Assignment 3: User-based Collaborative Filtering

(Adapted from University of Minnesota CSci 1901H Class project)

Assignment Overview

In this assignment you will implement a simple user-based collaborative filtering recommender system for predicting the ratings of an item using the data given. This prediction should be done using k nearest neighbors and Pearson correlation. Finally using the similarity of the k nearest neighbors, you are required to predict the ratings of the new item for the given user.

Format of ratings file

- The input file consists of one rating event per line. Each rating event is of the form: user_id\trating\tmovie_title
- The user_id is a string that contains only alphanumeric characters and hyphens and spaces (no tabs).
- The rating is one of the float values 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, and 5.0.
- The movie_title is a string that may contain space characters (to separate the words).
- The three fields -- user_id, rating, and the movie_title -- are separated by a single tab character (\t).

Submission Details

You need to turn in a python script <firstname_lastname_collabFilter.py>. This python code should contain the below functions. Make sure your python file name qualifies the format. The name is in lowercase and starts with firstname.

Requirements:

1. pearson_correlation(user1, user2) : **(30 points)**
    - This function calculates the pearson correlation between 2 users.
    - Return value is a float between 1 and -1.
    - For calculating the average for each user, include all the user's ratings and not just the intersection of the 2 user's ratings.
    - However when computing summation, use only items that both users have rated.

2. K_nearest_neighbors(user1, k) : **(30 points)**
    - This function calculates the k nearest neighbors of user1 based on pearson similarity.
    - Returns a list of k nearest neighbors and their similarity.
    - For calculating the average for each user, include all the user's ratings and not just the intersection of the 2 user's ratings.
    - However when computing summation, use only items that both users have rated.
    - When sorting similarities, if 2 users have the same similarity sort them by user id.

3. Predict(user1, item, k_nearest_neighbors) : **(40 points)**
    - This function calculates the final prediction for item for user1 using k nearest neighbors.
    - You will compute a simple weighted average of the ratings provided by the k nearest neighbors.
    - Use only the neighbors who have rated the input item.
    - Prediction = $\sum(W_{i,1})*(\text{rating}_{i,item}) / \sum(W_{i,1})$ where $W_{i,1}$ is the similarity of user i with user1 from the k nearest neighbors.

Running your code

The program takes 4 arguments

- Ratings input file
- User id (user name)
- Movie name to calculate prediction for
- K for k neighbors

Execution

Python weiwei_duan_ collabFilter.py ratings-dataset.tsv ‘Kluver’ ‘The Fugitive’ 10

ratings-dataset.tsv: input file
Kluver: User id
Movie: The Fugitive
K: 10

Output:

The program will output:

- K nearest neighbors with their user ids and similarity values separated by space as per the output file. They should be output in order. The K nearest neighbors should be ordered by descending order of pearson similarity and if 2 users have the same similarity sort them by user id in ascending order. If your format is not correct, you will lose **20% points**.
- Rating prediction for item.

The following snapshot is the output of running the above command line:

```
terveen 0.87230104569
JosephIsAwesome 0.803235974634
Nick 0.7618781994
Connor M 0.684334217784
8ccfa5d6-6f0b-407f-a463-0e1f745f9dad 0.679393489084
vikram 0.641162448303
cond0155 0.626566965549
edc4c49c-d263-4d37-bc57-da5f7a344800 0.586622396117
What makes you think I'm not? 0.576068994558
14684581-beae-4309-890f-2c64c4be8fc0 0.551008660972
3.88023994249
```

General Instructions:

1. Do not zip your files
2. Make sure to follow the output format and the file naming format.
3. Make sure not to write the output to any files. Use standard output to print them.
4. We will be using Moss for plagiarism detection.