Sign Out
Trending
All
Models
01-ai/Yi-34B
Text Generation
Follow
Follow
Follow
seungmin-baek
baek27
Profile
Account
Organizations
Billing
Access Tokens
SSH and GPG Keys
Webhooks
Papers
Notifications
Content Preferences
Connected Apps
Theme
Access Tokens
User Access Tokens
Access tokens programmatically authenticate your identity to the Hugging Face Hub, allowing applications to perform specific actions specified by the scope of permissions (read, write, or admin) granted. Visit the documentation to discover how to use them.
use LangChain with HuggingFace
READ
Manage
Show
koalpaca
WRITE
Manage
Show
New token

Practice





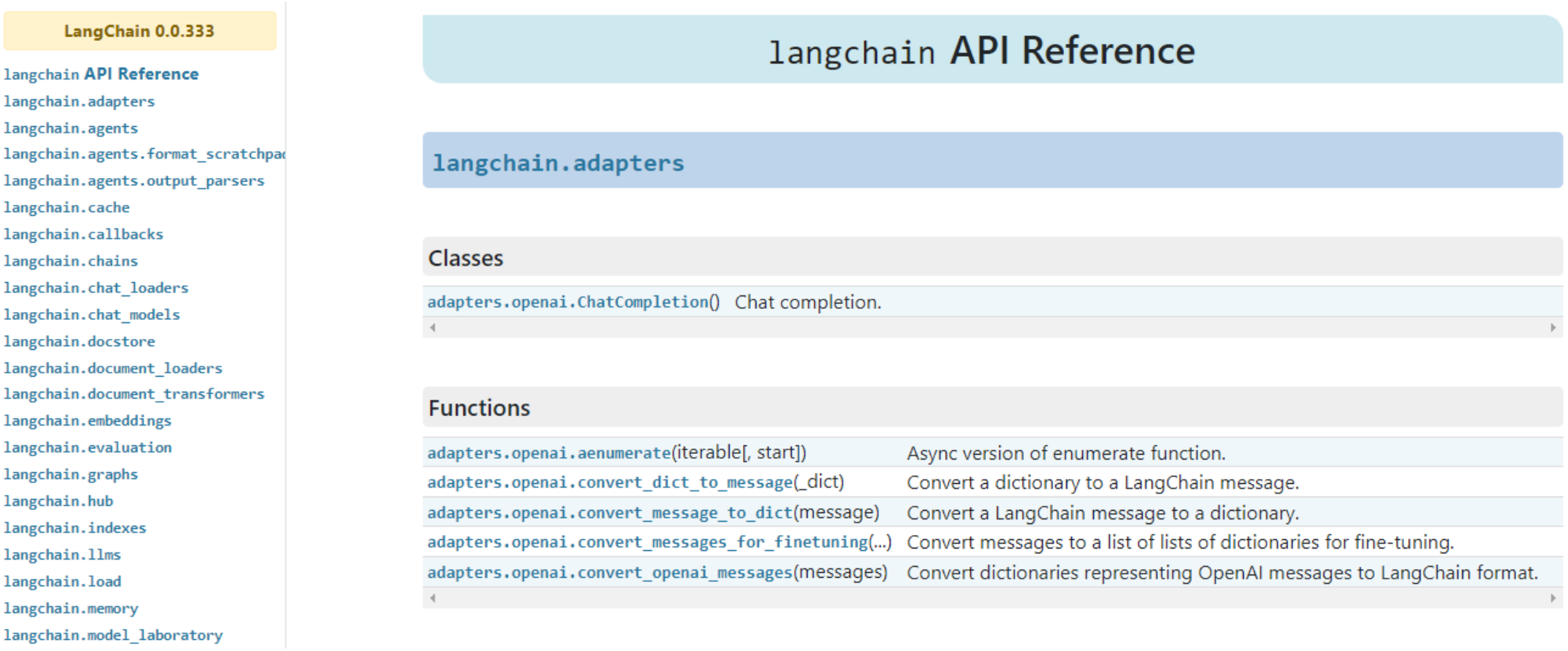

```
!pip install langchain

import os

os.environ['HUGGINGFACEHUB_API_TOKEN'] = "key"
```

Practice

- 모듈을 사용할 때는 api 문서에서 본인이 무엇을 불러와야 하는 지 고려해야 함.



LangChain 0.0.333

langchain API Reference
langchain.adapters
langchain.agents
langchain.agents.format_scratchpa
langchain.agents.output_parsers
langchain.cache
langchain.callbacks
langchain.chains
langchain.chat_loaders
langchain.chat_models
langchain.docstore
langchain.document_loaders
langchain.document_transformers
langchain.embeddings
langchain.evaluation
langchain.graphs
langchain.hub
langchain.indexes
langchain.llms
langchain.load
langchain.memory
langchain.model_laboratory

langchain API Reference

langchain.adapters

Classes

adapters.openai.ChatCompletion()	Chat completion.

Functions

adapters.openai.aenumerate(iterable[, start])	Async version of enumerate function.
adapters.openai.convert_dict_to_message(_dict)	Convert a dictionary to a LangChain message.
adapters.openai.convert_message_to_dict(message)	Convert a LangChain message to a dictionary.
adapters.openai.convert_messages_for_finetuning(...)	Convert messages to a list of lists of dictionaries for fine-tuning.
adapters.openai.convert_openai_messages(messages)	Convert dictionaries representing OpenAI messages to LangChain format.