Simple Approaches to Recommender Systems

a-Popularity Based Recommenders(available in https://www.linkedin.com/posts/zohreh-karimi-a21ba2163_popularityrecommendersystem-activity-6785962167796551681-2SQl)

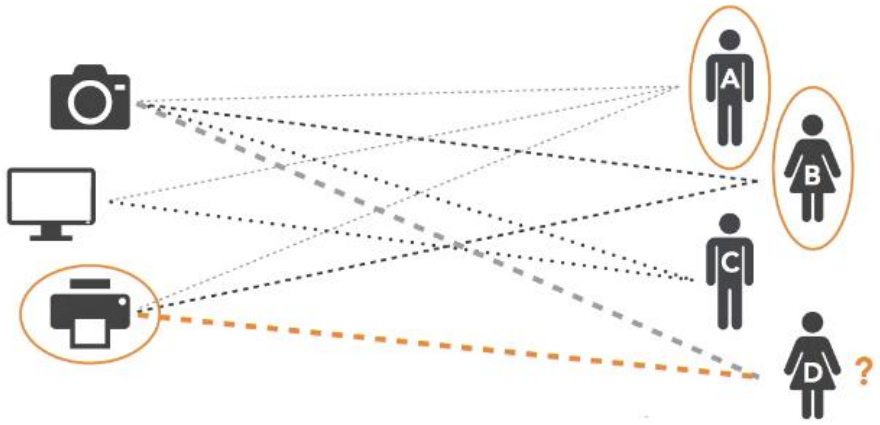
b-Correlation Based Recommendation

This recommendation sytem offer basic form of **Collaborative Filtering**, that's because with correlation based systems items are recommended based on similarities on their user reviews. in this system, use Pearson's r correlation to recommend an item that is most similar to the item a user has already chosen.

Overview of Pearson correlation coefficient(r):

- is a measure of linear correlation between two variables (in this case two item's ratings).
- r=1 -> Strong positive relationship
- r=0 -> Not linearly correlated
- r=-1 -> Strong negative linear relationship

Correlation based recommenders use item_based similarity, that is they recommend an item based on how well it correlates with other items with respect to user ratings, let's look at the logic of this:



as we see on picture, shopper D, see that she has already chosen and reviewed the camera. she gave a rating of 4 stars, Now let's see who else reviewed the camera. It looks like users A,B, and C also reviewed the camera, look at the ratings each of these users gave. user A = 4 stars, user B = 4 stars, user C = 2.5 stars. Based on correlations between user ratings, we'd say that user A's and user B's ratings are highly correlated with user D's ratings. Now look at what other items user A and user B liked. They both gave good ratings to the printer. so based on user A's and user B's review scores correlate with user D's review scores of the cameras, and based on the shared preferences user A and user B have for the the printer, we would recommend the printer to user D as well.

Here is an example for Correlation based recommender system with Python code. We use three datasets (Restaurant & consumer & geoplaces2 data Data Set)

Let's do this in jupyter Notebook:

```
#import libraries
import numpy as np
import pandas as pd

#import data
data=pd.read_csv('rating_final.csv')
cuisine=pd.read_csv('chefmozcuisine.csv')
geodata=pd.read_csv('geoplaces2.csv')


print(data.head())
```

```
  userID  placeID  rating  food_rating  service_rating
0  U1077   135085       2            2               2
1  U1077   135038       2            2               1
2  U1077   132825       2            2               2
3  U1077   135060       1            2               2
4  U1068   135104       1            1               2
```

each place in dataset gets rating of either zero,one or two. as we see, user ID is duplicate,

```
print(geodata.head(1))
```

```
   placeID   latitude  longitude  \
0   134999  18.915421 -99.184871

                                     the_geom_meter             name  \
0  0101000020957F000088568DE356715AC138C0A525FC46...  Kiku Cuernavaca

      address        city    state country fax  ...           alcohol  \
0  Revolucion  Cuernavaca  Morelos  Mexico   ?  ...  No_Alcohol_Served

  smoking_area dress_code     accessibility   price                   url  \
0         none   informal  no_accessibility  medium  kikucuernavaca.com.mx

  Rambience franchise    area other_services
0  familiar         f  closed           none

[1 rows x 21 columns]
```

the reason we want this dataset, is that is provides a name for each of the unique places that's been reviewed. we just use place ID and name columns.

```
places=geodata[['placeID','name']]
places.head()
```

	placeID	name
0	134999	Kiku Cuernavaca
1	132825	puesto de tacos
2	135106	El Rinc�n de San Francisco
3	132667	little pizza Emilio Portes Gil
4	132613	carnitas_mata

```
cuisine.head()
```

	placeID	Rcuisine
0	135110	Spanish
1	135109	Italian
2	135107	Latin_American
3	135106	Mexican
4	135105	Fast_Food

as we see, we have place ID and cuisine type.

Now, look at the ratings these places are getting.

```
rating=pd.DataFrame(data.groupby('placeID')['rating'].mean())
rating.head()
```

placeID	rating
132560	0.50
132561	0.75
132564	1.25
132572	1.00
132583	1.00

to the mean value we also need to look at how popular each of this places was. so add a column called rating count for how many reviews each place got.

```
rating['rating_count']=pd.DataFrame(data.groupby('placeID')['rating'].count())
rating.head()
```

placeID	rating	rating_count
132560	0.50	4
132561	0.75	4
132564	1.25	4
132572	1.00	15
132583	1.00	4

we use rating .describe() to statistics:

rating.describe()

	rating	rating_count
count	130.000000	130.000000
mean	1.179622	8.930769
std	0.349354	6.124279
min	0.250000	3.000000
25%	1.000000	5.000000
50%	1.181818	7.000000
75%	1.400000	11.000000
max	2.000000	36.000000

for count, taking a count of the rating data frame we get 130, that indicate they are 130 unique places that have been reviewed. you see the max value for rating count comes to 36. this means that the most popular place in dataset has got a total of 36 reviews. then we sort the dataset to see placeID with sorted ratings :

```
rating.sort_values('rating_count',ascending=False).head()
```

	rating	rating_count
placeID		
135085	1.333333	36
132825	1.281250	32
135032	1.178571	28
135052	1.280000	25
132834	1.000000	25

Let's find name of this place:

places[places['placeID']==135085]

	placeID	name
121	135085	Tortas Locas Hipocampo

and the type of cuisine this place serves:

cuisine[cuisine['placeID']==135085]

	placeID	Rcuisine
44	135085	Fast_Food

The next thing we need is to build a user by item utility matrix. To do that we call pivot table function.

```
places_crosstab=pd.pivot_table(data=data,values='rating',index='userID',columns='placeID')
places_crosstab.head()
```

placeID	132560	132561	132564	132572	132583	132584	132594	132608	132609	132613	...	135080	135081	135082	135085	135086	135088	135104
userID																		
U1001	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0.0	NaN	NaN	NaN
U1002	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	1.0	NaN	NaN	NaN
U1003	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2.0	NaN	NaN	NaN	NaN	NaN	NaN
U1004	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
U1005	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 130 columns

as you see, it's full of Nan values, That's because people never review that many places.let me show you how we can use this table to find places that are correlated.

```
Tortas_ratings=places_crosstab[135085]
Tortas_ratings[Tortas_ratings >=0]
```

```
userID
U1001    0.0
U1002    1.0
U1007    1.0
U1013    1.0
U1016    2.0
U1027    1.0
U1029    1.0
U1032    1.0
U1033    2.0
U1036    2.0
U1045    2.0
U1046    1.0
U1049    0.0
U1056    2.0
U1059    2.0
U1062    0.0
U1077    2.0
U1081    1.0
U1084    2.0
U1086    2.0
U1089    1.0
```

we got 36 review scores, and they range between zero and two. Not to find the correlation between each of this places and the Tortas restaurant, we call the core with method of our places cross tab, and then pass it the Tortas rating series. so generate a Pearson R correlation coefficient between Tortas and each other place that's been reviewed in the dataset(this correlation is based on similarities and user reviews that were given to each place).

```
similar_to_Tortas=places_crosstab.corrwith(Tortas_ratings)
corr_Tortas=pd.DataFrame(similar_to_Tortas,columns=['PearsonR'])
corr_Tortas.dropna(inplace=True)
corr_Tortas.head()
```

placeID	PearsonR
132572	-0.428571
132723	0.301511
132754	0.930261
132825	0.700745
132834	0.814823

we see the data frame that contain each placeID and a Pearson R correlation coefficient that indicate how well each place correlates with Tortas based on user rating.

if we've found some places that were really well correlated with Tortas but that had only, say, two ratings total, then those places probably wouldn't really be all that similar to Tortas. maybe those places got similar ratings as Tortas, but they wouldn't be very popular.therefore, that correlation really wouldn't be significant. we also need to take stock of how popular each of these places is and in addition to how wel the review scores correlate with the ratings that were given to other places in the dataset. so join our corr Tortas data frame with a rating state of frame. then create a filter so that we can see only the places from the data frame that have at least 10 user reviews, and for those places look at the Pearson R correlation coefficient sorted in descending order.

```
Tortas_corr_summery=corr_Tortas.join(rating['rating_count'])
Tortas_corr_summery[Tortas_corr_summery['rating_count']>=10].sort_values('PearsonR',ascending=False).head(10)
```

placeID	PearsonR	rating_count
135076	1.000000	13
135085	1.000000	36
135066	1.000000	12
132754	0.930261	13
135045	0.912871	13
135062	0.898933	21
135028	0.892218	15
135042	0.881409	20
135046	0.867722	11
132872	0.840168	12

since we sorted the data frame in descending order we have a list of top reviewd places that are most similar to Tortas. These Pearson R values of one aren't meaningful here. The reason you're seeing these is because for those places, there was only one user who gave a review to both places. That user gave both places the same score which is why you're seeing a Pearson R value of one. But a correlation that's based on similarities between only one review rating, that's not meaningful. The place need to have more than one reviewer in common, so we'll throw those place out.

Now take the top seven correlated results that remain and see if any of these places also serve fast food.

```
places_corr_Tortas=pd.DataFrame([135085,132754,135045,135062,135028,135042,135046],index=np.arange(7),columns=['placeID'])
summery=pd.merge(places_corr_Tortas,cuisine,on='placeID')
summery
```

	placeID	Rcuisine
0	135085	Fast_Food
1	132754	Mexican
2	135028	Mexican
3	135042	Chinese
4	135046	Fast_Food

The reason why you are only seeing five places hers, is that not all of the places were listed in the cuisine dataset.None the less, what we are seeing here, is that among the top six places that were correlated with Tortas,at least on of these places also serves fast food. Let's get a name for this place:

places[places['placeID']==135046]

	placeID	name
42	135046	Restaurante El Revecito

To evaluate how relevent the similarity metric really is though, consider the entire set of possibilities. Meaning how many cuisine types are served at places in this dataset.

cuisine['Rcuisine'].describe()

```
count          916
unique          59
top       Mexican
freq           239
Name: Rcuisine, dtype: object
```

as we see, there are 59 unique types of cuisines that are served. So in analysis, we got back were six top places that were similar to Tortas based on correlation and popularity.of these six places, one other place also serves fast food.

Code

```
import numpy as np
import pandas as pd
data=pd.read_csv('rating_final.csv')
cuisine=pd.read_csv('chefmozcuisine.csv')
geodata=pd.read_csv('geoplaces2.csv')
places=geodata[['placeID','name']]
rating=pd.DataFrame(data.groupby('placeID')['rating'].mean())
rating['rating_count']=pd.DataFrame(data.groupby('placeID')['rating'].count())
rating.sort_values('rating_count',ascending=False)
places_crosstab=pd.pivot_table(data=data,values='rating',index='userID',columns='placeID')
Tortas_ratings=places_crosstab[135085]
similar_to_Tortas=places_crosstab.corrwith(Tortas_ratings)
corr_Tortas=pd.DataFrame(similar_to_Tortas,columns=['PearsonR'])
corr_Tortas.dropna(inplace=True)
Tortas_corr_summery=corr_Tortas.join(rating['rating_count'])
Tortas_corr_summery[Tortas_corr_summery['rating_count']>=10].sort_values('PearsonR',ascending=False)
places_corr_Tortas=pd.DataFrame([135085,132754,135045,135062,135028,135042,135046],index=np.arange(7),columns=['placeID'])
summery=pd.merge(places_corr_Tortas,cuisine,on='placeID')
summery
```