Long term project

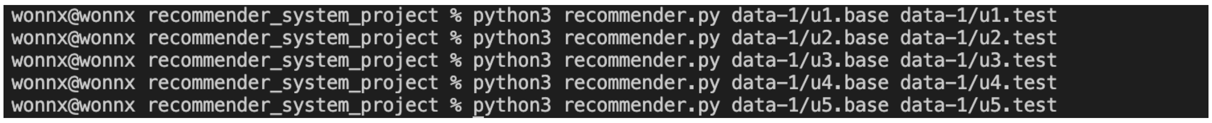
컴퓨터소프트웨어학과 2017029470 김종원

1. Environment

- Mac OS (Monterey) M1 chip
- Python 3.10.3 (release March 16, 2022)

2. How to compile

Before compiling dt.py, python version 3 must be installed in your system.



```
wonnx@wonnx recommender_system_project % python3 recommender.py data-1/u1.base data-1/u1.test
wonnx@wonnx recommender_system_project % python3 recommender.py data-1/u2.base data-1/u2.test
wonnx@wonnx recommender_system_project % python3 recommender.py data-1/u3.base data-1/u3.test
wonnx@wonnx recommender_system_project % python3 recommender.py data-1/u4.base data-1/u4.test
wonnx@wonnx recommender_system_project % python3 recommender.py data-1/u5.base data-1/u5.test
```

- Execution file name: recommender.py
- Training data file name: u#.test.txt
- Test data file name: u#.test.txt
- Output file name: u#.base_prediction.txt

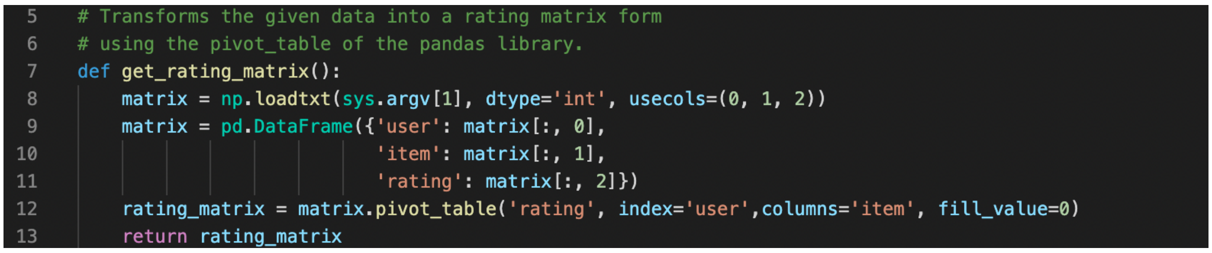
3. Summary of algorithm

I used the collaborate filtering method to predict the ratings of movies in test data. Collaborative filtering refers to a technology that predicts itself based on taste information collected from many users. Based on the user-user similarity neighbors that most similar to the corresponding user were extracted, and ratings were predicted based on the rates of the extracted neighbors.

- Using the pandas library, the ratings of movies left by users are transformed into a rating matrix.
- Based on the rating matrix, the Pearson correlation coefficient between user-user is obtained and converted into a similarity matrix.
- After sorting the similarity matrix, the neighbors corresponding to the top 5% of all users are obtained, and then the rating is predicted through a weighted sum.

4. Detailed description of codes

A. get_rating_matrix

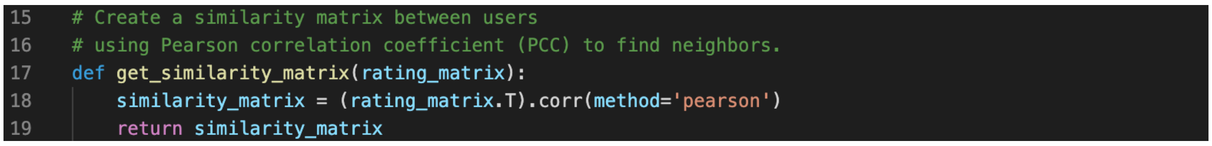


```python
# Transforms the given data into a rating matrix form
# using the pivot_table of the pandas library.
def get_rating_matrix():
    matrix = np.loadtxt(sys.argv[1], dtype='int', usecols=(0, 1, 2))
    matrix = pd.DataFrame({'user': matrix[:, 0],
                           'item': matrix[:, 1],
                           'rating': matrix[:, 2]})
    rating_matrix = matrix.pivot_table('rating', index='user',columns='item', fill_value=0)
    return rating_matrix
```

This is a function that transforms the given training data into a rating matrix. In the training data, the column corresponding to the time stamp was not used.

- Convert training data into dataFrame format of pandas library.
- Convert the dataFrame named **matrix** into the pivot_table type **rating_matrix**.

B. get_similarity_matrix

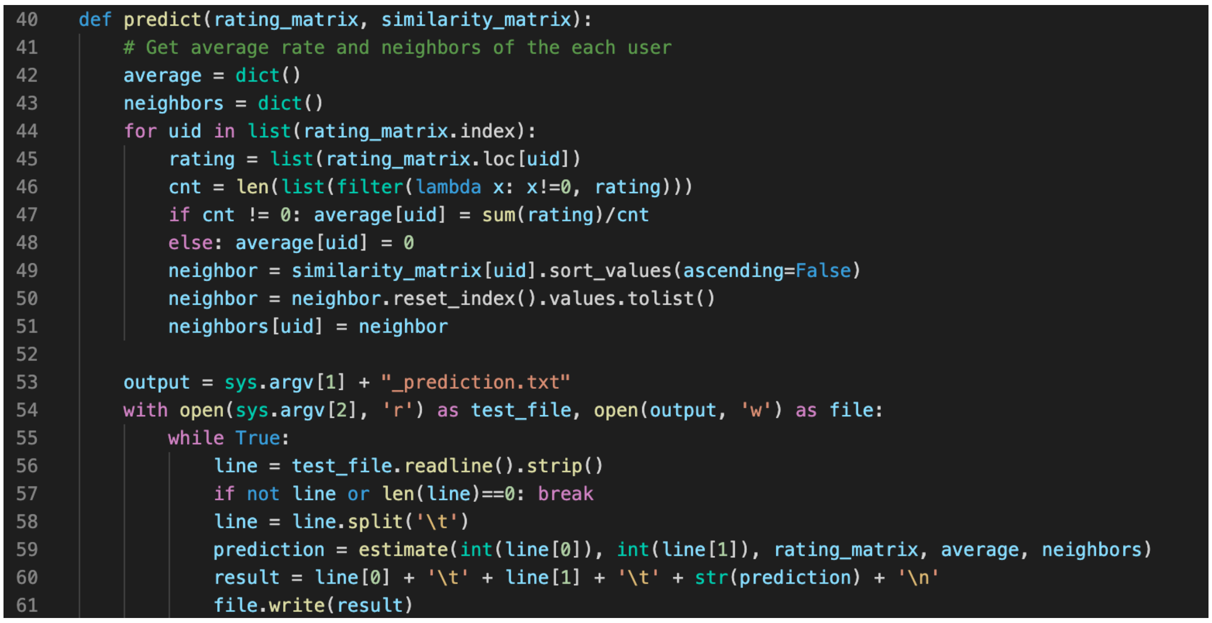


```python
# Create a similarity matrix between users
# using Pearson correlation coefficient (PCC) to find neighbors.
def get_similarity_matrix(rating_matrix):
    similarity_matrix = (rating_matrix.T).corr(method='pearson')
    return similarity_matrix
```

Based on the previously obtained rating matrix, it is a function that calculates the similarity between users and users though the Pearson correlation coefficient and stored as a similarity matrix.

- Using the built-in correlation method of pandas dataFrame, calculate similarity between users.

C. predict



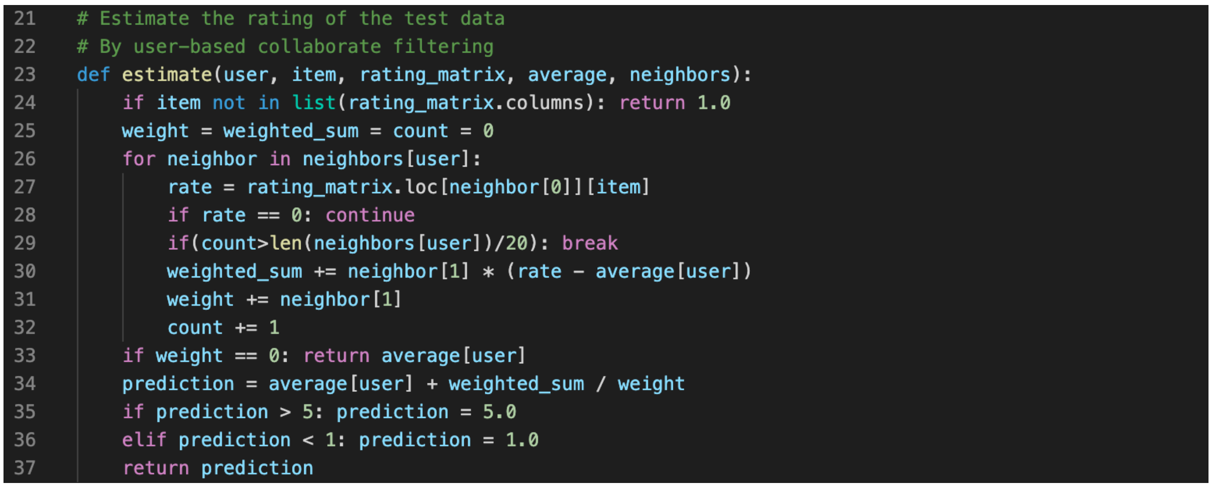
```python
def predict(rating_matrix, similarity_matrix):
    # Get average rate and neighbors of the each user
    average = dict()
    neighbors = dict()
    for uid in list(rating_matrix.index):
        rating = list(rating_matrix.loc[uid])
        cnt = len(list(filter(lambda x: x!=0, rating)))
        if cnt != 0: average[uid] = sum(rating)/cnt
        else: average[uid] = 0
        neighbor = similarity_matrix[uid].sort_values(ascending=False)
        neighbor = neighbor.reset_index().values.tolist()
        neighbors[uid] = neighbor

    output = sys.argv[1] + "_prediction.txt"
    with open(sys.argv[2], 'r') as test_file, open(output, 'w') as file:
        while True:
            line = test_file.readline().strip()
            if not line or len(line)==0: break
            line = line.split('\t')
            prediction = estimate(int(line[0]), int(line[1]), rating_matrix, average, neighbors)
            result = line[0] + '\t' + line[1] + '\t' + str(prediction) + '\n'
            file.write(result)
```

Average ratings of movies evaluated by each user and neighbors of each user are obtained. Using the average, neighbor, and rating matrix, we predict the user's rating for a movie that has not yet been rated. In addition, the output file name is set differently according to the input file, and the predicted rating is exported according to the format.

- **Average** and **neighbors** are in the form of dictionary, and each key contains average rating of the movies for the user and PCC for all users.

D. estimate

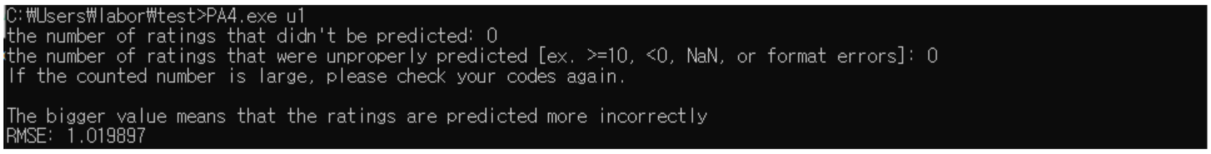


```python
# Estimate the rating of the test data
# By user-based collaborate filtering
def estimate(user, item, rating_matrix, average, neighbors):
    if item not in list(rating_matrix.columns): return 1.0
    weight = weighted_sum = count = 0
    for neighbor in neighbors[user]:
        rate = rating_matrix.loc[neighbor[0]][item]
        if rate == 0: continue
        if(count>len(neighbors[user])/20): break
        weighted_sum += neighbor[1] * (rate - average[user])
        weight += neighbor[1]
        count += 1
    if weight == 0: return average[user]
    prediction = average[user] + weighted_sum / weight
    if prediction > 5: prediction = 5.0
    elif prediction < 1: prediction = 1.0
    return prediction
```

5% of all users are set as neighbors, and if the neighbor has previously rated the movie, it is multiplied by similarity and weighted summed.

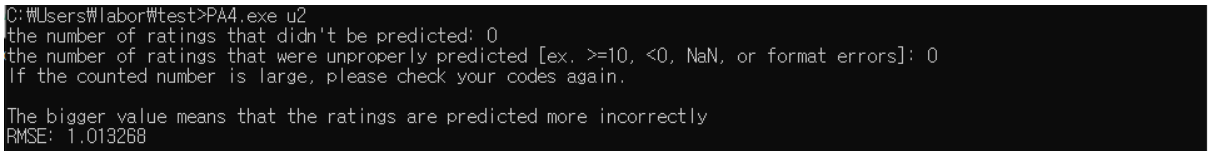
- If the movie does not exist in the existing rating matrix, the rating is predicted using the average rating value of the user.
- If the prediction result is less than 1, it is corrected to 1, and if it is greater than 5, corrected to 5.

5. Testing result



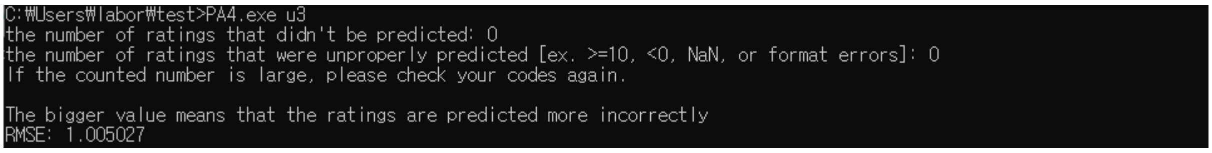
```
C:₩Users₩labor₩test>PA4.exe u1
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.019897
```



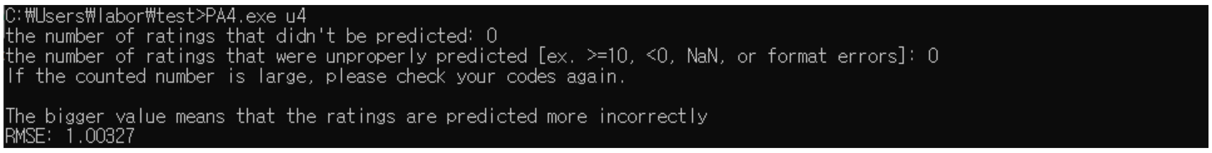
```
C:₩Users₩labor₩test>PA4.exe u2
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.013268
```



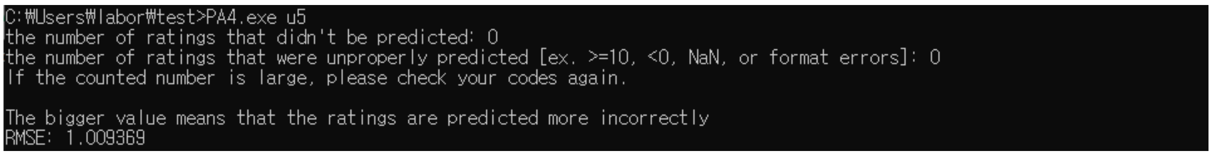
```
C:₩Users₩labor₩test>PA4.exe u3
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.005027
```



```
C:₩Users₩labor₩test>PA4.exe u4
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.00327
```



```
C:₩Users₩labor₩test>PA4.exe u5
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.009369
```

For testing, I used a measure called RMSE (Root Mean Square Error). The bigger value means that the rating are predicted more incorrectly. All five testing results came out close to 1, indicating that the user actually evaluated the movie and the predicted value differed by about 1.