Recommender System - 2016025950 정재용

Recommend Algorithm

- Item - based Collaborative Filtering
  - 아이템간의 유사도를 구하고, 유사한 아이템들에 대한 유저의 평점을 이용해서 평점을 예측한다.
- 예시
  - 1번 유저가 6번 아이템에 대해 몇점을 줄지 예상할 때
  - 6번 아이템과 나머지 아이템들의 유사도를 구한다.
  - 1번 유저가 해당 아이템들에 대해 매긴 평점을 구한다.
  - 유사도가 높은 아이템들을 가지고, 유저가 매긴 평점을 평균낸다.

Methods

get_sparse_matrix(user_item_list)

```
def get_sparse_matrix(user_item_list):
    # ----------
    # user-item-rating list를 sparse matrix로 변환해준다.
    #   user_item_list : user-item-rating을 row로 갖는 dataframe
    #   return : user-item을 행, 열로 갖고 rating을 나타내는 sparse matrix
    # ----------

    _list_columns = user_item_list.columns
    _index, _columns, _value = _list_columns

    _sparse = user_item_list.pivot_table(_value,index=_index, columns=_columns).fillna(0)

    return _sparse
```

- user-item 리스트를 sparse matrix로 변환한다.

cal_item_similarity(user_item_rating)

```python
def cal_item_similarity(user_item_rating):
    # ----------
    # item 간의 코사인 유사도를 구한다.
    #   user_item_rating : user-item-rating을 나타내는 sparse matrix
    #   return : item 간의 코사인 유사도를 계산한 array
    # ----------

    return cosine_similarity(user_item_rating.T)
```

- sparse matrix를 가지고 item간의 코사인 유사도를 구한다.
- sklearn 의 코사인 유사도 함수를 사용했다.

get_user_rating_avg(user_item_rating, user_id)

```python
def get_user_rating_avg(user_item_rating, user_id):
    # ----------
    # user가 item들에 대해 매긴 평점의 평균을 구한다.
    #   user_item_rating : user-item-rating을 나타내는 sparse matrix
    #   user_id : 구하고자 하는 유저의 id
    #   return : 유저가 매긴 평점들의 평균
    # ----------

    user_rating = user_item_rating.loc[user_id]
    user_rating_drop = user_rating[user_rating > 0]

    return int(round(np.average(user_rating_drop.values), 0))
```

- 유저가 매긴 평점들의 평균을 구한다.
- item이 training set에 없을 때 사용한다.

expect_user_item_rating(user_item_rating, sim_array, user_id, item_id)

```python
def expect_user_item_rating(user_item_rating, sim_array, user_id, item_id):
    # ----------
    # user가 item에 대해 매긴 평점을 예측한다.
    #   user_item_rating : user-item-rating을 나타내는 sparse matrix
    #   sim_array : 아이템 간의 코사인 유사도
    #   user_id : 구하고자 하는 유저의 id
    #   item_id : 예측하고자 하는 아이템의 id
    #   return : 아이템에 대한 평점의 예측값 (정수)
    # ----------

    # 유저가 아이템들에 매긴 평점과, 구하려는 아이템과 다른 아이템들간의 유사도를 가져온다.
    user_rating = user_item_rating.loc[user_id]
    item_index = np.where(user_item_rating.columns == item_id)[0][0]
    item_corr = sim_array[item_index]

    rating_corr =  np.array([user_rating, item_corr]).T
    rating_corr_drop = rating_corr[rating_corr[:, 0] > 0] # 유저가 평점을 매기지 않은 아이템들은 지운다.
    top_rating = rating_corr_drop[np.argsort(rating_corr_drop, axis=0).T[1]][::-1][0:13, :].T # 유사도 상위 13개 아이템을 가져온다.

    return int(round(np.average(top_rating[0],weights=top_rating[1]), 0))
```

- 유저가 다른 아이템들에 대해 매긴 평점과, 그 아이템들과의 유사도를 구한다.
- 유저가 평점을 매기지 않은 아이템들은 제외하고, 예측하고자 하는 아이템과 유사도가 높은 상위 13개 아이템을 구한다.
- 유사도를 가중치로 평점의 가중평균을 구한 후 정수 값으로 변환한다.

main function

```python
if __name__ == '__main__':
    base_file_name = sys.argv[1]
    test_file_name = sys.argv[2]

    columns = ['user_id', 'item_id', 'rating', 'time_stamp']
    base_file = pd.read_csv(base_file_name, sep='\t', names=columns)
    test_file = pd.read_csv(test_file_name, sep='\t', names=columns)

    user_item_list = base_file.drop('time_stamp', axis=1)
    test_rating = test_file['rating'].values
    test_user_item_list = test_file.drop(['time_stamp', 'rating'], axis=1)

    user_item_rating = get_sparse_matrix(user_item_list)
    sim_array = cal_item_similarity(user_item_rating)
```

- command line에서 파일 이름을 받고 해당 파일을 불러온다.
  - base 파일과 test 파일들을 실행파일과 같은 경로에 둔다.
- 데이터 전처리 과정을 거친다.
  - time_stamp 열은 사용하지 않기 때문에 삭제한다.
  - 주어진 list를 sparse matrix로 바꾸고, item 간의 유사도를 코사인 유사도를 사용해서 구한다.

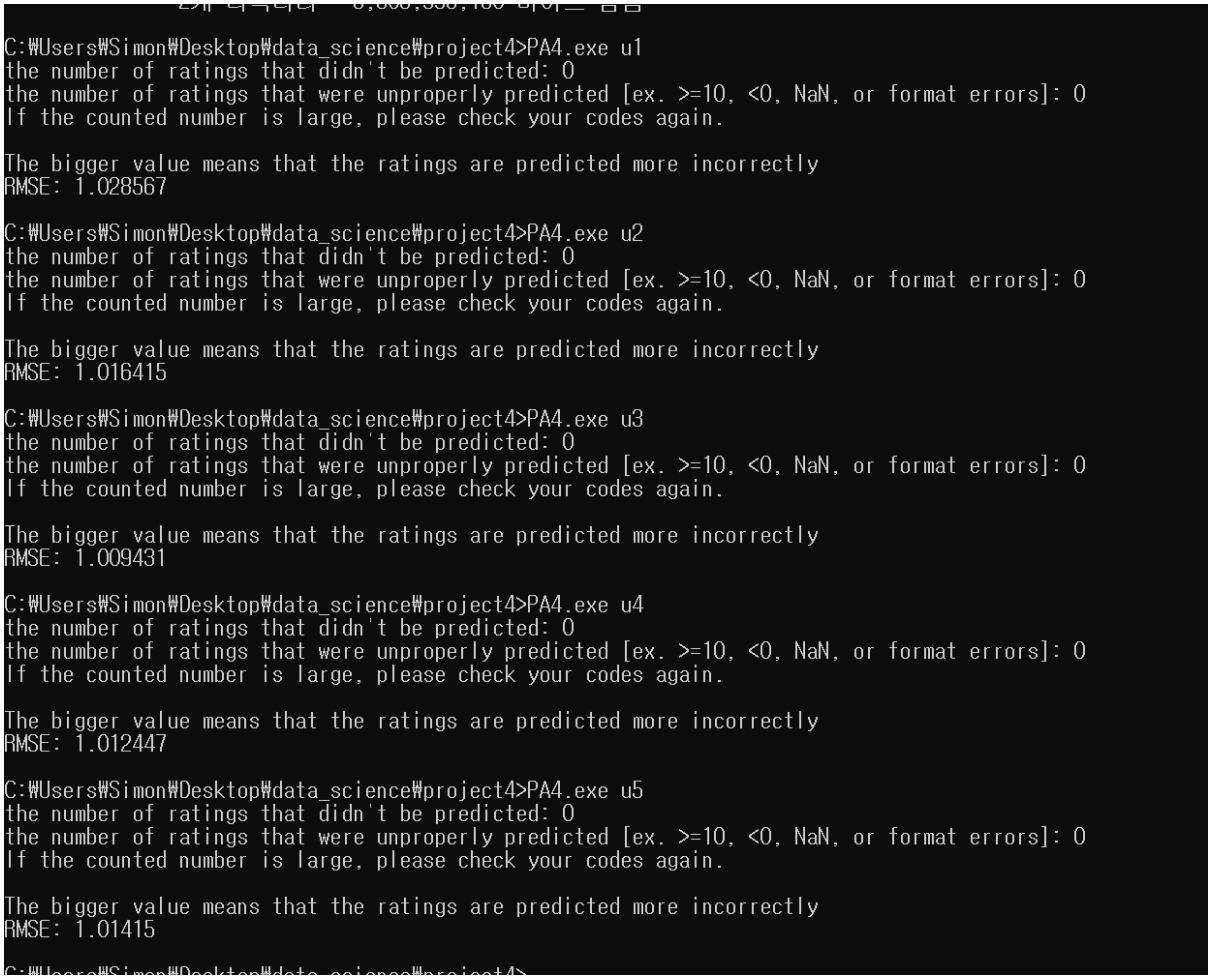
```python
result = []
# 각 user-item 쌍에 대해 예측값을 구한다.
# 기존 데이터에 없던 item에 대해서는 유저가 매긴 평점들의 평균으로 예측한다.
for idx, user_id, item_id in test_user_item_list.itertuples():
    if item_id in user_item_rating.columns:
        result.append(expect_user_item_rating(user_item_rating, sim_array, user_id, item_id))
    else:
        result.append(get_user_rating_avg(user_item_rating, user_id))

test_user_item_list['rating'] = result
test_user_item_list.to_csv(f'{base_file_name}_prediction.txt', sep='\t', index=False, header=False)
```

- user가 item에 대해 남길 평점을 예측한다.
    - 만약 해당 item이 training set에 없던 것이면, 유저가 매긴 평점들의 평균을 구해서 사용한다.

Test Result

```
(base)  simon  ~/Desktop/data_science/assignment/project4_recommend_system  ls
2020_DM_Longterm project.pdf test                    u2.base                 u3.base_prediction.txt  u4.test
Untitled.ipynb               u1.base                 u2.base_prediction.txt  u3.test                 u5.base
data                         u1.base_prediction.txt  u2.test                 u4.base                 u5.base_prediction.txt
recommender.py               u1.test                 u3.base                 u4.base_prediction.txt  u5.test
(base)  simon  ~/Desktop/data_science/assignment/project4_recommend_system  python3 recommender.py u1.base u1.test
(base)  simon  ~/Desktop/data_science/assignment/project4_recommend_system
```

- base 파일과 test 파일을 실행 프로그램과 같은 경로에 둔다.


C:₩Users₩Simon₩Desktop₩data_science₩project4>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: 1.028567
C:₩Users₩Simon₩Desktop₩data_science₩project4>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: 1.016415
C:₩Users₩Simon₩Desktop₩data_science₩project4>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: 1.009431
C:₩Users₩Simon₩Desktop₩data_science₩project4>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: 1.012447
C:₩Users₩Simon₩Desktop₩data_science₩project4>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: 1.01415