Python - Web Scraping with OpenAI

Setup

API Key from OpenAI

```
https://platform.openai.com/organization/api-keys
```

```
python3 -m pip install openai
```

```
python3 -m pip install requests
```

```
python3 -m pip install beautifulsoup4
```

```
python3 -m pip install feedparser
```

```
python3 -m pip install youtube-transcript-api
```

Tested on MacOS. Should work fine on Ubuntu.

Auto Post

This lab allows you to scrape a web page and then have OpenAI rewrite the post, and create a featured image to go with the new post.

We then download the image to our computer, and write the post and the image imbed to an HTML file.

lab-auto-post.py

```
from openai import OpenAI
from requests import get
from bs4 import BeautifulSoup

client = OpenAI()

```

```
# openai_key ='API Key from OpenAI'
# client = OpenAI(api_key=openai_key)

def scrape(url):
    page = get(url).text
    soup = BeautifulSoup(page, 'html.parser')
    post = soup.find_all('p')
    text =''
    for line in post: #Create a string without HTML tags
        text = f'{text} {line.text}'
    return text

def ai(query):
    completion = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "system", "content": "You are a blogger."},
        {"role": "user", "content": f"Create a 200 word blod post about -- {query}"},
        {"role": "user", "content": f"Do not mention author"},
        {"role": "user", "content": f"Do not mention when post was written"},
    ]
    )
    response = completion.choices[0].message.content
    return response

def ai_image(query):
    response = client.images.generate(
        model="dall-e-3",
        prompt=query,
        n=1,
        size="1792x1024"
    )

    pic_name = f'{response.created}.png' #Using Timestamp for Name
    response_image = get(response.data[0].url)
    with open(pic_name, 'wb') as file: #Mode 'wb' allows for writing non text files
```

```
        file.write(response_image.content) #.content is the full file from a get request

    return pic_name

url = 'https://arstechnica.com/space/2024/08/china-deploys-first-satellites-for-a-broadband-
network-to-rival-starlink/'
query = 'provide a 20 word summary'
result_bs = scrape(url)
result_ai = ai(result_bs)
pic_name = ai_image(result_ai)

with open('auto-post.html', 'a') as file:
    file.write(f'<img style="height:300px; width:auto;" src="{pic_name}">')
    file.write(f'<p>{result_ai}</p>')

print(pic_name)
print(result_ai)
```

Autoblog

This lab has your script scrape an RSS feed for links to articles. We then use this list to iterate through and have OpenAI rewrite the articles and create new titles for them.

We then write the output to an HTML file.

IMPORTANT!!! -->> We limit the number of posts rewritten to 5 by using [:5] in the for loop. Standard RSS feeds can have dozens of items so for cost, and time purposes it's best to limit the loop.

lab-auto-blog.py

```
from openai import OpenAI
import feedparser
from bs4 import BeautifulSoup
from requests import get
```

```
client = OpenAI()

# openai_key ='API Key from OpenAI'
# client = OpenAI(api_key=openai_key)

def build_list(url):
    rss_feed = get(url).text
    feed = feedparser.parse(rss_feed)

    url_list = []
    for post in feed['entries']:
        url_list.append(post['link'])

    return url_list

def scrape(url):
    page = get(url).text
    soup = BeautifulSoup(page, 'html.parser')
    post = soup.find_all('p')
    text =''
    for line in post: #Create a string without HTML tags
        text = f'{text} {line.text}'
    return text

def ai(query):
    completion_post = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "system", "content": "You are a Blogger."},
        {"role": "user", "content": f"Write a 100 word blog post on -- {query}"},
        {"role": "user", "content": f"Do not mention author"},
        {"role": "user", "content": f"Do not mention when post was written"},
    ]
    )
    response_post = completion_post.choices[0].message.content
    completion_title = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "system", "content": "You are a Blogger."},
        {"role": "user", "content": f"Create a Title for a blog post about -- {response_post}"},
    ]
    )
    response_title = completion_title.choices[0].message.content

    return response_title, response_post

url = 'https://feeds.arstechnica.com/arstechnica/index'

url_list = build_list(url)

for page in url_list[:5]: #[:5] limits the loop to 5 iterations
    response_bs = scrape(page)
    response_ai = ai(response_bs)

    print(f'{response_ai[0]}\n {response_ai[1]}')

    with open('auto-blog.html', 'a') as file:
        file.write(f'<h1>{response_ai[0]}</h1>')
        file.write(f'<p>{response_ai[1]}<p>')
```

Latest News

This lab allows you to scrape an RSS feed and ask OpenAI questions about what is in the feed.

Note: for os.system function we use 'clear' on Mac and Linux, but the command is 'cls' on Windows.

lab-news.py

```
from openai import OpenAI
import feedparser
from requests import get
import os

client = OpenAI()

# openai_key ='API Key from OpenAI'
# client = OpenAI(api_key=openai_key)

def scrape_feed(url):
    rss_feed = get(url).text
    feed = feedparser.parse(rss_feed)
    text = ''
    for post in feed['entries']:
        text = f'{text} {post["title"]} - {post["description"]}'

    return text

def ai(query, text):
    completion = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": f"provide answer from this document {text}"},
        {"role": "user", "content": query}
    ]
    )
    response = completion.choices[0].message.content
    return response

while True:
    url = 'https://feeds.arstechnica.com/arstechnica/index'
    query = input('Question:  ')
    os.system('clear')
    result_text = scrape_feed(url)
    result_ai = ai(query, result_text)

    print(query)
    print(result_ai)
```