추천시스템 프로젝트 [도서 추천 시스템]

이웃기반 협업필터링 〈User-based Collaborative Filtering〉

2018111298 조승연

1. 개요

유저가 도서에 매긴 평점 데이터를 활용해 2D sparse matrix를 만들고,
코사인 유사도를 이용해 유저를 중점으로 추천 점수를 계산한다.

2. 진행 코드

(1) 데이터 전처리

In [3]:
```
ratings_df['UserID'] = ratings_df['Dummy'].str.split(';').str[0]
ratings_df['ISBN']= ratings_df['Dummy'].str.split(';').str[1]
ratings_df['Book-Rating']= ratings_df['Dummy'].str.split(';').str[2]
ratings_df = ratings_df.drop(columns=['Dummy'], axis=1)
ratings_df = ratings_df.drop(index=0, axis= 0)
# 새 컬럼에 적용 후 기존 컬럼 삭제, 불필요한 헤더 부분 삭제
ratings_df['Book-Rating'] = ratings_df['Book-Rating'].str.replace('"','')
ratings_df['Book-Rating'] = ratings_df['Book-Rating'].str.replace(',','')
ratings_df['ISBN'] = ratings_df['ISBN'].str.replace('"','')
# 큰따옴표, 콤마 등 삭제
ratings_df.head()
```

Out[3]:

	UserID	ISBN	Book-Rating
1	276725	034545104X	0
2	276726	0155061224	5
3	276727	0446520802	0
4	276729	052165615X	3
5	276729	0521795028	6

In [5]:
```
ratings_df['UserID'] = ratings_df['UserID'].apply(pd.to_numeric)
ratings_df['Book-Rating'] = ratings_df['Book-Rating'].apply(pd.to_numeric)
ratings_df.dtypes
# rating 타입 숫자형으로 변경
```

Out[5]:
```
UserID          int64
ISBN           object
Book-Rating    float64
dtype: object
```

In [6]:
```
user_count = ratings_df.groupby('UserID').agg({'Book-Rating' : [np.size, np.mean, np.std]})
book_count = ratings_df.groupby('ISBN').agg({'Book-Rating' : [np.size, np.mean, np.std]})

user_count.loc[user_count[('Book-Rating', 'size')] > 2, 'heavy'] = 1
user_filtered = user_count[user_count['heavy']==1]
user_filtered.describe()
```

Out[6]:

	Book-Rating			heavy
	size	mean	std	
count	30447.000000	30447.000000	30447.000000	30447.0
mean	31.931225	3.940958	3.171838	1.0
std	162.955434	2.307209	1.411712	0.0
min	3.000000	0.000000	0.000000	1.0
25%	4.000000	2.250000	2.516611	1.0
50%	7.000000	3.800000	3.534945	1.0
75%	17.000000	5.555556	4.112988	1.0
max	13602.000000	10.000000	5.773503	1.0

데이터 타입 변경, 2개 이상의 평점만 포함시키도록 데이터 정제

(2) 데이터 셋 분할

In [22]:
```
ratings = ratings_df.loc[ratings_df['UserID'].isin(user_filtered.index)]
#ratings_df.to_csv(os.path.join(path, 'BX-Book-Ratings_updated2.csv'), index=False)

ratings = ratings[:80000]
```

In [23]:
```
train_df, test_df = train_test_split(ratings, test_size=0.3, random_state=1234)

print(train_df.shape)
print(test_df.shape)
```

```
(56000, 3)
(24000, 3)
```

최대 8만 데이터 활용 가능(개발 환경 문제)
테스트 셋 : 트레인 셋 = 3 : 7

(3) Sparse Matrix 생성

```
In [38]: user_sparse_matrix = sparse_matrix.fillna(0).transpose()
         user_sparse_matrix
```

Out [38]:

ISBN	9022906116	0 7336 1053 6	0 907 062 008	00000000	000000000	0000000000	00000000000	0000000124911	000000078245	000000168274	...	O590603647	O68481545
UserID													
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	
278843	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.
278844	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.

(4) 코사인 유사도(유저 간)

```
In [27]: user_cossim_df = cossim_matrix(user_sparse_matrix, user_sparse_matrix)
         #user_cossim_df.describe()
         user_cossim_df
```

Out [27]:

	8	9	14	17	22	32	39	44	53	56	...	278774	278782	278798	278832	278838	278843	278844	278849	278851	278854
UserID																					
8	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
278843	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
278844	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
278849	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
278851	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

(5) 유저 예측 점수

```
In [39]: user_prediction_result_df = user_prediction_result_df.transpose()
         user_prediction_result_df
```

Out [39]:

UserID	8	9	14	17	22	32	39	44	53	56	...	278774	278782	278798	278832	278838	278843	278844	278849	278851	278854
9022906116	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0.141927	0	0.101721	0	0	0.0272716	0
0 7336 1053 6	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0 907 062 008	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
00000000	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
000000000	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
O9088446X	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0.162202	0	0.116253	0	0	0.0311675	0
X000000000	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0.202752	0	0.145316	0	0	0.0389594	0
ZR903CX0003	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0.0202752	0	0.0145316	0	0	0.00389594	0
\0432534220\	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0.121651	0	0.0871895	0	0	0.0233756	0
\2842053052\	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0.141927	0	0.101721	0	0	0.0272716	0

40637 rows × 2305 columns

(6) RMSE 평가 및 해당 추천 도서 목록 출력

```
In [34]: result_df = evaluate(test_df, user_prediction_result_df)
         print(result_df)
         print(f"RMSE: {sqrt(mean_squared_error(result_df['actual_rating'].values, result_df['pred_rating'].values))}")
```

```
6135
1981
<ipython-input-32-9e340391fed6>:16: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  for userId, group in tqdm(grouped):
100% 1996/1996 [00:17<00:00, 116.12it/s]
   actual_rating        ISBN pred_rating
0            0.0  0515087122           0
1            7.0  042516098X           0
RMSE: 4.949747468305833
```

```
In [36]: result_df_with_title = pd.merge(result_df, book_name, how='inner', on='ISBN')
         result_df_with_title
```

Out [36]:

	actual_rating	ISBN	pred_rating	title
0	0.0	0515087122	0	The Cat Who Ate Danish Modern (Cat Who... (Pap...
1	7.0	042516098X	0	Hornet's Nest

3. 데이터 양에 따른 결과 변화

(1) 최대 데이터 8만, 테스트 : 트레인 셋 = 3 : 7

```
In [34]: result_df = evaluate(test_df, user_prediction_result_df)
         print(result_df)
         print(f"RMSE: {sqrt(mean_squared_error(result_df['actual_rating'].values, result_df['pred_rating'].values))}")
```

```
6135
1981
<ipython-input-32-9e340391fed6>:16: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  for userId, group in tqdm(grouped):
100% |████████████████████████████████| 1996/1996 [00:17<00:00, 116.12it/s]
   actual_rating        ISBN pred_rating
0            0.0  0515087122           0
1            7.0  042516098X           0
RMSE: 4.949747468305833
```

```
In [36]: result_df_with_title = pd.merge(result_df, book_name, how='inner', on='ISBN')
         result_df_with_title
```

Out [36]:

	actual_rating	ISBN	pred_rating	title
0	0.0	0515087122	0	The Cat Who Ate Danish Modern (Cat Who... (Pap...
1	7.0	042516098X	0	Hornet's Nest

RMSE 점수 4.94

(2) 최소 데이터 1만, 테스트 : 트레인 셋 = 3 : 7

```
In [53]: result_df = evaluate(test_df, user_prediction_result_df)
         print(result_df)
         print(f"RMSE: {sqrt(mean_squared_error(result_df['actual_rating'].values, result_df['pred_rating'].values))}")
```

```
236
240
<ipython-input-50-9e340391fed6>:16: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  for userId, group in tqdm(grouped):
100% |████████████████████████████████| 244/244 [00:01<00:00, 127.88it/s]
   actual_rating        ISBN pred_rating
0            7.0  042516098X           0
RMSE: 7.0
```

```
In [54]: result_df_with_title = pd.merge(result_df, book_name, how='inner', on='ISBN')
         result_df_with_title
```

Out [54]:

	actual_rating	ISBN	pred_rating	title
0	7.0	042516098X	0	Hornet's Nest

RMSE 점수 7.0

4. 결론 및 개선점

2개 이상의 평가를 내린 유저의 데이터만을 적용하는 정제, 데이터 양, 테스트 셋과 트레인 셋의 비율에 따라 RMSE 결과값이 각각 다르게 나오는 걸 확인할 수 있었다.

RMSE 결과 비교			
데이터 정제	데이터 양	테스트 셋 : 트레인 셋	**RMSE 점수**
X	8만	2 : 8	0.0
X	8만	3 : 7	0.0
X	10만	4 : 6	0.0
O	7만	3 : 7	5.5
O	7만	4 : 6	6.13
O	8만	2 : 8	0.0
O	8만	3 : 7	4.9497
O	8민	4 : 6	6.13
O	8만	5 : 5	6.13

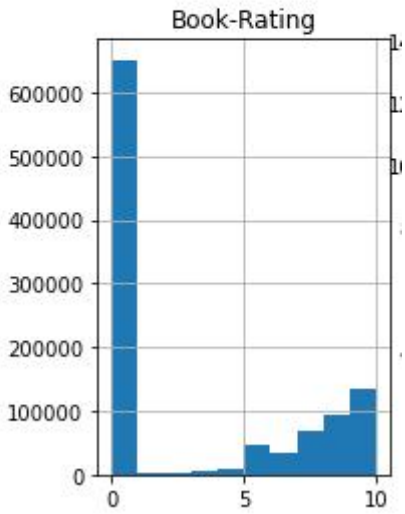
0.0의 경우 예측된 값이 0인 데이터가 단 하나만 존재해 0.0이란 점수가 나온 것으로 추측

(1) 문제점

- 데이터 활용 양을 늘릴 때마다 램 부족 현상이 일어나는 개발 환경의 한계로 전체적으로 사용한 데이터가 적어 예측된 결과들도 적었다. 때문에 RMSE 점수도 평균적으로 높았으며, 성능 측정이 제대로 이루어지지 않는 문제들이 발생했다.
- 사용한 데이터 셋의 평점 분포를 확인했을 때 0인 평점이 압도적이었다.
- rating 데이터 프레임의 8만개 값을 단순 슬라이싱이 아닌 sample 메소드를 활용한 랜덤 표본 추출 작업을 하려고 하니 개발 PC의 램 메모리 부족 문제로 진행할 수 없었다.

```
In [56]: ratings_df.hist()

Out[56]: array([[<matplotlib.axes._sub
                 <matplotlib.axes._sub
        dtype=object)
```



-

(2) 개선점

다양한 방법으로 RMSE 성능 측정을 시도해본 결과 활용할 데이터양을 늘리고 테스트 셋 비율을 적정 수준으로 설정해야 보다 확실한 결과를 구할 수 있을 것으로 예상된다. 더 많은 데이터를 활용하는 것이 프로젝트의 개선점이 될 것이라 생각한다.