ARTIFICIAL
INTELLIGENCE

AMAZON WEB SCRAPPER

Efficient Amazon Web Scraper: Extracting Key Product Data for Market Analysis and Competitive Insights.

Mansoor Bukhari

Presented for:
Sir Muhammad Rizwan

Task: Web Scraping Project - Amazon Books Data Extraction

Objective: Your task is to develop a Python script that can scrape data from multiple pages of Amazon's search results for books and store the extracted information in a CSV file. The script should handle pagination effectively and extract specific details from each book listing.

Website: Amazon Search Results for "Books"

Link: (https://www.amazon.com/s?k=boiks&page=4&crid=3FTUGPSBLEW3Z&qid=1725369915&sprefix=boik%2Caps%2C490&ref=sr_pg_4)

Requirements:

1. **Scraping the First Page:**
   - Start by scraping the first page of the search results.
   - Extract the book title, author, price, rating, number of reviews, and link to each book.
2. **Handling Pagination:**
   - Identify the pagination mechanism on Amazon.
   - Implement a loop to scrape data from multiple pages (decide the number of pages to scrape).
   - Handle potential errors, such as encountering CAPTCHAs or page not found (404) errors.
3. **Data Storage:**
   - Store the extracted data (title, author, price, rating, number of reviews, and link) in a Python dictionary.
   - Convert the dictionary into a DataFrame using pandas.
4. **Saving to CSV:**
   - Save the data into a CSV file named `amazon_books_data.csv`.
   - Optionally, save the data into an Excel file named `amazon_books_data.xlsx`.
5. **Additional Requirements:**
   - Ensure the script is well-commented, explaining each step of the process.
   - The script should be efficient and handle any exceptions that may occur during the scraping process.
   - Due to Amazon's strict anti-scraping measures, consider adding random delays between requests and using a user-agent header.

Steps to Complete the Task:

1. **Write the Code:**

   - Begin by writing a script to scrape data from the first page of Amazon's search results.
   - Implement the pagination mechanism using a loop to iterate through multiple pages.
   - Extract the necessary details: title, author, price, rating, number of reviews, and link.
2. **Handle Errors:**
   - Add error handling for CAPTCHAs, 404 errors, and potential blocking by Amazon.
   - Consider using proxies or a rotating user-agent to avoid being blocked.
3. **Save the Data:**
   - Use pandas to convert the extracted data into a DataFrame.
   - Save the DataFrame to a CSV file and optionally to an Excel file.
4. **Testing:**
   - Run the script to ensure it scrapes data from multiple pages successfully.
   - Check the CSV and Excel files to confirm the data is correctly stored.
5. **Submit the Script:**
   - Submit your completed Python script along with the generated `amazon_books_data.csv` and optionally `amazon_books_data.xlsx` files.

Expected Output:

- A Python script (`amazon_books_scraper.py`) that successfully scrapes book data from multiple Amazon search results pages.
- A CSV file (`amazon_books_data.csv`) and optionally an Excel file (`amazon_books_data.xlsx`) containing the scraped data.

Important Note: Amazon has strict terms of service regarding web scraping, and they actively block scrapers. Ensure that your script is used responsibly, considering ethical guidelines, and for educational purposes only.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
from time import sleep
import random

data = []

data = []  # Initialize list to store scraped data

# Loop through pages
for page in range(1, 50):
    url = f'https://www.amazon.com/s?
k=boiks&page={page}&qid=1725444238&ref=sr_pg_{page}'
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.96
Safari/537.36",
        "Accept-Language": "en-US,en;q=0.9",
        "Accept-Encoding": "gzip, deflate, br",
        "Connection": "keep-alive",
        "Upgrade-Insecure-Requests": "1",
        "Referer": "https://www.amazon.com/",
        "Origin": "https://www.amazon.com/"
    }

    # Make the request
    r = requests.get(url, headers=headers)

    if r.status_code == 200:  # Check if the request was successful
        soup = BeautifulSoup(r.content, 'html.parser')

        # Find all product items
        for item in soup.select('.s-main-slot .s-result-item'):
            # Extract book name
            title_element = item.select_one('.a-size-medium.a-color-
base.a-text-normal')
            title = title_element.get_text(strip=True) if
title_element else 'N/A'

            # Extract author name
            author_element = item.select_one('.a-size-base.a-link-
normal')
            author = author_element.get_text(strip=True) if
author_element else 'N/A'

            # Extract price
            price_element = item.select_one('.a-price .a-offscreen')
            price = price_element.get_text(strip=True) if
price_element else 'N/A'
```

```
            # Extract rating
            rating_element = item.select_one('.a-icon-star-small')
            rating = rating_element.get_text(strip=True) if
rating_element else 'N/A'

            # Extract number of reviews
            reviews_element = item.select_one('.a-size-base')
            reviews = reviews_element.get_text(strip=True) if
reviews_element else 'N/A'

            # Extract book link
            link_element = item.select_one('.a-link-normal.s-
underline-text')
            link = f"https://www.amazon.com{link_element['href']}" if
link_element else 'N/A'

            # Append data to the list
            data.append({
                'Title': title,
                'Author': author,
                'Price': price,
                'Rating': rating,
                'Reviews': reviews,
                'Link': link
            })

        # Random sleep interval between 1 and 5 seconds
        sleep(random.uniform(1, 5))
    else:
        print(f"Failed to retrieve page {page} with status code
{r.status_code}")
        break
```

While Scraping this website 99% of time we got `503 error`

What is 503 error?

- A 503 Service Unavailable error indicates that the server is currently unable to handle the request. This can happen for various reasons, and it usually signifies that the server is overloaded or undergoing maintenance. When dealing with web scraping, encountering a 503 error often means that your requests are being blocked or rate-limited by the server due to excessive traffic or other protective measures.

Common Causes of 503 Errors in Web Scraping:

- **Rate Limiting:** The server might have detected too many requests coming from your IP address and is temporarily blocking further requests.
- **IP Blocking:** Your IP might be blocked due to suspicious activity or scraping behavior.
- **Server Overload:** The server might be experiencing high traffic or other issues, causing it to temporarily refuse connections.

- **Bot Protection:** The site might have mechanisms in place to detect and block bots or automated scraping tools.

Which Methods I used to tackle this issue

- Their are alot of methods which we can use to handle this issue like
  - Use Proxies
  - Handle Cookies and Session
  - Respect Robots.txt
  - Respect Rate Limit
- But I use two methods to handle it
  - Adding Headers
  - Sleep For Some Time

1. Adding Headers Headers can make your request look more like a genuine browser request, which can sometimes help avoid detection and rate-limiting. Ensure your headers include:
   - **User-Agent:** Identifies the browser making the request.
   - **Accept-Language:** Specifies the preferred language.
   - **Accept-Encoding:** Indicates the accepted content encoding.
   - **Connection:** Controls whether to keep the connection open or close it.
2. Implementing Sleep Delays
   - Adding delays between requests helps in preventing rate-limiting by spacing out your requests. Implementing sleep is crucial for mimicking natural browsing behavior and avoiding detection.

```
# Create a DataFrame from the collected data
df = pd.DataFrame(data)

# Save the DataFrame to a CSV file
df.to_csv('amazon_books.csv', index=False)
print("Data has been saved to amazon_books.csv")
```

```
Data has been saved to amazon_books.csv
```

```
df = pd.read_csv('amazon_books.csv')

df.head()
```

```
                                                Title        Author
Price  \
0                                                 NaN           NaN
NaN
1                            If He Had Been with Me  Laura Nowlin
$7.27
```

```
2  Lessons from the Greatest Stock Traders of All...     John Boik
$14.73
3  How Legendary Traders Made Millions: Profiting...     John Boik
$15.55
4                                                  NaN            NaN
NaN

                Rating
Link
0                  NaN
NaN
1  4.2 out of 5 stars
https://www.amazon.com/If-He-Had-Been-Me/dp/17...
2  4.7 out of 5 stars  https://www.amazon.com/Lessons-Greatest-
Stock-...
3  4.6 out of 5 stars  https://www.amazon.com/How-Legendary-Traders-
M...
4                  NaN
NaN

df.shape

(275, 5)
```