Assignment 3: Item-based Collaborative Filtering

Assignment Overview

In this assignment you will implement a simple item-based collaborative filtering recommender system. The ratings data you will use for developing and testing your program was generated by students.

Write your own code!

For this assignment to be an effective learning experience, you must write your own code! I emphasize this point because you will be able to find Python implementations of most or perhaps even all of the required functions on the web. Please do not look for or at any such code! **Do no share code with other students in the class!!**

Here's why:

- The most obvious reason is that it will be a huge temptation to cheat: if you include code written by anyone else in your solution to the assignment, you will be cheating. As mentioned in the syllabus, this is a very serious offense, and may lead to you failing the class.
- However, even if you do not directly include any code you look at in your solution, it surely will influence your coding. Put another way, it will short-circuit the process of you figuring out how to solve the problem, and will thus decrease how much you learn.

So, just don't look on the web for any code relevant to these problems. Don't do it.

Format of ratings file

The 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 (no whitespace, 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

What you will turn in: `lastname_firstname_collabFilter.py`

Program Description

You will read in the ratings file and compute the similarity between all pairs of items using the Pearson correlation equations we discussed in class:

For the item-based algorithm, denote the set of users $u \in U$ who rated both items i and j, then the *Pearson Correlation* will be

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \overline{r}_i)\left(r_{u,j} - \overline{r}_j\right)}{\sqrt{\sum_{u \in U} (r_{u,i} - \overline{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \overline{r}_j)^2}}, \qquad (2)$$

where $r_{u,i}$ is the rating of user u on item i, $\overline{r}_i$ is the average rating of the ith item by those users, see Figure 2 [40].

Where:

- **Note: Sum over set of users *U* who rated both items *i, j***
- $r_{u,i}$ is rating of user u on item i
- r_i is average rating of i^{th} item by those users

Clarification. Sometimes there is not enough data to compute the similarity between two items. In the worst case -- for example, if only one user rated both items -- the denominator in the formula will be zero. In such cases, your similarity function should just return 0.0.

Use the prediction equation we discussed in class to predict the rating of a user u on an item i that the user has not already rated:

- Predict rating for user u on item i, where:
  - $w_{i,n}$ is weight between items *i and n*
  - $r_{u,n}$ is rating for user u on item n
  - Summation over **neighborhood set *N* of items** rated by u that are most similar to i
  - Note: if there are fewer than N items with nonzero similarity to item i, then return a prediction of 0.0

$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} |w_{i,n}|}$$

where the summations are over all other rated items $n \in N$ for user u, $w_{i,n}$ is the weight between items i and n, $r_{u,n}$ is the rating for user u on item n.

Finally, your program should output a **list of k items to recommend to the user** specified as an input argument to the program. You should identify the k items with the highest predicted rating for this user. Be sure not to include any items that the user has already rated.

Some notes from the grader:

1. To truncate the prediction to 5 digits.
2. Whenever we use the sort function (to take the N neighbors and take k top recommendations) if the value we are sorting on is same consider alphabetical ordering of the movie names. This is to avoid inconsistency in the output.
3. Do not redirect the output to a file instead print it on stdout.

Running your code

The program takes 4 arguments: the name of the ratings file, the id of user1 (a string) for whom you will make recommendations, the size of the neighborhood n to be used in the prediction equation, and the number of recommendations that will be made for user1.

```
python collabFilter.py ratingsFileName user1 n k
```

The program will output the k recommended movies for user1 and the predicted rating for each of those movies.

Testing your code

The command to execute is :

python3.4 reco.py ratings-dataset.tsv Kluver 3 3