David Wiszowaty
CS 547, Spring 2016; Project 2 Evaluation

Memory-Based Collaborative Filtering Algorithm: The first thing that is done is the mean of all the ratings for each user is computed. Once the means have been computed, the system would take in a user and a movie, and then start to compute the prediction for the given input. The first thing that is done is computing vector similarity between the given user and all other users in the dataset. The denominator will use items that a user has voted for, and this is independent from the other user. Once the similarities have been computed, the prediction can be computed. The predicted rating would then be returned.

Model-Based Collaborative Filtering Algorithm: The first thing that is done is to precompute the mean for all the ratings of an item, and this would be done for each item. Once this is complete, the means and ratings would be used to compute correlation-based similarity between all pairs of items. Instead of storing all pairs, each item would store the k highest rated items, this is done to save on space.
Once this has been computed, then the user can input a movie and a user. Given the input, the system would simply have to compute the weighted sum and return the prediction.

Memory-Based Collaborative Filtering Algorithm Extension: The third algorithm I decided to use is an extension of the first one. The extension is implementing inverse user frequency. This works similar to inverse document frequency, where an additional weight is attached to the value, the weight depends on the popularity of the value or category. Inverse user frequency would do the same by attaching a weight to each vote. The weight is determined by the number of users that voted for the same movie. If more users voted for the same movie then its a very common movie and therefor less weight should be given.

The algorithm works almost the same at the standard memory-base algorithm. The extended algorithm has to perform additional steps. The first thing that is done is to compute the frequency of each item. This is done during the pre-processing stage since its not reliant on any specific users.

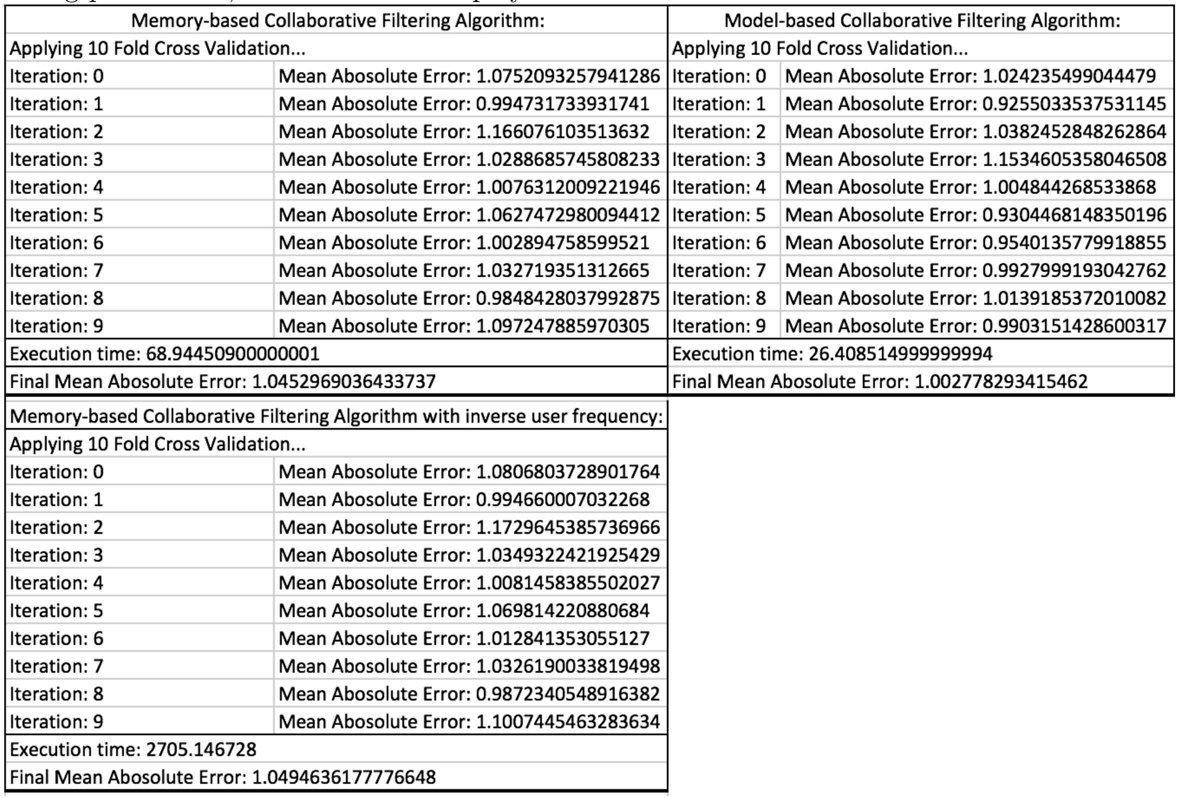
The algorithm also uses vector similarity to compute the similarity scores among the users. However the algorithm has been modified to take advantage of the inverse user frequency values. Every rating used in the formula will be multiplied by the frequency. Once the similarity scores have been executed, the prediction values would be calculated the same way as the original memory based algorithm.

The goal for this algorithm is that using the frequencies, the accuracy would improve because there is more focus on unique movies that users watched.

Evaluation: To evaluate all three of the algorithms I used K Fold Cross Validation on each algorithm to predict ratings. Using this method, the data is split into two parts, one is for training and the other for testing. I would go through each user in the testing data set and predict movie ratings for movies that the user has already voted for. Movies with no ratings were not used since there is no way of determining what the actual value is.

The number of folds I used for cross validation is 10. I took the test data set and applied all three of the algorithms. To compare the accuracy of all three algorithms I computed the Mean Absolute Error (MAE). The MAE value is used to compare the predicted values with the actual values, the lower the value the better the accuracy of the algorithm. I also kept track of the execution time to determine the execution time of each algorithm. To compare all three algorithms, both the execution time and the MAE value will be examined.

After running cross validation on all three algorithms and computing rating predictions, the results are displayed below:



Memory-based Collaborative Filtering Algorithm:	
Applying 10 Fold Cross Validation...	
Iteration: 0	Mean Abosolute Error: 1.0752093257941286
Iteration: 1	Mean Abosolute Error: 0.994731733931741
Iteration: 2	Mean Abosolute Error: 1.166076103513632
Iteration: 3	Mean Abosolute Error: 1.0288685745808233
Iteration: 4	Mean Abosolute Error: 1.0076312009221946
Iteration: 5	Mean Abosolute Error: 1.0627472980094412
Iteration: 6	Mean Abosolute Error: 1.002894758599521
Iteration: 7	Mean Abosolute Error: 1.032719351312665
Iteration: 8	Mean Abosolute Error: 0.9848428037992875
Iteration: 9	Mean Abosolute Error: 1.097247885970305
Execution time: 68.94450900000001	
Final Mean Abosolute Error: 1.0452969036433737	

Model-based Collaborative Filtering Algorithm:	
Applying 10 Fold Cross Validation...	
Iteration: 0	Mean Abosolute Error: 1.024235499044479
Iteration: 1	Mean Abosolute Error: 0.9255033537531145
Iteration: 2	Mean Abosolute Error: 1.0382452848262864
Iteration: 3	Mean Abosolute Error: 1.1534605358046508
Iteration: 4	Mean Abosolute Error: 1.004844268533868
Iteration: 5	Mean Abosolute Error: 0.9304468148350196
Iteration: 6	Mean Abosolute Error: 0.9540135779918855
Iteration: 7	Mean Abosolute Error: 0.9927999193042762
Iteration: 8	Mean Abosolute Error: 1.0139185372010082
Iteration: 9	Mean Abosolute Error: 0.9903151428600317
Execution time: 26.408514999999994	
Final Mean Abosolute Error: 1.002778293415462	

Memory-based Collaborative Filtering Algorithm with inverse user frequency:	
Applying 10 Fold Cross Validation...	
Iteration: 0	Mean Abosolute Error: 1.0806803728901764
Iteration: 1	Mean Abosolute Error: 0.994660007032268
Iteration: 2	Mean Abosolute Error: 1.1729645385736966
Iteration: 3	Mean Abosolute Error: 1.0349322421925429
Iteration: 4	Mean Abosolute Error: 1.0081458385502027
Iteration: 5	Mean Abosolute Error: 1.069814220880684
Iteration: 6	Mean Abosolute Error: 1.012841353055127
Iteration: 7	Mean Abosolute Error: 1.0326190033819498
Iteration: 8	Mean Abosolute Error: 0.9872340548916382
Iteration: 9	Mean Abosolute Error: 1.1007445463283634
Execution time: 2705.146728	
Final Mean Abosolute Error: 1.0494636177776648	

All three algorithms were executed using the same test and training datasets.

At each iteration of the K fold validation, the MAE value is recorded, and in the end the final MAE value is computed by taking the mean of all MAE's values. Based on the results, the model-based algorithm scored the lowest MAE value and the fasted execution run. This means that the model based algorithm is the most accurate among the other algorithms.

One thing that was noticed is that the extended memory-based algorithm was the least accurate out of all the algorithms and the longest running. The difference between the error value from the original memory-based algorithm and the modified algorithm is very small. The difference in values can potentially be affected by the input dataset. The number of differences between popular movies and less popular movies can potentially have an effect on the frequency value which would then affect the prediction values.

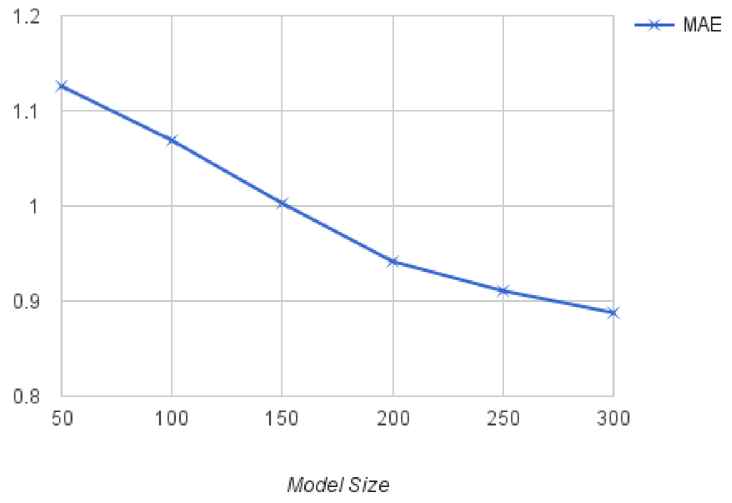
I believe these values can give a user a pretty good estimate of what a certain user would vote for a certain movie. The better algorithm to use would be to use the model based algorithm since it provides more accurate results and at a faster rate. Depending on the parameters which are set, and the data set, then the results from the model based algorithm are pretty reasonable.

In terms of efficiency, the model based algorithm is best. The reason for this is that most of the time is spend pre-computing data. The similarities between movies is precomputed which would decrease the amount of time spent computing the predictions. If we compare this to the memory based algorithm, the memory-based algorithm only precomputes the user means. For each user, the similarities have to be computed among all users and then the prediction can be computed. Both algorithms have there advantages and disadvantages. The advantage of the memory-based algorithm is that it can quickly compute a rating for a specific movie, for a specific person. However this is dependent on the number of users in the dataset. The more users, the more computations. This is also a disadvantage. If the number of users is high, or the end-user is interested in computing predictions for a large amount of users then the memory-based algorithm can be slow. The model-based algorithm would be more preferred in this situation. A disadvantage of the model-based algorithm is if the number of movies is large. This would increase the amount of time needed to compute the correlation-based similarity among the items, and increase the amount of memory required to store the similarities. To take full advantage of the algorithm, the model size would be the same size as the number of items. However this would result in a space complexity of $O(n^2)$, where n is the number of items.

The run time for the memory-based extended algorithm is quite large. The reason for the large run time is applying the inverse user frequency values to the vector similarity algorithm. The original algorithm only required the ratings between two users. The product of both ratings would be taken and then all the ratings would be added together. This required little computations, and could efficiently be done using python's *numpy*. With the addition of frequency values, more computations is required for the vector similarity. This is manually done, which causes the algorithm to run much slower. This is the disadvantage of the algorithm. Another disadvantage is if there wasn't a lot of diversity among all the user votes on the movies. This would cause the frequency values to all be similar which would minimize the impact towards the prediction.

The accuracy of the algorithms can be improved. According to the paper given titled, "Item-Based Collaborative Filtering Recommendation Algorithms", there are several ways to improve the accuracy of the results. One of the ways is the ratio between the test and training data. The more training data that is provided, the better the accuracy. The recommended ratio between test and training is 0.8.

Another way to improve the accuracy is to change the model size. This would only have an effect on the model-based algorithm. The model size stores the number of the most similar movies for each movie. The more values stored, the better the prediction. In the graph below, it represents MAE values of different model sizes. When I set the model size to be 300, the MAE value was 0.8877, which is an improvement over 1.0027.



When I set the model size to be 300, the MAE value was 0.8877, which is an improvement over 1.0027. In the original test, the model size was set to 150. The reason why the model size can't be the large is because of the space complexity. The larger the model, the more memory is required, and it would take a little longer to compute the prediction.
The reason why I choose 150 is that it was a nice balance between the amount of memory required and the MAE value.
Based on the analysis, the model-based collaborative filtering algorithm performs more accurately and faster when compared to the other two algorithms.