BLM19202E Data Structures Programming Assignment #3
Movie Recommendation System using Heap-Based Collaborative Filtering
(31 May 2023, 23:59)

In this project, you will implement a movie recommendation system using heap-based collaborative filtering. Collaborative filtering is a popular technique used in recommender systems to make recommendations based on the preferences of similar users. In a heap-based approach, the top-N recommendations are computed by maintaining a heap of the most similar users to the target user.

You will use a dataset of user-movie matrix where each row represents a user and each column represents a movie, and the matrix values represent the user's ratings with the movies.

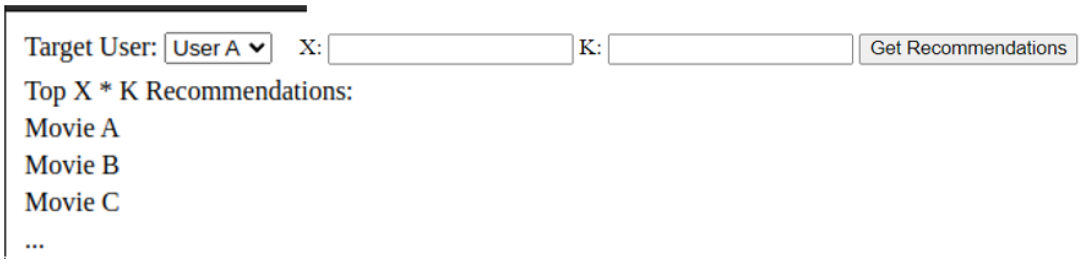
Next, you will compute the similarity between each pair of users using a cosine similarity metric. You will then construct a heap of the most similar users to the target user, where the heap is ordered by the similarity score. Top-N movie recommendation should contain the X highest rated movies from the K similar users to the target user selected. **For example: Assume that X is 5 and K is 3, you should list 15 movie recommendations. The first 5 movies are selected from the highest rated movies by the most similar user to the target user. Note: Each row in the target_user.csv represents the target user.**

The Project will include the following requirements:

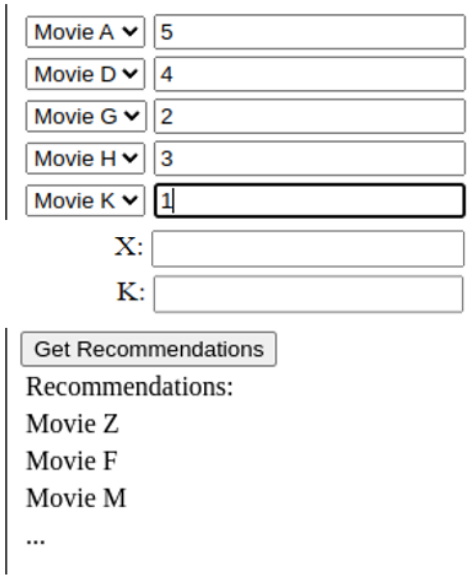
- The program should be written in Java.
- The program should read a user-movie matrix from a file and store it in a data structure properly. You can use LinkedList, BST, HashTable or 2D matrix. You should determine the data structure that can store the user-movie matrix.
- The program should be able to compute the similarity between each pair of users using a cosine similarity metric.
- The program should be able to maintain a heap of the most similar users to the target user.
- The program should be able to retrieve the X * K movie recommendations using most similar users (should store in heap) to the target user.
- X and K values should be parametric. These values can be changed by the user.

GUI Design:

- You should design two different user interfaces:
    a. Get Recommendations according to the target user:
        i. In the first screen, the user only selects the target user from combo box (or you can use another selection component to list the target users). Target usernames should be read from the “target_user.csv” file.
        ii. After this selection (you can start your calculations after the selection, or you can use the get recommendation button), your algorithm computes the similarity between each pair of users in the “main_data.csv” file and the target user using cosine similarity.
        iii. Then, each user is stored in a heap according to their similarity to the target user. Most similar users can be stored in the root.
        iv. Get the most similar X user from the heap and list K highest rated movies for these users. In the total, you should retrieve X * K movies.
        v. You do not list movie ids, you should list the movie names. You can read the movie names from the “movies.csv” file.



b. Get Recommendation according to the movies:
   i. To create a movie rating grid, design a 5x5 layout where each row represents a movie, and each column represents the rating given by the user. The movie names should be retrieved from the "movies.csv" file and displayed in a combo box or another selection component. The combo box should contain ten randomly selected movies from the file, rather than all of them. Users can enter ratings for each movie in the corresponding text fields.
   ii. User should select 5 different movie names using the selection component and enter corresponding rating value between [1-5] (should be integer value).
   iii. Then, these selections are converted to the vector (user vector).
   iv. Similarities are computed between each pair of users in the “main_data.csv” file and user vector using cosine similarity.
   v. Then, each user is stored in the heap according to their similarity to the target user. Most similar users can be stored in the root.
   vi. Get the most similar X user from the heap and list K highest rated movies for these users. In the total, you should retrieve X * K movies.
   vii. You do not list movie ids, you should list the movie names. You can read the movie names from the “movies.csv” file.



IMPLEMENTATION DETAILS

Insert and search operations will be done using heap. Please don't use arrays instead, **the projects prepared with array or other data structures will not be evaluated**.

You must write a report including your design and implementation details. The report also includes the output of your program.

You must design your GUI and perform each operation on this GUI.

GRADING:

1. Read user-movie matrix and selection of data structure properly (20 points).
2. Correct implementation of a heap to store similar users (25 points).
3. Correct implementation movie retrieval (20 points).
4. GUI design and working correctly (20 points).
5. Report (15 points)

Notes:
By the due date, please submit the source code of your program, on the submission on LMS. You must export the project to a JAR file and zip it with code files and report.

In this assignment you must work in a group. There must be a maximum of two students in each group.
Note that projects submitted after the project's due date will not be accepted and evaluated. Please keep this in mind and **promptly start working on your projects**! Similarity test is applied with all projects. Any potential violation of this rule will lead everyone involved to failing from all projects.

Good luck 😊