CS 324: Data Mining **Dave Musicant**
Recommendation Systems **Winter 2015**

For this assignment, you'll build two collaborative filtering systems for movie recommendation. The first will be user-based; the second will be item-based. Build both systems using the small version (100k) of the MovieLens data, so that your code can run reasonably quickly. One convenient thing about this dataset is that the MovieLens people have already separated the datasets into into training sets and test sets. The idea is that you use the ratings from the training sets to make predictions for the users and movies in the test sets. You can then compare the error from ratings that you generate for the test sets against the actual ratings provided.

Datasets

Use the `ua.base` and `ua.test` files for your recommender system; you should produce recommendations for all of the user-movie pairs in `ua.test`, and compare your recommendations against the ones given. The README file explains the layouts of the files.

Specific details

You should focus on section 9.3. Use cosine distance to measure the distance between users or items on the values that they actually share in common that are not missing. Normalize user ratings by subtracting out each user's average rating. (I tried doing it without, and my results were considerably worse.)

What you should turn in

Turn in two programs: one determines movie ratings via a user-based algorithm, and the other does so via an item-based algorithm. You'll have lots of shared code, so you will likely have other files that contain code shared by both. Your programs should print out the RMSE error between the ratings you produce and the actual ratings given in the test file.

For both methods, you should also print out the 10 movies for which you most closely estimated the ratings, on average, and the 10 movies for which you had the most trouble. Do you have something like the Napoleon Dynamite problem going on? Submit a short writeup describing what you find.

As you write your code, you will have to play with a number of parameters. Some that I can think of are:

- the number of movies that two users share in common in order to consider them potentially nearest neighbors; if they share too few movies in common, you may want to exclude them
- the number of ratings you needed to give a movie before you consider it to be "best estimate" movie or a "worst estimate" movie. If you only guessed the rating for a particular movie once, and did really well at it, it's probably a poor candidate as a movie that you can estimate really well

- the number of nearest neighbors that you want to use in either the user or item based similarity metrics
- others?

Enjoy! I had a lot of fun with this, and I was really curious to see which movies my program could predict well, and which ones it had a lot of trouble with.