CIT 594 – Homework 4

Milestones

This homework has three milestones with the following points and deadlines:

~~Milestone 1 – Design (30 points)~~
~~Due Monday, Mar. 21, at 12pm~~

~~Milestone 2 – Implementation (70 points)~~
~~Due Monday, Mar. 28, at 12pm~~

Milestone 3 – Changing Requirements (30 points)
Due Sunday, Apr. 3, at 12pm

Note: Late submissions are not allowed for Milestone 3.

Recommender Systems – User-User Collaborative Filtering

Milestone 3 – Changing Requirements

For milestone 3, the goal is to modify your recommender system to support the following requirements. (Note: Your system should support both sets, the old ones from previous milestones and the new ones from this milestone, of requirements.) You can choose any 30 points out of the 50 points available. The remaining 20 points are available as Extra Credit:

1. **Changing the file type**
    a. Our Recommender system has become very popular and other companies want to use it. The BookCrossing community (http://www.bookcrossing.com/) has made their dataset available (http://www2.informatik.uni-freiburg.de/~cziegler/BX/) and want to see how well our system works with their data. The dataset contains ~1.1M ratings for ~270K users for ~271K books. (10 points)
    b. Our original MovieLens dataset was from 2009. Since then, we have a new dataset that was released in 2016 that has a new file format and a lot more data. The "full" dataset contains 22M ratings for 33K movies for 240K users. The "small" dataset contains 100K ratings for for 10K movies for 700 users. Use the "small" dataset for this part - http://grouplens.org/datasets/movielens/latest/ (10 points)
    c. If you attempt both (a) and (b) here, you also *have* to attempt part (c). Pick one of the following two options (5 points):

i. Now that you have support for at least two additional file formats, you want to now support any future file formats as well. Use (at least) one of the the design patterns discussed in class to make your program more flexible for supporting other file types. (If you attempt this part, parts (a) and (b) get weighted down to 7.5 points each.)

ii. As the datasets grow larger, your programs will take a longer time to run. For this part, make (at least) two performance improvements to your system so that your code runs faster. (If you attempt this part, you have to use the "full" dataset for part (b).)

2. Changing the recommendation engine

a. Cosine Similarity (10 points)

This model is somewhat different than the Pearson correlation, as it is a vector-space approach based on linear algebra rather than a statistical approach. Users are represented as |I|-dimensional vectors and similarity is measured by the cosine distance between two rating vectors. This can be computed efficiently by taking their dot product and dividing it by the product of their L2 (Euclidean) norms.

$$s(u,v) = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{\|\mathbf{r}_u\|_2 \|\mathbf{r}_v\|_2} = \frac{\sum_i r_{u,i} r_{v,i}}{\sqrt{\sum_i r_{u,i}^2} \sqrt{\sum_i r_{v,i}^2}}$$

Unknown ratings are considered to be 0; this causes them to effectively drop out of the numerator. If the user mean baseline is subtracted from the ratings prior to computing the similarity, cosine similarity is equivalent to Pearson correlation when the users have rated the same set of items.

b. Baseline Predictor (10 points)

We have a new employee in our company who says he's studied Recommender Systems in the past and recommends using a much simpler technique (compared to k-Nearest Neighbors and Pearson Correlation). His suggestion is a Baseline Predictor as described below:

$$b_{u,i} = \mu + b_u + b_i$$

$$b_u = \frac{1}{|I_u|} \sum_{i \in I_u} (r_{u,i} - \mu)$$

$$b_i = \frac{1}{|U_i|} \sum_{u \in U_i} (r_{u,i} - b_u - \mu)$$

where $b_{u,i}$ is the baseline prediction for user u and item i; b_u and b_i are the user and item baseline predictors, respectively; μ is the overall average rating.

c. If you attempt both (a) and (b) here, you also *have* to attempt part (c). There are now several different options for our Recommender System. How do we determine which approach works best? Design and implement a (small) experiment to find out which amongst the three (Baseline, kNN with Pearson Correlation, kNN with Cosine Similarity) works best. (5 points)

Extra Credit

You can attempt the remaining 20 points from above for EC. As always, you cannot have any help from the TAs/instructor for the EC.

Grading Criteria for Milestone 3

100% for functionality – Does the changes work as required? Are there bugs? …

Programming – General Comments

Here are some guidelines with respect to programming style.

Please use Javadoc-style comments.

For things like naming conventions, please see Appendix I (Page A-79) of the Horstmann book. You can also install the Checkstyle plugin (http://eclipse-cs.sourceforge.net/) in Eclipse, which will automatically warn you about style violations.

Submission Instructions

You should also submit a text file titled readme.txt. That is, write in plain English, instructions for using your software, explanations for how and why you chose to design your code the way you did. The

readme.txt file is also an opportunity for you to get partial credit when certain requirements of the assignment are not met. Think of the readme as a combination of instructions for the user and a chance for you to get partial credit.

Additionally, in your readme file, focusing on the changes you had to make, please explain how easy/hard the change was and what you could have done in the earlier milestones to make it easier for you.

Please create a folder called YOUR_PENNKEY. Places all your files inside this – the Java files, the readme.txt file. Zip up this folder. It will thus be called YOUR_PENNKEY.zip. So, e.g., my homework submission would be swapneel.zip. Please submit this zip file via canvas.