First Completion

13 min

We can initiate a chat completion with an AI model by making a request to the API. An API request can contain a series of messages. Each of the messages in the series of messages sent via the API request can be one of the following types:

- **System**: this role is optional and typically provides context for how the AI model should respond to questions. We will cover it in more detail in the next exercise.
- **User**: this role represents the prompts and messages the user sends. The user message may be a question, a request, or a follow-up response to provide additional context to the AI model. This message is commonly known as a "prompt" for the AI model. For instance, a user message could be:

"Tell me how many seconds there are in a day in a Shakespearean tone."

- **Assistant**: this role follows the user message and represents the AI model's reply to the prompt. For instance, an assistant message could be:

"Hark, tis a question of time, where a day be composed of four and twenty hours, and each hour possesseth three thousand six hundred seconds. Thus, a day unfoldeth in the span of eighty six thousand four hundred seconds, as the sun traverseth the sky."

The assistant message can also be sent with a user message to shape the type of reply we would want the model to respond with for future prompts.

To start a chat we will use the chat.completions.create() method of the OpenAI instance. This method has 2 required arguments:

- model: the model name, either "gpt-3.5-turbo" or "gpt-4-turbo-preview" at the time of this writing
- messages: a list of message dictionaries each with the following keys:
    - role: the role of the message, "system", "user" or "assitant"
    - content: the text of the message

Use the OpenAI instance in client to run the following code and start a chat:

```
from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {
      "role": "user",
      "content": "Tell me how many seconds there are in a day in a Shakespearean tone?"
    }
  ])
```

You'll notice the 2 arguments, model and messages. The messages list contains one dictionary with "role": "user". The "content" string will be the prompt for the AI model.

Running this code will return a ChatCompletion object that contains the model's response. Let's look at what the returned response looks like:

Chatbot: ChatCompletion(id='chatcmpl-8km20vYxq1ecQJ0yKI1agikEnEeet', choices=[Choice(finish_reason='stop', index=0, logprobs=None, message=ChatCompletionMessage(content='Hark, tis a question of time, where a day be composed of four and twenty hours, and each hour possesseth three thousand six hundred seconds. Thus, a day unfoldeth in the span of eighty six thousand four hundred seconds, as the sun traverseth the sky.', role='assistant', function_call=None, tool_calls=None))], created=1706158412, model='gpt-3.5-turbo-0613', object='chat.completion', system_fingerprint=None, usage=CompletionUsage(completion_tokens=57, prompt_tokens=87, total_tokens=144))

The response object represents the completion of a chat interaction with the GPT-3.5 Turbo model. It provides information like a unique chat ID, created timestamp, the model used, and the number of input and output tokens used in the conversation. It also provides a list of Choice() objects that indicate how the completion ended, the content of the reply, and more.

We can access and print out the content of the reply by running the following:

print(response.choices[0].message.content)

Output (Model Response):

Hark, tis a question of time, where a day be composed of four and twenty hours, and each hour possesseth three thousand six hundred seconds. Thus, a day unfoldeth in the span of eighty six thousand four hundred seconds, as the sun traverseth the sky.

Note that the model replies listed throughout this lesson are the verbose replies from the model but may be slightly trimmed in length.

Let's practice a chat completion within the code editor.

Instructions

1. Checkpoint 1 Passed

1.

Create a chat completion using either the gpt-3.5-turbo model or the gpt-4-turbo-preview model with a "user" message of your choosing.

Save the chat completion response in a variable called response.

 Use the following syntax:

```
response = client.chat.completions.create(
  model=MODEL_STRING,
  messages=[
    {
      "role": ROLE_STRING,
      "content": PROMPT_STRING
    }
  ])
```

 Checkpoint 2 Passed

2.

Now, retrieve the content from the model’s response. Feel free to output it to the terminal.

 Use the following syntax:

response.choices[0].message.content

 Checkpoint 3 Passed

3.

If you want to, change the model to the one you didn’t use before and see if you’re prompt receives a different completion.

Change the following argument:

model=MODEL_STRING

script.py

```
from openai import OpenAI, types


client = OpenAI()


# Your code below:
response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {
      "role": "user",
      "content": "Is the emerging of generative artificial intelligence affecting the possibilities of being hired of full stack web developers?"
    }
  ])


print(response.choices[0].message.content)
```

Response

The emerging of generative artificial intelligence (AI) may have some impact on the possibilities of being hired as a full stack web developer, but it is not necessarily negative. Here are a few points to consider:

1. Enhanced productivity: Generative AI systems can assist full stack web developers by automating repetitive tasks, such as code generation or testing. This can boost productivity and allow developers to focus on more complex and high-value tasks.

2. New opportunities: With the advancements in AI, new domains and industries are emerging. Full stack web developers can explore and leverage AI technologies to develop innovative solutions in areas like machine learning, natural language processing, or computer vision.

3. Evolving skill set: As AI technologies progress, full stack web developers may need to enhance their skills to stay relevant. Understanding the principles and applications of AI, machine learning frameworks, and integrating AI components into web applications can become valuable additions to a developer's skill set.

4. Demand for problem-solving abilities: While AI technologies are advancing, they cannot replace the problem-solving abilities, creativity, and adaptability of human developers. Full stack web developers with a deep understanding of business requirements, user experience, and architecture design will continue to be in demand.

Ultimately, the impact of generative AI on the possibilities for full stack web developers depends on how developers adapt to this emerging technology. Embracing AI and developing complementary skills can help developers remain competitive and open up new opportunities in the evolving landscape.