Hongsen He
w1079638
hhe@scu.edu

COEN 272 Project 2
Results Discussion

Please provide the following information
1. Compare the accuracy of the various algorithms. Do you think your results are reasonable? How can you justify the results by analyzing the advantages and disadvantages of the algorithms?
2. How long does each algorithm take to complete the prediction? Discuss the efficiency of the algorithms.

Summary:
I implemented cosine similarity and Pearson correlation and item-based with adjusted cosine. Also I made improvement with IUF and case amplification.
My own algorithm is 0.6 * cosine + 0.4 * Pearson

Environment:
Programming language is Python 3.4
IDE is PyCharm CE
Mac OS X

Log:
In the beginning I tried k = 80 and no case amplification, no IUF, the result 5 is 0.845.
Then I tried k = 80 and with case amplification p = 2.5 but no IUF. The result5 is better: 0.843566337376516, 0.815333333333333, and 0.818558888781711.
Overall is 0.825972746675423. It is better than before.

Then I tried p = 1.5 but no IUF and k = 100 with test 5. The result 5 is 0.830561460547705. And with IUF, p = 1.5 and k = 100. The result 5 is better: 0.830436413655121.
So I can get the result: either with IUF or with case amplification is better without.
And in case amplification p = 1.5 is better than p = 2.5.
Above all was testing with cosine similarity algorithm

	test5	test10	test20	Overall
cosine	0.822	0.782	0.779	0.7813
Pearson-IUF	0.823	0.771	0.746	0.778
Pearson-case	0.833	0.785	0.770	0.804
Item-based	0.884	0.821	0.782	0.835
Combine	0.771	0.737	0.738	0.7489

So when I ran Pearson I found that cosine is a little better than Pearson. Also IUF and case amplification and Dirichlet smoothing will improve on these algorithms.

Also I don't think item based is better because the training matrix is sparse matrix. It means it is difficult to find similar movies in the matrix.

And I tried several K value, as when I printed out cosine similarity I realized at least top 50 was 1. So I want my K value bigger (more relative neighborhoods) From 80 to 120, even total is 200, and finally I make K to 100.
And 0.6 * cosine + 0.4 * Pearson is my own way. As I found this is the best one so far.

Finally let's talk about the efficiency.
Basically my program will run about 20 minutes. That is too long I know.
And Pearson is longer than Cosine. If I combine them together the running time will be about one hour.
I think the reason is: I made a big matrix 200*1000 for training matrix and it is a sparse matrix. For searching it is too slow. Another reason is I made 3 functions: def prediction_movie_list(), def given_list_in_test(usr_id), def avg_usr_rating_in_test(usr_id). And for prediction rates there will be about 8000 lines. Each line will use these 3 functions. So I have to read 3 * 8000 = 24,000 times IO. And at the beginning I loaded a matrix, so more than 24,000 IO kill the efficiency.
In addition, for these three algorithms, the time complexity is O (N^2), even I combine two algorithms, it is just addition. So overall the time complexity is O (N^2).
Unfortunately I found this IO mistake at the last minute, so maybe I can improve it later. This is my first Python program. I enjoy it!
Source code is attached with rich comments.