Python tutorial: How to use the OpenAI API?

Learning Curve

Things you can do with the OpenAI API

In this tutorial, we will explore how to use the OpenAI API to perform various tasks using Python. The OpenAI API allows you to access powerful natural language processing models, such as ChatGPT, which can be used for a wide range of applications, including chatbots, language translation, text completion, and more.

We'll look at just two different use cases: text completion and image generation, although it's possible to do much more.

Prerequisites

Before we begin, make sure you have the following prerequisites:

- Python 3.6 or higher installed
- OpenAI Python library (openai) installed. You can install it using pip:

```
pip install openai
```

Libraries

```python
import openai #openAI api
import requests #get content of the generated image
import matplotlib.pyplot as plt #display generated image
from PIL import Image #create an image object
import io #make the image readable
print("hello world")
```

```
## hello world
```

Define your API key

Since the use of the API is not free, you have to create an OpenAI account and add a credit card. However, the text completion is cheap (0.002$/1k token with GPT-3.5) and the image generation is ok if you don't abuse of it (0.02$/image with DALLE). These values may change depending on the model you use.

```python
openai.api_key = "Your api key goes here"
```

Text completion with GPT-3.5

Define parameters for the model

They are mutliple parameters that you can modify when working with GPTs. It's important to know what they do in order to customize as much as possible the way the model should behave in your use case. The description below are from the API reference. I only put the less context-specific parameters here and let you check the documentation if you are looking for something in particular.

- `temperature` (between 0 and 2, default 1): What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.
- `top_p` (between 0 and 1, default 1): An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered.
- `n` (default 1): How many chat completion choices to generate for each input message.
- `presence_penalty` (between -2 and 2, default 0): Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.
- `frequency_penalty` (between -2 and 2, default 0): Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.

To get an idea of the importance of these parameters, I recommend you try out different combinations. Start with a simple, deterministic model, then modify it until the randomness is at its maximum.

NB: it is not recommended to set the `temperature` AND the `top_p` at the same time.

```
#Define parameters for the model
temperature = 1.5
frequency_penalty = 0.5
presence_penalty = 0.5
model = "gpt-3.5-turbo" #you can choose another model
```

Define the inputs

Now we want to define 2 different inputs. The `system_msg` must indicate how the model should behave. The `prompt` is just the message to which the model should *respond*.

```
system_msg = """You're an AI assistant.
              All your answers start with
              'That is a good question my friend...'."""

prompt = "What are the main ideas behind quantum physics?"

messages=[{"role": "system","content": system_msg},
          {"role": "user","content": prompt}]
```

Get the completion

```
#call API
completion = openai.ChatCompletion.create(model=model,
                             messages=messages,
                             temperature=temperature,
                             frequency_penalty=frequency_penalty,
                             presence_penalty=presence_penalty)

#get output
output = completion["choices"][0]["message"]["content"]
```

```
## Input:
##   What are the main ideas behind quantum physics?
```

```
## Output:
##   That is a good question my friend! The main ideas behind quantum physics involve the
##  study of the behavior and interactions of subatomic particles, such as atoms and ph
## otons. Quantum physics explains that particles can exist as waves and exhibit both p
## article-like and wave-like properties, allowing for concepts such as entanglement, s
## uperposition and indeterminacy.
##
## It brings about two principles that changed everyth
## ing in our undserstanding, Heisenberg's uncertainty principle about how you can’t si
## multaneously know anything precisely both where its at and where it’d going at the s
## ame time.
## Another is Max Planck discovered that energy exists in discreet chunks or
## packages rather than just floatind around evenly on all scales of reality. This chan
## ges how also see things like wavelengths make photons behave sometimes more like bit
## s thenwaves., black body circulation has mostly contributed to unnderstandinghow hot
##  things graved emits infrared radiation really creating anything above bog ick reali
## tive life physics to incredible new material designing types beneath hughsmagnificat
## ion containing vitural honey cuts in qubic graphite each inspiring remarkages everyd
## ay tasks difficult comingof fuels incorporating batteries heating spesffuc germcinid
##  bases nanoparticles polarizerfilters routers magnobatics dnasonomatic studies trans
## formativeencryption behind expensive equipment using empty dyak cold power settings.
```

Image generation with DALLE

Define prompt and parameters

```
#Define input
prompt = "high-quality image of the cutest cat in the world"
size = "1024x1024" #256x256, 512x512, or 1024x1024
n = 2 #number of image to generate
```

Get the images

```
#call API
response = openai.Image.create(
            prompt=prompt,
            n=n,
            size=size)

#get first image
image_url1 = response['data'][0]['url']
image_data1 = requests.get(image_url1).content
img1 = Image.open(io.BytesIO(image_data1))

#get second image
image_url2 = response['data'][1]['url']
image_data2 = requests.get(image_url2).content
img2 = Image.open(io.BytesIO(image_data2))
```

Display the images

```
## (-0.5, 1023.5, 1023.5, -0.5)
```

```
## (-0.5, 1023.5, 1023.5, -0.5)
```

Prompt:
high-quality image of the cutest cat in the world