Kyle Kovacik

Part 1: Collaborative Filtering

1. My first model in ./MODELPART1 uses Memory-Based Collaborative Filtering in order to predict ratings and generate recommendations. This uses the vector space similarity approach. It builds rating vectors using the ratings of all users in the train file and calculates the Cosine Simularity of them all. Then, to predict the rating, a user weights similar users ratings of the dish based on the similarity value to get the result. Task 2 just predicts all unrated dishes, and sorts the list to provide the recommendations.

2. MAE: 0.526984 on user_ratings_test.json

3. Precision@10: 1.0
   Precision@20: 1.0
   Recall@10: 0.479093
   Recall@20: 0.958185

4. Based on the results above, my model one is relatively effective at predicting the correct rating of a dish using similar users. The MAE being 0.526894 is decent for ratings from 1-5. It is a little off, but it will never be a while rating off. The Precision values are very good for this test set. All of the relevant files are retrieved. Though, the test data might represent too small a size to show a more accurate precision. The recall is very good also. Since we retrieve all the relevant files so early, the recall only goes up as the number returned increases.

Part 2: Build your own model

1. My model in ./MODELPART2 uses a model-based clustering approach to predict ratings. It uses Gaussian Mixture Model soft clustering in order to find the probability a dish belongs in each cluster. The Gaussian Mixture Model generates random gaussians for each cluster and through enough iterations or convergence, estimates the best mean and covariance for the amount of clusters. Then, I use part of what was done in Part 1 to predict the ratings. I use a memory based, vector space similarity model, in order to determine the cosine similarity between the dish to predict and all of the already predicted dishes the user has rated. They are compared using a vector of the probabilities of the dish being in each cluster. Therefore, dishes that are in a similar cluster to the dish to predict will be weighted higher in the prediction. Similarly to above, Task 2 just predicts all unrated dishes, and sorts the list to provide the recommendations.

2. MAE: 0.671997

3. Precision@10: 0.712871
   Precision@20: 0.712871
   Recall@10: 0.333780

Recall@20: 0.0.667560

4. Based on the results above, this model 2 is not amazingly effective at predicting ratings and predictions, but it does alright. First of all, the MAE is 0.671997. For the rating scale of 1-5, this is a pretty decent value. It is off by a fair margin, but still relatively close overall. The recall and precision is much worse. The precision for both 10 and 20 is 0.712871. This is not very good, and for this dataset, is disappointing it stays so low at 20 results. The recall is alright at 0.333780 and 0.667569, but not great values. Overall, the values are not absolutely terrible, but not great.

Part 3: Comparison

	Model 1	**Model 2**
MAE	0.526984	0.671997
Precision@10	1.0	0.712871
Precision@20	1.0	0.712871
Recall@10	0.479093	0.333780
Recall@20	0.958185	0.667560
Time	20 seconds	3 minutes

1.

2. The above table shows the different MAE, Precision, Recall, and approximate time it took for the models to complete. It has a smaller MAE, higher precision, higher recall, and significantly smaller time. In this case, the memory approach was significantly better than the clustering approach.

3. For Task 1, the model 1 approach is better than model 2 by the drop in MAE. Using the similar users ratings seemed to generate a better estimate of the correct values. Since the first model does not need to parse over 320 ingredients, the speed is significantly better too for data with a lot of features. For Task 2, the model 1 approach was also better the model 2 by the increased precision and recall. Since the predictions were better when using similar users so were the recommendations. Having a smaller user base also can shrink computation time.

4. For Task 1, the model 2 approach, while not better above, would perform better predictions with users that performed many ratings and dishes that had better defined clusters. While the above MAE was lacking, it is relatively close to model 1, and could overtake it given enough data although slower. For Task 2, the model 2 approach, while similarly not better, would still perform better given clusters that were more specific and users with high ratings. Because it does not use similar users like 1, the model is very reliant on the ratings of the user it is trying to predict. Therefore, resulting in not all relevant files being found. Though, with smaller iterations and ingredients, the speed could be faster than model 1.

5. When adding a new recipe to the dataset, little is changed in both models. Since model 1 is based on users and their dish ratings. Adding a new dish, changes little in the overall dish values and computation time. Though, when trying to predict that new dish for a user, since no other user has rated it, the prediction would simply guess the average rating of that user. As for model 2, the computation time increases a non-negligent amount. Since cluster processing is based on ingredients and dishes, a new dish is factored into each iteration of the gaussian model. Since 1 dish of 1000 does not affect the clusters very much, when trying to predict the dish for a user, the result is relatively accurate because the ingredients are relevant in other models. The clusters are able to capture the new dish whereas user similarity is unable to.

Note: How to Run Models

Make sure pandas is updated enough:

pip3 install –upgrade pandas

Run chmod +x MODELPART1

Run chmod +x MODELPART2

Then just:

./MODELPART1

./MODELPART2

Model 1 takes about 1 minute to finish on mc18

Model 2 takes about 5 minutes to finish on mc18