STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

2020 ECE SUMMER HONORS RESEARCH PROGRAM -

Industrial Recommender Systems in Media & Entertainment

• *By Siddharth Mandgi*
under Prof Hong Man

WEEK-4

Table of Contents

Collaborative Filtering for Movie and Music Recommender Engines

What is Collaborative Filtering?

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users.

It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. It looks at the items they like and combines them to create a ranked list of suggestions.

Our Approach

There are many ways to decide which users are similar and combine their choices to create a list of recommendations.
In our approach we have used the Unsupervised Machine Learning Algorithm of **K- Nearest Neighbors.**

Data

The datasets used of this recommender engine are Amazons Movies & TV and Digital Music datasets

Detailed Steps

Importing the Datasets

We begin our approach by importing the datasets in python by unzipping their respective json files using a custom function.

```
#For Extracting data from JSON file
import gzip

def parse(path):
  g = gzip.open(path, 'rb')
  for l in g:
    yield eval(l)

def getDF(path):
  i = 0
  df = {}
  for d in parse(path):
    df[i] = d
    i += 1
  return pd.DataFrame.from_dict(df, orient='index')

df = getDF('/Users/siddharthmandgi/Desktop/2020-Summer-Honors-Research/Datasets/reviews_Movies_and_TV_5.json.gz')

df.to_csv('/Users/siddharthmandgi/Desktop/2020-Summer-Honors-Research/Datasets/Amazon_Movies.csv')
```

Creating Relevant Variables

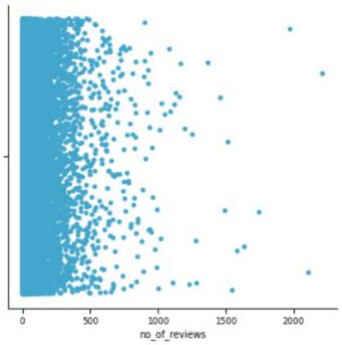
We the create a count variable to count the no of reviews for each movie id along with mean_ratings variable to generate the mean ratings for each movie and music track.

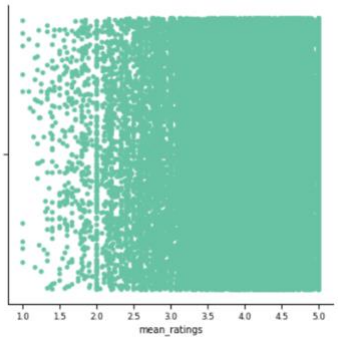
Movie_ID	mean_ratings	no_of_reviews
0005019281	4.458716	109
0005119367	4.793478	92
0307141985	4.800000	5
0307142469	4.750000	40
0307142477	2.333333	6
...	...	...
B00L8QP082	2.888889	9
B00LCAD24S	2.875000	8
B00LG7VVPO	4.833333	12
B00LH9ROKM	2.777778	9
B00LT1JHLW	4.272727	11

Music_ID	mean_ratings	no_of_reviews
5555991584	4.739130	23
B0000000ZW	4.133333	15
B00000016T	4.851064	47
B00000016W	4.629630	135
B00000017R	4.650000	20
...	...	...
B00JJOG5D4	4.428571	7
B00JRBLSR2	3.875000	8
B00JTHVWO8	4.714286	7
B00JYKU6BK	4.307692	13
B00KILDVEI	3.600000	5

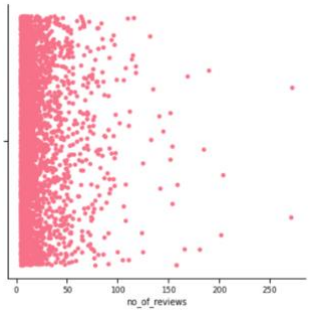
Seaborn Plots for these Variables

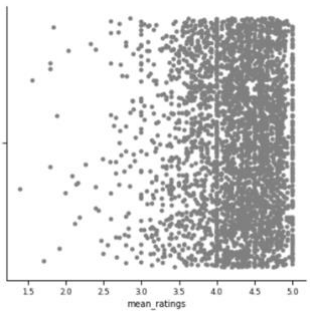
Movies & TV





Digital Music





From these plots and EDA, we find that Movie ID B003EYVXV4 has the **max total no of reviews = 2213** and a **mean rating = 4.11528** making it the most popular movie. **Average number of reviews per movie = 33.905** and similarly Music ID B0007NFL18 has the **max total no of reviews = 272** and a **mean rating = 2.558824**. **Average number of reviews per track = 18.135090.**

Popularity Threshold

We filter out the movies and music tracks with low no of reviews by defining popularity thresholds.

Pivot Table

We create a pivot tables for the users, movies or music tracks, no_of_reviews for our datasets.

reviewerID	A08161909WK3HU7UYTMW	A1020L7BWW9RAX	A10323WWTFPSGP	A103KNDW8GN92L
Music_ID				
5555991584	0.0	0.0	0.0	0.0
B00000016T	0.0	0.0	0.0	0.0
B00000016W	0.0	0.0	0.0	0.0
B00000017R	0.0	0.0	0.0	0.0
B0000004UM	0.0	0.0	0.0	0.0
...	...	...	...	...
B00DYFCYSO	0.0	0.0	0.0	0.0
B00EH49FRE	0.0	0.0	0.0	0.0
B00ERMICY8	0.0	0.0	0.0	0.0
B00F1CRRIU	0.0	0.0	0.0	0.0
B00FAEQ22G	0.0	0.0	0.0	0.0

reviewerID	A00295401U6S2UG3RAQSZ	A00348066Q1WEW5BMESN	A0040548BPHKXMHH3NTI
Movie_ID			
0005019281	0.0	0.0	0.0
0005119367	0.0	0.0	0.0
0307142469	0.0	0.0	0.0
0307142485	0.0	0.0	0.0
0307142493	0.0	0.0	0.0
...	...	...	...
B0006FO8E8	0.0	0.0	0.0
B0006FO8HA	0.0	0.0	0.0
B0006FO8IY	0.0	0.0	0.0
B0006FO8NY	0.0	0.0	0.0
B0006FO9B0	0.0	0.0	0.0

KNN Algorithm (Cosine Similarities)

We then convert these pivot tables into sparse matrix which contains only the non-zero ratings of for each movie and user as a vector which is then used in the KNN model.

We use cosine similarities to calculate distance between each neighbor and thus generate recommendations.

```
from scipy.sparse import csr_matrix as csr
movies_csr = csr(movie_features_df.values)
from sklearn.neighbors import NearestNeighbors as NN
model = NN(metric = 'cosine', algorithm = 'brute')

model.fit(movies_csr)
```

Results

Generating top 5 Recommendations: -

Movies & TV

```
Recommendations for Movie_ID: 0790744473

1 : 0780623746 with a cosine distance of 0.8951558003262833
2 : 6302993687 with a cosine distance of 0.9205581911370118
3 : 6303824358 with a cosine distance of 0.9345337468831068
4 : 6300251004 with a cosine distance of 0.9531553810066405
5 : 6305513406 with a cosine distance of 0.9585269837602576
```

Digital Music

```
Recommendations for Music_ID: B000B5QWNI

1 : B000002IHQ with a cosine distance of 0.8520662093574536
2 : B000001DYY with a cosine distance of 0.8708843048496521
3 : B000001EXB with a cosine distance of 0.8754212979841561
4 : B00001QGQI with a cosine distance of 0.884166491884198
5 : B00006ISBT with a cosine distance of 0.8869802419571245
```

- ***The best-case scenario: Cos0 = 1 which implies maximum similarity***
- ***The worst-case scenario: Cos90 = 0 which implies maximum dissimilarity***

Our distance values are around 0.9 which implies that our recommendations are very similar in features to our reviewed movie or music track.

Weighted Average Method for Generating Recommendations

Often times some movies or music tracks have a very high rating of 4.0 or 5.0 but have very low no of reviews. This implies that the high ratings of track have been provided by very few users and thus its ratings are not completely reliable. To tackle such a situation, we use weighted average ratings which provides ratings considering both the ratings and the no of reviewers

IMDB's Algorithm

$$W = \frac{Rv + Cm}{v + m}$$

- W = weighted score
- R = Average Rating
- C = the mean of all votes across the dataset
- v = no of votes or reviews for the movie
- m = minimum votes or number of reviews required to be in the top 250

Results

Our Recommendations based on this algorithm: -

Music_ID	mean_ratings	no_of_reviews	weighted_average
B000003AEK	4.882353	102	4.792109
B00003002C	4.890110	91	4.789453
B000002WR5	4.924242	66	4.786277
B0000025BA	4.876289	97	4.782991
B000001A6N	4.884615	78	4.771168
...	...	...	...
B0009SCVTG	2.494253	87	2.781355
B00004XOWM	2.473214	112	2.707449
B0007NFL18	2.558824	272	2.658342
B0000CD5F1	1.921053	38	2.641109
B00006ZCFJ	1.842857	70	2.313344

Movie_ID	mean_ratings	no_of_reviews	weighted_average
B006W9KNXC	4.933835	665	4.903520
B004NSUXHU	4.890323	1085	4.872430
B0007N1BBC	4.924460	278	4.856043
B00006C7G9	4.870321	748	4.845240
B003TO541O	4.876712	511	4.840234
...	...	...	...
B00BTFK07I	1.425532	47	2.317254
0790732475	2.075000	320	2.213736
B001RIYVYK	1.619048	84	2.162269
B0007XBM5W	1.762712	118	2.151585
B0000YEE6C	1.655367	177	1.942911

These recommendations are prioritized based on max-to-min weighted average scores.