CSC111 Project 2: Bored? Chapter Closed? Let Me Fix That!

Dorsa Rohani, Jiamei Huo, Ivy Huang, Behnaz Ghazanfari

Sunday, March 30, 2025

Introduction

With the rise of social media, the internet and video games, it seems like nowadays, many people are choosing other hobbies over reading. For instance, research from The Reading Agency has shown that 50% of adults in the UK don't read regularly as a hobby. Their survey conducted showed that the primary challenges to reading included a lack of time, difficulty in focusing attention, and difficulty in choosing titles to read (The Reading Agency). Our goal is to address these problems by quickly helping people choose a book and tailoring the book to their interests to improve concentration. Our project question is, **How can we recommend new books to people based on their preferred genres and star ratings?** To do this, our program will narrow down a dataset of books to 10 titles based on the user's preferences. The user states their preferred genres from the given list and inputs their minimum star rating in the Python console. The recommended books are based on how many genres overlap with the user input as well as how high their star rating is.

Dataset and Computational Overview

The Goodreads book dataset we found is structured as a CSV file, which contains a header and rows representing a book (BrightData). Each column represents the book information: its title, author, star rating, number of ratings, number of reviews, an array of genres, and a first published date.
Dataset Source: https://huggingface.co/datasets/BrightData/Goodreads-Books

Sample data:

```
name,author,star_rating,num_ratings,num_reviews,genres,first_published
"The Hobbit","['J.R.R. Tolkien']",4.28,3307,2677,"['Fantasy', 'Adventure', 'Fiction']",1937
"1984","['George Orwell']",4.18,3955,3125,"['Science Fiction', 'Dystopia', 'Classics']",1949
```

To store the book data, we made a `save_data.py file`, which takes the dataset from Hugging Face and creates a `data.csv` file. The primary data processing was done in the `process_and_validate_data` function in the `main.py` file, where we used the pandas library to read the CSV file and load it into a dataframe. Books with missing titles or authors were filtered out due to insufficient information. Missing ratings were filled with the average star ratings. Rows with duplicate book titles were dropped and the processed data then replaced the given CSV file. Lastly, the data is filtered using the `get_recommendations_for_user` method, under the `BookGraphVisualizer` class in `main.py`, based on the user input (as described in the introduction).

The dataset was converted into a graph structure in the `BookGraph` class of the `graph.py file`, where:

- **Vertices (Nodes)** represent individual books, containing the following information: title, author, star rating, number of ratings, number of reviews, genres, and a publishing date.
- **Edges (Connections between Nodes)** represent relationships between books based on rating quality and genre similarity.

Book Score Calculation

Two books are connected if they share at least one genre. The score of a book is determined using a combination of review-weighted scores and genre similarity. These calculations were done in the `_build_graph` method under the `BookGraph` class in `graph.py`.

1. Review-Weighted Score Calculation

Each book is assigned a score based on its community reviews:

$$S = (100 \times R_5) + (80 \times R_4) + (60 \times R_3) + (40 \times R_2) + (20 \times R_1)$$

where R_n represents the number of reviews for each star rating (1-5). The coefficients 100, 80, 60, 40, and 20 are the values assigned to ratings 5 through 1, respectively.

2. Genre Similarity Calculation

Each book is assigned a set of string values corresponding to its genres.

Example Calculation:
If:

- Book A has genres: {Fantasy, Adventure, Young Adult}
- Book B has genres: {Fantasy, Adventure, Sci-Fi}

Then: the number of shared genres between the two books is:

$$|Genres_A \cap Genres_B| = 2 \quad (Fantasy, Adventure)$$

The total number of unique genres across both books is:

$$|Genres_A \cup Genres_B| = 4 \quad (Fantasy, Adventure, YoungAdult, Sci-Fi)$$

Using the genre similarity formula:

$$G(A,B) = \frac{|Genres_A \cap Genres_B|}{|Genres_A \cup Genres_B|}$$

$$G(A,B) = \frac{2}{4} = 0.5$$

This means that Book A and Book B have a 50% genre similarity, which will affect their relevance score for the recommended books in the console. A higher genre similarity coefficient results in a stronger connection between books, making them more likely to be recommended together. This ensures that highly-rated books with shared themes are more strongly connected. The final graph consists of books nodes and edges, ensuring quality recommendations.

For this project, we used Plotly. Plotly.com was utilized in parts of our code (indicated as comments) to visualize the graph. We used Plotly's networkx graph plotting capabilities to display the book recommendation network (the method `_create_nextworkx_graph` under the `BookGraphVisualizer` class in `main.py`). The nodes and edges were plotted in a 2D '`plotly.graph_objects.Scatter,`' in the `visualize_full_graph` method. The colours for each node were randomized and then assigned based on the book's main genre. For example, if the color teal was randomly selected for the fantasy genre, a fantasy book would have a teal colored node.

Users can hover over nodes to view book details such as title, author, rating, and genres. The program will load a network graph with two subplots in your browser where each subplot shows a maximum of 10 multicolored book nodes. The **top of each subplot will show the highest rated book** in the preferred genres given, away from the rest of the nodes. They are adjacent to other book nodes, which are additional recommendations (other options that may not necessarily match the user's preferences, unlike the 10 recommended books on the console that are closely related).

Instructions for Obtaining Datasets and Running the Program

1. Install Python libraries under `requirements.txt` file by running
   `pip install -r requirements.txt`

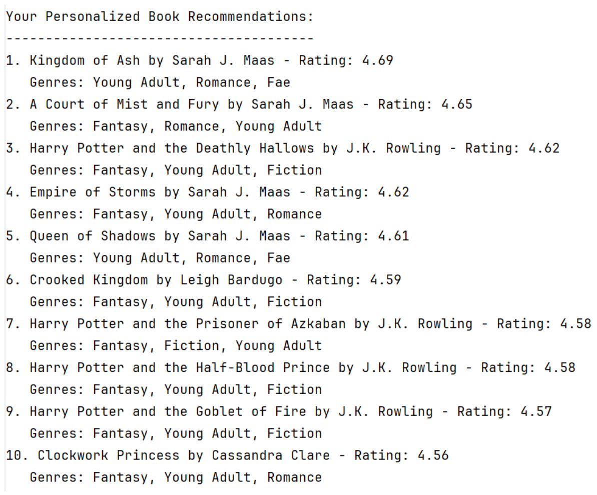
2. Download `data.zip`, the zip file which contains the required datasets.

3. Run `main.py`.

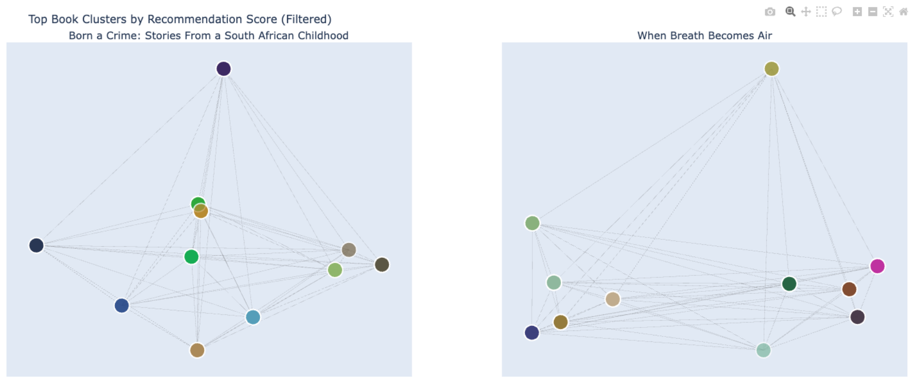
After running the program, you should see two questions related to your preferred genres and the minimum book rating. The console will output 10 personalized title recommendations, as well as recommendation quality metrics (relevance and diversity), where:

- Relevance: a score representing the match between book recommendations and genre preferences
- Diversity: a score representing the number of unique genres among all genres

It should look as described in the last paragraphs of Computational Overview.
Example: the console output and the visualized graph after selecting the genre of 'Young Adult' and a minimum book rating of 3.0. Every vertex's book information (title, author, rating, and genres) are displayed once hovered over with a cursor.



```
Your Personalized Book Recommendations:
----------------------------------------
1. Kingdom of Ash by Sarah J. Maas - Rating: 4.69
   Genres: Young Adult, Romance, Fae
2. A Court of Mist and Fury by Sarah J. Maas - Rating: 4.65
   Genres: Fantasy, Romance, Young Adult
3. Harry Potter and the Deathly Hallows by J.K. Rowling - Rating: 4.62
   Genres: Fantasy, Young Adult, Fiction
4. Empire of Storms by Sarah J. Maas - Rating: 4.62
   Genres: Fantasy, Young Adult, Romance
5. Queen of Shadows by Sarah J. Maas - Rating: 4.61
   Genres: Young Adult, Romance, Fae
6. Crooked Kingdom by Leigh Bardugo - Rating: 4.59
   Genres: Fantasy, Young Adult, Fiction
7. Harry Potter and the Prisoner of Azkaban by J.K. Rowling - Rating: 4.58
   Genres: Fantasy, Fiction, Young Adult
8. Harry Potter and the Half-Blood Prince by J.K. Rowling - Rating: 4.58
   Genres: Fantasy, Young Adult, Fiction
9. Harry Potter and the Goblet of Fire by J.K. Rowling - Rating: 4.57
   Genres: Fantasy, Young Adult, Fiction
10. Clockwork Princess by Cassandra Clare - Rating: 4.56
   Genres: Fantasy, Young Adult, Romance
```



Changes to the Project Plan

When discussing the project plan together, we decided to make the program more interactive by including user input and dropping some features we deemed unnecessary (e.g., color gradients, weighted edges, and “clicking a node will display a list of similar books with a dynamically updated visualization and a filtering option will allow users to select books by genre or rating). The features we dropped would clutter the graph and are not entirely relevant to the final book recommendations. We also implemented some of the TA feedback: we used pandas to load and process the data into the CSV file type, and in the report further explained the data processing and data storage. We also used subplots to compare recommendation groups and highlighted the top-rated books for the given genres. For the subplots, we decided to separate the top-rated book from the other recommended books so that it would be easy to distinguish.

Discussion

The main goal of our project was to see if we could give people good book recommendations by combining their input (preferred genres and star ratings) with graph computations. In general, our results appear to show that this approach works well and gives useful suggestions. The program consistently outputs 10 recommended books that relevantly match the user’s chosen genres and minimum rating. The interactive graph also makes it easy to explore more book connections, helping users not only find top-rated books but also discover similar ones through clusters.

A strength of our program is how it scores books in two ways. First, it uses review scores to reflect how popular and well-liked a book is. Second, it measures how similar the book’s genres are to the user’s preferences. Together, these two scores help recommend books that are both high-quality and a good match. By testing our program, the recommendations seemed accurate, especially when choosing popular genres like Fantasy or Sci-Fi.

However, there were some limitations with the dataset we found and algorithms we used. The dataset is large but focuses mostly on well-known books, so it may exclude more niche or less popular titles. It was not fully comprehensive, as we were unable to find a dataset that contained book descriptions/summaries, languages, user reading history, or behavioural data. Also, sometimes the graph included book suggestions that did not entirely match the user’s input. This is likely due to the calculations used for book scores; rating largely affects the scores of top book node’s neighbours, which in turn can affect the relevancy of the books in terms of genre. Furthermore, it was difficult to determine what the actual primary genres of a book would be based on the dataset, thus we took the first three genres as the main genres of each book.

Lastly, one challenge we faced was balancing the complexity of our program and its practicality for the user, as we needed to ensure that the graph visualisation would look aesthetically pleasing, not overly cluttered, and easy to interpret. To overcome this, we decided on providing 10 books with coloured nodes, which would make the graph look organized and would provide a good amount of recommendations.

We also noticed slower performance when working with a bigger dataset. Our system works well with smaller sets, but handling tens of thousands of books would need better optimization. Thus, in the future, we plan to use faster data structures or parallel processing to improve speed. We also want to make our genre-matching better by using more information, like book summaries or users' reading histories.

Overall, the project shows how graphs and data analysis can be used to make personalized book recommendations. The system is user-friendly and visually clear, making it a solid tool that can be improved further. Next steps include adding machine learning algorithms to make the recommendations even more accurate and helpful.

References

1. Bright Data. "Goodreads-Books." *Huggingface.co*, June 2024, `https://huggingface.co/datasets/BrightData/Goodreads-Books`. Accessed 30 Mar. 2025.

2. Plotly. "Plotly for Python." *Plotly*, `https://plotly.com/python/`. Accessed 30 Mar. 2025.

3. The Reading Agency. *The State of the Nation's Adult Reading: 2024 Report.* 2024, `https://readingagency.org.uk/wp-content/uploads/2024/07/State-of-the-Nations-Adult-Reading_2024-Overview-Report.pdf`.