improved_cosine_recommeder_system

October 30, 2019

[1]:
```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tqdm import tqdm
import heapq
```

[2]:
```
df=pd.read_csv('ratings.csv')
```

[3]:
```
user_movie_rating = df.pivot_table(index='userId', columns='movieId',␣
 ↪values='rating')
```

[4]:
```
# dist=[]
# klen=[]
```

[5]:
```
# for j in tqdm(range(1,len(user_movie_rating.index)+1)):
#     dist_temp=[]
#     klen_temp=[]
#     for i in range(1,len(user_movie_rating.index)+1):
#         user1_rating=user_movie_rating.iloc[j-1][user_movie_rating.iloc[j-1].
 ↪isna()==False]
#         user1_movieId=list(user_movie_rating.iloc[j-1][user_movie_rating.
 ↪iloc[j-1].isna()==False].index)
#         user_i_rating=user_movie_rating.iloc[i-1][user_movie_rating.iloc[i-1].
 ↪isna()==False]
#         user_i_1_rating=user_i_rating[user_i_rating.index.
 ↪isin(user1_movieId)]
#         user_i_1_movieId=list(user_i_1_rating.index)
#         user1_i_rating=user1_rating[user1_rating.index.
 ↪isin(user_i_1_movieId)]
#         a=sum(user1_i_rating*user_i_1_rating)
#         b=np.sqrt(sum(np.square(user_i_1_rating)))
#         c=np.sqrt(sum(np.square(user1_rating)))
#         k=a/(b*c)
#         klen_temp.append(k)
#         dist_temp.append(np.cos(k))
#     klen.append(klen_temp)
#     dist.append(dist_temp)
```

[6]:
```
# df1=pd.DataFrame(klen)
```

[7]:
```
# df1
```

[8]:
```
# df2=pd.DataFrame(dist)
```

[9]:
```
# df2
```

[10]:
```
# df1.to_csv('cos_similarity_improved.csv', index=False, header=False)
```

[11]:
```
# df2.to_csv('angle_similarity_improved.csv', index=False, header=False)
```

[12]:
```
similarity_df=pd.read_csv('cos_similarity_improved.csv',header=None)
```

[13]:
```
similarity_df.fillna(0.0,inplace=True)
```

[14]:
```
user_similarity_dict={}
for i in tqdm(range(similarity_df.shape[0])):
    sorted_similarity=heapq.nlargest(10,similarity_df.iloc[i])[1:]
    dict1={}
    for j in sorted_similarity:
        dict1[similarity_df.iloc[i][similarity_df.iloc[i]==j].index.
 ↪values[0]+1]=j
    user_similarity_dict[i+1]=dict1
```

```
100%|| 610/610 [00:06<00:00, 92.33it/s]
```

[15]:
```
abs_error_sum=0
root_square_sum=0
count=0

for i in tqdm(user_movie_rating.index):
    non_null_movies=list(user_movie_rating.loc[i][user_movie_rating.loc[i].
 ↪isna()==False].index)
    k0=user_movie_rating.loc[list(user_similarity_dict[i].keys())[0]]
    k0=k0[k0.index.isin(non_null_movies)]
    k0.fillna(0,inplace=True)
    k1=user_movie_rating.loc[list(user_similarity_dict[i].keys())[1]]
    k1=k1[k1.index.isin(non_null_movies)]
    k1.fillna(0,inplace=True)
    k2=user_movie_rating.loc[list(user_similarity_dict[i].keys())[2]]
    k2=k2[k2.index.isin(non_null_movies)]
    k2.fillna(0,inplace=True)
    a=list(user_similarity_dict[i].values())[0]
    b=list(user_similarity_dict[i].values())[1]
    c=list(user_similarity_dict[i].values())[2]
    predicted_data=((k0*a+k1*b+k2*c)/(a+b+c))
    predicted_data=np.ceil(predicted_data*2)/2
    predicted_data=predicted_data.replace(6.0,5.0)
    predicted_data=predicted_data.replace(5.5,5.0)
    actual_data=user_movie_rating.loc[i][user_movie_rating.loc[i].isna()==False]
    abs_error_sum=abs_error_sum+sum(np.abs(predicted_data-actual_data))
    root_square_sum=root_square_sum+sum(np.square(predicted_data-actual_data))
    count=count+len(non_null_movies)
```

```
100%|| 610/610 [00:03<00:00, 160.58it/s]
```

[16]:
```
np.sqrt(root_square_sum/count)
```

[16]:
```
1.8193189813206385
```

[17]:
```
abs_error_sum/count
```

[17]:
```
1.3945961759689
```

[ ]:
```
```