# Programming Assignment #4

## Index



**학번: 2017041930**

**이름: 김선동**

**1. Environment & Requirements**

OS는 Windows, language는 python을 사용하였다. Editor는 visual studio code를 사용하였다. 사용한 python의 버전은 3.10.4이다.
별도의 패키지를 사용하였는데 requirements.txt를 첨부하였으니 아래 명령어로 설치하면 된다.

> pip3 install -r requirements.txt

**2. How to run?**

> ./recommender.py u1.base u1.test
> ./recommender.py u2.base u2.test
> ./recommender.py u3.base u3.test
> ./recommender.py u4.base u4.test
> ./recommender.py u5.base u5.test

위 명령어를 실행하면 u#.base를 기반으로 예측을 해서 u#.test의 데이터에 맞게 rating을 예측하여 u#.base_prediction.txt에 저장해준다.

### 3. Summary of my implementation

```
if __name__ == '__main__':
    if len(sys.argv) != 3:
        print("Execute the program with two arguments: train file name, test file name")
        print("Train file name = 'u#.base, test file name = 'u#.test'")

    train_file_name = sys.argv[1]
    test_file_name = sys.argv[2]
    output_file_name = sys.argv[1][0:2] + ".base_prediction.txt"

    # Dataframe로 표현
    train_set = pd.read_csv(train_file_name, sep="\t", names=["user_id", "item_id", "rating", "time_stamp"])
    test_set = pd.read_csv(test_file_name, sep="\t", names=["user_id", "item_id", "rating", "time_stamp"])

    # Time stamp는 drop
    train_set = train_set.drop(columns="time_stamp")
    test_set = test_set.drop(columns="time_stamp")

    # 유저와 아이템의 수 파악
    num_of_users = max(train_set.iloc[:, 0])
    num_of_items = max(train_set.iloc[:, 1])

    # rating matrix 만들기 - Df사용하면 없는 정보 누락할 수 있음
    rating_matrix = np.full((num_of_users, num_of_items), -1)
    for user_id, item_id, rating in train_set.values.tolist():
        rating_matrix[user_id-1][item_id-1] = rating
```

Input 파일을 읽어서 dataframe으로 바꿔준다. 이 때, time_stamp는 사용하지 않기 때문에 drop해주었다. 유저와 아이템의 최대 번호를 저장해준다. 이 정보가 필요한 이유는 pivot_table같은 것으론 중간 정보가 누락되기 때문에 이를 통해 훈련 데이터에 상한 데이터까지 표현하기 위함이다. (유사도 포함) 또한, rating matrix를 만드는데 평가하지 않은 것은 -1로 초기화한다. (다른 값들은 결과에 방해된다)

```
# 각 유저가 모든 item에 부여한 rating의 평균
avg_rating = np.zeros(num_of_users)
for user in range(num_of_users):
    rated_item = rating_matrix[user] >= 0
    avg_rating[user] = rating_matrix[user][rated_item].sum() / len(rated_item[rated_item])

# 유저 간 cosine similarity로 neighbors 구하기
user_similarity_matrix = np.ones([num_of_users, num_of_users])
for i in range(num_of_users):
    for j in range(num_of_users):
        if i > j:
            user_similarity_matrix[i][j] = user_similarity_matrix[j][i]
        else:
            user_similarity_matrix[i][j] = get_cosine_similarity(rating_matrix[i], rating_matrix[j])
```

이후에 prediction을 할 때, 적절한 neighbor가 없다면 그 유저가 아이템들에 부여한 평균을 부여해야 하므로 rating을 구해준다. 뿐만 아니라, aggregation에도 필요하다. 유저 간의 코사인 유사도를 구해서 이웃을 구할 것이기에 모든 유저

페어 끼리의 코사인 유사도를 구해준다. 대칭 행렬이기 때문에 삼각행렬 부분만 구해주고 나머지는 복사해주면 된다. 연산 속도를 줄이기 위함이다.

```
def get_cosine_similarity(a, i):
    same_item_idx = (a>= 0) * (i >= 0)
    if len(same_item_idx[same_item_idx]) == 0:
        return 0
    # 양수 데이터만 남김
    vec1 = a[same_item_idx]
    vec2 = i[same_item_idx]
    return vec1.dot(vec2) / (np.sqrt(sum(vec1 ** 2)) * np.sqrt(sum(vec2 ** 2)))
```

코사인 유사도는 알려진 공식을 사용하였다. 이 때 연산의 편의를 위해, 양수값만 남겨서 사용한다.

```
# u#.base_prediction.txt에 넘겨줄 정보 만들기
result = list()
for user_id, item_id, rating in test_set.values.tolist():
    try:
        itemInfo = rating_matrix[:, item_id-1]
        rated = itemInfo >= 0
        rated_user_similarity = user_similarity_matrix[user_id-1, rated]
        # 적절한 Neighbor가 없는 경우 유저가 부여한 rating의 평균값을 취한다.
        if rated_user_similarity.sum() == 0:
            result.append([user_id, item_id, avg_rating[user_id-1]])
        # 적절한 Neighbor가 있다면, Aggregation
        else:
            prediction = avg_rating[user_id-1] + sum(rated_user_similarity * (itemInfo[rated] - avg_rating[rated])) / sum(rated_user_similarity)
            result.append([user_id, item_id, np.clip(prediction, 1, 5)])
    # 없던 아이템인 경우 평균값
    except IndexError as e:
        result.append([user_id, item_id, avg_rating[user_id-1]])

# save file
np.savetxt(output_file_name, result, fmt='%d\t%d\t%s')
```

이제 코사인 유사도를 통해 각 user들의 이웃들을 구해주었다. Test 파일을 읽어서 라인별로 rating을 예측해주면 된다. 만약 유저의 적절한 neighbor가 없다면 해당 유저가 부여한 rating의 평균값을 취한다. 반대로 적절한 neighbor들이 있따면 aggregation 해주면 된다. 이 때 예측값이 1보다 작으면 1로 5보다 크면 5로 바꾸어 저장해준다. 또한, 기존의 train set에서 얻은 id를 초과한다면 없던 아이템이므로 그냥 평균값을 취해준다. 마지막으로 이를 양식에 맞게 저장해주면 된다.

## 4. Testing

```
C:₩Users₩mok03₩GitHub₩ITE4005_Data-Science₩Recommendation Systems₩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: 0.9729288

C:₩Users₩mok03₩GitHub₩ITE4005_Data-Science₩Recommendation Systems₩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: 0.9636573

C:₩Users₩mok03₩GitHub₩ITE4005_Data-Science₩Recommendation Systems₩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: 0.9569385

C:₩Users₩mok03₩GitHub₩ITE4005_Data-Science₩Recommendation Systems₩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: 0.954217

C:₩Users₩mok03₩GitHub₩ITE4005_Data-Science₩Recommendation Systems₩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: 0.9540529
```