COEN 169: Web Information Management
Project 2

Caroline Liongosari

Results:

	Cosine Similarity	Pearson Correlation	Pearson w/ IUF	Pearson w/ Case Amp	Pearson w/ IUF and Case Amp	Item based Collaborative Filtering	Custom Algorithm
MAE of Given 5	0.823933975240715	0.81530573965237	0.822558459422283	0.818556958859572	0.831686882580968	0.902213329998749	0.793422533450044
MAE of Given 10	0.789166666666667	0.758833333333333	0.7695	0.768333333333333	0.778	0.805666666666667	0.747333333333333
MAE of Given 20	0.769364329121254	0.751519243754228	0.782772258126748	0.770425388251182	0.805729719301630	0.796566026815858	0.738690074274139
Overall MAE	**0.792152355934986**	**0.774257100640289**	**0.792562797570186**	**0.785708422262354**	**0.807420784764406**	**0.833483828599573**	**0.758783450993269**

Discussion:

Overall, I believe my results were reasonable since Professor Fang said all the algorithms should have comparable mean average errors of around 0.8. In only 2 of 7 cases shown in the table had overall MAE results of 0.8 or higher. I wasn't surprised that the MAE for Pearson correlation was lower than in cosine similarity since Pearson correlation takes into account cases such that if a user rated completely the opposite as another user, it still can use that information while cosine similarity wouldn't.

One interesting note I figured out was that in cases where a user had rated all the movies they had rated the same, Pearson correlation would fail since it would have resulted a division by 0, so in cases where this happens, I just had my function return a Pearson correlation of 0.5. This definitely helped the algorithm's performance since when I originally had it return 0, I had an MAE of about 0.835, while not bad, it's certainly worse than 0.774.

Another thing to note is at first I decided to implement prediction using cosine similarity with all possible neighbors since I believed that it's better to calculate the weights with as many neighbors as possible and I got an MAE of ~0.79 as listed in the table. I also used this approach for the algorithms. However, since it was mentioned in class to use k-nearest neighbors, I tried implementing cosine in this manner with a variety of values of k: k= 10 (MAE: ~0.97), k=50 (MAE: ~0.92) and k=100 (MAE: ~0.82) but it became tedious to test more values of k to find the optimal number of k especially when I only can submit my result files 30 times to get MAE results, so I kept my original method of using all possible neighbors, which seemed to work well as the table above shows.

I was surprised to see that IUF and case amplification seemed to worsen the original Pearson predictions, especially when they're combined together. I was surprised since these modifications were supposed to improve the Pearson predictions. IUF is supposed to take into account the overall popularity of a movie and case amplification is supposed to emphasize high weights while punishing the lower weights. It may be possible that this is because of the small training dataset. If given the chance, I would like to implement these algorithms with a much larger dataset to see if there's differences in results. IUF depends on the number of users (m=200) and number of users ratings those movies (which ranged from 0 to 102 with 500 of the 1000 movies having 10 or less ratings), these small numbers may have led to skewed results. This also applies with case amplification since weights tend to be really close values to each other. What I also found strange was that the MAE became worse when given 20 ratings for each tested user in these cases, I'm not really sure why this happened since for the rest of the algorithms, the MAE decreased as more data is given for each user, which is what is supposed to happen.

I was also surprised to see item-based collaborative filtering doing worse than user-based collaborative filtering since item similarity is supposed to be more stable than user similarity. I think this happened because I was using all possible neighbors, which may have been too many. I probably should have tried some smaller values of k. Perhaps a larger training dataset would help?

For my custom-made algorithm, I was inspired by the fact that the winning team for the Netflix prize combined multiple algorithms to get their predicted results, so I decided to do something similar by combining the predicted ratings of the algorithms I had already implemented. I tried multiple combinations, but the best turned out to be using the average of the predicted ratings from my simple cosine, Pearson, and item-base collaborative filtering algorithms which got me an overall MAE of ~0.756, which was a lot better than I had expected.

Another note: I also tried implementing what happens if I just used the users' average ratings as their predicted ratings and ended up getting an MAE of ~0.82, which was surprisingly good. Though I bet once a larger dataset is used, this result would become a lot worse.

**Last minute note: I just realized that I had implemented Pearson correlation incorrectly this entire time. In my simple_cos_similarity function (which I implemented as a helper function for Pearson correlation), I had the function return 0 if the calculated similarity was less than 0. I realized this shouldn't be the case since the range for pearson correlation is [-1,1]. not [0,1] like cosine similarity. I ran my Pearson correlation function again with this fix and got an overall MAE of ~0.790 which is worse than my result in the table with the error, which was strange to me. I am unfortunately almost out of tries to submit files to the online system so I wouldn't be able to test all of my other algorithms like IUF, Case Amplification, the combination of both, and my custom algorithm which uses this function so my results in the table contain that error. But, I still really find it bizarre that my error ended up improving the results - maybe it's something I'll look into more in the future. This error might also explain the worse results for IUF, Case amplification, and the combination for both.