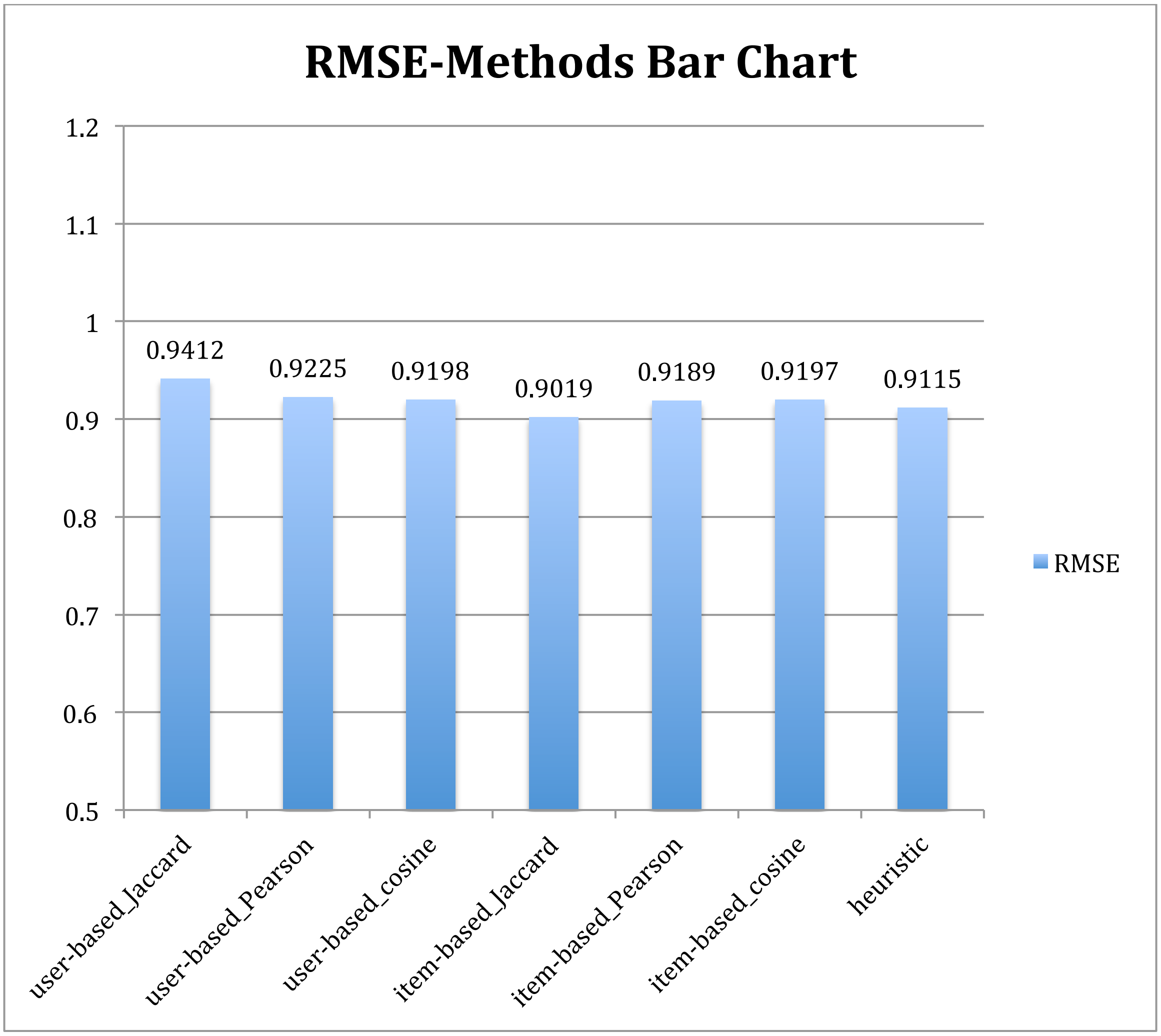
A. RMSE for each method



Observations: Item-based methods all outperform user-based methods. Maybe this is because there are more than 6 thousand users and more than 3 thousand movies, so for two movies, there are more information they can extract.

My heuristic methods that combine user-information, movie-genre, and user ratings outperform all three user-based approaches. This may be because we combine more information.

Surprisingly, Jaccard is the best among the three when I use item-based CF. Intuitively, Jaccard may be bad when there are lots of information. So I think when more data are availability, cosine or Pearson will give better result.

B. Briefly explain your result1/result2. Explain how you get the result3 using what methods and which type of similarity and what information. Explain why using these settings will give the best result.

Result1 is accomplished by user-based collaborative filtering, and for the specific test case, cosine measurement is the best. Generally, using Pearson will give better result. But as the training set is not that large (only 3000*6000), and the RMSE does not diff much, I think the result is quite satisfactory. By user-based collaborative filtering, first I compare the two users' similarity based on one of three measurements. Then given a (toRateUser, toRateMovie), I first try to figure out those users that are similar to toRateUser, and have already rated toRateMovie. Then I pick a subset of them as neighbors of toRateUser. By taking into consideration average rating by toRateUser, and similar users' rating, I derive the rating for (toRateUser, toRateMovie).

Result2 is accomplished by item-based collaborative filtering, and for the specific test case, Jaccard measurement is the best. And actually, three measurements are all quite good. This is probably because the number of users is larger than then number of movies. By item-based collaborative filtering, first I compare the two movies' similarity based on one of three measurements. Then given a (toRateUser, toRateMovie), I figure out the movies that are similar to toRateMovie, and have been rated by toRateUser. Then I pickup a subset of those movies, and combine their rating and the average rating of toRateMovie.

I build a user profile for each user using user information and movie information.
The movies file is separated considered according to their genres. Their genres are translated into integers in the system. Each movie corresponds to a set of integers of genres except that some movies are missing genres. Movies of each genre are regarded separately. That is, in the original user-based approach, an average rating is computed to each user. And now, several average ratings, each corresponds to a genre, is computed. These average ratings are saved in each user's profile.
Besides, each user profile contains gender, age and occupation.
I make use of user profile similarity and pure user-based approach to calculate similarity between two users. If they are similar in both user profiles and rating history, then they have a higher possibility to rate other files.
The similarity calculation between two user profiles is based on observation and intuition. I discuss with a psychology student to give weights to each attribute.
Specifically, initially two user profiles have similarity (called amplification) 1.
If they are same in gender, then amplification *= 5
If they are same in occupation, then amplification *= 1.1

More difference in ages, more likely two users tend to prefer different movies,
so amplification *= (5 - 0.05 * abs(age1-age2))
Besides, average ratings for each genre of movies are also taken into consideration:
amplification *= 1 + (0.5 - diffInEachGenre)
The result amplification can be multiplied to other similarity results.
This method can give slightly better result than pure user-based approach because it not only considers history ratings but also takes into consideration user and item attributes. I think with more data, this approach can be better than best item-based approach as well.

C. How do you do the cross validation?

I used 10-fold cross validation, as 10 is a recommended number. When read ratings file, for each rating, I assign a random number from 0 to 9. This random value designates which partition the rating belongs to. Then it generates some temporary files: training_set_{i}, test_set_{i}, ground_truth_{i} (i is from 0 to 9). Then I will compute RMSE ten times and get an average number. Each iteration, I use file training_set_{i}, which contains partition 0 to 9 except i, and test_set_{i}, which contains partition i. I use the recommender system I wrote to make prediction for test_set_{i}, and get test_results_{i}. Then I compare test_results_{i} with ground_truth_{i}, and get one iteration's RMSE. The average of ten rounds' RMSE is the final result.