190c-9a-chatGPT-API-2023-8-28.odp

CTI One Corporation
Project Lead: Harry Li, Ph.D.
Team Member: Yusuke Yakuwa
Youran Zheng
Updated by Haomeng Zhang
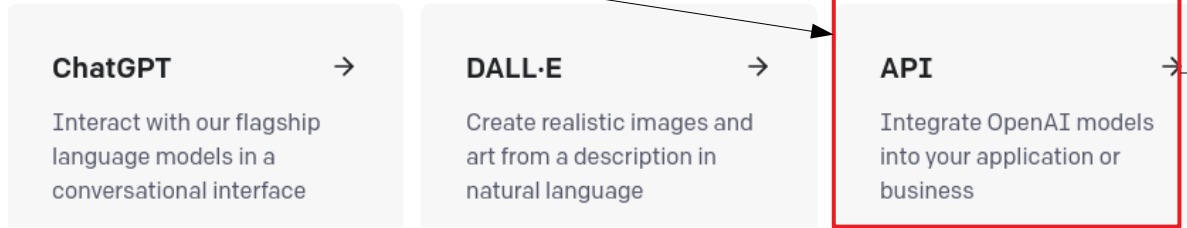
July 02, 2024

ChatGPT API from OpenAI (8/28,2023)

https://community.openai.com/t/is-chat-gpt-provided-for-free/86249

Step 1. Log in ChatGPT, then click on API

OpenAI

Step 2. Design your prompt to program your model

Example: Conversation with API: a conversation is formatted with a system message first, followed by alternating user and assistant message, see below:

ChatGPT →	DALL·E →	API →
Interact with our flagship language models in a conversational interface	Create realistic images and art from a description in natural language	Integrate OpenAI models into your application or business

GPT-3.5 Turbo

GPT-3.5 Turbo is optimized for dialogue.

Learn about GPT-3.5 Turbo ↗

GPT-3.5-Turbo Pricing:

Model	Input	Output
4K context	$0.0015 / 1K tokens	$0.002 / 1K tokens
16K context	$0.003 / 1K tokens	$0.004 / 1K tokens

https://openai.com/pricing#language-models

```
import openai

openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
      {"role": "system", "content": "You are a assistant."},
      {"role": "user", "content": "Who won the world series in 20?"},
      {"role": "assistant", "content": "The Los Angeles Dodgers."},
      {"role": "user", "content": "Where was it played?"}
  ]
)
```

ChatGPT API Model "gpt-4o" (07/02,2024)

https://platform.openai.com/docs/models/gpt-4o

GPT-4o

GPT-4o ("o" for "omni") is our most advanced model. It is multimodal (accepting text or image inputs and outputting text), and it has the same high intelligence as GPT-4 Turbo but is much more efficient—it generates text 2x faster and is 50% cheaper. Additionally, GPT-4o has the best vision and performance across non-English languages of any of our models. GPT-4o is available in the OpenAI API to paying customers. Learn how to use GPT-4o in our text generation guide.
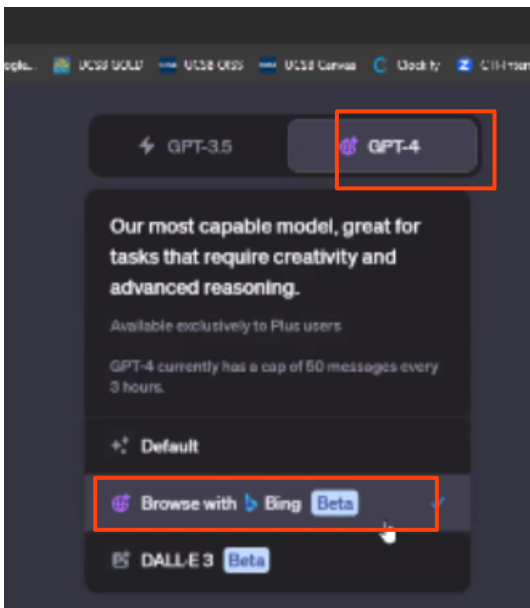
GPT-4o Pricing:

Model	Pricing	Pricing with Batch API*
gpt-4o	$5.00 / 1M input tokens	$2.50 / 1M input tokens
	$15.00 / 1M output tokens	$7.50 / 1M output tokens

https://openai.com/pricing#language-models

Conda Environment Setting for the ChatGPT API (10/27-8/28,2023)

Note 2023-10-27, Version 4.0 to support browsing with Bing is tested by YZ, $20 per month for the membership;

2023-10-27: A Python version of 3.7.1 or newer is required. You can install the OpenAI Python library via pip with the command pip install openai. It's important to ensure that your environment meets this minimum Python version requirement before installing and using the OpenAI library to interact with OpenAI's API.

The yml for conda environment configuration is created, check the file below.

```
#---------------------------------------------
# CTI One Corporation
# for Chat-GPT
# Version x0.1
# Coded by: Youran Zheng, 2023-10-27
# Create a Anaconda environment;
# Open a terminal, then
# $ conda env create -f chatgpt-2023-10-27.yml
# Activate the Anaconda environment: $ conda activate chatgpt
#-----------------------------------------
name: chatgpt-2023-10-27


dependencies:
  - python==3.7.1
  - pip
  - pip:
    - openai==0.27.9
```

ChatGPT API Customer Development Folder Structure and Source Code (11-11-8/28,2023)

:~/PycharmProjects/chatGPT$ tree -L 2
.

```
.
├── fine-tuning
│       ├── fine-tuning-10QA-2023-9-14
│       ├── fine-tuning.jsonl
│       ├── finetuning.py
│       └── sample
├── hello-the-world
│       ├── API_Key.json
│       ├── ChatHistory.json
│       └── GPTWithHistorySaved.py
└── web-integration
```

Created a new key: User-support-2023-11-11

Create new secret key

Name Optional

User-support-2023-11-11

Cancel   **Create secret key**

API keys

Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically disable any API key that we've found has leaked publicly.

NAME	KEY	CREATED	LAST USED ⓘ		
User-support-2023-11-...	sk - . . . JADo	Nov 11, 2023	Never	✏	🗑

Hello World Test Code on GPT-3.5-Turbo (8/31, 2023)

1. Save the API key into API_Key.json file

2. Choose the language model as "gpt-3.5-turbo"

3.The variable "messages" is used to send chatting content in the list of dictionaries form.

4. "role" : "system" sets the personality and goal for GPT.

5. "role" : "user" sends inquires.

6. "role" : "assistant" GPT responses to the inquires.

```python
OpenAI_Practice >  HelloWorld.py > ...
1  import openai, json, os
2
3  workdir = os.path.abspath(os.path.dirname(__file__))
4  print(workdir)
5  with open(workdir + "/API_Key.json", "r") as f:
6      keyDict = json.load(f)
7
8  MODEL = "gpt-3.5-turbo"
9  API_KEY = keyDict.get("personalTestKey")
10 openai.api_key = API_KEY
11
12 response = openai.ChatCompletion.create(
13     model = MODEL,
14     messages = [
15         {"role": "system", "content": "You are a helpful assistant."},
16         {"role": "user", "content": "Hello Chat-GPT. First tell me your language model and then introdu
17     ]
18 )
19 responseContent = response['choices'][0]['message']['content']
       = response['usage']['total_tokens']
   -GPT: ")
   onseContent)
   e total token cost is: {tokenCost}")
```

```
harry@harrys-gpu-laptop: ~/PycharmProjects/chatGPT
File  Edit  View  Search  Terminal  Help
(base) harry@harrys-gpu-laptop:~/PycharmProjects/chatGPT$ tree -L 2

    API_Key.json
    ChatHistory.json
    fine-tuning
        fine-tuning.jsonl
        finetuning.py
        sample
    fine-tuning-10QA-2023-9-14
        GPT_Fine_Tuning.zip
    GPTWithHistorySaved.py

3 directories, 6 files
(base) harry@harrys-gpu-laptop:~/PycharmProjects/chatGPT$
```

Hello World Test Code on GPT-3.5-Turbo With Chat History (8/31)

1. Save the API key into API_Key.json file

2. Choose the language model as "gpt-3.5-turbo"

3.The variable "messages" is used to send chatting content in the list of dictionaries form.

4. "role" : "system" can set the personality and goal for GPT.

5. "role" : "user" is used to send questions.

6. "role" : "assistant" is used to let GPT remember the last answer to make GPT memorize the chat history.

7. The pricing of GPT-3.5-Turbo is:

GPT-3.5 Turbo is optimized for dialogue.

Learn about GPT-3.5 Turbo ↗

Model	Input	Output
4K context	$0.0015 / 1K tokens	$0.002 / 1K tokens
16K context	$0.003 / 1K tokens	$0.004 / 1K tokens

8. Create a list of dictionaries as "chatHistory", replace it with "messages" sent to GPT. Every time it receives a response, the assistant response (GPT answer) is appended to the "chatHistory", so that the GPT remembers what user has asked.

```
OpenAI_Practice > 🐍 WithChatHistory.py > ...
1   import openai, json, os
2
3   workdir = os.path.abspath(os.path.dirname(__file__))
4   print(workdir)
5   with open(workdir + "/API_Key.json", "r") as f:
6       keyDict = json.load(f)
7
8   MODEL = "gpt-3.5-turbo"
9   API_KEY = keyDict.get("personalTestKey")
10  openai.api_key = API_KEY
11
12  chatHistory = [{"role": "system", "content": "You are a helpful assistant."}]
13
14  while True:
15      userMessage = input("User: "); print()
16      if userMessage.lower() == "quit" : break
17
18      chatHistory.append({"role": "user", "content": userMessage})
19      response = openai.ChatCompletion.create(
20          model = MODEL,
21          messages = chatHistory
22      )
23      assistantResponse = response['choices'][0]['message']['content']
24      print(f"Chat-GPT: {assistantResponse}\n\n")
25      chatHistory.append(response['choices'][0]['message'])
```

Test result:

```
User: What are Python and Java?

Chat-GPT: Python and Java are both programming languages.

Python is a high-level, interpreted programming language that emphasizes code readability and simplicity. It is often used for web development, scientific computing, and data analysis. Python has a large standard library and a rich ecosystem of third-party libraries, making it highly versatile and widely adopted.

Java, on the other hand, is a general-purpose, object-oriented programming language that is designed to be platform-independent. It is known for its "write once, run anywhere" principle, which means that Java programs can run on any device or operating system that has a Java Virtual Machine (JVM). Java is commonly used in back-end development, Android app development, and large enterprise systems.

Both Python and Java have their own strengths and can be used to build a wide range of applications, depending on the context and requirements.


User: What is the major difference then?

Chat-GPT: There are several major differences between Python and Java:

1. Syntax: Python has a clean and concise syntax, with a focus on readability, using indentation and whitespace to structure code. Java, on the other hand, has a more verbose and formal syntax, with the use of semicolons and curly braces.

2. Type system: Python is dynamically typed, which means that variable types are determined at runtime. Java, on the other hand, is statically typed, where variable types are declared explicitly and checked at compile-time.

3. Memory management: Python uses automatic memory management through garbage collection, meaning that developers don't have to worry about allocating or deallocating memory. Java also uses garbage collection, but it provides more control over memory management through the use of explicit memory deallocation using the "finalize()" method.

4. Performance: Java is often considered to have better performance than Python because it is compiled to bytecode and runs on the Java Virtual Machine (JVM). Python, being an interpreted language, is generally slower. However, Python can be faster in certain cases, especially when utilizing optimized libraries.

5. Usage: Python is commonly used in areas like web development, data analysis, scientific computing, machine learning, and artificial intelligence. Java is popular for enterprise applications, Android app development, server-side development, and large-scale systems.

Ultimately, the choice between Python and Java depends on the requirements of the project, developer preferences, and the specific domain in which the language will be used.


User:
```

In this example, I first asked what Python and Java are. The GPT introduces Python and Java. Second, I asked chat GPT what the major difference is. Instead of asking it "what is the major difference between Python and Java", I used "then" to omit "between Python and Java". The GPT still answered me about the difference between Python and Java because it remembered the previous question I asked.

1. Save the API key into api_key.txt file

2. Choose the language model as "gpt-4o"

3. The variable "messages" is used to send chatting content in the list of dictionaries form.

4. "role" : "system" sets the personality and goal for GPT.

5. "role" : "user" sends inquires.

6. "role" : "assistant" GPT responses to the inquires.

```python
from openai import OpenAI
import os
import json


with open('api_key.txt', 'r') as f:
    OPENAI_API_KEY = f.read()

client = OpenAI(api_key=OPENAI_API_KEY)

completion = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "system", "content": "You are a poetic assistant, skilled in explaining complex programming con
        {"role": "user", "content": "Compose a poem that explains the concept of recursion in programming."}
    ]
)

print(completion.choices[0].message)
gpt_answer = response.choices[0].message.content

with open("response.txt", "w") as file:
    file.write(gpt_answer)
```

Hello World Test Code on GPT-4o with Chat History (07/02/24)

1. Save the API key into API_Key.json file

2. Choose the language model as "gpt-4o"

3. The variable "messages" is used to send chatting content in the list of dictionaries form.

4. "role" : "system" can set the personality and goal for GPT.

5. "role" : "user" is used to send questions.

6. "role" : "assistant" is used to let GPT remember the last answer to make GPT memorize the chat history.

7. The pricing of GPT-4o is:

Model	Pricing	Pricing with Batch API*
gpt-4o	$5.00 / 1M input tokens	$2.50 / 1M input tokens
	$15.00 / 1M output tokens	$7.50 / 1M output tokens

8. Create a list of dictionaries as "chatHistory", replace it with "messages" sent to GPT. Every time it receives a response, the assistant response (GPT answer) is appended to the "chatHistory", so that the GPT remembers what user has asked.

```python
from openai import OpenAI
import os
import json


with open('api_key.txt', 'r') as f:
    OPENAI_API_KEY = f.read()

client = OpenAI(api_key=OPENAI_API_KEY)


completion = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Who won the world series in 2020?"},
        {"role": "assistant", "content": "The Los Angeles Dodgers won the World Series in 2020."},
        {"role": "user", "content": "Where was it played?"}
    ]
)

print(completion.choices[0].message)
gpt_answer = completion.choices[0].message.content

with open("response.txt", "w") as file:
    file.write(gpt_answer)
```

Return:

```
ChatCompletionMessage(content='The 2020 World Series was played at Globe Life Field in Arlington, Texas. This was the first time the World Series was held at a neutral site due to the COVID-19 pandemic.', role='assistant', function_call=None, tool_calls=None)
```