PROJECT OVERVIEW - RMP Web Scraper and Database

GENERAL IDEA

Webscrapes RateMyProfessor website for UT Austin professors and ranks them by score.

- Can be filtered by the major each professor teaches.
- Allows data to be looked up by typing in a professor's first and last name
- Outputs a list of professors ranked by their score given a certain major.

Data Structures Used:

- Hash Table
- Queue

Algorithm Type Used:

- Merge Sort (Divide and Conquer Sorting Algorithm)

DATASETS

Datasets will be generated by the web scraping code and stored as a .csv file to be analyzed by the main portion of the code. This is because ratings constantly change with each new review, so the dataset can't stay stagnant.

METHODS & LIBRARIES

The main packages I used were Selenium to webscrape RMP and Pandas to create the .csv file. I also used Chrome and chromedriver in order to webscrape. In the main portion of the code, I used the tabulate module in order to be able to format the data collected in a table for easier reading.

The two classes I've included in the main portion of the code are the Queue class and the Hash Table class. The queue class is used to scroll through the top n professors while the hash table class is used to store the professor data, using their names as the key. These were taken from the GitHub repository provided by Professor Teymourian. I've included the merge_sort() function in order to sort the data based on their rating and reversed this list to have the highest rating be at the front. Especially with a large dataset, a divide-and-conquer algorithm such as merge sort was a better call than other sorting algorithms. The code for this algorithm was provided by chatgpt. There are two print functions that I've added–one for printing the professor's rating, name, department, total number of ratings, percentage who would retake the course, and difficulty in table format. The other simply prints the professor's name and department in table format. The purpose of this is so the user can find a professor by name to look them up if they only know the professor's department. There is also a filtered_list() function which takes in two parameters with the most important one being the department the user wishes to filter the data by. The main_menu() function reprints the list of commands available in case the user needs to refer to them again. The command_menu() function takes in the command the user inputs and executes it–whether that be searching a professor by name, clearing a filter, or adding a filter. Finally, the main() function is the one that takes in the user command input and calls the command_menu() function. If the command doesn't exist, it loops and asks the user again for a valid command.

As a note: any professors with a total number of ratings equating to zero won't be included in the final dataset. I've also discarded any professor's scores if they had less than 5 reviews since their ratings are likely to skew the final results.

TEST CASES

For each .csv file generated, the test cases would need to be modified since the data is constantly changing. However, when running the code, I checked to make sure the ranking is correct and that the filter was properly working whether the user was adding one or removing one.

EXPECTATIONS

For now, I expect the software to be very elementary since it's only able to generate rankings and organize the data by the values obtained. Since RMP isn't an official university website, the amount of professors on there is dependent on the students who write reviews for them, so there will be many professors who are not included in the dataset.

Additionally, past professors who no longer work at UT may also be present on the board, and any professors who have transferred from another university to UT and haven't had that information updated on RMP won't be included in the count. Unfortunately, Professor Teymourian is a part of that statistic since it shows that he is still a professor at Boston University. However, I went ahead and manually added him at the end of the .csv file attached since I was curious where he ranked (though running the web scraper won't produce that same result). Moreover, some professors who didn't appear on the main board (but do appear when the filter for their department is turned on) will also be missing from the final count.

The main weakness of this is that the ranking system is a bit inaccurate–let's say two professors are ranked 5, but professor A has 5 reviews while professor B has 100 reviews. Of course, professor B should be ranked higher than professor A, but it is possible that professor A's name will be on top. For improvement purposes, having the

professors ranked by rating and then having professors of the same score further ranked by rating number would provide a more accurate representation.
Moreover, the queue that I used only allows for the user to scroll down the queue since the previous n professors have lined up again at the end of the line. However, the user should be able to see the previous n professors by scrolling back up in the terminal.

As for the hash table, I set the starting bucket size to 1000, so it's possible for some buckets to have multiple entries or to have no data at all. However, I would have liked to have the bucket size be the same number as the number of entries–however, I could not find a way to do this, but it is a good area for further improvement.

And lastly, I used the file functions such as reading a file and closing it rather than using the input method that we have been using all semester with the sys module. This is because I was more comfortable with the former method, but the latter method would have the option for the user to choose a professor file to analyze rather than hard-coding one in. This is the last place of improvement that I could think of.

NOTES
*** In the submission, I've included this project overview file, the term_project_main.py file which is where the ranking and other functions takes place after the .csv file is created, the webscraper_rmp.py file which when run will web scrape the information on UT Austin professors, and the .csv file that I generated when web scraping on November 24, 2023 in case the web scraper takes too long or to test the term_project_main.py file immediately.

The web scraper itself requires a little help from the user (just clicking X at the bottom banner ad once). The other large pop-up ad and cookies popup have been dealt with in the code. I have run a test on the web scraper to make sure it works (although it took roughly 40-50 minutes before I was given back the file with the information I scraped. For time's sake, I have uploaded it under the name "professors.csv". This file was created on November 24, 2023, so a few ratings might change between the interval of its creation and the date of the project presentation.

Also, while running the webscraper on terminal, sometimes the page rendering would time out and raise the TimeoutException. However, I have dealt with this by forcing the program to rerun until the page fully loads, so running the web scraper might take quite a bit of time. The longest run while I was testing it was around 2 hours.
I noticed too that the number that shows how many professors are at the university when the url is clicked (~ 4600) is inaccurate to how many are actually present. I thought it was an issue with my web scraper, but I tested it on a smaller scale (filtered it

to a department with less professors in it) and physically counted the data and realized there was a mismatch between the two numbers. Moreover, despite there being ~3500 entries in the .csv file, when the entries with less than 5 total ratings are removed, only ~1500 remain, so many professors are not included in the final list due to this reason as well.