Scraping GitHub Trending Topics and Repositories

Overview

This notebook demonstrates how to scrape trending topics and popular repositories from GitHub. It retrieves:

- Trending topics and their descriptions.
- Top 25 repositories for each topic, along with relevant details such as repository name, owner, stars, and URL.

Libraries Used

- `requests` to fetch web pages
- `BeautifulSoup` (from `bs4`) to parse HTML and extract information
- `Pandas` to organize and save data

Objectives

1. Extract trending GitHub topics and their descriptions.
2. Retrieve details for the top repositories associated with each topic.
3. Save the scraped information into structured CSV files for each topic.

> **Note**: Please ensure that your scraping adheres to GitHub's Terms of Service, especially regarding request rates.

```
# Install required libraries if they are not already installed
!pip install requests --upgrade --quiet
!pip install beautifulsoup4 --upgrade --quiet

# Import necessary libraries for web scraping and data processing
import requests
from bs4 import BeautifulSoup
import pandas as pd
import os
```

Fetching the GitHub Topics Page

The following function retrieves the HTML content of GitHub's Topics page.

```
def get_topics_page():
    """
    Retrieves the HTML content of GitHub's Topics page.

    Returns:
        BeautifulSoup object containing parsed HTML of the page.
    Raises:
        Exception: If the page cannot be loaded.
    """
    topics_url = 'https://github.com/topics'
    response = requests.get(topics_url)
    if response.status_code != 200:
        raise Exception(f"Failed to load page {topics_url}")

    # Parse the page content using BeautifulSoup
    doc = BeautifulSoup(response.text, 'html.parser')
    return doc

doc = get_topics_page()
```

Extracting Topic Titles

This function fetches and returns a list of topic titles from the GitHub Topics page.

```
def get_topic_titles(doc):
    """
    Extracts and returns a list of topic titles from the GitHub Topics
page.

    Parameters:
        doc (BeautifulSoup): Parsed HTML of the topics page.

    Returns:
        list: Topic titles as strings.
    """
    # Find elements containing the topic titles based on HTML
structure
    selection_class = 'f3 lh-condensed mb-0 mt-1 Link--primary'
    topic_title_tags = doc.find_all('p', {'class': selection_class})
    topic_titles = []
    for tag in topic_title_tags:
        topic_titles.append(tag.text)
    return topic_titles

titles = get_topic_titles(doc)

len(titles)

30

titles[:6]

['3D', 'Ajax', 'Algorithm', 'Amp', 'Android', 'Angular']
```

Extracting Topic Descriptions

This function retrieves descriptions for each GitHub topic.

```python
def get_topic_descs(doc):
    """
    Extracts and returns descriptions for each GitHub topic.

    Parameters:
        doc (BeautifulSoup): Parsed HTML of the topics page.

    Returns:
        list: Topic descriptions as strings.
    """
    # Find elements containing the topic descriptions based on HTML
structure
    desc_selector = 'f5 color-fg-muted mb-0 mt-1'
    topic_desc_tags = doc.find_all('p', {'class': desc_selector})
    topic_descs = []
    for tag in topic_desc_tags:
        topic_descs.append(tag.text.strip())
    return topic_descs
```

Extracting Topic URLs

The function below extracts and returns URLs for each GitHub topic.

```python
def get_topic_urls(doc):
    """
    Extracts and returns URLs for each GitHub topic.

    Parameters:
        doc (BeautifulSoup): Parsed HTML of the topics page.

    Returns:
        list: Topic URLs as strings.
    """
    # Construct full URLs for each topic
    topic_link_tags = doc.find_all('a', {'class': 'no-underline flex-1
d-flex flex-column'})
    topic_urls = []
    base_url = 'https://github.com'
    for tag in topic_link_tags:
        topic_urls.append(base_url + tag['href'])
    return topic_urls
```

Main Scraping Function for Topics

The function below aggregates all topic data (title, description, URL) into a DataFrame.

```
def scrape_topics():
    """
    Scrapes the GitHub Topics page for topic titles, descriptions, and
URLs.

    Returns:
        pd.DataFrame: DataFrame containing 'title', 'description', and
'url' columns.
    """
    # Fetch and parse the GitHub topics page
    topics_url = 'https://github.com/topics'
    response = requests.get(topics_url)
    if response.status_code != 200:
        raise Exception('Failed to load page {}'.format(topic_url))
    doc = BeautifulSoup(response.text, 'html.parser')
    topics_dict = {
        'title': get_topic_titles(doc),
        'description': get_topic_descs(doc),
        'url': get_topic_urls(doc)
    }
    return pd.DataFrame(topics_dict)
```

Extracting Repositories for a Topic

This function retrieves the top repositories for a specific GitHub topic.

```
def get_topic_page(topic_url):
    # Download the page
    response = requests.get(topic_url)
    # Check successful response
    if response.status_code != 200:
        raise Exception('Failed to load page {}'.format(topic_url))
    # Parse using Beautiful soup
    topic_doc = BeautifulSoup(response.text, 'html.parser')
    return topic_doc

def parse_star_count(stars):
    stars=stars.strip()
    if stars[-1]=='k':
        return int(float(stars[:-1])*1000)
    return(int(stars))

def get_repo_info(h1_tag, star_tag):
    # returns all the required info about a repository
    base_url = 'https://github.com'
    a_tags = h1_tag.find_all('a')
    username = a_tags[0].text.strip()
    repo_name = a_tags[1].text.strip()
    repo_url =  base_url + a_tags[1]['href']
    stars = parse_star_count(star_tag.text.strip())
    return username, repo_name, stars, repo_url

def get_topic_repos(topic_doc):
    # Get the h1 tags containing repo title, repo URL and username
    repo_tags = topic_doc.find_all('article',{'class':'border rounded 
color-shadow-small color-bg-subtle my-4'})

    # Get star tags
    star_tags=topic_doc.find_all('span',{'id':'repo-stars-counter-
star'})

    topic_repos_dict = { 'username': [], 'repo_name': [], 'stars': 
[],'repo_url': []}

    # Get repo info
    for i in range(len(repo_tags)):
        repo_info = get_repo_info(repo_tags[i], star_tags[i])
        topic_repos_dict['username'].append(repo_info[0])
        topic_repos_dict['repo_name'].append(repo_info[1])
        topic_repos_dict['stars'].append(repo_info[2])
        topic_repos_dict['repo_url'].append(repo_info[3])

    return pd.DataFrame(topic_repos_dict)

def scrape_topic(topic_url, path):
    if os.path.exists(path):
        print("The file {} already exists. Skipping...".format(path))
        return
    topic_df = get_topic_repos(get_topic_page(topic_url))
    topic_df.to_csv(path, index=None)

def scrape_topics_repos():
    print('Scraping list of topics')
    topics_df = scrape_topics()

    os.makedirs('data', exist_ok=True)
    for index, row in topics_df.iterrows():
        print('Scraping top repositories for 
"{}"'.format(row['title']))
        scrape_topic(row['url'], 'data/{}.csv'.format(row['title']))
```

Complete Scraping Function

The `scrape_topics_repos` function orchestrates the scraping process by:

1. Scraping topics from the GitHub Topics page.
2. Extracting details of top repositories for each topic.
3. Saving the repository data for each topic to individual CSV files.

```
scrape_topics_repos()
```

```
Scraping list of topics
Scraping top repositories for "3D"
Scraping top repositories for "Ajax"
Scraping top repositories for "Algorithm"
Scraping top repositories for "Amp"
Scraping top repositories for "Android"
Scraping top repositories for "Angular"
Scraping top repositories for "Ansible"
Scraping top repositories for "API"
Scraping top repositories for "Arduino"
Scraping top repositories for "ASP.NET"
Scraping top repositories for "Awesome Lists"
Scraping top repositories for "Amazon Web Services"
Scraping top repositories for "Azure"
Scraping top repositories for "Babel"
Scraping top repositories for "Bash"
Scraping top repositories for "Bitcoin"
Scraping top repositories for "Bootstrap"
Scraping top repositories for "Bot"
Scraping top repositories for "C"
Scraping top repositories for "Chrome"
Scraping top repositories for "Chrome extension"
Scraping top repositories for "Command-line interface"
Scraping top repositories for "Clojure"
Scraping top repositories for "Code quality"
Scraping top repositories for "Code review"
Scraping top repositories for "Compiler"
Scraping top repositories for "Continuous integration"
Scraping top repositories for "C++"
Scraping top repositories for "Cryptocurrency"
Scraping top repositories for "Crystal"
```