



# ChatGPT Workshop

Steven Van Vaerenbergh

 @steven2358

April 28, 2023

 Treasure Hacks 3.5 hackathon

# Whoami

- Based in Spain (Europe)

- Associate professor at   
UNIVERSIDAD  
DE CANTABRIA

-  @steven2358

- Repo for this workshop:  
<https://github.com/steven2358/chatgpt-workshop>



Illustration: Inkscape Diffusion + ControlNet

# What is Generative AI

“Generative AI is a type of AI technology that can create unique, original content automatically, including text, code, images, audio, and video.”

- Uses machine learning (ML), large language models (LLM)
- OpenAI is a pioneer in the field of generative AI.  
E.g. GPT, DALL·E 2, GitHub Copilot, ChatGPT, ...
- More resources: <https://github.com/steven2358/awesome-generative-ai>

# Getting started with the OpenAI API

1. Sign up for an OpenAI account:  
<https://platform.openai.com/account/>

\$5 in free credit

2. Check the playground  
<https://platform.openai.com/playground>



## Playground

[Save](#)[View code](#)[Share](#)

Write a recipe for a strawberry cake.



Looking for ChatGPT?

[Try it now](#)[Submit](#)

8

Top P

1

5





## Playground

[Save](#)[View code](#)[Share](#)

Write a recipe for a strawberry cake.



Strawberry Cake

Ingredients:

- 2 cups all-purpose flour
- 1 teaspoon baking powder
- 1 teaspoon baking soda
- 1/2 teaspoon salt
- 1/2 cup butter, softened
- 1 1/2 cups white sugar



Looking for ChatGPT?

[Try it now](#)[Submit](#)

264

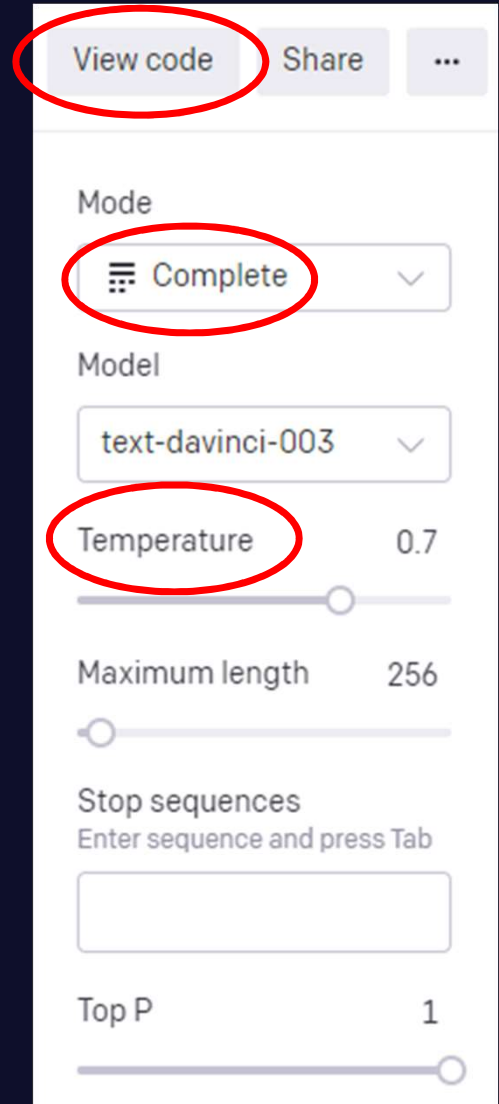
Top P

1

6

# Sidebar

- Mode: “Complete”, “Chat”, or other.
- Complete: GPT models (prompt + response)
- Chat: ChatGPT models (conversation)
- “View code” button...



The image shows a sidebar for configuring the OpenAI API. At the top, there are three buttons: 'View code' (circled in red), 'Share', and a three-dot menu. Below these, the 'Mode' is set to 'Complete' (circled in red). The 'Model' is set to 'text-davinci-003'. The 'Temperature' is set to 0.7 (circled in red). The 'Maximum length' is set to 256. The 'Stop sequences' section has a text input field. The 'Top P' is set to 1.

View code Share ...

Mode

Complete

Model

text-davinci-003

Temperature 0.7

Maximum length 256

Stop sequences  
Enter sequence and press Tab

Top P 1

You can use the following code to start integrating your current prompt and settings into your application.

POST /v1/completions

python  Copy

```
1  import os
2  import openai
3
4  openai.api_key = os.getenv("OPENAI_API_KEY")
5
6  response = openai.Completion.create(
7      model="text-davinci-003",
8      prompt="Write a recipe for a strawberry cake.",
9      temperature=0.7,
10     max_tokens=256,
11     top_p=1,
12     frequency_penalty=0,
13     presence_penalty=0
14 )
```

Your API Key can be found [here](#). You should use environment variables or a secret management tool to expose your key to your applications.



```
openai.api_key = os.getenv("OPENAI_API_KEY")
```

Get an OpenAI API key here: <https://platform.openai.com/account/api-keys>

Store it as an environment variable (or using an API key manager).

E.g. in conda: <https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#saving-environment-variables>



## Playground

[Save](#)[View code](#)[Share](#)

### SYSTEM

You are a helpful assistant.  
You answer questions in the  
style of Shakespeare.

**USER**

What is daylight savings time?

Add message

[Submit](#)**Mode**

Chat

Beta

**Model**

Temperature

0.7



Maximum length

256



Top P

1



Frequency penalty

0



Presence penalty

0





## Playground

[Save](#)[View code](#)[Share](#)

### SYSTEM

You are a helpful assistant.  
You answer questions in the  
style of Shakespeare.

### USER

What is daylight savings time?

### ASSISTANT

Fair question, good sir or madam,  
Daylight savings time is a temporal  
program,  
Wherein we alter the clock's forward or  
back,  
So that the hours of daylight we may  
hack.  
'Tis done to save energy, or so we're  
told,  
And make use of the sun's light, bright  
and bold.

[Submit](#)[Give us feedback](#)

### Mode

[Chat](#)[Beta](#)

### Model



Temperature

0.7

Maximum length

256

Top P

1

Frequency penalty

0

Presence penalty

0

# script01\_helloworld.py

```
1  import os
2  import openai
3
4  # get API key from environment variable
5  openai.api_key = os.getenv("OPENAI_API_KEY")
6
7  # send the question to the API and receive the response
8  completion = openai.ChatCompletion.create(
9      model="gpt-3.5-turbo",
10     messages=[
11         {
12             "role": "user",
13             "content": "Say Hello World in the style of a pirate."
14         }
15     ]
16 )
17
18 # print the response
19 print("\n"+completion.choices[0].message.content)
```

```
(generative) c:\git\chatgpt-workshop>python script01_helloworld.py
```

```
(generative) c:\git\chatgpt-workshop>python script01_helloworld.py
```

```
Ahoy mateys, avast ye! Listen up me hearties, fer I shall declare this to ye, 'tis the hour to r  
aise yer anchor and hoist the Jolly Roger high, arrr! With me hearty voice, I shout to ye, "Ahoy  
World!"
```

```
(generative) c:\git\chatgpt-workshop>_
```



# script02\_chatbot.py

```
1  # based on https://github.com/mckaywrigley/takeoff-school-your-1st-ai-app
2
3  import os
4  import openai
5
6  # get API key from environment variable
7  openai.api_key = os.getenv("OPENAI_API_KEY")
8
9  # loop until user types 'exit'
10 while True:
11     question = input("What is your question? ")
12
13     if question.lower() == "exit":
14         print("Goodbye!")
15         break
16
17     # send the question to the API and receive a response
18     completion = openai.ChatCompletion.create(
19         model="gpt-3.5-turbo",
20         messages=[
21             {"role": "system", "content": "You are a helpful assistant. Answer the given question."},
22             {"role": "user", "content": question}
23         ]
24     )
25
26     # print the response
27     print(completion.choices[0].message.content + "\n")
28
```

```
(generative) c:\git\chatgpt-workshop>python script02_chatbot.py
```

```
(generative) c:\git\chatgpt-workshop>python script02_chatbot.py  
What is your question? _
```

```
(generative) c:\git\chatgpt-workshop>python script02_chatbot.py  
What is your question? Why is the sky blue?_
```

```
(generative) c:\git\chatgpt-workshop>python script02_chatbot.py
```

What is your question? Why is the sky blue?

The sky appears blue because of a phenomenon called Rayleigh scattering. Blue light from the sun is scattered in all directions by the gases and particles in the Earth's atmosphere. This causes the blue light to be seen from all directions, making the sky appear blue during the day.

What is your question?

# ChatGPT memory

- The chatbot in script02\_chatbot.py has no memory
- Each call to `openai.ChatCompletion.create()` starts from scratch
- We must include the entire conversation history to get the new response:

```
openai.ChatCompletion.create(  
    model="gpt-3.5-turbo",  
    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?"}  
    ]  
)
```



# Practical chatbot memory

## Option 1: Keep a running memory of the previous conversation

```
completion = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
        {"role": "system", "content": "You are a helpful assistant. Answer the given question."},
        {"role": "user", "content": memory[0]}
        {"role": "assistant", "content": memory[1]}
        {"role": "user", "content": question}
    ]
)
memory.append(question)
memory.append(completion.choices[0].message.content)
```

## Option 2: Store interactions in a vector database such as <https://www.pinecone.io/>

# script03\_summarize\_pdf.py

```
1  import os
2  import PyPDF2
3  import openai
4
5  # get API key from environment variable
6  openai.api_key = os.getenv("OPENAI_API_KEY")
7
8  # initialize summary
9  pdf_summary_text = ""
10
11 # read the pdf
12 pdf_file = open("doc/whitepaper.pdf", 'rb')
13 pdf_reader = PyPDF2.PdfReader(pdf_file)
14
15 # loop over pages
16 for page_num in range(len(pdf_reader.pages)):
17     page_text = pdf_reader.pages[page_num].extract_text().lower()
18
19     # request the summary of one page
20     response = openai.ChatCompletion.create(
21         model="gpt-3.5-turbo",
22         messages=[
23             {"role": "user", "content": f"Summarize this: {page_text}"},
24         ],
25     )
26
27     # append the page summary
28     pdf_summary_text += response["choices"][0]["message"]["content"] + "\n\n"
29
30     # write the results to file
31     with open("doc/whitepaper_summary.txt", "w+") as file:
32         file.write(pdf_summary_text)
33
34 pdf_file.close()
```

# Some tips on models

- `gpt-3.5-turbo`: performance similar to `text-davinci-003` but 10x cheaper
- `gpt-4`: much better performance than `gpt-3.5-turbo` but 30x more expensive

# More scripts

- Ask questions about a GitHub repository:  
<https://github.com/mckaywrigley/repo-chat>
- Autonomous agents
  - Auto-GPT <https://github.com/Significant-Gravitas/Auto-GPT>
  - BabyAGI <https://github.com/yoheinakajima/babyagi>
  - MicroGPT <https://github.com/muellerberndt/micro-gpt>

# OpenAI API

- Documentation: <https://platform.openai.com/docs>

```
openai.ChatCompletion.create()
```

```
openai.Completion.create()
```

```
openai.Audio.transcribe()
```

```
openai.Image.create()
```

## Guides



### Chat Beta

Learn how to use chat-based language models



### Text completion

Learn how to generate or edit text



### Embeddings

Learn how to search, classify, and compare text



### Speech to text Beta

Learn how to turn audio into text



### Image generation Beta

Learn how to generate or edit images



### Fine-tuning

Learn how to train a model for your use case

# script04\_generate\_image.py

```
1  import os
2  import openai
3
4  # get API key from environment variable
5  openai.api_key = os.getenv("OPENAI_API_KEY")
6
7  # generate an image
8  response = openai.Image.create(
9      prompt="award-winning picture of T-rex playing a ukulele",
10     n=1,
11     size="512x512"
12 )
13 image_url = response['data'][0]['url']
14 print(image_url)
```



# Thank you for your attention!

