

✓ Getting Started with GPT-4 API

copyright 2024 Denis Rothman

✓ May 14,2024 update to from gpt-4 to gpt-4o

OpenAI released gpt-4o which is now deployed in this notebook

```
gmodel="gpt-4o"
```

```
from IPython.display import Image      #This is used for rendering images in the notebook
```

✓ Step 1: Installing & importing OpenAI

```
!pip install tiktoken
```

```
Collecting tiktoken
  Downloading tiktoken-0.6.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.8 MB)
    1.8/1.8 MB 9.6 MB/s eta 0:00
Requirement already satisfied: regex<=2022.1.18 in /usr/local/lib/python3.10/dist-packages (from tiktoken)
Requirement already satisfied: requests<=2.26.0 in /usr/local/lib/python3.10/dist-packages (from tiktoken)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->tiktoken)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->tiktoken)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->tiktoken)
Requirement already satisfied: certifi<=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->tiktoken)
Installing collected packages: tiktoken
Successfully installed tiktoken-0.6.0
```

```
!pip install cohere
```

```
Collecting cohere
  Downloading cohere-5.3.3-py3-none-any.whl (151 kB)
    151.2/151.2 kB 2.5 MB/s eta 0:00
Collecting fastavro<2.0.0,>=1.9.4 (from cohere)
  Downloading fastavro-1.9.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.1 MB)
    3.1/3.1 MB 13.2 MB/s eta 0:00
Collecting httpx<=0.21.2 (from cohere)
  Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
    75.6/75.6 kB 6.0 MB/s eta 0:00
Collecting httpx-sse<0.5.0,>=0.4.0 (from cohere)
  Downloading httpx_sse-0.4.0-py3-none-any.whl (7.8 kB)
Requirement already satisfied: pydantic<=1.9.2 in /usr/local/lib/python3.10/dist-packages (from httpx-sse->cohere)
Requirement already satisfied: requests<3.0.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from httpx-sse->cohere)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from httpx-sse->cohere)
```

```

Collecting types-requests<3.0.0,>=2.0.0 (from cohere)
  Downloading types_requests-2.31.0.20240406-py3-none-any.whl (15 kB)
Requirement already satisfied: typing_extensions>=4.0.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: anyio in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages
Collecting httpcore==1.* (from httpx>=0.21.2->cohere)
  Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
----- 77.9/77.9 kB 5.0 MB/s eta
Requirement already satisfied: idna in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx>=0.21.2->cohere)
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
----- 58.3/58.3 kB 2.8 MB/s eta
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pydantic-core==2.18.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages
Installing collected packages: types-requests, httpx-sse, h11, fastavro, httpcore
Successfully installed cohere-5.3.3 fastavro-1.9.4 h11-0.14.0 httpcore-1.0.5 ht

```

```

try:
    import openai
except:
    !pip install openai
    import openai

```

```

⇒ Collecting openai
  Downloading openai-1.23.6-py3-none-any.whl (311 kB)
----- 311.6/311.6 kB 5.3 MB/s et
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packages
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: typing_extensions<5,>=4.7 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pydantic-core==2.18.1 in /usr/local/lib/python3.10/dist-packages
Installing collected packages: openai
Successfully installed openai-1.23.6


```

Step 1-May 14,2024 update to gpt-4o

OpenAI released gpt-4o which is now deployed in this notebook

✓ Step 2: Entering the API KEY

```
from google.colab import drive
drive.mount('/content/drive')
```

 Mounted at /content/drive

```
f = open("drive/MyDrive/files/api_key.txt", "r")
API_KEY=f.readline()
f.close()
```

Authentication

Setting the environment variable OPENAI_API_KEY to the value of API_KEY

```
import os
os.environ['OPENAI_API_KEY'] =API_KEY
openai.api_key = os.getenv("OPENAI_API_KEY")
```

Step 3: Running an NLP tasks with the default parameters

✓ Step 4: Example 1: Grammar correction

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

response = client.chat.completions.create(
    model=gmodel,
    messages=[
        {
            "role": "system",
            "content": "You will be provided with statements, and your task is to convert them to correct grammar."
        },
        {
            "role": "user",
            "content": "She no went to the market."
        }
    ],
    temperature=0,
    max_tokens=256,
```

```

top_p=1,
frequency_penalty=0,
presence_penalty=0
)
print(response.choices[0].message.content)

```

↔ She didn't go to the market.

✓ Example 2: Translation

```

from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
    model=gmodel,
    messages=[
        {
            "role": "system",
            "content": "You will be provided with sentences, and your task translate from English to French."
        },
        {
            "role": "user",
            "content": "She did not go to the market."
        }
    ],
    temperature=0,
    max_tokens=256,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0
)
print(response.choices[0].message.content)

```

↔ Elle n'est pas allée au marché.

✓ Example 3: Time Complexity

<https://platform.openai.com/examples/default-time-complexity>

```

from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
    model=gmodel,
    messages=[
        {
            "role": "system",
            "content": "You will be provided with Python code, and your task is to calculate its time complexity."
        },
        {
            "role": "user",
            "content": "def foo(n, k):\n    accum = 0\n    for i in range(n):\n

```

```

    }
],
temperature=0,
max_tokens=256,
top_p=1,
frequency_penalty=0,
presence_penalty=0
)
print(response.choices[0].message.content)

```

➡ To determine the time complexity of the function `foo(n, k)`, let's analyze the

1. ****Outer Loop****: The outer loop `for i in range(n)` iterates `n` times, where
2. ****Inner Loop****: Inside the outer loop, there is another loop `for l in range
3. ****Operation Inside Inner Loop****: Inside the inner loop, the operation `accum

Now, let's calculate the total number of times the innermost operation is executed.

- For each of the `n` iterations of the outer loop, the inner loop runs `k` times.
- Therefore, the statement `accum += i` is executed `n * k` times.

Since the time complexity of the operation inside the loops is $O(1)$, the overall time complexity is $O(n * k)$.

✓ Example 4: Text to emoji

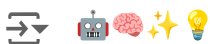
<https://platform.openai.com/examples/default-emoji-translation>

```

from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
    model=gmodel,
    messages=[
        {
            "role": "system",
            "content": "You will be provided with text, and your task is to translate it into emoji."
        },
        {
            "role": "user",
            "content": "Artificial intelligence is a technology with great promise."
        }
    ],
    temperature=0.8,
    max_tokens=256,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0
)
print(response.choices[0].message.content)

```



✓ Example 5: Spreadsheet creator

<https://platform.openai.com/examples/default-spreadsheet-gen>

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

response = client.chat.completions.create(
    model=gmodel,
    messages=[
        {
            "role": "user",
            "content": "Create a two-column CSV of top science fiction movies along with the year"
        }
    ],
    temperature=0.5,
    max_tokens=300,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0
)
print(response.choices[0].message.content)
```

⇒ Here's a sample CSV content for top science fiction movies along with their year

```
```
Title,Year
"Blade Runner",1982
"2001: A Space Odyssey",1968
"Star Wars: Episode IV - A New Hope",1977
"The Matrix",1999
"Inception",2010
"Back to the Future",1985
"Aliens",1986
"Metropolis",1927
"Terminator 2: Judgment Day",1991
"Interstellar",2014
"Arrival",2016
"Ex Machina",2015
"Minority Report",2002
"Star Trek",2009
"The Fifth Element",1997
"District 9",2009
"Her",2013
"Gravity",2013
"Edge of Tomorrow",2014
"The War of the Worlds",1953
```
```

To create a CSV file, you can paste this content into a plain text editor and save it.

✓ Example 6: Advanced Tweet classifier

<https://beta.openai.com/examples/default-tweet-classifier>

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

response = client.chat.completions.create(
    model=gmodel,
    messages=[
        {
            "role": "system",
            "content": "You will be provided with a tweet, and your task is to classify its sent
        },
        {
            "role": "user",
            "content": "I loved the new Batman movie!"
        }
    ],
    temperature=0,
    max_tokens=256,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0
)
print(response.choices[0].message.content)
```

⇒ positive

✓ Example 7: Natural Language to SQL

<https://platform.openai.com/examples/default-sql-translate>

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

response = client.chat.completions.create(
    model=gmodel,
    messages=[
        {
            "role": "system",
            "content": "Given the following SQL tables, your job is to write queries given a use
        },
        {
            "role": "user",
            "content": "Write a SQL query which computes the average total order value for all o
        }
    ],
    temperature=0,
    max_tokens=1024,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0
)
print(response.choices[0].message.content)
```

➡ To compute the average total order value for all orders on a specific date, we can use the following SQL query:

```
```sql
SELECT AVG(TotalOrderValue) AS AvgTotalOrderValue
FROM (
 SELECT o.OrderID, SUM(od.Quantity * p.UnitPrice) AS TotalOrderValue
 FROM Orders o
 JOIN OrderDetails od ON o.OrderID = od.OrderID
 JOIN Products p ON od.ProductID = p.ProductID
 WHERE o.OrderDate = '2023-04-01'
 GROUP BY o.OrderID
) AS OrderValues;
```
```

This query works as follows:

1. ****FROM Orders o****: Start by selecting from the `Orders` table.
2. ****JOIN OrderDetails od ON o.OrderID = od.OrderID****: Join the `OrderDetails` table to the `Orders` table using the `OrderID` column.
3. ****JOIN Products p ON od.ProductID = p.ProductID****: Join the `Products` table to the `OrderDetails` table using the `ProductID` column.
4. ****WHERE o.OrderDate = '2023-04-01'****: Filter the orders to include only those placed on April 1, 2023.
5. ****GROUP BY o.OrderID****: Group the results by `OrderID` to calculate the total value for each order.
6. ****SUM(od.Quantity * p.UnitPrice) AS TotalOrderValue****: Calculate the total value for each order by summing the product of quantity and unit price.
7. ****SELECT AVG(TotalOrderValue) AS AvgTotalOrderValue****: Finally, calculate the average total order value across all orders.

This query will give you the average total value of all orders placed on April 1, 2023.