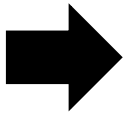
LangChain 소개

Prompt

API 호출 형태

```
client.chat.completions(
    model="gpt-4",
    messages=[
        {"role": "system",
         "content": "너는 도움이 되는 AI 어시스턴트이다
."},
        {"role": "user”,
         "content": "OpenAI의 CEO가 누구야?"},
    ]
)
```



OpenAI GPT에 실제 입력되는 형태

```
<|im_start|>system
너는 도움이 되는 AI 어시스턴트이다.
<|im_end|>
<|im_start|>user
OpenAI의 CEO가 누구야?
<|im_end|>
<|im_start|>assistant
```

Prompt

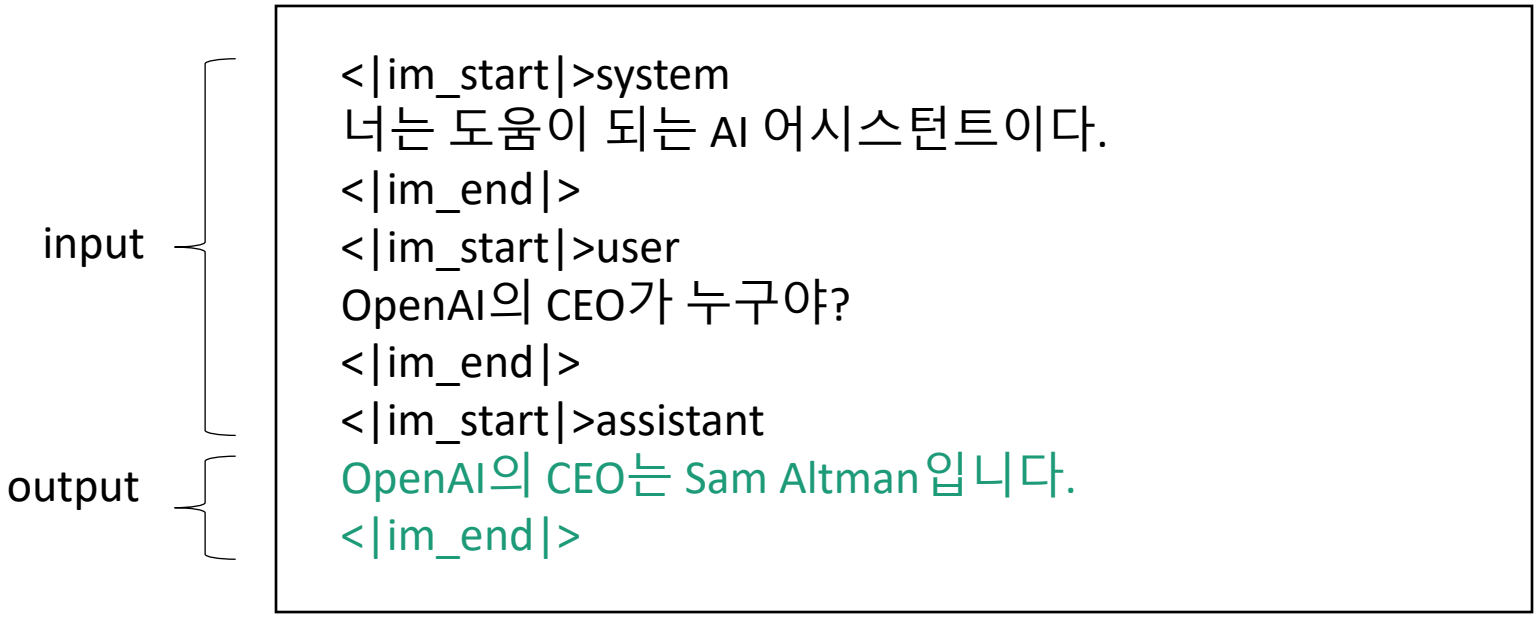
system instruction/prompt

messages

response

<|im_start|>system
너는 도움이 되는 AI 어시스턴트이다.
<|im_end|>
<|im_start|>user
OpenAI의 CEO가 누구야?
<|im_end|>
<|im_start|>assistant
OpenAI의 CEO는 Sam Altman입니다.
<|im_end|>

Prompt


input
output
<|im_start|>system
너는 도움이 되는 AI 어시스턴트이다.
<|im_end|>
<|im_start|>user
OpenAI의 CEO가 누구야?
<|im_end|>
<|im_start|>assistant
OpenAI의 CEO는 Sam Altman입니다.
<|im_end|>

Prompt

input

output

<|im_start|>user
“**이 영화 너무 재미 없다.**” 이 문장이 긍정이면 positive 부정이면 negative
<|im_end|>
<|im_start|>assistant
negative
<|im_end|>

Prompt

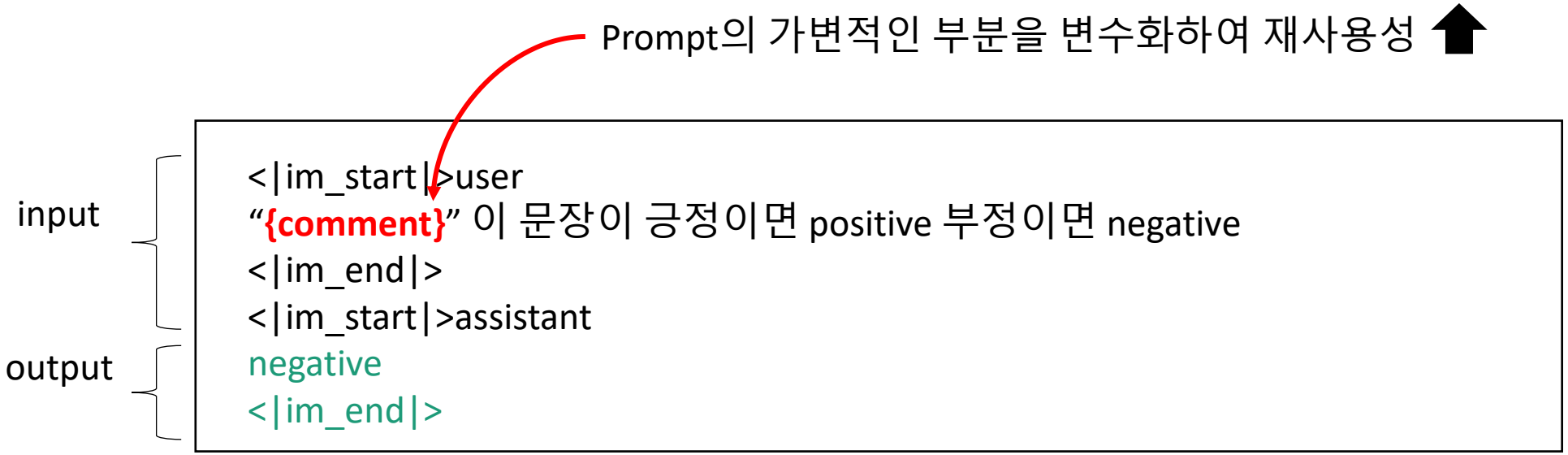
input / output

<|im_start|>user
“**이 영화 너무 재미 없다.**” 이 문장이 긍정이면 positive 부정이면 negative
<|im_end|>
<|im_start|>assistant
negative
<|im_end|>

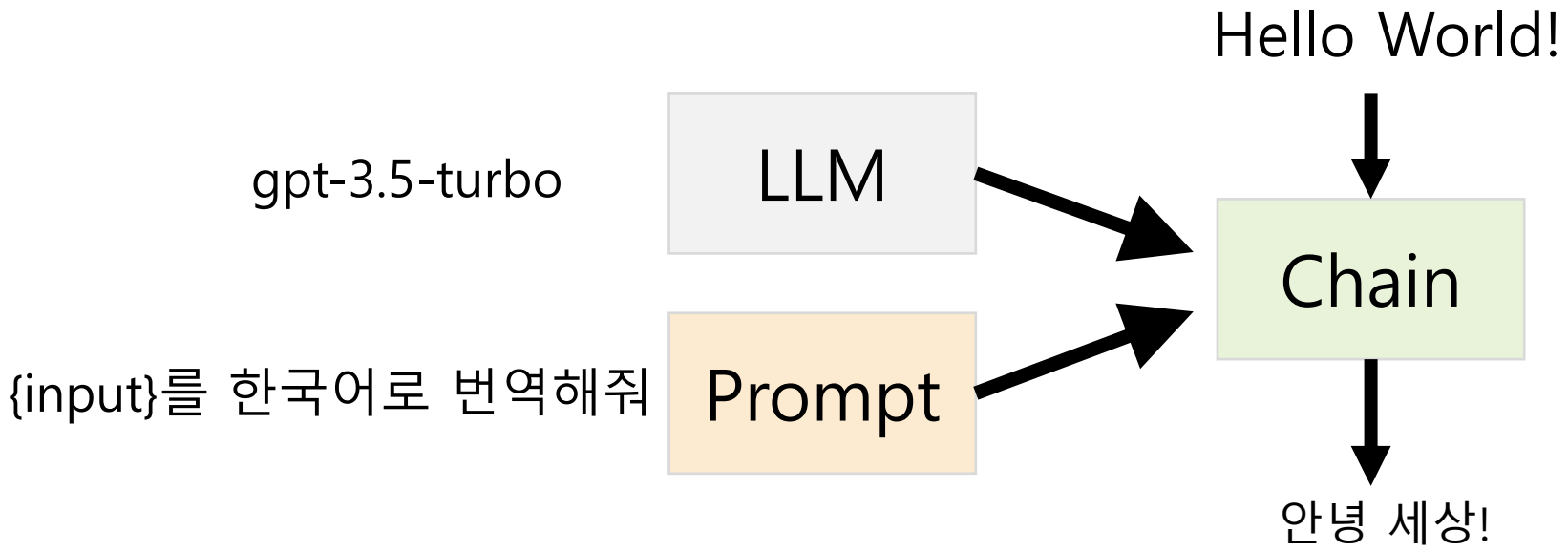
input / output

<|im_start|>user
“**이 영화 너무 재미있다.**” 이 문장이 긍정이면 positive 부정이면 negative
<|im_end|>
<|im_start|>assistant
positive
<|im_end|>

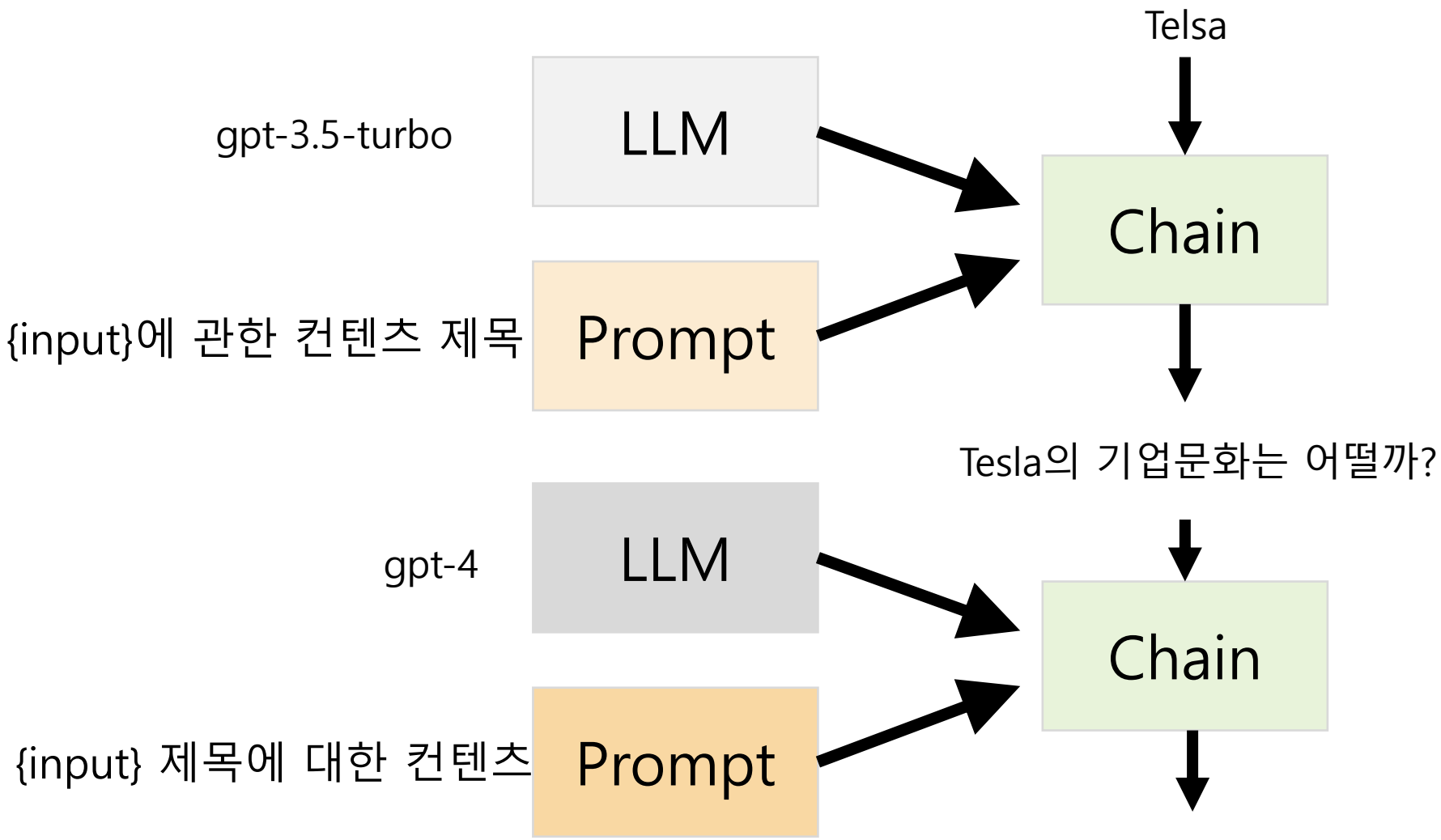
Prompt Template



Chain


Hello World!
gpt-3.5-turbo
LLM
Chain
{input}를 한국어로 번역해줘
Prompt
안녕 세상!

Chain


Telsa
gpt-3.5-turbo
LLM
Chain
{input}에 관한 컨텐츠 제목
Prompt
Tesla의 기업문화는 어떨까?
gpt-4
LLM
Chain
{input} 제목에 대한 컨텐츠
Prompt

LangChain

- LLM을 쉽게 호출하게 해주고 Prompt engineering과 Chaining을 도와주는 라이브러리

without LangChain

```python
from typing import List

import openai


prompt_template = "Tell me a short joke about {topic}"
client = openai.OpenAI()

def call_chat_model(messages: List[dict]) -> str:
    response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=messages,
    )
    return response.choices[0].message.content

def invoke_chain(topic: str) -> str:
    prompt_value = prompt_template.format(topic=topic)
    messages = [{"role": "user", "content": prompt_valu
    return call_chat_model(messages)

invoke_chain("ice cream")
```

with LangChain

```python
from langchain_core.runnables import RunnablePassthrou


prompt = ChatPromptTemplate.from_template(
    "Tell me a short joke about {topic}"
)
output_parser = StrOutputParser()
model = ChatOpenAI(model="gpt-3.5-turbo")
chain = (
    {"topic": RunnablePassthrough()}
    | prompt
    | model
    | output_parser
)

chain.invoke("ice cream")
```

Prompt Template 사용

without LangChain

```python
from typing import Iterator


def stream_chat_model(messages: List[dict]) -> Iterato
    stream = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=messages,
        stream=True,
    )
    for response in stream:
        content = response.choices[0].delta.content
        if content is not None:
            yield content

def stream_chain(topic: str) -> Iterator[str]:
    prompt_value = prompt.format(topic=topic)
    return stream_chat_model([{"role": "user", "conten


for chunk in stream_chain("ice cream"):
    print(chunk, end="", flush=True)
```

with LangChain

```python
for chunk in chain.stream("ice cream"):
    print(chunk, end="", flush=True)
```

Streaming

without LangChain

```python
from concurrent.futures import ThreadPoolExecutor


def batch_chain(topics: list) -> list:
    with ThreadPoolExecutor(max_workers=5) as executor
        return list(executor.map(invoke_chain, topics)

batch_chain(["ice cream", "spaghetti", "dumplings"])
```

with LangChain

```python
chain.batch(["ice cream", "spaghetti", "dumplings"])
```

Batch

without LangChain

```python
import anthropic

anthropic_template = f"Human:\n\n{prompt_template}\n\nA
anthropic_client = anthropic.Anthropic()

def call_anthropic(prompt_value: str) -> str:
    response = anthropic_client.completions.create(
        model="claude-2",
        prompt=prompt_value,
        max_tokens_to_sample=256,
    )
    return response.completion

def invoke_anthropic_chain(topic: str) -> str:
    prompt_value = anthropic_template.format(topic=top
    return call_anthropic(prompt_value)

invoke_anthropic_chain("ice cream")
```

with LangChain

```python
from langchain.chat_models import ChatAnthropic

anthropic = ChatAnthropic(model="claude-2")
anthropic_chain = (
    {"topic": RunnablePassthrough()}
    | prompt
    | anthropic
    | output_parser
)

anthropic_chain.invoke("ice cream")
```

다른 LLM API도 동일한 인터페이스로 사용 가능