Product Review Aggregator and Sentiment Classifier

Overview

This tool aggregates product reviews and classifies sentiments for specific aspects across multiple e-commerce platforms. Using web scraping, NLP, and sentiment analysis, it processes review data and generates structured CSV files with both overall and aspect-specific sentiment labels.

Objective

- **Aggregate** reviews from e-commerce websites.
- **Classify** sentiments related to key product aspects like “battery life,” “delivery,” and “price.”
- **Output**: CSV files with review content and sentiment classifications.

Tools Used

- **Selenium & BeautifulSoup**: Web scraping.
- **FlashText**: Recognizing entities and extracting product-specific keywords.
- **NLTK**: Preprocessing reviews (tokenization, stop word removal).
- **Flair**: Sentiment analysis and aspect-based classification.

Setup and Execution

1. Install Dependencies

Install required libraries before starting:

bash

Copy code

pip install selenium beautifulsoup4 pandas nltk flair FlashText

2. Scrape Product Reviews

Using Selenium and BeautifulSoup, reviews are scraped from given URLs for each product.

python

Copy code

```
from selenium import webdriver
from bs4 import BeautifulSoup
import time


# Set up Selenium driver
driver = webdriver.Chrome()
```

```
urls = ["URL1", "URL2", "URL3"]  # Add product URLs here
for url in urls:
    driver.get(url)
    time.sleep(2)  # Wait for page load

    # Scroll to load more reviews if needed
    scroll_pause = 2
    for _ in range(5):
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(scroll_pause)

    # Parse HTML with BeautifulSoup
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    reviews = soup.find_all('div', class_='a-expander-content reviewText')  # Adjust selector as needed

    driver.quit()  # Close driver
```

3. Save Reviews to CSV

Extracted reviews are stored in a DataFrame and saved to a CSV file for further processing.

python

Copy code

```
import pandas as pd

# Convert to DataFrame and save as CSV
df = pd.DataFrame(reviews_text, columns=["Review"])
df.to_csv('reviews.csv', index=False)
```

Data Preprocessing

NLTK is used to preprocess reviews, including tokenization, stop word removal, and lemmatization.

python

Copy code

```
import nltk
```

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

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import string

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
punctuation_table = str.maketrans('', '', string.punctuation)

def preprocess_review(text):
    text = text.lower().translate(punctuation_table)  # Remove punctuation
    tokens = word_tokenize(text)
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word.isalpha() and word not in stop_words]
    return tokens
```

Aspect-Based Sentiment Analysis

1. Initialize Flair Sentiment Model

Flair's pre-trained sentiment model is used to analyze each review for specific aspects.

python

Copy code

```
import flair
from flair.data import Sentence

sentiment_model = flair.models.TextClassifier.load('en-sentiment')
```

2. Classify Aspect-Based Sentiments

Define product aspects and classify sentiment for each aspect within each review.

python

Copy code

```
aspects = ['battery life', 'delivery', 'display quality', 'price', 'durability']


def classify_aspect_sentiments(reviews, aspects):
    aspect_sentiments = {aspect: [] for aspect in aspects}
    for review in reviews:
        sentence = Sentence(review)
        sentiment_model.predict(sentence)
        sentiment = sentence.labels[0]


        for aspect in aspects:
            if aspect in review:
                aspect_sentiments[aspect].append(str(sentiment))
    return aspect_sentiments


aspect_sentiments = classify_aspect_sentiments(reviews, aspects)
```

3. Save Aspect-Based Sentiments to CSV

Store the results in a structured CSV file.

python

Copy code

```
import csv


with open('aspect_sentiments.csv', mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(["Aspect", "Sentiment"])


    for aspect, sentiments in aspect_sentiments.items():
        for sentiment in sentiments:
            writer.writerow([aspect, sentiment])
```

Example Output

- reviews.csv: Contains all scraped product reviews.
- aspect_sentiments.csv: Contains aspect-specific sentiments for each review.

Troubleshooting

- **Selector Adjustments**: Adjust soup.find_all selectors to match each site's HTML structure.
- **Scroll Count**: Tweak Selenium scroll settings if the review page requires more or fewer scrolls to load reviews.

This solution offers an efficient way to collect and analyze product review sentiments, providing valuable insights for evaluating product quality across multiple platforms.