scraping-amazon

December 11, 2023

[ ]:
```
import requests
from bs4 import BeautifulSoup
import csv

def extract_product_info(soup):
    try:
        title = soup.find("span", attrs={"id": 'productTitle'}).text.strip().
 ↪replace(',', '')
    except AttributeError:
        title = "NA"

    try:
        price = soup.find("span", attrs={'id': 'priceblock_ourprice'}).text.
 ↪strip().replace(',', '')
    except AttributeError:
        price = "NA"

    try:
        rating = soup.find("i", attrs={'class': 'a-icon a-icon-star␣
 ↪a-star-4-5'}).text.strip().replace(',', '')
    except AttributeError:
        try:
            rating = soup.find("span", attrs={'class': 'a-icon-alt'}).text.
 ↪strip().replace(',', '')
        except AttributeError:
            rating = "NA"

    try:
        review_count = soup.find("span", attrs={'id': 'acrCustomerReviewText'}).
 ↪text.strip().replace(',', '')
    except AttributeError:
        review_count = "NA"

    try:
        available = soup.find("div", attrs={'id': 'availability'}).find("span").
 ↪text.strip().replace(',', '')
    except AttributeError:
```

```
        available = "NA"

    return title, price, rating, review_count, available

def extract_reviews(soup):
    reviews = soup.find_all("div", class_="review")

    review_info = []
    for review in reviews:
        title_element = review.find("a", class_="review-title")
        rating_element = review.find("span", class_="review-rating")
        author_element = review.find("span", class_="a-profile-name")
        date_element = review.find("span", class_="review-date")
        body_element = review.find("div", class_="review-text")

        title = title_element.text.strip() if title_element else None
        rating = float(rating_element.text.strip()) if rating_element else None
        author = author_element.text.strip() if author_element else None
        date = date_element.text.strip() if date_element else None
        body = body_element.text.strip() if body_element else None

        review_info.append({"title": title, "rating": rating, "author": author,␣
 ↪"date": date, "body": body})

    return review_info

def save_review_info_to_csv(review_info, filename):
    with open(filename, "w", encoding="utf-8", newline='') as f:
        writer = csv.DictWriter(f, ["title", "rating", "author", "date",␣
 ↪"body"])
        writer.writeheader()
        writer.writerows(review_info)

def main(URL):
    # Specifying user agent
    HEADERS = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.
 ↪36 (KHTML, like Gecko) Chrome/44.0.2403.157 Safari/537.36',
               'Accept-Language': 'en-US, en;q=0.5'}

    # Making the HTTP Request
    webpage = requests.get(URL, headers=HEADERS)
    soup = BeautifulSoup(webpage.content, "html.parser")

    # Extract product information
    title, price, rating, review_count, available = extract_product_info(soup)

    # Save product information to a CSV file
```

```
    with open("product_info.csv", "a", encoding="utf-8", newline='') as f:
        writer = csv.writer(f)
        writer.writerow([title, price, rating, review_count, available])

    # Extract and save reviews to a CSV file
    review_info = extract_reviews(soup)
    save_review_info_to_csv(review_info, f"reviews_product_{review_count}.csv")

if __name__ == '__main__':
    # URLs to scrape
    urls = [
        "https://www.amazon.com/
 ↪Soundcore-Cancelling-Headphones-Comfortable-Bluetooth/dp/B08HMWZBXC/
 ↪ref=sr_1_10?keywords=headphones&sr=8-10&th=1",
        "https://www.amazon.com/dp/B0828PYKZN/ref=sspa_dk_detail_0?
 ↪ie=UTF8&s=electronics&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhaWxfdGhlbWF0aWM&pd_rd_i=B0828PYKZN&th=
    ]

    # Iterating over the URLs
    for url in urls:
        main(url)
```

[13]:
```
from textblob import TextBlob

def sentiment_analysis(text):
    analysis = TextBlob(text)
    return analysis.sentiment.polarity
```

[31]:
```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

nltk.download('stopwords')
nltk.download('punkt')


def extract_keywords(text):
    if isinstance(text, str):  # Check if text is a string
        # Tokenize the text
        words = word_tokenize(text)

        # Remove stop words
        stop_words = set(stopwords.words('english'))
        filtered_words = [word.lower() for word in words if word.isalnum() and␣
 ↪word.lower() not in stop_words]
```

```
        # Perform frequency distribution
        freq_dist = nltk.FreqDist(filtered_words)

        # Get the top 3 keywords
        top_keywords = freq_dist.most_common(3)

        return [keyword[0] for keyword in top_keywords]
    else:
        return []
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[32]:
```
def extract_features(text):
    # Count the number of words
    num_words = len(text.split())

    # Count the number of characters
    num_characters = len(text)

    # Return a dictionary with extracted features
    return {
        'num_words': num_words,
        'num_characters': num_characters,
        # Add more features as needed
    }
```

[33]:
```
import math

def extract_features(text):
    if isinstance(text, str) and not math.isnan(text):
        # Count the number of words
        num_words = len(text.split())

        # Count the number of characters
        num_characters = len(text)

        # Return a dictionary with extracted features
        return {
            'num_words': num_words,
            'num_characters': num_characters,
            # Add more features as needed
        }
    else:
        # Return a default dictionary for NaN values
```

```
        return {
            'num_words': 0,
            'num_characters': 0,
        }
```

[30]:
```
import pandas as pd

# Read review data from CSV files
product_1_reviews = pd.read_csv("/content/reviews_product_60337 ratings.csv")
product_2_reviews = pd.read_csv("/content/reviews_product_69060 ratings.csv")

# Analyze average rating
avg_rating_1 = product_1_reviews["rating"].mean()
avg_rating_2 = product_2_reviews["rating"].mean()

# Analyze sentiment
sentiment_1 = product_1_reviews["body"].fillna("").apply(sentiment_analysis)
sentiment_2 = product_2_reviews["body"].fillna("").apply(sentiment_analysis)


# Analyze common keywords
keywords_1 = product_1_reviews["body"].apply(extract_keywords)
keywords_2 = product_2_reviews["body"].apply(extract_keywords)

# Compare product features
product_1_features = []
product_2_features = []

for review in product_1_reviews["body"]:
    product_1_features.append(extract_features(review))

for review in product_2_reviews["body"]:
    product_2_features.append(extract_features(review))
```