The Process of Crawling Douban Movie Data

1. Technical Approach

The main technologies and tools used in this code are:

* **requests**: Used to send HTTP requests and retrieve HTML content from the target pages.
* **pyquery**: Used for parsing HTML and extracting the necessary data fields.
* **Proxy Pool**: Randomly selects proxies to reduce the likelihood of IP blocking.
* **Random Delays**: Adds random delays between requests to simulate human behavior and avoid being flagged as a bot.

1. Data Fields Extracted

Based on project requirements, each comment’s data includes the following fields:

* **ID**: Comment’s sequence number (starting from 0).
* **Movie\_Name\_EN**: Movie’s English title.
* **Movie\_Name\_CN**: Movie’s Chinese title.
* **Crawl\_Date**: Date when the data was crawled.
* **Number**: Unique ID for each comment on Douban.
* **Username**: Username of the commenter.
* **Date**: Date the comment was posted.
* **Star**: User’s rating for the movie (1-5 stars).
* **Comment**: Content of the comment.
* **Like**: Number of likes on the comment.

1. Code Implementation
   1. Initialization and Movie List

The code defines the IDs of 28 movies and their corresponding English and Chinese titles. The template for each movie’s comment page URL is as follows:

[https://movie.douban.com/subject/{movie\_id}/comments?start={start}](https://movie.douban.com/subject/%7bmovie_id%7d/comments?start=%7bstart%7d)

Each page contains 20 comments, and by controlling the start parameter (0, 20, 40, ...), multiple pages of comments can be retrieved.

* 1. Pagination Crawling

To retrieve 100,000 comments, the code implements pagination:

1. **Loop through each movie**: For each movie ID, the code retrieves comments from the specified number of pages.
2. **20 Comments per Page**: After each request, the code increments the start parameter by 20 to get the next page’s comments.
3. **Set Page Limit**: The max\_pages parameter controls the maximum number of pages for each movie, preventing excessive requests for a single movie.
   1. Data Extraction and Parsing

For each page of HTML data, the code uses the pyquery library to extract comment details:

1. **Parse HTML**: The HTML content is converted to a pyquery object, doc.
2. **Locate Comments**: Each comment is identified using the .comment-item selector.
3. **Field Extraction**: For each comment, the following data is extracted:
   * **ID**: Sequential number (incremental).
   * **Movie\_Name\_EN** and **Movie\_Name\_CN**: Movie’s English and Chinese titles.
   * **Crawl\_Date**: Date when the data was crawled.
   * **Number**: Unique ID of the comment.
   * **Username**: Username of the commenter.
   * **Date**: Date the comment was posted.
   * **Star**: User’s rating (from CSS class name).
   * **Comment**: Content of the comment.
   * **Like**: Number of likes on the comment.

Each comment’s data is stored in a dictionary and added to the list of comments.

1. Anti-Scraping Mechanisms

The code includes the following anti-scraping measures to reduce the risk of being blocked:

1. **Proxy Pool**: A predefined list of proxies, from which one is randomly selected for each request.
2. **Dynamic Headers**: Adds User-Agent and language settings to mimic a real browser request.
3. **Random Delays**: Each request is followed by a random delay of 1 to 3 seconds, simulating human browsing behavior to prevent frequent access from being flagged as bot activity.
4. Results

During execution, the program successfully retrieved comments for 28 movies, totaling around 100,000 comments. Each comment contains details like username, date, rating, comment content, and number of likes, fulfilling the data requirements.

Movie Data Cleaning Process

1. Data Import

The dataset is imported from a CSV file located in Google Drive using pandas.read\_csv, with UTF-8 encoding to handle any special characters. Initial exploration of the data is conducted with df.head() and df.info() to inspect the structure and identify potential issues such as missing values or formatting inconsistencies.

1. **Handling Missing Values**

Missing values are addressed by removing rows where critical fields (Comment or Star) are empty, ensuring only complete entries are retained in the dataset. This is essential for data reliability, as missing comments or ratings could skew the results in further analysis.

df.dropna(subset=['Comment', 'Star'], inplace=True)

1. Handling Duplicate Entries

Duplicate rows are removed using drop\_duplicates, ensuring that each entry in the dataset represents a unique comment or rating. This step prevents data from being counted multiple times, which could otherwise impact the accuracy of any descriptive statistics or modeling performed on the dataset.

df.drop\_duplicates(inplace=True)

1. Date Format Conversion

The Date column is converted to datetime format to facilitate accurate time-based analyses. If any date conversion fails due to incorrect formats, these rows are removed to maintain consistency in the dataset.

df['Date'] = pd.to\_datetime(df['Date'], errors='coerce')

df.dropna(subset=['Date'], inplace=True)

1. Text Cleaning in Comments

The Comment field undergoes text cleaning, where any extraneous symbols or emojis are removed using regular expressions. This process ensures that the comments contain only alphanumeric characters and whitespace, facilitating more reliable text analysis, such as sentiment analysis or keyword extraction.

df['Comment'] = df['Comment'].apply(lambda x: re.sub(r'[^\w\s]', '', x))

1. Preventing Chinese Encoding Issues

import matplotlib.pyplot as plt plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

1. Exporting Cleaned Data

The cleaned dataset is saved as a new CSV file with UTF-8 encoding. This file can be used for further analysis without needing additional data cleaning steps.

df.to\_csv('/content/drive/MyDrive/cleaned\_DMSC.csv', index=False, encoding='utf-8-sig')