LitLabyrinth: Your Next Reading Adventure

Janelle Yan, Abigail Burke, Niala Samlalsingh, Tanush Arora

April 2024

1 Introduction

Brief problem description and project question/goal:
Our goal: **How can we recommend new books to people based on previous community reviews and/or the person's preferences?**

We have always been captivated by the ability of books to transport us into different worlds, broaden our perspectives, and deepen our understanding of things. According to the Washington Post, about 51% of Americans read more than one book in 2023 and that's just one country (Andrew Van Dam) After a person finishes one book, they're faced with the inevitable problem of needing to find another book if they want to continue reading. They could scour bookshelves in the library or ask friends for hopeful recommendations. However, this could be time-consuming and may result in books that don't fit their preferences. Our project aims to fix that problem so that readers can continue to read for pleasure and easily find a book that they would enjoy based on the book that they have just finished and enjoyed.

Our project was born out of this passion for reading and the desire to contribute to global literacy efforts which align with Sustainable Development Goal (SDG) number 4 which is focused on ensuring inclusive and equitable quality education (United Nations). A system that recommends books could potentially be used in schools worldwide and ensure that students develop their reading skills and receive a quality education.

2 Datasets

There are two major datasets that we used for this project.

The first is named "Large book file" and the second is named "Large user file". The large book file consists of all of our books and the author, genres, number of pages, and a short summary of each book. For this dataset, the first column is the book titles, the second column is the authors, the third column is the number of pages each book has, and the fourth column has a short summary of the book. The book titles, authors, and number of pages were used from the STA130 GitHub (Schwartz). The genres and summaries were generated using ChatGPT.

The large user file contains each review that users made. This file consists of many users that have reviewed each book from the book file. Each book has between one and five reviews. This dataset contains the users, the book titles reviewed, the rating given, and a short review from the user. The books were taken from the book file. The usernames, ratings, and reviews were generated using ChatGPT.

All of this data was formatted in multiple Excel sheets and then exported into multiple CVS files.

3 Computational Overview

Our book recommendation system uses a graph as its main data structure to cohesively link every piece of information about our books and users. Our project uses book information and user review data to achieve this.

Our major computations include:

- Functions that calculate various similarity scores between books based on fields like author, genre, and rating.
- A function to calculate the average rating for a book based on reviews
- Adding vertices, edges, and vertex information using a combination of multiple functions to load the graph from our datasets
- Functions that perform minor data retrieval from the graph
- Functions that perform more major data retrieval for specific book information
- Multiple sorting and retrieval functions (along with their helpers) to recommend books based on multiple fields such as a preferred genre, author, or even just a list of liked titles (further discussed in the next point)
- The pride of our project is the function able to take in a list of titles rather than specific attributes and return titles similar to them either in terms of genre or author based on its ranking in scoring functions called upon in its function body
- Another function can take in a user ID and return books liked by that user if the searcher knows that the user has similar tastes
- Another function returns the most popular books in the system based on a scoring system similar to the one described above

For the functions in our program that do some form of similarity scoring, the scoring method is similar to what was done in exercise 3 where the score is calculated based on items in common divided by the total item set.

Our program reports the results in an interactive way by opening a pop-up with Tkinter that a user can use to get specific information about the books or reviewers. The user can also get a list of all the books and a list of all the users in case they are stuck on what the valid books and users to input are. When you first run the main.py file, a list of the valid actions the user can enter, general instructions on how to use the interface, a light-purple input box, and a submit button show up in the form of a GUI. The user can input anything into the input box and it will either give the user the info they requested or inform the user that they imputed an invalid input. Possible actions that the user can do include: - get a specific piece of information from a book(such as author or page number) - get books in a specific genre - get books by a specific author - get books reviewed by a specific reviewer - get all the books in the dataset - get all the reviewers in the dataset - book recommendations from a specific book - a specific number of the books with the highest rating in the data set Each time the user inputs something in the initial first window, a new pop-up will appear that will get any further information that the user needs to enter to get the desired information and will output the information the user desires. We also used Networkx and Plotly to create a visualization of the graph that shows whenever u first run the program and has purple dots being books, blue nodes being reviewers, and faint lines connecting books to reviewers.

As described above, our program uses the Python libraries tkinter, Networkx, and Plotly to create a GUI and visualization for our graph. To create a visualization, we utilize the libraries Networkx and plotly. By converting our graph into a Networkx graph, we are allowed to use the flexible Networkx and the large number of functions the library has to offer for manipulating the graph (Hagberg et al). We made use of specifically the node attribute and remove_edge to acquire the needed data and discard unneeded data to then utilize Scatter from the Plotly library to create 2 scatter plots to visualize the graph on the screen (Hagberg et al; Plotly Technologies Inc.). We then used Figure from Plotly to combine these 2 scatter plots and used functions such as .update_layout, .update_xaxes, and .update_yaxes to change the scatter plots visualization to our liking by turning off certain features such as the legend, the zero line, and the grid (Plotly Technologies Inc.). With all of this combined, Networkx and Plotly allowed us to display a slightly interactive, as users could hover over each dot in the visualization to see reviewer or book names.

In Tkinter, we first called Tk() to allow us access to the Tkinter's functions (freeCodeCamp.org; Van Rossum). We then displayed text on screen by making use of tkinter's Label() function, displayed interactive buttons with tkinter's Button() function, and allowed users to input text by creating a field that they could type in with tkinter's Entry() function (freeCodeCamp.org; Van Rossum). We also utilized the function bind() to configure the text in the labels such that the text did not get cut off (Van Rossum). Tkinter allowed us to create a GUI that the user can fully interact with and allowed us to display our information and results.

4 Obtaining datasets and running the program

All the necessary files will be uploaded into a zip folder on MarkUs. This includes our main.py, project2-part1 containing our main functions, project2-visualization with our visualizer code, requirements file, and four data sets. There will be a book dataset and a user data set pair; a smaller pair for testing, and a larger pair for seeing full-sized results.

At least one dataset pair will already be coded into the program and main.py can be run immediately in the Python console upon download. To switch between data sets, simply replace the 2 instances where they appear in main.py (near the top and bottom of the code) and run again in the Python console.

Once the code is run, you should expect to see a browser pop-up with the visualization of our main graph, as well as a window providing access to the graphic user interface of our choice for this project. From there, instructions on using the interface will appear on the screen.

Please note that to check our code, you may feel free to use the graph visualization which would have opened in your browser to access book/user names that exist in the dataset. Simply hover over a vertex to see the item stored in it. Books are in purple and users are in blue.

Things to Test:

Get information
- Author
- Genre
- Pages
- Summary
- Reviews
- Average rating
- General information

Books in (genre)

All books

All users

Find books read by (user)

Books by (author)

Book recs (w/ book)
- Singular book
- Multi Books

Most popular

5 Changes to project plan

- Our code roughly reflects our proposal. We did not make that many changes from the original idea that we submitted in the proposal. For example, we kept the idea of the book and user classes as vertices for our graph, and several of the ideas for our computations were implemented as functions. However, we used a completely new dataset from our original plan and adjusted functions accordingly, such as discarding the idea of displaying the date of review (since there was none). We also added functions/actions the user could do that we did not have in the proposal, such as a helpful list of all the books and reviewers. Our biggest difference came from our visualization aspect. We did not utilize most of the functions that we mentioned in the proposal, however, we did use

Networkx as we mentioned there. We combined this with the Plotly library to create our visualization for the graph (instead of just using Networkx as we mentioned in the proposal). With our TA's advice on the GUI in mind, we also introduced a concept that we didn't touch on at all during the proposal and that is using Tkinter to create an interactive GUI. We did not mention at all during the proposal how we were going to get the user to interact with our code except stating that the user could somehow get certain information from our datasets and graphs. For our final copy of the project, we utilized Tkinter to allow the user to do this and it was involved in a large amount of our code and creation of GUI especially.

6 Discussion

Our journey began with a clear vision: to develop a book recommendation system capable of suggesting books based on various criteria, similar to weighted graphs in our lectures. From the outset, we encountered a significant challenge – sourcing an adequate dataset. Despite our best efforts, finding a single dataset encompassing all the necessary information proved to be incredibly difficult, consequently, time had to be set aside to just merge datasets, format, and generate some usable data.

With our dataset challenges addressed, we shifted our focus to implementation. We devised methods for recommending books based on diverse criteria, drawing inspiration from our lectures and exercises. The process echoed the strategies we employed in handling weighted graphs, underscoring the practical relevance of our theoretical knowledge. As we delved deeper into coding and testing, we encountered the inevitable debugging challenges. However, armed with patience and teamwork, we tackled each issue methodically and resolved what we presently assume to be all of our errors. The implementation of the GUI came with a few complications which took more time than we had expected to be resolved, but those problems resolved themselves once we realized we overlooked a few steps in linking the interface to the functions. It was smooth sailing after these complications got smoothed out.

One of our proudest achievements was the successful implementation of our main function. This function, designed to accept single or multiple books and utilize our customized criteria for recommendations, represented the culmination of our collective efforts. Seeing our code come to life and produce meaningful results was immensely gratifying, reaffirming our belief in the potential of our project. This method was the main goal we had in mind before adding any additional filtering criteria. The idea was to ensure that users have access to books like the ones they'd read before without necessarily knowing what specific criteria to filter based on. The filtration system is quite primitive at the moment. However, we expect that as we grow as programmers, we'll learn about more tools that can be used to upgrade our filtration criteria and make steps toward perfecting our program.

Continuing to look ahead, we recognize that there is still room for growth and innovation in multiple areas of our program. We aspire to explore additional optional parameters, enhancing the versatility and accuracy of our recommendation system. Furthermore, we are eager to refine the user experience by developing more intuitive and visually appealing graphical user interfaces (GUIs). Some members of our group are particularly enthusiastic about integrating interactive features directly into the graph visualizer, envisioning a seamless and immersive user interaction.

Reflecting on this project, we discovered the importance of adaptability in the face of challenges, the power of collaboration in achieving shared goals, and the satisfaction that comes from seeing our efforts translate into tangible outcomes. Our project not only honed our technical skills and gave us greater exposure to new ways of implementing and interacting with our code but also fostered a sense of camaraderie and accomplishment within our group.

7 References

Tkinter Course - Create Graphic User Interfaces in Python Tutorial. Directed by freeCodeCamp.org, 2019. YouTube, https://www.youtube.com/watch?v=YXPyB4XeYLA. Accessed 4 Apr. 2024

Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008 Accessed 4 Apr. 2024

Plotly Technologies Inc. "Scatter." Collaborative Data Science, https://plotly.com/python/line-and-scatter/. Accessed 4 Apr. 2024.

Plotly Technologies Inc. "Plotly.Graph_objects.Figure — 5.20.0 Documentation." Collaborative Data Science, https://plotly.com/python-api- reference/generated/plotly.graph_objects.Figure.html. Accessed 4 Apr. 2024.

Van Rossum, G. "Tkinter — Python Interface to Tcl/Tk." Python Documentation, https://docs.python.org/3/library/tkinter.html. Accessed 4 Apr. 2024.

NetworkX Team. "Drawing — NetworkX 3.2.1 Documentation." NetworkX, https://networkx.org/documentation/stable/reference/drawing.html. Accessed 6 Mar. 2024.

Sadia Sharmin. "CSC111 Project 2 / Phase 1: Proposal." CSC111 Project 2 / Phase 1: Proposal, https://www.teach.cs.toronto.edu/c̃sc111h/winter/assignments/project2/phase1/. Accessed 6 Mar. 2024.

Scott Schwartz. "STA130_F23/Data/Amazonbooks.Csv at Main · point0five/STA130_F23." GitHub, https://github.com/point0five/STA130_F23/blob/main/Data/amazonbooks.csv. Accessed 6 Mar. 2024.

The United Nations. “Goal 4 — Department of Economic and Social Affairs.” United Nations,
https://sdgs.un.org/goals/goal4. Accessed 6 Mar. 2024.