Retrieve and Analyze Goodreads Data

Genre considered: **War**

Introduction:

In this task, we aim to perform data extraction, transformation, and loading (ETL) from the Goodreads website, focusing on books related to the genre of "**war**" published in the last five years. Using a Python script, we iteratively retrieve data from Goodreads' search results pages. This structured data will subsequently undergo exploratory data analysis (EDA) to identify trends and insights, such as highly rated authors and potential correlations between book length and ratings, helping us uncover valuable patterns within the dataset.

Scrapping tool used:

The script utilizes BeautifulSoup to parse the HTML content and extract relevant information such as book titles, authors, publication years, page counts, book types, ratings, and the counts of ratings and reviews. The retrieved data is then filtered to include only books published between 2019 and 2024.

Step-by-Step Approach:

Initialization:

- Import necessary libraries such as requests for making HTTP requests, BeautifulSoup from bs4 for parsing HTML content, and tqdm for progress tracking.
- Initialize lists (book_titles, book_links, pages, whole_data) to store data points and an integer (data_points) to count the total number of data points retrieved.

Iterate Over Pages:

- Use a for loop to iterate through Goodreads search result pages (from page 1 to 2000). For each page:
- Construct the URL for the current page by concatenating the base URL with the current page number.

Send HTTP Request:

- Use the requests.get method to send a GET request to the constructed URL with custom headers to mimic a real browser request.

Parse HTML Content:

- Decode the response content and parse the HTML using BeautifulSoup.

Extract Book Titles and Links:

- Find all elements with the class bookTitle which contain the book titles and links to detailed book pages.

For each book link:

- Increment the data_points counter.
- Extract and clean the book title and store it in the book_titles list.
- Construct the full book link and store it in the book_links list.

Extract Detailed Book, Publication Year and Author Information:

- Find the author names associated with each book and store them in a list.
- Send a GET request to the detailed book link.
- Parse the detailed book page HTML content using BeautifulSoup.
- Locate the publication date element and extract the publication year.
- Store the publication year in the row data if it falls within the last five years (2019-2024).

Extract Page Count and Book Type:

- Locate and extract the page count and book type information and store them in the row data.

Extract Ratings Information:

- Locate and extract the book rating and store it as a floating-point number.
- Locate and extract the number of ratings and reviews and store them as integers.

Store Extracted Data:

- Append the cleaned and structured row data to the whole_data list.

Repeat:

- Continue the process for all pages and books found on each page.

Final Output:

- After the loop completes, the whole_data list will contain structured data for books related to the "war" genre published in the last five years, ready for exploratory data analysis (EDA).

Observations after EDA:

1) Based on the heat map shown, we can say that the review_count and rating_count are very strongly positively correlated. A correlation coefficient (r) of 0.93 suggests a very strong linear relationship between the two variables. This means that as one variable increases, the other variable tends to also increase in a predictable manner.

2) Seeing the univariate analysis, we can say the following:
   - Hardcover books are mostly considered.
   - Most frequent rating is between 4.1 and 4.2.
   - # of pages are mostly between 310 and 360.

- Highest number of books are published in 2022.

3) Kindle has gained popularity in the year 2021 as this type has the highest count only in the year throughout the past 5 years. Except for 2021, Hardcover books have been on the top hand.

4) The hardcover book count over the years has been fluctuating although it has been the most common book considered over the past 5 years.

5) To analyse the distribution over the points, violin plot is considered.
   - The review_count and rating_count depicts a lot of similar distribution as they are strongly positively correlated to each other.
   - Coming to rating, hardcover books are uniformly rated between 3.8 and 4.4 and kindle books are uniformly rated between 4.1 and 4.6.

6) Considering the aggregate values, we have took bar plots.
   - The highest mean rating is for Mass market paperback books. But this information might not be reliable as the count of mass market paperback books constitutes about 0.5% of the whole dataset. The most reliable mean rating is for the hardcover books as they constitute as the highest count.
   - Most of the books have only a single author.

7) To visualize the magnitude of values in a two-dimensional way, heatmaps are considered.
   - In 2021, the mean page count was highest for the books in Good category.
   - Since the rating_count and review_count are corelated, we can see that the density of both the vaiables are high in the years 2020 for Average books and 2019 for Good books.
   - Even though the mean page_count is highest for Mass Market Paperback for Good books, we cannot say that it is the most reliable information as the number of samples considered are very less compared to any other book_type.
   - For Good books of eBook book_type, the mean rating_count and review_count for the books are significantly higher than any other book_type.
   - When page_count has been category is introduced, the Long books have the highest mean rating.
   - The long books 2024 have the highest rating when compared to short and medium-length books over any other years.
   - The review_count and rating_count is the highest for Long book category in the year 2020.
   - The rating count for Long books is highest when considering eBook book_type.

   Here, we can say that, when categorized based on page_count, Long books stand in the highest position and these books have most number of ratings and reviews which made them more popular comparatively.

8) Analysis on top 5 books.
    - Although there is no significant difference between the ratings of the top 5 books, we can see that “God of War” has the highest rating.
    - The book Rhythm of War has the highest rating_count and review_count among all the five books.

Assumptions taken into consideration while performing EDA:

Assumption 1: Converting the Ratings feature to categorical feature

- if ratings is < 3.9 it is categorized as Average
- if rating si more than 3.9, it is categorized as Good

Assumption 2: Converting the page_count feature to categorical feature

- if page_count is < 100 it is categorized as short
- if page_count si >=100 and less than 400, it is categorized as Medium-Length
- if page_count is >=400, and less than 1220, it is categorized as long books

Challenges:

- *Resource intensive and time consuming:*

  Parsing and processing HTML content for up to 2000 pages can be time-consuming and resource intensive. Optimizing the code to handle such large-scale data scraping efficiently, including proper error handling and resource management, is crucial.

- *# IP requests consideration:*

  In this task, since Goodreads often have rate limiting and anti-scraping measures in place, sending many requests may result in IP blocking. So, therefore, I have scrapped through 2000 pages considering the number of IP requests for the task.

- *Handling NaN or None values:*

  The structure of the web pages might not be consistent. For instance, some books might not have all the fields (e.g., publication date, rating, page count) resulting in NaN or None values. These rows consisting of NaN or None were removed to avoid inconsistences in the EDA.

- *Formatting issues:*

  The script needs to handle cases where expected data is missing or incorrectly formatted. For instance, not all books might have a rating or a publication date, which can cause issues if not properly handled.

Future improvements:

- *Use Proxies and Rotate User Agents:*

To prevent getting blocked, use a pool of proxies and rotate user agents. Libraries like requests and fake_useragent can help with this.

- *Implementing Delays:*

Considering delays between loading the pages might ensure that the content is loaded and captured effectively.

- *Parallelize Requests:*

Use threading or asynchronous requests to speed up the scraping process. This ensures that the time consumption is reduced and the resources are properly utilized.