User Prompting

10 min

Providing examples within our input prompts can help generate an output that best suits our needs. All of our user prompts in this lesson so far have been *zero-shot prompts*, which are broad prompts where the model must use its general understanding and capabilities to generate a response.

In the previous exercise, we asked our model for travel ideas for a family of four. The model's response was lengthy in suggesting eight different Carribean destinations as it had to generate its response based on its general knowledge and understanding of the Carribean.

Now, imagine that we changed our prompt to constrain the destination to include waterfall hiking:

```
response = client.chat.completions.create(
  model="gpt-3.5-turbo",

messages=[
   {
     "role": "system",
     "content": "You are a friendly travel guide excited to help users travel the Caribbean. Your responses should only include destinations that are in the Caribbean."
   },
   {
     "role": "user",
     "content": "Give me some travel ideas for a family of four where there are also waterfall hiking excursions."
   }
 ])

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

Output (Model Response):

Sure! The Caribbean offers plenty of travel options for families looking to combine outdoor adventure with waterfall hiking excursions. Here are a few destinations to consider:

1. Dominica: Known as the "Nature Isle of the Caribbean," Dominica boasts lush rainforests and numerous waterfalls. Visit the Trafalgar Falls or hike the challenging but rewarding Boiling Lake trail.

2. Grenada: This beautiful island is home to the Annandale Falls, where you can experience a short, family-friendly hike through tropical foliage to reach the picturesque cascade. Additionally, you can explore other waterfalls like Seven Sisters and Concord Falls.

Here, we introduce additional criteria and context (specifically, waterfall hiking) for the model when recommending its Caribbean destinations. The model's response is now more specific.

We can also use *few-shot* prompts. Few-shot prompting is a specific technique within prompt engineering that involves providing a language model with examples or demonstrations of the desired behavior to guide its replies. It allows users to specify the task or context they want the model to focus on.

Here is a user prompt that gives the model some formatting guidance on how the Caribbean destination list should be returned:

```
response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {
      "role": "system",
      "content": "You are a friendly travel guide excited to help users travel the Caribbean. Your responses should only include destinations that are in the Caribbean."
    },
    {
      "role": "user",
      "content": 'Give me some travel ideas for a family of four. I want the results to be in CSV format: \n'
            'Destination,AirportCode,PopulationSize,Description.\n'
            'Bermuda,BDA,"63,856","Bermuda, a picturesque island paradise in the North Atlantic Ocean, captivates travelers with its pink sand beaches, crystal-clear turquoise waters, and charming pastel-colored architecture."'
    }
  ]
)
print(response.choices[0].message.content)
```

Output (Model Response):

Aruba,AUA,"106,766","Aruba, a Caribbean island just off the coast of Venezuela, is known for its stunning white-sand beaches, crystal-clear blue waters, and year-round warm weather. It offers a wide range of family-friendly activities such as snorkeling, scuba diving, and exploring natural attractions like Arikok National Park."

Jamaica,MBJ,"2,890,299","Jamaica, the birthplace of reggae music, invites families to enjoy its beautiful beaches, vibrant local culture, and thrilling adventure activities. From stunning waterfalls and lush rainforests to friendly locals and delicious cuisine, Jamaica has something for everyone."

The model used the example provided within the user prompt to return properly formatted and ordered data. Specifying the formatting is also great for users who need to export the data into a file with a specific format. We could alternatively modify our prompt to request the data in a bulleted list instead of in CSV format.

Few-show prompts are helpful in providing instructions, and for specifying tasks for the model. It is best to use this technique when:

- We have complex tasks that the model may not understand
- We want customized or specifically formatted replies
- We want the model to be able to adapt to unique or edge cases
- We want to narrow the focus of the model output

Instructions

1. Checkpoint 1 Passed

1.

Create and run a chat completion with two prompts within the messages list:

- A system prompt that instructs the model to find patterns in data and to solve the next value in the sequence.
- The following user prompt: A=1, D=4, J=10; What is T?

Be sure to put each prompt in a dictionary with the keys role and content.

Use the following syntax:

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

script.py

```
from openai import OpenAI


client = OpenAI()


response = client.chat.completions.create(
 model="gpt-3.5-turbo",
 messages=[
  # Your code below
   {
    "role": "system",
    "content": "Find patterns in data and solve the next value in the sequence."
   },
   {
    "role": "user",
    "content": "A=1, D=4, J=10; What is T?"
   }
 ]
```

)

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

Response

To find the pattern in the given sequence (A=1, D=4, J=10), we can look at the position of the letters in the alphabet and their corresponding values.

A = 1 (1st letter in the alphabet)

D = 4 (4th letter in the alphabet)

J = 10 (10th letter in the alphabet)

We can see that the values are increasing by consecutive numbers: 1, 2, 3, 4, ...

Using this pattern, we can determine the value for the letter T (20th letter in the alphabet):

T = 1 + 2 + 3 + 4 + ... + 20

To find this sum, we can use the formula for the sum of an arithmetic series:

Sum = (n/2)(first term + last term)

In this case, n = 20, the first term = 1, and the last term = 20.

Sum = (20/2)(1 + 20)

= 10(21)

= 210

Therefore, T would equal 210.