LangChain: Models, Prompts and Output Parsers

Outline

- Direct API calls to OpenAI
- API calls through LangChain:
  - Prompts
  - Models
  - Output parsers

Get your OpenAI API Key

In [1]:
```
!pip install python-dotenv
!pip install openai
```

```
Collecting python-dotenv
  Using cached python_dotenv-1.0.0-py3-none-any.whl (19 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.0.0
Collecting openai
  Using cached openai-0.27.7-py3-none-any.whl (71 kB)
Requirement already satisfied: requests>=2.20 in /opt/homebrew/lib/python3.10/site-packages (from openai) (2.31.0)
Collecting tqdm (from openai)
  Using cached tqdm-4.65.0-py3-none-any.whl (77 kB)
Collecting aiohttp (from openai)
  Using cached aiohttp-3.8.4-cp310-cp310-macosx_11_0_arm64.whl (336 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/homebrew/lib/python3.10/site-packages (from request
s>=2.20->openai) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in /Users/abakh005/Library/Python/3.10/lib/python/site-packages (from r
equests>=2.20->openai) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/homebrew/lib/python3.10/site-packages (from requests>=2.2
0->openai) (2.0.2)
Requirement already satisfied: certifi>=2017.4.17 in /opt/homebrew/lib/python3.10/site-packages (from requests>=2.2
0->openai) (2023.5.7)
Requirement already satisfied: attrs>=17.3.0 in /Users/abakh005/Library/Python/3.10/lib/python/site-packages (from
aiohttp->openai) (22.1.0)
Collecting multidict<7.0,>=4.5 (from aiohttp->openai)
  Using cached multidict-6.0.4-cp310-cp310-macosx_11_0_arm64.whl (29 kB)
Collecting async-timeout<5.0,>=4.0.0a3 (from aiohttp->openai)
  Using cached async_timeout-4.0.2-py3-none-any.whl (5.8 kB)
Collecting yarl<2.0,>=1.0 (from aiohttp->openai)
  Using cached yarl-1.9.2-cp310-cp310-macosx_11_0_arm64.whl (62 kB)
Collecting frozenlist>=1.1.1 (from aiohttp->openai)
  Using cached frozenlist-1.3.3-cp310-cp310-macosx_11_0_arm64.whl (34 kB)
Collecting aiosignal>=1.1.2 (from aiohttp->openai)
  Using cached aiosignal-1.3.1-py3-none-any.whl (7.6 kB)
Installing collected packages: tqdm, multidict, frozenlist, async-timeout, yarl, aiosignal, aiohttp, openai
Successfully installed aiohttp-3.8.4 aiosignal-1.3.1 async-timeout-4.0.2 frozenlist-1.3.3 multidict-6.0.4 openai-0.
27.7 tqdm-4.65.0 yarl-1.9.2
```

In [2]:
```
import os
import openai

from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file
openai.api_key = os.environ['OPENAI_API_KEY']
```

Chat API : OpenAI

Let's start with a direct API calls to OpenAI.

In [3]:
```
def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=0,
    )
    return response.choices[0].message["content"]
```

In [4]:
```
get_completion("What is 1+1?")
```

Out[4]:
```
'As an AI language model, I can tell you that the answer to 1+1 is 2.'
```

In [5]:
```
customer_email = """
Arrr, I be fuming that me blender lid \
flew off and splattered me kitchen walls \
with smoothie! And to make matters worse,\
the warranty don't cover the cost of \
cleaning up me kitchen. I need yer help \
right now, matey!
"""
```

In [6]:
```
style = """American English \
in a calm and respectful tone
"""
```

In [7]:
```
prompt = f"""Translate the text \
that is delimited by triple backticks
into a style that is {style}.
text: ```{customer_email}```
"""

print(prompt)
```

```
Translate the text that is delimited by triple backticks
into a style that is American English in a calm and respectful tone
.
text: ```
Arrr, I be fuming that me blender lid flew off and splattered me kitchen walls with smoothie! And to make matters w
orse,the warranty don't cover the cost of cleaning up me kitchen. I need yer help right now, matey!
```
```

In [8]:
```
response = get_completion(prompt)
```

In [9]:
```
response
```

Out[9]:
```
'I am quite upset that my blender lid came off and caused my smoothie to splatter all over my kitchen walls. Additi
onally, the warranty does not cover the cost of cleaning up the mess. Would you be able to assist me, please? Thank
you kindly.'
```

## Chat API : LangChain

Let's try how we can do the same using LangChain.

In [10]:
```
!pip install --upgrade langchain
```
```

```
Collecting langchain
  Downloading langchain-0.0.189-py3-none-any.whl (975 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 975.6/975.6 kB 1.3 MB/s eta 0:00:00[31m1.6 MB/s eta 0:00:01
Requirement already satisfied: PyYAML>=5.4.1 in /opt/homebrew/lib/python3.10/site-packages (from langchain) (6.0)
Collecting SQLAlchemy<3,>=1.4 (from langchain)
  Using cached SQLAlchemy-2.0.15-cp310-cp310-macosx_11_0_arm64.whl (2.0 MB)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /opt/homebrew/lib/python3.10/site-packages (from langchain)
(3.8.4)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in /opt/homebrew/lib/python3.10/site-packages (from lang
chain) (4.0.2)
Collecting dataclasses-json<0.6.0,>=0.5.7 (from langchain)
  Using cached dataclasses_json-0.5.7-py3-none-any.whl (25 kB)
Collecting numexpr<3.0.0,>=2.8.4 (from langchain)
  Using cached numexpr-2.8.4-cp310-cp310-macosx_11_0_arm64.whl (89 kB)
Collecting numpy<2,>=1 (from langchain)
  Using cached numpy-1.24.3-cp310-cp310-macosx_11_0_arm64.whl (13.9 MB)
Collecting openapi-schema-pydantic<2.0,>=1.2 (from langchain)
  Using cached openapi_schema_pydantic-1.2.4-py3-none-any.whl (90 kB)
Collecting pydantic<2,>=1 (from langchain)
  Using cached pydantic-1.10.8-cp310-cp310-macosx_11_0_arm64.whl (2.5 MB)
Requirement already satisfied: requests<3,>=2 in /opt/homebrew/lib/python3.10/site-packages (from langchain) (2.31.
0)
Collecting tenacity<9.0.0,>=8.1.0 (from langchain)
  Using cached tenacity-8.2.2-py3-none-any.whl (24 kB)
Requirement already satisfied: attrs>=17.3.0 in /Users/abakh005/Library/Python/3.10/lib/python/site-packages (from
aiohttp<4.0.0,>=3.8.3->langchain) (22.1.0)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /opt/homebrew/lib/python3.10/site-packages (from aio
http<4.0.0,>=3.8.3->langchain) (3.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /opt/homebrew/lib/python3.10/site-packages (from aiohttp<4.0.
0,>=3.8.3->langchain) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in /opt/homebrew/lib/python3.10/site-packages (from aiohttp<4.0.0,>=
3.8.3->langchain) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in /opt/homebrew/lib/python3.10/site-packages (from aiohttp<4.0.0,
>=3.8.3->langchain) (1.3.3)
Requirement already satisfied: aiosignal>=1.1.2 in /opt/homebrew/lib/python3.10/site-packages (from aiohttp<4.0.0,>
=3.8.3->langchain) (1.3.1)
Collecting marshmallow<4.0.0,>=3.3.0 (from dataclasses-json<0.6.0,>=0.5.7->langchain)
  Using cached marshmallow-3.19.0-py3-none-any.whl (49 kB)
Collecting marshmallow-enum<2.0.0,>=1.5.1 (from dataclasses-json<0.6.0,>=0.5.7->langchain)
  Using cached marshmallow_enum-1.5.1-py2.py3-none-any.whl (4.2 kB)
Collecting typing-inspect>=0.4.0 (from dataclasses-json<0.6.0,>=0.5.7->langchain)
  Using cached typing_inspect-0.9.0-py3-none-any.whl (8.8 kB)
Requirement already satisfied: typing-extensions>=4.2.0 in /opt/homebrew/lib/python3.10/site-packages (from pydanti
c<2,>=1->langchain) (4.6.3)
Requirement already satisfied: idna<4,>=2.5 in /Users/abakh005/Library/Python/3.10/lib/python/site-packages (from r
equests<3,>=2->langchain) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/homebrew/lib/python3.10/site-packages (from requests<3,>=
2->langchain) (2.0.2)
Requirement already satisfied: certifi>=2017.4.17 in /opt/homebrew/lib/python3.10/site-packages (from requests<3,>=
2->langchain) (2023.5.7)
Requirement already satisfied: packaging>=17.0 in /Users/abakh005/Library/Python/3.10/lib/python/site-packages (fro
m marshmallow<4.0.0,>=3.3.0->dataclasses-json<0.6.0,>=0.5.7->langchain) (21.3)
Collecting mypy-extensions>=0.3.0 (from typing-inspect>=0.4.0->dataclasses-json<0.6.0,>=0.5.7->langchain)
  Using cached mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /Users/abakh005/Library/Python/3.10/lib/python/site-pack
ages (from packaging>=17.0->marshmallow<4.0.0,>=3.3.0->dataclasses-json<0.6.0,>=0.5.7->langchain) (3.0.9)
Installing collected packages: tenacity, SQLAlchemy, pydantic, numpy, mypy-extensions, typing-inspect, openapi-sche
ma-pydantic, numexpr, marshmallow, marshmallow-enum, dataclasses-json, langchain
Successfully installed SQLAlchemy-2.0.15 dataclasses-json-0.5.7 langchain-0.0.189 marshmallow-3.19.0 marshmallow-en
um-1.5.1 mypy-extensions-1.0.0 numexpr-2.8.4 numpy-1.24.3 openapi-schema-pydantic-1.2.4 pydantic-1.10.8 tenacity-8.
2.2 typing-inspect-0.9.0
```

Model

In [24]:
```
from langchain.chat_models import ChatOpenAI
```

In [25]:
```
# To control the randomness and creativity of the generated
# text by an LLM, use temperature = 0.0
chat = ChatOpenAI(temperature=0.0)
chat
```

Out[25]:
```
ChatOpenAI(verbose=False, callbacks=None, callback_manager=None, client=<class 'openai.api_resources.chat_completio
n.ChatCompletion'>, model_name='gpt-3.5-turbo', temperature=0.0, model_kwargs={}, openai_api_key=None, openai_api_b
ase=None, openai_organization=None, request_timeout=None, max_retries=6, streaming=False, n=1, max_tokens=None)
```

Prompt template

In [13]:
```
template_string = """Translate the text \
that is delimited by triple backticks \
into a style that is {style}. \
text: ```{text}```
"""
```

In [14]:
```
from langchain.prompts import ChatPromptTemplate

prompt_template = ChatPromptTemplate.from_template(template_string)
```

In [15]:
```
prompt_template
```

Out[15]:
```
ChatPromptTemplate(input_variables=['text', 'style'], output_parser=None, partial_variables={}, messages=[HumanMess
agePromptTemplate(prompt=PromptTemplate(input_variables=['style', 'text'], output_parser=None, partial_variables=
{}, template='Translate the text that is delimited by triple backticks into a style that is {style}. text: ```{tex
t}```\n', template_format='f-string', validate_template=True), additional_kwargs={})])
```

In [16]:
```
prompt_template.messages[0].prompt
```

Out[16]:
```
PromptTemplate(input_variables=['style', 'text'], output_parser=None, partial_variables={}, template='Translate the
text that is delimited by triple backticks into a style that is {style}. text: ```{text}```\n', template_format='f-
string', validate_template=True)
```

In [17]:
```
prompt_template.messages[0].prompt.input_variables
```

Out[17]:
```
['style', 'text']
```

In [18]:
```
customer_style = """American English \
in a calm and respectful tone
"""
```

In [19]:
```
customer_email = """
Arrr, I be fuming that me blender lid \
flew off and splattered me kitchen walls \
with smoothie! And to make matters worse, \
the warranty don't cover the cost of \
cleaning up me kitchen. I need yer help \
right now, matey!
"""
```

In [20]:
```
customer_messages = prompt_template.format_messages(
                    style=customer_style,
                    text=customer_email)
```

In [21]:
```
print(type(customer_messages))
print(type(customer_messages[0]))
```

```
<class 'list'>
<class 'langchain.schema.HumanMessage'>
```

In [22]:
```
print(customer_messages[0])
```

```
content="Translate the text that is delimited by triple backticks into a style that is American English in a calm a
nd respectful tone\n. text: ```\nArrr, I be fuming that me blender lid flew off and splattered me kitchen walls wit
h smoothie! And to make matters worse, the warranty don't cover the cost of cleaning up me kitchen. I need yer help
right now, matey!\n```\n" additional_kwargs={} example=False
```

In [26]:
```
# Call the LLM to translate to the style of the customer message
customer_response = chat(customer_messages)
```

In [27]:
```
print(customer_response.content)
```

```
I'm really frustrated that my blender lid flew off and made a mess of my kitchen walls with smoothie. To add to my
frustration, the warranty doesn't cover the cost of cleaning up my kitchen. Can you please help me out, friend?
```

In [28]:
```
service_reply = """Hey there customer, \
the warranty does not cover \
cleaning expenses for your kitchen \
because it's your fault that \
you misused your blender \
by forgetting to put the lid on before \
starting the blender. \
Tough luck! See ya!
"""
```

In [29]:
```
service_style_pirate = """\
a polite tone \
```

```
that speaks in English Pirate\
"""
```

In [30]:
```
service_messages = prompt_template.format_messages(
    style=service_style_pirate,
    text=service_reply)

print(service_messages[0].content)
```

```
Translate the text that is delimited by triple backticks into a style that is a polite tone that speaks in English
Pirate. text: ```Hey there customer, the warranty does not cover cleaning expenses for your kitchen because it's yo
ur fault that you misused your blender by forgetting to put the lid on before starting the blender. Tough luck! See
ya!
```
```

In [31]:
```
service_response = chat(service_messages)
print(service_response.content)
```

```
Ahoy there, me hearty customer! I be sorry to inform ye that the warranty be not coverin' the expenses o' cleaning
yer galley, as it be yer own fault fer misusin' yer blender by forgettin' to put the lid on afore startin' it. Aye,
tough luck! Farewell and may the winds be in yer favor!
```

Output Parsers

Let's start with defining how we would like the LLM output to look like:

In [32]:
```
{
  "gift": False,
  "delivery_days": 5,
  "price_value": "pretty affordable!"
}
```

Out[32]:
```
{'gift': False, 'delivery_days': 5, 'price_value': 'pretty affordable!'}
```

In [33]:
```
customer_review = """\
This leaf blower is pretty amazing.  It has four settings:\
candle blower, gentle breeze, windy city, and tornado. \
It arrived in two days, just in time for my wife's \
anniversary present. \
I think my wife liked it so much she was speechless. \
So far I've been the only one using it, and I've been \
using it every other morning to clear the leaves on our lawn. \
It's slightly more expensive than the other leaf blowers \
out there, but I think it's worth it for the extra features.
"""

review_template = """\
For the following text, extract the following information:

gift: Was the item purchased as a gift for someone else? \
Answer True if yes, False if not or unknown.

delivery_days: How many days did it take for the product \
to arrive? If this information is not found, output -1.

price_value: Extract any sentences about the value or price,\
and output them as a comma separated Python list.

Format the output as JSON with the following keys:
gift
delivery_days
price_value

text: {text}
"""
```

In [34]:
```
from langchain.prompts import ChatPromptTemplate

prompt_template = ChatPromptTemplate.from_template(review_template)
print(prompt_template)
```
```

```
input_variables=['text'] output_parser=None partial_variables={} messages=[HumanMessagePromptTemplate(prompt=Prompt
Template(input_variables=['text'], output_parser=None, partial_variables={}, template='For the following text, extr
act the following information:\n\ngift: Was the item purchased as a gift for someone else? Answer True if yes, Fals
e if not or unknown.\n\ndelivery_days: How many days did it take for the product to arrive? If this information is
not found, output -1.\n\nprice_value: Extract any sentences about the value or price,and output them as a comma sep
arated Python list.\n\nFormat the output as JSON with the following keys:\ngift\ndelivery_days\nprice_value\n\ntex
t: {text}\n', template_format='f-string', validate_template=True), additional_kwargs={})]
```

In [35]:
```
messages = prompt_template.format_messages(text=customer_review)
chat = ChatOpenAI(temperature=0.0)
response = chat(messages)
print(response.content)
```

```
{
    "gift": true,
    "delivery_days": 2,
    "price_value": ["It's slightly more expensive than the other leaf blowers out there, but I think it's worth it
for the extra features."]
}
```

In [36]:
```
type(response.content)
```

Out[36]:
```
str
```

In [37]:
```
# You will get an error by running this line of code
# because'gift' is not a dictionary
# 'gift' is a string
response.content.get('gift')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[37], line 4
      1 # You will get an error by running this line of code
      2 # because'gift' is not a dictionary
      3 # 'gift' is a string
----> 4 response.content.get('gift')

AttributeError: 'str' object has no attribute 'get'
```

Parse the LLM output string into a Python dictionary

In [38]:
```
from langchain.output_parsers import ResponseSchema
from langchain.output_parsers import StructuredOutputParser
```

In [39]:
```
gift_schema = ResponseSchema(name="gift",
                             description="Was the item purchased\
                             as a gift for someone else? \
                             Answer True if yes,\
                             False if not or unknown.")
delivery_days_schema = ResponseSchema(name="delivery_days",
                                      description="How many days\
                                      did it take for the product\
                                      to arrive? If this \
                                      information is not found,\
                                      output -1.")
price_value_schema = ResponseSchema(name="price_value",
                                    description="Extract any\
                                    sentences about the value or \
                                    price, and output them as a \
                                    comma separated Python list.")

response_schemas = [gift_schema,
                    delivery_days_schema,
                    price_value_schema]
```

In [40]:
```
output_parser = StructuredOutputParser.from_response_schemas(response_schemas)
```

In [41]:
```
format_instructions = output_parser.get_format_instructions()
```

In [42]:
```
print(format_instructions)
```

The output should be a markdown code snippet formatted in the following schema, including the leading and trailing "\`\`\`json" and "\`\`\`":

```json
{
        "gift": string  // Was the item purchased                              as a gift for someone else?
Answer True if yes,                              False if not or unknown.
        "delivery_days": string  // How many days                                          did it take for the product
to arrive? If this                                        information is not found,
output -1.
        "price_value": string  // Extract any                                        sentences about the value or
price, and output them as a                                    comma separated Python list.
}
```

In [43]:
```
review_template_2 = """\
For the following text, extract the following information:

gift: Was the item purchased as a gift for someone else? \
Answer True if yes, False if not or unknown.

delivery_days: How many days did it take for the product\
to arrive? If this information is not found, output -1.

price_value: Extract any sentences about the value or price,\
and output them as a comma separated Python list.

text: {text}

{format_instructions}
"""

prompt = ChatPromptTemplate.from_template(template=review_template_2)

messages = prompt.format_messages(text=customer_review,
                                  format_instructions=format_instructions)
```

In [44]:
```
print(messages[0].content)
```

```
For the following text, extract the following information:

gift: Was the item purchased as a gift for someone else? Answer True if yes, False if not or unknown.

delivery_days: How many days did it take for the productto arrive? If this information is not found, output -1.

price_value: Extract any sentences about the value or price,and output them as a comma separated Python list.

text: This leaf blower is pretty amazing.  It has four settings:candle blower, gentle breeze, windy city, and torna
do. It arrived in two days, just in time for my wife's anniversary present. I think my wife liked it so much she wa
s speechless. So far I've been the only one using it, and I've been using it every other morning to clear the leave
s on our lawn. It's slightly more expensive than the other leaf blowers out there, but I think it's worth it for th
e extra features.


The output should be a markdown code snippet formatted in the following schema, including the leading and trailing
"\`\`\`json" and "\`\`\`":

```json
{
        "gift": string  // Was the item purchased                              as a gift for someone else?
Answer True if yes,                              False if not or unknown.
        "delivery_days": string  // How many days                                          did it take for the product
to arrive? If this                                        information is not found,
output -1.
        "price_value": string  // Extract any                                        sentences about the value or
price, and output them as a                                    comma separated Python list.
}
```
```

In [45]:
```
response = chat(messages)
```

In [46]:
```
print(response.content)
```

```
```json
{
        "gift": true,
        "delivery_days": "2",
        "price_value": ["It's slightly more expensive than the other leaf blowers out there, but I think it's worth it for the extra features."]
}
```
```

In [47]:
```
output_dict = output_parser.parse(response.content)
```

In [48]:
```
output_dict
```

Out[48]:
```
{'gift': True,
 'delivery_days': '2',
 'price_value': ["It's slightly more expensive than the other leaf blowers out there, but I think it's worth it for the extra features."]}
```

In [49]:
```
type(output_dict)
```

Out[49]:
```
dict
```

In [50]:
```
output_dict.get('delivery_days')
```

Out[50]:
```
'2'
```