Report

Amazon Review Sentiment Analyzer Using ChatGPT

Continuous Assessment

By

Nishant Kumar

11902722

Roll no: 5

School Of Computer Science and Engineering

Lovely Professional University, Punjab

STUDENT DECLARATION

This is to bring to notice that this report has been written by Nishant Kumar. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. we aver that if any part of the report is found to be copied, we shall take full responsibility for it.

We would like to express our special thanks to our mentor Mr. Dipen Saini for his time and efforts he provided throughout the year. Your useful advice and suggestions were helpful to us during the project's completion. In this aspect, we are eternally grateful to you.

Nishant Kumar

11902722

INTRODUCTION

Project is about of extracting reviews of a particular product from Amazon's website and writing them into a CSV (Comma Separated Values) file involves the process of scraping data from Amazon's website and converting it into a structured format that can be easily processed and analyzed. This can be achieved using web scraping techniques, ‘Selenium’ which involves using software tools to extract data from websites and convert them into a machine-readable format.

Selenium is a powerful automation testing tool that can be used for web scraping. It allows us to automate interactions with a web browser and extract information from a web page. Amazon is a popular e-commerce website that has a vast collection of products and user reviews. Extracting product reviews from Amazon is useful for market research, sentiment analysis, and customer feedback analysis.

To extract reviews from Amazon using Selenium in Python, the following steps can be followed:

1.Install the Selenium library in Python and the web driver for your preferred web browser (e.g., Chrome, Firefox, etc.).

2.Navigate to the Amazon product page that you want to extract reviews from using Selenium.

3.Locate the HTML elements that contain the reviews using Selenium's find_element_by_xpath or find_element_by_css_selector methods.

4.Extract the review text, rating, date, and other relevant information from the HTML elements.

5.Write the extracted data into a CSV file using the csv module in Python.

The ChatGPT API is a tool that allows developers to integrate the capabilities of the GPT-3.5 language model into their own applications. This API allows users to interact with the model and generate natural language responses to a variety of inputs.

To use the ChatGPT API, developers must first sign up for an API key and authenticate their requests. Once authenticated, developers can send requests to the API with various inputs, such as a prompt or a question, and the API will return a response generated by the model.

The extracted data can then be used as input for the ChatGPT API, allowing the model to generate responses based on the extracted data.

Then extract responses and rewrite into the same CSV file using the csv module in Python.

Explanation of the Python code

```
from selenium import webdriver

from selenium.webdriver.common.keys import Keys

import time

import csv

from selenium.webdriver.common.by import By

driver =webdriver.Chrome()
```

“from selenium import webdriver”

This line imports the Selenium webdriver module, which allows Python to control a web browser.

“from selenium.webdriver.common.keys import Keys”

This line imports the Keys class from the Selenium webdriver module, which provides methods for simulating keyboard keys (e.g., ENTER, TAB).

“import time”

This line imports the time module, which provides methods for adding delays or pausing the execution of a Python script

“import csv”

This line imports the csv module, which provides methods for reading and writing CSV files.

“from selenium.webdriver.common.by import By”

This line imports the By class from the Selenium webdriver module, which provides methods for locating elements on a web page.

“driver =webdriver.Chrome()”

This line creates an instance of the Chrome webdriver, which allows Python to control a Chrome browser window.

```
url = "https://amzn.eu/d/7fNyfUW"


driver = webdriver.Chrome()


driver.get(url)
time.sleep(5)
reviews_button = driver.find_element(By.XPATH, '//*[@id="reviews-medley-footer"]/div[2]/a')
time.sleep(2)
reviews_button.click()
time.sleep(1)
```

1. url = "https://amzn.eu/d/7fNyfUW" - This line defines a variable url and assigns it the value of the Amazon product page URL.

2. driver = webdriver.Chrome() - This line creates a new instance of the Chrome webdriver, which is used to automate interactions with a web browser.

3. driver.get(url) - This line instructs the web browser controlled by the driver to navigate to the URL specified by url.

4. time.sleep(5) - This line causes the program to pause for 5 seconds before continuing. This is a common technique used in web scraping to allow time for the page to fully load before attempting to interact with it.

5. reviews_button = driver.find_element(By.XPATH, '//*[@id="reviews-medley-footer"]/div[2]/a') - This line uses the find_element method of the driver to locate an element on the page using an XPATH expression. Specifically, it searches for the "Write a customer review" button on the product page.

6. time.sleep(2) - This line causes the program to pause for 2 seconds before continuing.

7. reviews_button.click() - This line simulates a click on the "Write a customer review" button, which should open the product review page.

8. time.sleep(1) - This line causes the program to pause for 1 second before continuing.

```
while True:
    try:
        load_more_button = driver.find_element(By.XPATH, '//*[@id="cm_cr-pagination_bar"]/ul/li[last()]/a')
        load_more_button.click()
        time.sleep(2)
        driver.find_element(By.TAG_NAME, "body").send_keys(Keys.END)
    except:
        break
time.sleep(5)
```

1. while True: - This starts a loop that will continue indefinitely until it is explicitly broken.

2. try: - This starts a try block, which contains code that may raise an exception.

3. load_more_button = driver.find_element(By.XPATH, '//*[@id="cm_cr-pagination_bar"]/ul/li[last()]/a') - This line uses the find_element method of the driver to locate an element on the page using an XPATH expression. Specifically, it searches for the "Load More" button on the product review page.

4. load_more_button.click() - This line simulates a click on the "Load More" button, which should load additional reviews on the page.

5. time.sleep(2) - This line causes the program to pause for 2 seconds before continuing.

6. driver.find_element(By.TAG_NAME, "body").send_keys(Keys.END) - This line selects the body element of the page using the find_element method, and then simulates a press of the END key using the send_keys method. This should scroll the page to the bottom, which may trigger the loading of additional reviews.

7. except: - This starts an except block, which is executed if an exception is raised in the try block.

8. break - This statement breaks out of the while loop, ending the loop.

9. time.sleep(5) - This line causes the program to pause for 5 seconds before continuing. This is a common technique used in web scraping to allow time for the page to fully load before attempting to interact with it again.

```
# Extract the reviews from the page source

reviews = driver.find_elements(By.XPATH, '//div[@data-hook="review"]')

review_list = []

for review in reviews:

    text_element = review.find_element(By.XPATH, './/span[@data-hook="review-body"]')

    text = text_element.text

    review_list.append({'text': text})
```

1. reviews = driver.find_elements(By.XPATH, '//div[@data-hook="review"]'): This line uses Selenium's find_elements() method to locate all of the div elements on the page with a data-hook attribute equal to "review". This should return a list of elements, with each element representing a single review.

2. review_list = []: This line initializes an empty list called review_list that will be used to store the scraped reviews.

3. for review in reviews:: This line begins a for loop that will iterate over each review element in the reviews list.

4. text_element = review.find_element(By.XPATH, './/span[@data-hook="review-body"]'): This line uses the find_element() method to locate the span element within the current review element that has a data-hook attribute equal to "review-body". This element should contain the text of the review.

5. text = text_element.text: This line gets the text content of the text_element using its text attribute and assigns it to the variable text.

6. review_list.append({'text': text}): This line creates a new dictionary object containing the text of the review (text)

```
# Save the reviews to a CSV file

with open('amazon_reviews.csv', 'w', encoding='utf-8', newline='') as csvfile:

    fieldnames = ['Reviews']

    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    writer.writeheader()

    for review in review_list:

        writer.writerow(review)
```

1. with open('amazon_reviews.csv', 'w', encoding='utf-8', newline='') as csvfile:: This line opens the CSV file named amazon_reviews.csv for writing, using the open() function. The w mode specifies that the file should be opened for writing (and any existing data in the file should be overwritten). The encoding parameter specifies the character encoding to use when writing to the file (in this case, UTF-8). The newline parameter specifies the line ending character(s) to use when writing rows to the file. The with statement ensures that the file is automatically closed when the block is exited, even if an error occurs.

2. fieldnames = ['Reviews']: This line creates a list of field names for the CSV file. In this case, there is only one field: "Reviews".

3. writer = csv.DictWriter(csvfile, fieldnames=fieldnames): This line creates a DictWriter object named writer, which can write rows to the CSV file. The csvfile parameter specifies the file to write to, and the fieldnames parameter specifies the names of the fields in the CSV file.

4. writer.writeheader(): This line writes the header row to the CSV file, using the field names specified in fieldnames.

5. for review in review_list:: This line begins a loop over a list of review objects. Presumably, review_list is a list of dictionaries, with each dictionary representing a single review.

6. writer.writerow(review): This line writes a single row to the CSV file, using the data in the review dictionary.

```
import pandas as pd

# Read the CSV file into a DataFrame

df = pd.read_csv('amazon_reviews.csv')

# Write the DataFrame to an Excel file

df.to_excel('output_file.xlsx', index=False)


# Read in Excel file

df = pd.read_excel('output_file.xlsx')
```

1. import pandas as pd: This line imports the Pandas library and assigns it the alias pd, which is a commonly used convention in the Python data science community.

2. df = pd.read_csv('amazon_reviews.csv'): This line reads in a CSV file called amazon_reviews.csv and creates a Pandas DataFrame object called df. The CSV file is assumed to be in the same directory as the Python script.

3. df.to_excel('output_file.xlsx', index=False): This line exports the DataFrame df to an Excel file called output_file.xlsx. The index=False argument specifies that the row index should not be included in the output file.

4. df = pd.read_excel('output_file.xlsx'): This line reads in the Excel file output_file.xlsx

```
import openai

from pyChatGPT import ChatGPT

session_token='eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIn0..9RpPaBoFT7Ywmt5v.wm8et77abhRvJFrzHHjrm1QnU1DVr-
9BwTvHqd3XE2ftMFnn5EVc3b_B4jQTql2O3kZ97ubgC7HuB4Q6Eudz6VFd3I6n8kTDoLbYpIA29pEf65CrQ_zMbRhdqroVa0ByQhwLzxvxrYXpuHQiDJx
TjtVf6CA4hg8zoCLyypY6nUCSqO2Q0Jr4kTZ6_Fqg48NlAAs84dqOtCIUq6FvbGwlkgu8a0zHwDKW8Ov6qQDObAC8A43uZBfv-
8XwZ9PjZk3onLfHGNBnAuY0Nl-
aIzMl5HcvJ9ukus2uHn0U0HxAdklenAruoN1BDCp48Sm8uBNBLVYretDotCBDwplVqiIWyPWSIDsualq4ZXDJEcoCEn-
PYs9e8hOeTcRJ_KVUpkquchyTweya1OovzDsMy8AoB-N0SLG7g9BNoJ5NLoNeJo56qM2G-Puz8z27jRUxrDZNY7sbiGE6GYKbYe827K0Gw-
Mj4vEqSli0zT7gSV_pPb_QnvxFMx6rtbOs0KtccYnyw5X_Ep7bIBhmol88cVjrb1vP9rKMyGuzcQRY7a4s6tqPk4_Q9rs4NDMVnL6IYK8u8y-
DUD8sJLZlaGst-evy0WeEMA_wFT-TlSvNCoMgpGP7U6lCt9URu0jAwhZJzJw1LeOo8Y7RjZxnrR38aoITkLPzXgE0Ip-
T9oS_YRv8qCQ7UvAHxNrPoWQY2FoDqg3mxpvrysCIO4WV7EaYfvwZD8FeaYkA6XUP1A3Wj1LDQcDndnbc02gPn2ksmpxasb2CN0lV0ZZXmpuxwzxQSAaM
GsWcfNp4ShTWbxrsfXty9gRWBQHFCOjmPJ6NFcnk1raTDCoL6CVwGrDx3xStx_VXKwCmx1lWuCcqbH8nPMsgLr9tdZX6kQQCijOAKaAQdAqwoQeb795al
BoK6u1DTYcPV5YvIwtaMhg0lPWYBAbNZedsLlQOb2Gl_EgGNMUMuW3Fwt1AJI-spjlE2qD53G200wxmkTS_BeLFFDmloD3aLTQ-
qV7Bsz4ehalQFNRPts2zIQtvEcE7j931BZTvW8zO2ns6WlzBPf9DtaZjUUGLVRgFpl9GqIRfc8AKZv4BsVq4ddoWpuqO3u9lv50ltuL9Bu3ga13jNlxFd
sQv40WK3rJMKOXC6Lej7f8lVsBYMaoenGQ8LntH6WAqdDtrkLd5fpD_ZAGktK7VBE4TI49WfYGoCXQIOWlLgXsVRBtWbW70w3Yxye9peiQhO3dzREyNL1
itor3ZLYy4LPCv2hiiJyRaF_WDNVgYlV-zPvX4eU70Duvd8t3-6Jot73ygcAGForq2fzAKjoYv1358W_CHmy-
cKig2knvxv_l7b6M7lmG9YUtguMGN2CIPX5VOPidtHi3M46p3l7jHvBaywxgMAbCchrdpPTZDI5Nr-
UcfWbwyUNdfLOheVpt3b7bsAvhLM4SRTdIZVNTEM64tk3Z-
aTGxVv_Ha3xxPl4ECB6IFLQSjct2ON8G7C2IxDosSqoM_duJCRIyHuxi3OqLQ03PbVr07uEUR3ULZA_HLzplPqyq-NBsL4qtlzvj-
wM_dmVT4U0ITD2Icv-yk5X7cgIiMDoZlLwBvtIPL8Cr1ICFlKfhh2ZIRq-NqsBgBQnbQ5npsWGAmcxhvJ7ZrX3Fk0oJRAIdevYYew4YWoFFF-
6_7w_Bay8Teek7OjcMd6XOy87ame-
8XLaUdfD3KMm7Bewp3OQejuEx5sH5WO7h881pbvz4G8WTY3totV4Kah4cWeYHWlDvHWdlnUUgjP_ArcU7DNxKw5Uk51OO0Mgd_m__NId11f_CpMQsb783
wPB33KYDxbrQBsurmDjiuQYhhzrtBX6tYH_e0W8ILlnz1pNtMJ6pnlT8TVa4P57dgwaE2JIKt0ByAC2oWONm8Wur9mcLUkPmxbOTkpfmzCNTmBALM8219
gCnG8PxQJQeVOPGUO-61riqyEtUwEu_Fml0UpgK4Xm6kOikd9NSgjx9Tuz0Fbx_x__gn5bhY9b2FKWIUPvSCGi8b9-be2GOgufkNKDfjn3tt42qPI3-
VqA2jJDEeSYkU9K_GNGxAmKFGiWmP5S-rNMHhM79_gxJt9jJTG9zkXv6L022kYMvzKzQsqITPcUOI2RD1syu6GrCFZ7s4rArhRzEK8i3CK9-lw-u-
I2jHdj-
b84KDHMlYcVRV7OjhYtNg6Nn8G1Vj9vHyJdxIHQg8I3KbZu85dKV1OUnVbw8RlAsSoqvjIZyYtwCR0zZ47g1Mi3QqOt_UXti94nw5tgn5DleSE27qTjwR
69UyhS1luwU38zKKcgWzbatgApad2c6Za97C1jo55fzdwMp2-
HbJSKlVfHrKeLCw_FmnoslNzKWrcO0W1_S4lD6zX7q8NQxHXBUEF1wwQrPCKr44Yju880Ms_g94zpxxKHfOEUwX4mSfAQpZN1w5Se4FMkPCQk6sknVgPM
nY7NakUlJ5H_PF5O-WL5gNxI0WMcB9v5KP-DESUA-qe_LrgVs6Gwgzg6N48ULVY1ajZt60yzvVg.l7ZyYi_MbvtQrs6VkEh2Dw'

api=ChatGPT(session_token)

import time

time.sleep(10)

# Iterate over each row in the Excel file

for index, row in df.iterrows():

    # Get the value of the 'input' cell in this row

    input_text = row['Reviews']

    time.sleep(3)

    resp=api.send_message("Give me answer in one word, this is postive or negative or netural sentance "+input_text)

    time.sleep(3)

    output_text=resp['message']

    print(output_text)

    text = output_text

    clean_text = re.sub(r'\.', '', text)

    print(clean_text)

    # Write the generated text to the 'output' cell in this row
```

```
    df.at[index, 'Sentiments'] = clean_text

    api.refresh_chat_page()

    api.reset_conversation()
# Write the updated dataframe back to the same Excel file

df.to_excel('output_file.xlsx', index=False)
```

1. api=ChatGPT(session_token): This line creates an instance of the ChatGPT API object, which is used to communicate with the OpenAI GPT model. The session_token argument is a unique identifier that is used to track the conversation state between requests.

2. import time: This line imports the Python time module, which provides functions for working with time and delays.

3. time.sleep(10): This line pauses the program for 10 seconds using the time.sleep() function. This is likely done to allow the ChatGPT API to initialize and establish a connection to the OpenAI servers.

4. for index, row in df.iterrows():: This line begins a for loop that iterates over each row in the Pandas DataFrame df. The index variable contains the index of the current row, while row contains the actual row data.

5. input_text = row['Reviews']: This line extracts the value of the Reviews column for the current row and assigns it to the input_text variable.

6. time.sleep(3): This line pauses the program for 3 seconds. This delay may be intended to avoid overwhelming the ChatGPT API with too many requests at once.

7. resp=api.send_message("Give me answer in one word, this is positive or negative or neutral sentence "+input_text): This line sends a message to the ChatGPT API using the send_message() method of the api object. The message text is constructed by concatenating the string "Give me answer in one word, this is positive or negative or neutral sentence " with the value of input_text. The response from the API is stored in the resp variable.

8. time.sleep(3): This line pauses the program for 3 seconds.

9. output_text=resp['message']: This line extracts the message text from the response returned by the ChatGPT API and assigns it to the output_text variable.

10. print(output_text): This line prints the output_text variable to the console.

11. text = output_text: This line assigns the value of output_text to the text variable.

12. clean_text = re.sub(r'\.', '', text): This line uses the Python re (regular expression) module to remove all occurrences of the "." character from the text variable. The resulting string is assigned to the clean_text variable.

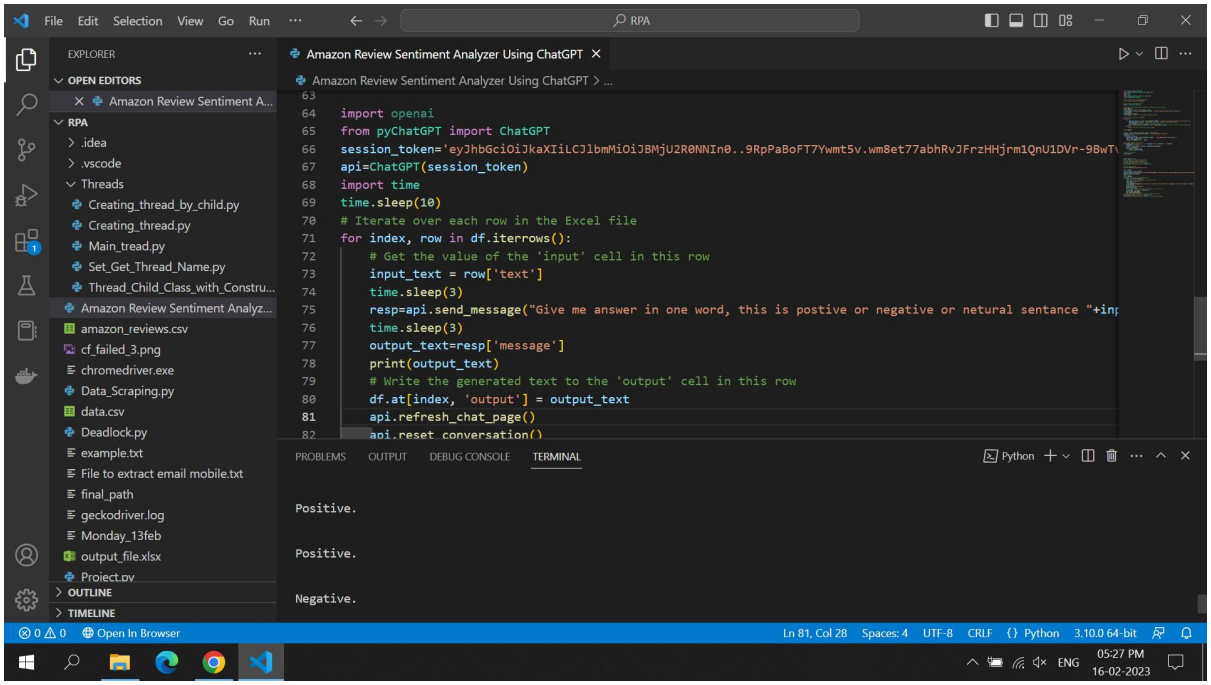
13. print(clean_text): This line prints the clean_text variable to the console.

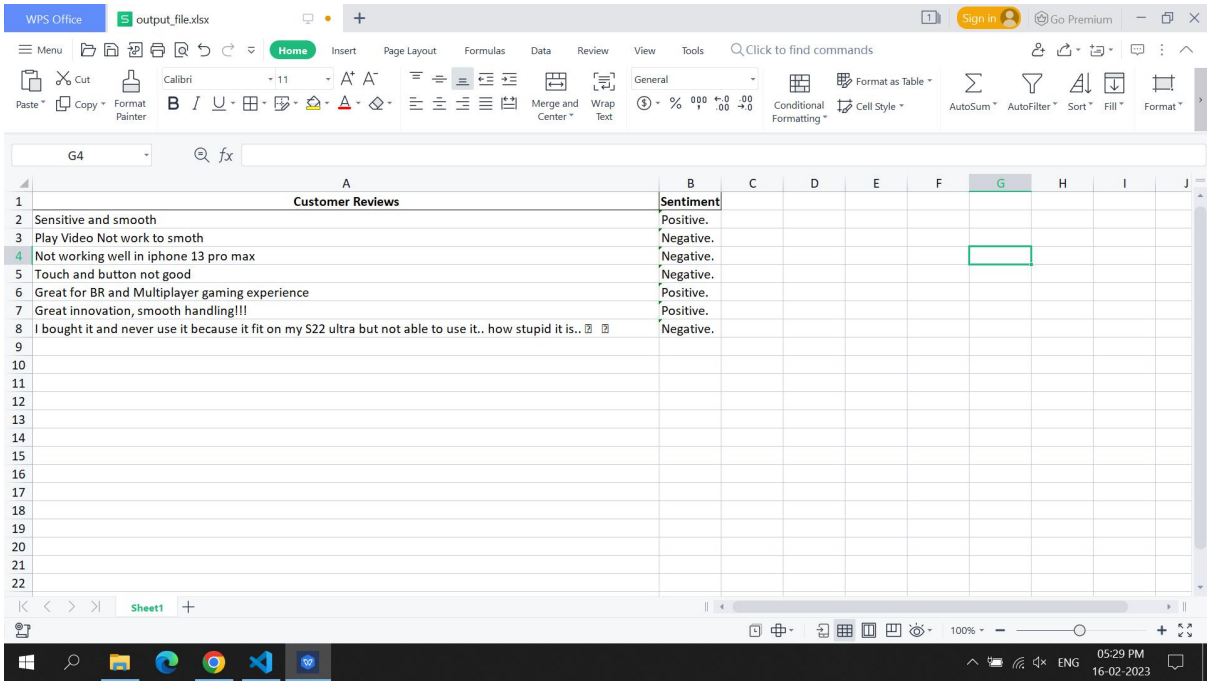
14. df.at[index, 'Sentiments'] = clean_text: This line updates the value of the Sentiments column for the current row in the Pandas DataFrame df to the value of clean_text.

15. api.refresh_chat_page(): This line refreshes the ChatGPT API page, which may be necessary to avoid errors or issues with the API connection.

16. api.reset_conversation(): This line resets the conversation state of the ChatGPT API, which clears any context or memory from previous requests.

17. df.to_excel('output_file.xlsx', index=False): This line exports the updated Pandas DataFrame df to an Excel file called output_file.xlsx, with the index=False


File Edit Selection View Go Run
RPA
EXPLORER
OPEN EDITORS
Amazon Review Sentiment A...
RPA
.idea
.vscode
Threads
Creating_thread_by_child.py
Creating_thread.py
Main_tread.py
Set_Get_Thread_Name.py
Thread_Child_Class_with_Constru...
Amazon Review Sentiment Analyz...
amazon_reviews.csv
cf_failed_3.png
chromedriver.exe
Data_Scraping.py
data.csv
Deadlock.py
example.txt
File to extract email mobile.txt
final_path
geckodriver.log
Monday_13feb
output_file.xlsx
Project.py
OUTLINE
TIMELINE
Amazon Review Sentiment Analyzer Using ChatGPT
Amazon Review Sentiment Analyzer Using ChatGPT > ...
63
64 import openai
65 from pyChatGPT import ChatGPT
66 session_token='eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIn0..9RpPaBoFT7Ywmt5v.wm8et77abhRvJFrzHHjrm1QnU1DVr-9BwT
67 api=ChatGPT(session_token)
68 import time
69 time.sleep(10)
70 # Iterate over each row in the Excel file
71 for index, row in df.iterrows():
72 # Get the value of the 'input' cell in this row
73 input_text = row['text']
74 time.sleep(3)
75 resp=api.send_message("Give me answer in one word, this is postive or negative or netural sentance "+inp
76 time.sleep(3)
77 output_text=resp['message']
78 print(output_text)
79 # Write the generated text to the 'output' cell in this row
80 df.at[index, 'output'] = output_text
81 api.refresh_chat_page()
82 api.reset_conversation()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python
Positive.
Positive.
Negative.
0 0
Open In Browser
Ln 81, Col 28 Spaces: 4 UTF-8 CRLF Python 3.10.0 64-bit
ENG
05:27 PM
16-02-2023


WPS Office
output_file.xlsx
Sign in
Go Premium
Menu
Home
Insert
Page Layout
Formulas
Data
Review
View
Tools
Click to find commands
Paste
Cut
Copy
Format Painter
Calibri
11
Merge and Center
Wrap Text
General
Conditional Formatting
Format as Table
Cell Style
AutoSum
AutoFilter
Sort
Fill
Format
G4
Customer Reviews
Sentiment
Sensitive and smooth
Positive.
Play Video Not work to smoth
Negative.
Not working well in iphone 13 pro max
Negative.
Touch and button not good
Negative.
Great for BR and Multiplayer gaming experience
Positive.
Great innovation, smooth handling!!!
Positive.
I bought it and never use it because it fit on my S22 ultra but not able to use it.. how stupid it is..
Negative.
Sheet1
100%
ENG
05:29 PM
16-02-2023