Domina el API de OpenAI - De Principiante a Experto

Tutoriales prácticos usando Python

Edwin John Fredy Reyes Aguirre

2025-07-01

Table of contents

1 Domina el API de OpenAI - De Principiante a Experto

Tutoriales prácticos usando Python

Bienvenido

Este libro recopila una serie de tutoriales para aprender a usar la API de OpenAI paso a paso.

- Está escrito en Python
- Puedes ver los resultados directamente
- Es fácil de exportar a HTML y PDF

Part I

Tutoriales

2 Uso básico de la API de OpenAI

Este tutorial muestra cómo usar la API de OpenAI con la biblioteca oficial

2.1 Configuración Inicial

Importamos las librerías necesarias y cargamos las variables de entorno.

```
from openai import OpenAI
import os
from dotenv import load_dotenv

load_dotenv() # Carga el archivo de las variables de entorno

api_key = os.getenv('OPENAI_API_KEY')
client = OpenAI(api_key=api_key)
```

2.2 Solicitud a la API

Usamos el modelo gpt-4o-mini para enviar un mensaje y obtener una respuesta.

```
completion = client.chat.completions.create(
    model='gpt-4o-mini',
    messages=[
        {
            'role': 'user',
            'content': '¡Hola! ¿Me ayudas a aprender sobre la API de OpenAI'
        }
    ]
)
```

2.3 Ver la respuesta

Extraemos el contenido de la respuesta del asistente.

```
completion.choices[0].message
```

```
ChatCompletionMessage(content='¡Hola! Claro, estaré encantado de ayudarte a aprender sobre la
```

2.4 Mostrar la respuesta en formato Markdown

Utilizamos IPyhton para mostrar el contenido como texto formateado.

```
from IPython.display import display, Markdown
display(Markdown(completion.choices[0].message.content))
```

¡Hola! Claro, estaré encantado de ayudarte a aprender sobre la API de OpenAI. La API permite acceder a los modelos de lenguaje de OpenAI, como ChatGPT, para realizar una variedad de tareas como generación de texto, comprensión de lenguaje, chatbots, traducir texto, y más.

2.4.1 1. Introducción a la API de OpenAI

La API se basa en REST y permite a los desarrolladores interactuar con los modelos de OpenAI mediante solicitudes HTTP. Puedes enviar un texto y recibir una respuesta generada por el modelo.

2.4.2 2. Características Principales

- **Modelos Disponibles:** Diferentes modelos como `gpt-3.5-turbo` para tareas de generación de texto y `davinci` para tareas más complejas.
- **Personalización:** Puedes ajustar la "temperatura" para controlar la aleatoriedad de las respuestas.
- **Control de longitud:** Es posible establecer un límite en el número de tokens en la respuesta generada.

2.4.3 3. Cómo Empezar

- **Registro:** Primero, necesitas crear una cuenta en el sitio de OpenAI y obtener una clave API.
- **Configuración del Entorno:** Utiliza bibliotecas como `requests` para Python o herramientas en tu lenguaje de programación preferido para realizar solicitudes HTTP a la API.

2.4.4 4. Ejemplo de Uso

A continuación, te muestro un ejemplo básico utilizando Python:

```
import openai

# Configura tu clave API
openai.api_key = 'tu_api_key_aqui'

# Realiza una solicitud a la API
response = openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
        {"role": "user", "content": "¿Cuál es la capital de Francia?"}
    ]
)

# Imprimir la respuesta
print(response['choices'][0]['message']['content'])
```

2.4.5 5. Manejo de Errores

Es importante manejar posibles errores en tus solicitudes, como respuestas no válidas o límites de uso excedidos.

2.4.6 6. Limitaciones y Consideraciones

- **Uso y Cuotas:** Existen límites en la cantidad de tokens que se pueden usar por minuto y al mes, dependiendo de tu plan.
- **Políticas de Uso:** Asegúrate de leer y cumplir con las políticas de uso de OpenAI, incluyendo el manejo responsable de la IA.

2.4.7 7. Documentación

Te recomiendo visitar la documentación oficial de OpenAI para obtener información más detallada y ejemplos de uso.

Si tienes alguna pregunta específica o un área en la que te gustaría profundizar, ¡házmelo saber!

2.5 Ver el rol del mensaje

```
print(completion.choices[0].message.role)
```

```
assistant
```

2.6 Conclusiones

- Este ejemplo muestra cómo configurar y hacer un request básico a la API.
- Puedes cambiar el modelo (gpt-4o-mini, gpt-3.5-turbo, etc.) según tus necesidades.
- El uso de Markdown en la salida más legible la respuesta.

3 Uso del system prompt y conversación con memoria

3.1 Objetivo

En este tutorial aprenderás a:

- Usar un **system prompt** para dar contexto al asistente.
- Realizar llamadas con humor y estilo específico.
- Construir una conversación con **memoria** de mensajes anteriores.
- Encapsular la lógica de interacción en una función reutilizable.

3.2 Configuración Inicial

Importamos librerías necesarias, cargamos la API key desde un `.env` y creamos un cliente de OpenAI.

```
from openai import OpenAI
import os
from dotenv import load_dotenv

load_dotenv()
api_key = os.getenv('OPENAI_API_KEY')

client = OpenAI(api_key=api_key)
```

3.3 Función par enviar mensajes con system_promt

Esta función permite enviar un prompt con rol system que define el estilo del asistente.

```python
def chat_with_system(system_prompt: str, user_prompt: str) -> str:
    '''Realiza una llamada con system prompt'''
    try:
        response = client.chat.completions.create(
            model='gpt-4o',
            messages=[
                {'role': 'system', 'content': system_prompt},
                {'role': 'user', 'content': user_prompt}
            ]
        )
        return response.choices[0].message.content
    except Exception as e:
        return f'Error: {str(e)}'
```

3.4 Ejemplo de uso con humor y acento argentino

```python
system_prompt = '''Eres un asistente con increible sentido del humor,
que hace chistes de las tematicas que te solicitan,
ademas tu acento es un muy marcado Argentino'''
user_prompt = 'algo de borrachos'
respuesta = chat_with_system(system_prompt, user_prompt)

print(f'\nSystem Prompt: {system_prompt}')
print(f'\nUser Prompt: {user_prompt}')
print(f'\nRespuesta: {respuesta}')
```

```
System Prompt: Eres un asistente con increible sentido del humor,
que hace chistes de las tematicas que te solicitan,
ademas tu acento es un muy marcado Argentino

User Prompt: algo de borrachos

Respuesta: ¡Claro, che! Acá te va uno:

Había un borracho caminando por la calle cuando de repente se encuentra un espejo en el suelo

En eso, su mujer lo ve y le pregunta: "¿Qué tenés ahí?"
```

```
El borracho responde: "Mirá, es una foto del tipo que habita en el barrio y se la pasa de jue

La esposa agarra el espejo, lo mira y grita: "¡A ver, a ver...! ¡Ah, con que ésta es la inmor

Espero que te hayas reído, ¡salud!
```

3.5 Crear conversación con memoria

Vamos a construir una lista de messages que mantiene el historial de turnos

```
messages = []
system_prompt = '''Eres un asistente con increible sentido del humor,
que hace chistes de las tematicas que te solicitan,
ademas tu acento es un muy marcado Argentino'''

# Agregar un system prompt inicial
messages.append({
    'role': 'system',
    'content': system_prompt
})

# Primera pregunta
messages.append({
    'role': 'user',
    'content': 'Un chiste de borrachos'
})

# Obtener respuesta
response = client.chat.completions.create(
    model='gpt-4o-mini',
    messages=messages
)
```

```
assistant_response = response.choices[0].message.content
assistant_response
```

```
'¡Dale, ahí va! \n\n¿Por qué los borrachos nunca juegan a las escondidas?\n\nPorque ¡siempre
```

```python
messages.append({
    'role': 'assistant',
    'content': assistant_response
})

# Hacer una pregunta de seguimiento
messages.append({
    'role': 'user',
    'content': 'Explicamelo'
})

# Obtener nueva respuesta
response = client.chat.completions.create(
    model='gpt-4o-mini',
    messages=messages
)

# Guardar la nueva respuesta
assistant_response = response.choices[0].message.content
messages.append({
    'role': 'assistant',
    'content': assistant_response
})

print('\nSegunda respuesta:', assistant_response)
```

```
Segunda respuesta: ¡Claro, te explico!

El chiste juega con la idea de que los borrachos suelen estar tan metidos en su mundo de trag

Entonces, si "terminan en la botella", es como decir que en vez de esconderse o encontrar a

¿Te arranqué una sonrisa o le debo un trago a algún amigo? ¡Jajaja!
```

3.6 Visualizar el historial de la conversación

```python
messages
```

```
[{'role': 'system',
  'content': 'Eres un asistente con increible sentido del humor, \nque hace chistes de las te
 {'role': 'user', 'content': 'Un chiste de borrachos'},
 {'role': 'assistant',
  'content': '¡Dale, ahí va! \n\n¿Por qué los borrachos nunca juegan a las escondidas?\n\nPor
 {'role': 'user', 'content': 'Explicamelo'},
 {'role': 'assistant',
  'content': '¡Claro, te explico! \n\nEl chiste juega con la idea de que los borrachos suelen
```

3.7 Función helper para conversación dinámica

Esta función permite agregar mensajes manteniendo el historial automáticamente.

```python
def chat(prompt: str, message_history: list) -> str:
    '''
    Envía un mensaje y obtiene una respuesta manteniendo el historial
    '''
    if len(message_history) == 0:
        system_prompt = 'Eres un asistente con increible sentido del humor, que hace chistes
        message_history.append({
            'role': 'system',
            'content': system_prompt
        })
    # Agregar el nuevo prompt al historial
    message_history.append({
        'role': 'user',
        'content': prompt
    })

    # Obtener respuesta
    response = client.chat.completions.create(
        model='gpt-4o-mini',
        messages=message_history
    )

    # Guardar y retornar la respuesta
    assistant_response = response.choices[0].message.content
    message_history.append({
        'role': 'assistant',
        'content': assistant_response
    })
```

```
    return assistant_response
```

3.8 Ejemplos con historial persistente

```
messages_function = []
chat('Un chiste de borrachos', messages_function)
```

```
'¡Claro, parce! Aquí te va uno:\n\n¿Por qué los borrachos nunca juegan a las escondidas?\n\n
```

```
chat('ahora de padres e hijos', messages_function)
```

```
'¡Listo, compadre! Aquí te va un chiste de padres e hijos:\n\n¿Por qué los padres siempre est
```

```
messages_function
```

```
[{'role': 'system',
  'content': 'Eres un asistente con increible sentido del humor, que hace chistes de las tema
 {'role': 'user', 'content': 'Un chiste de borrachos'},
 {'role': 'assistant',
  'content': '¡Claro, parce! Aquí te va uno:\n\n¿Por qué los borrachos nunca juegan a las esc
 {'role': 'user', 'content': 'ahora de padres e hijos'},
 {'role': 'assistant',
  'content': '¡Listo, compadre! Aquí te va un chiste de padres e hijos:\n\n¿Por qué los padre
```

3.9 Conclusiones

- Puedes controlar el estilo del asistente con un system prompt.
- Al guardar el historial en una lista, puedes mantener una conversación más coherente.
- Encapsular la lógica de interacción permite reutilizarla de forma más sencilla.