RESEARCH PAPER RECOMENDATTION SYSTEM USING K-MEANS CLUSTERING

18BIT0126

18BIT0150

18BIT0471

Submitted to

Prof. Geraldine Bessie Amali, SCOPE

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Contents

Abstract

HERE WE USE THE BOOKS DATASET INSTEAD OF PAPERS AS WE ARE UNABLE TO FIND A DATASET BASED ON RESEARCH PAPERS AND WE TRIED SCRAPPING BUT IT WAS OF NO USE. BUT THE PROCESS OF EXECUTION IS SAME FOR BOTH JUST DATASET CHANGES.

Now a days we are going towards the development of technology and bringing the A.I and M.L into scope of every aspect. Here in our project we propose an algorithm that helps the user to find the books that can be read by his from his previous ratings and from the ratings of other users.

Typically in all the areas we go for keyword search now a days but in our project its not keyword search its about we collect the data and cluster them based on the similarity of the users. After that we go for the book which the user was searching here the people who read maximum no of similar books are grouped in to a cluster so we can give better suggestions.

Recommendation systems are widely used to recommend products to the end users that are most appropriate. Online book selling websites now-a-days are competing with each other by many means. Recommendation system is one of the stronger tools to increase profit and retaining buyer. The book recommendation system must recommend books that are of buyer's interest.

INTRODUCTION

Recommendation systems were evolved as intelligent algorithms, which can generate results in the form of recommendations to users. They reduce the overhead associated with making best choices among the plenty. Now, Recommender systems can be implemented in any domain from E-commerce to network security in the form of personalized services. They provide benefit to both the consumer and the manufacturer, by suggesting items to consumers, which can't be demanded until the recommendations. Every recommender system comprises of two entities, one is user and other is item. A user can be any customer or consumer of any product or items, who get the suggestions. Input to recommendation algorithm can be a database of user and items and output obliviously will be the recommendations. As in our case, inputs consist of database of customers and database of books and output denotes the book recommendations.

In our project here we give the recommendations based on k-mean clustering.

HARDWARE AND SOFTWARE REQUIREMENTS

SOFTWARE:Anaconda 3 with jupiternotebook and python shell installed

HARDWARE:

- Need a storage device to be installed on your machine

- Ram should be of atleast 4gb

EXISTING SYSTEM METHOD

Present system is mainly based on keyword search that the books that are recommended to the user are of same authors or publishers it mainly comes under the keyword recommendation.
DRAWBACKS IN SYSTEM:

- The system will not give accurate recommendations.
- Cannot understand the user intention.
- Cannot find user favourites.

DESIGN

- Here we design the system so that it takes the data for training itself.
- And after that based on user searches it gives the recommendations.
- And it also recommend based on to which cluster the user belongs.
- Here we use skitlearn library to cluster the data.
- And in the processing based on the search we go through the clusters to find the book searched and then we recommend the books of that cluter.

In [14]:
```
#mam we had used the books instead of the reasearch papers as we cannot find t
he relavent dataset.
#but the processing of data and execution is same for both so we considered bo
oks dataset.
# IMPORTING THE LIBRARIES THAT ARE ALL NEEDED
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.sparse import csr_matrix
from mpl_toolkits.axes_grid1 import make_axes_locatable
from sklearn.cluster import KMeans
from sklearn.metrics import mean_squared_error
import itertools
# here we import the kmeans which help us to cluster the papers
from sklearn.cluster import KMeans
#here the silhouette_score helps us to effectively define the clusters
from sklearn.metrics import silhouette_samples, silhouette_score
%matplotlib inline
#here we read the papers dataset from the given directory
papers = pd.read_csv(r'C:\Users\narur\Desktop\fpapers.csv')
papers.head(10)
```

Out[14]:

	paperId	title	genres
0	1	The Hunger Games (The Hunger Games, #1)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Harry Potter and the Sorcerer's Stone (Harry P...	Adventure|Children|Fantasy
2	3	Twilight (Twilight, #1)	Comedy|Romance
3	4	To Kill a Mockingbird	Comedy|Drama|Romance
4	5	The Great Gatsby	Comedy
5	6	The Fault in Our Stars	Action|Crime|Thriller
6	7	The Hobbit	Comedy|Romance
7	8	The Catcher in the Rye	Adventure|Children
8	9	Angels & Demons (Robert Langdon, #1)	Action
9	10	Pride and Prejudice	Action|Adventure|Thriller

In [15]:
```
#here we read the ratings dataset from the given directory
ratings = pd.read_csv(r'C:\Users\narur\Desktop\pratings.csv')
#here we print the top most entries in the dataset
ratings.head(10)
```

Out[15]:

	userId	paperId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
5	1	70	3.0	964982400
6	1	101	5.0	964980868
7	1	110	4.0	964982176
8	1	151	5.0	964984041
9	1	157	5.0	964984100

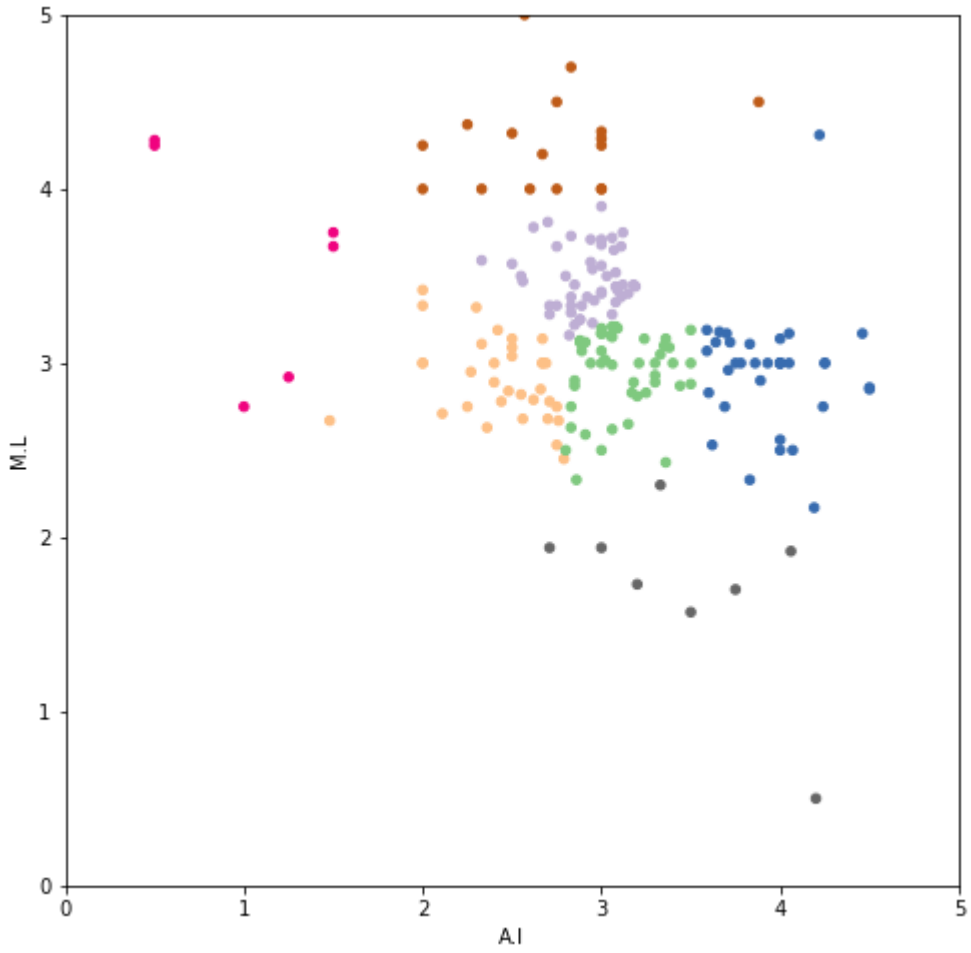
In [16]:
```
def draw_clusters(biased_dataset, predictions, cmap='viridis'):
    fig = plt.figure(figsize=(8,8))
    ax = fig.add_subplot(111)
    #defining the x-limit and y-limit in the graph
    plt.xlim(0, 5)
    plt.ylim(0, 5)
    #labeling the axis
    ax.set_xlabel('A.I')
    ax.set_ylabel('M.L')
    #concatination along the axis of the data pd.concat
    #arthematic align of rows and columns in the dataset pd.dataframe
    #reseting the indexes reset_index()
    #scatter plot by the matlab function plt.scatter()
    clustered = pd.concat([biased_dataset.reset_index(), pd.DataFrame({'group'
:predictions})], axis=1)
    plt.scatter(clustered['avg_scifi_rating'], clustered['avg_romance_rating'
], c=clustered['group'], s=20, cmap=cmap)


def draw_paper_clusters(clustered, max_users, max_papers):
    for cluster_id in clustered.group.unique():
        # To improve visibility, we're showing at most max_users users and max
_papers papers per cluster.
        d = clustered[clustered.group == cluster_id].drop(['index', 'group'],
axis=1)
        #If Y has n rows and m columns, then Y.shape is (n,m). So Y.shape[0] i
s n.
        n_users_in_cluster = d.shape[0]
        #sorting the papers by rating density papers rated most are top of clu
ster
        d = sort_by_rating_density(d, max_papers, max_users)
        #reindexing by the help of mean values
        d = d.reindex(d.mean().sort_values(ascending=False).index, axis=1)
        d = d.reindex(d.count(axis=1).sort_values(ascending=False).index)
#getting upto max_users and max_papers into d
#.iloc[] is primarily integer position based (from 0 to length-1 of the axis),
but may also be used with a boolean array.
        d = d.iloc[:max_users, :max_papers]
        n_users_in_plot = d.shape[0]
# We're  selecting to show all clusters to have some restriction we can have l
ike len(d)>n where n is no of users in cluster
        if len(d) >0 :
            print('cluster # {}'.format(cluster_id))
            print('# of users in cluster: {}.'.format(n_users_in_cluster), '#
 of users in plot: {}'.format(n_users_in_plot))
            #figure outlook makings
            fig = plt.figure(figsize=(15,4))
            ax = plt.gca()
            ax.invert_yaxis()
            ax.xaxis.tick_top()
            labels = d.columns.str[:40]
#set_yticks takes 2 arguments 1-y labels y_tick locations 2-minor:bool,optiona
l:: if True sets minor ticks. Default is False.
#set_yticklables takes 2 arguments 1-lables 2-minor:bool,optional:: if True se
ts minor ticks. Default is False.
            ax.set_yticks(np.arange(d.shape[0]) , minor=False)
```

```
            ax.set_xticks(np.arange(d.shape[1]) , minor=False)
            ax.set_xticklabels(labels, minor=False)
            ax.get_yaxis().set_visible(False)
            # Heatmap plotting of the clusters
            heatmap = plt.imshow(d, vmin=0, vmax=5, aspect='auto')
            #labeling of the axis
            ax.set_xlabel('papers')
            ax.set_ylabel('User id')
            #divider making at the outline
            divider = make_axes_locatable(ax)
            #clour axis dividion allocation
            cax = divider.append_axes("right", size="5%", pad=0.05)
            # Color bar divided from the heat map
            cbar = fig.colorbar(heatmap, ticks=[5, 4, 3, 2, 1, 0], cax=cax)
            #colour axis labels or ticklabels
            cbar.ax.set_yticklabels(['5 stars', '4 stars','3 stars','2 stars',
'1 stars','0 stars'])
            plt.setp(ax.get_xticklabels(), rotation=90, fontsize=9)
            plt.tick_params(axis='both', which='both', bottom='off', top='off'
, left='off', labelbottom='off', labelleft='off')
            plt.show()

def get_most_rated_papers(user_paper_ratings, max_number_of_papers):
    # 1- Count =appending the no of users for each paper
    user_paper_ratings = user_paper_ratings.append(user_paper_ratings.count(),
ignore_index=True)
    # 2- sorting the papers ratings
    #.drop() function in Pandas be used to delete rows from a DataFrame, with
 the axis set to 0
    user_paper_ratings_sorted = user_paper_ratings.sort_values(len(user_paper_
ratings)-1, axis=1, ascending=False)
    user_paper_ratings_sorted = user_paper_ratings_sorted.drop(user_paper_rati
ngs_sorted.tail(1).index)
    # 3- slice getting the required no of papers from the array
    most_rated_papers = user_paper_ratings_sorted.iloc[:, :max_number_of_paper
s]
    return most_rated_papers

def get_users_who_rate_the_most(most_rated_papers, max_number_of_papers):
    # Get most voting users
    # 1- Count= count the users who rated most of the papers
    #converting into a one-dimensional labeled array to store any type of data
    most_rated_papers['counts'] = pd.Series(most_rated_papers.count(axis=1))
    # 2- Sort=sorting the users by user ratings values(counts)
    most_rated_papers_users = most_rated_papers.sort_values('counts', ascendin
g=False)
    # 3- Slice=selecting the max_users from the array(selecting the max_papers
for operation)
    most_rated_papers_users_selection = most_rated_papers_users.iloc[:max_numb
er_of_papers, :]
    #.drop() function in Pandas be used to delete rows from a DataFrame, with
 the axis set to 0
    most_rated_papers_users_selection = most_rated_papers_users_selection.drop
(['counts'], axis=1)
    return most_rated_papers_users_selection

#function to sort by density i.e returns most_rated_papers are papers which ar
```

```
e rated most and useres who rate most
def sort_by_rating_density(user_paper_ratings, n_papers, n_users):
    #calling the functions defined above
    most_rated_papers = get_most_rated_papers(user_paper_ratings, n_papers)
    most_rated_papers = get_users_who_rate_the_most(most_rated_papers, n_users
)
    return most_rated_papers

def get_genre_ratings(ratings, papers, genres, column_names):
    genre_ratings = pd.DataFrame()
    for genre in genres:
        genre_papers = papers[papers['genres'].str.contains(genre) ]
        avg_genre_votes_per_user = ratings[ratings['paperId'].isin(genre_paper
s['paperId'])].loc[:, ['userId', 'rating']].groupby(['userId'])['rating'].mean
().round(2)
        genre_ratings = pd.concat([genre_ratings, avg_genre_votes_per_user], a
xis=1)
    genre_ratings.columns = column_names
    return genre_ratings

def bias_genre_rating_dataset(genre_ratings, score_limit_1, score_limit_2):
    biased_dataset = genre_ratings[((genre_ratings['avg_romance_rating'] < sco
re_limit_1) & (genre_ratings['avg_scifi_rating'] > score_limit_2)) | ((genre_r
atings['avg_scifi_rating'] < score_limit_1) & (genre_ratings['avg_romance_rati
ng'] > score_limit_2))]
    biased_dataset = pd.concat([biased_dataset[:300], genre_ratings[:2]])
    biased_dataset = pd.DataFrame(biased_dataset.to_records())
    return biased_dataset
```

In [17]:
```
genre_ratings = get_genre_ratings(ratings, papers, ['Romance', 'Sci-Fi'], ['av
g_romance_rating', 'avg_scifi_rating'])
# Bias the dataset with high bound=3.2 and low bound=2.5
biased_dataset = bias_genre_rating_dataset(genre_ratings, 3.2, 2.5)
# Printing the resulting number of records & the head of the dataset
print( "Number of records: ", len(biased_dataset))
#printing the head values or topmost values from the biased_datset
print(biased_dataset.head(10))
#x is the set of avg_scifi and roamnce ratings biased means enforced some rule
s
X = biased_dataset[['avg_scifi_rating','avg_romance_rating']].values
# Create an instance of KMeans to find seven clusters
kmeans_2 = KMeans(n_clusters=7)
# Use fit_predict to cluster the dataset
predictions_2 = kmeans_2.fit_predict(X)
draw_clusters(biased_dataset, predictions_2, cmap='Accent')
#cmap=colour map consists of accent,virdis,etc...
#kmeans is a package in the sklearn
```

```
Number of records:  193
   index  avg_romance_rating  avg_scifi_rating
0      3                0.50              4.20
1      4                3.38              2.83
2      5                3.09              2.50
3      7                2.65              3.15
4      9                3.17              3.00
5     10                3.33              2.00
6     19                2.68              2.56
7     26                3.00              2.00
8     28                2.89              3.18
9     34                3.10              3.35
```



In [18]:
```
# Merge the two tables then pivot so we have Users X papers dataframe
ratings_title = pd.merge(ratings, papers[['paperId', 'title']], on='paperId' )
# pd.pivot_table()= create a spreadsheet-style pivot table as a DataFrame.
user_paper_ratings = pd.pivot_table(ratings_title, index='userId', columns= 't
itle', values='rating')
# Print he number of dimensions and a subset of the dataset
print('dataset dimensions: ', user_paper_ratings.shape, '\n\nSubset example:')
#printing the subset of the user_paper_ratings
user_paper_ratings.iloc[:6, :10]
```

```
dataset dimensions:  (610, 9690)

Subset example:
```

Out[18]:

title	Angels (Walsh Family, #3)	"حكايات فرغلي المستكاوي "حكايتى مع كفر السحلاوية	#GIRLBOSS	'Salem's Lot	'Tis (Frank McCourt, #2)	1,000 Places to See Before You Die	1/4 جرام	10% Happier: How I Tamed the Voice in My Head, Reduced Stress Without Losing My Edge, and Found Self-Help That Actually Works	100 Bullets, Vol. 1: First Shot, Last Call	L Sonn
userId										
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

In [19]:
```
n_papers = 30
n_users = 18
#sort_by_rating_density function call to get the most_rated_papers_users_selection data
most_rated_papers_users_selection = sort_by_rating_density(user_paper_ratings, n_papers, n_users)
# Print the result
print('dataset dimensions: ', most_rated_papers_users_selection.shape)
#printing the top most in the most_rated_papers_users_selection.head()
most_rated_papers_users_selection.head()
```

```
dataset dimensions:  (18, 30)
```

Out[19]:

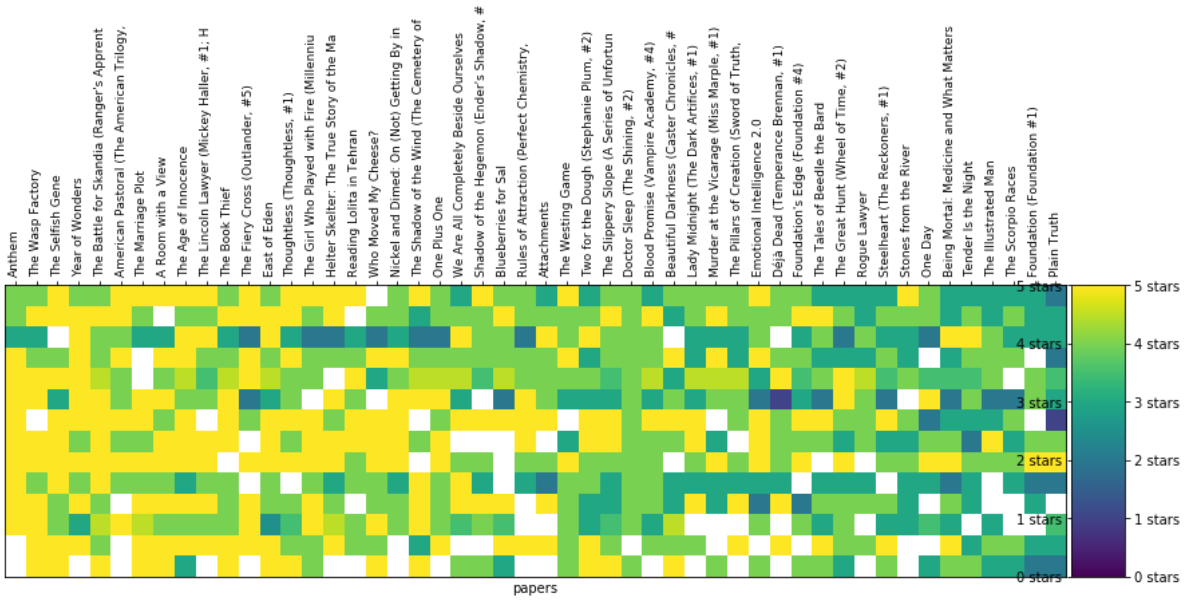
title	Who Moved My Cheese?	A Light in the Attic	The Shadow of the Wind (The Cemetery of Forgotten Books, #1)	Postmortem (Kay Scarpetta, #1)	The Wasp Factory	East of Eden	Blood Promise (Vampire Academy, #4)	The Girl Who Played with Fire (Millennium, #2)	Nickel and Dimed: On (Not) Getting By in America
413	5.0	5.0	5.0	4.0	5.0	5.0	4.0	5.0	5.0
589	5.0	4.5	4.5	3.5	4.0	5.0	4.0	4.0	4.5
473	3.0	5.0	4.0	4.5	4.5	4.0	4.5	3.0	4.0
479	5.0	5.0	4.0	4.5	5.0	4.5	5.0	5.0	4.5
67	3.5	3.0	2.0	3.5	4.5	5.0	3.5	2.5	3.5

5 rows × 30 columns

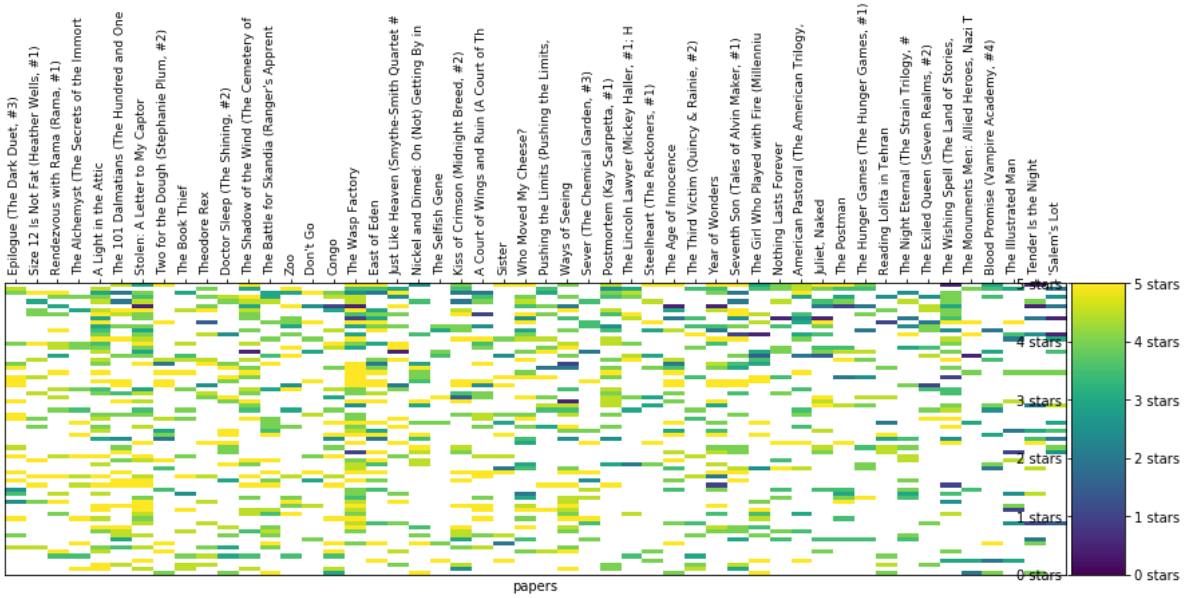
In [20]:
```
#this below line was from above cell
user_paper_ratings =  pd.pivot_table(ratings_title, index='userId', columns=
'title', values='rating')
total=1000
#getting the thousand papers that are rated most by the users
most_rated_papers_1k = get_most_rated_papers(user_paper_ratings, total)
#creating the sparse matrix for most_paper_ratings
sparse_ratings = csr_matrix(pd.SparseDataFrame(most_rated_papers_1k).to_coo())
#no of clusters to be made as input to n
n=20
#kmeans clustering
predictions = KMeans(n_clusters=n, algorithm='full').fit_predict(sparse_rating
s)
max_users = 70
max_papers = 50
#concatination along the axis of the data pd.concat
#arthematic align of rows and columns in the dataset pd.dataframe
clustered = pd.concat([most_rated_papers_1k.reset_index(), pd.DataFrame({'grou
p':predictions})], axis=1)
#drawing the clusters with max_users and max_papers
draw_paper_clusters(clustered, max_users, max_papers)
```

```
C:\Users\narur\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: FutureWar
ning: SparseDataFrame is deprecated and will be removed in a future version.
Use a regular DataFrame whose columns are SparseArrays instead.

See http://pandas.pydata.org/pandas-docs/stable/user_guide/sparse.html#migrat
ing for more.

  import sys
C:\Users\narur\Anaconda3\lib\site-packages\pandas\core\frame.py:3456: FutureW
arning: SparseSeries is deprecated and will be removed in a future version.
Use a Series with sparse values instead.

    >>> series = pd.Series(pd.SparseArray(...))

See http://pandas.pydata.org/pandas-docs/stable/user_guide/sparse.html#migrat
ing for more.

  return klass(values, index=self.index, name=items, fastpath=True)
```

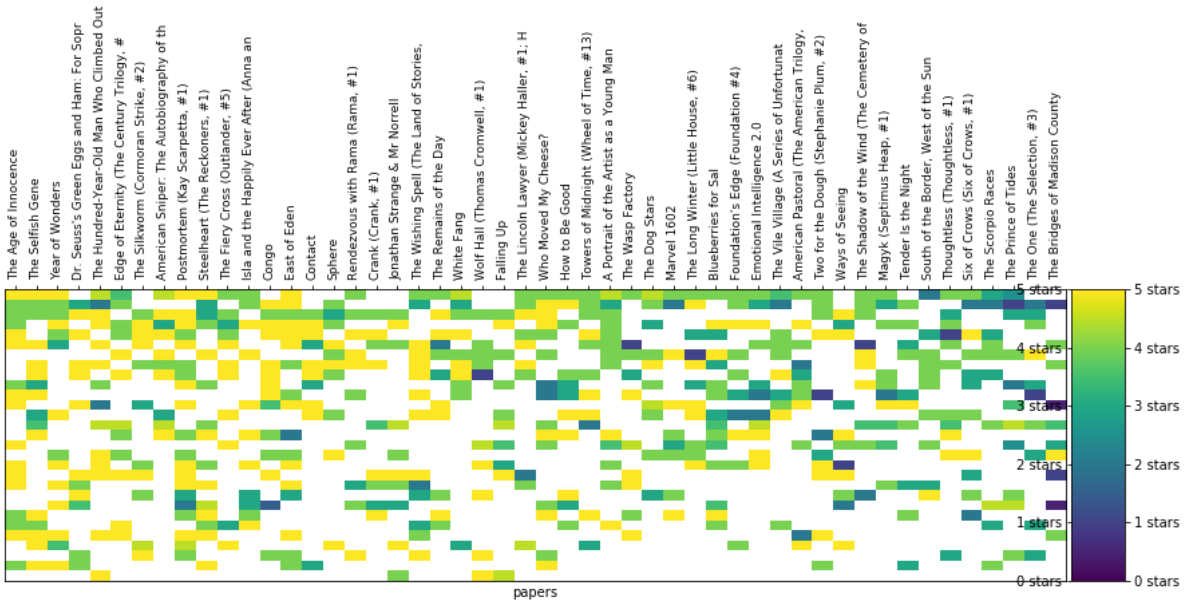
```
cluster # 4
# of users in cluster: 14. # of users in plot: 14
```



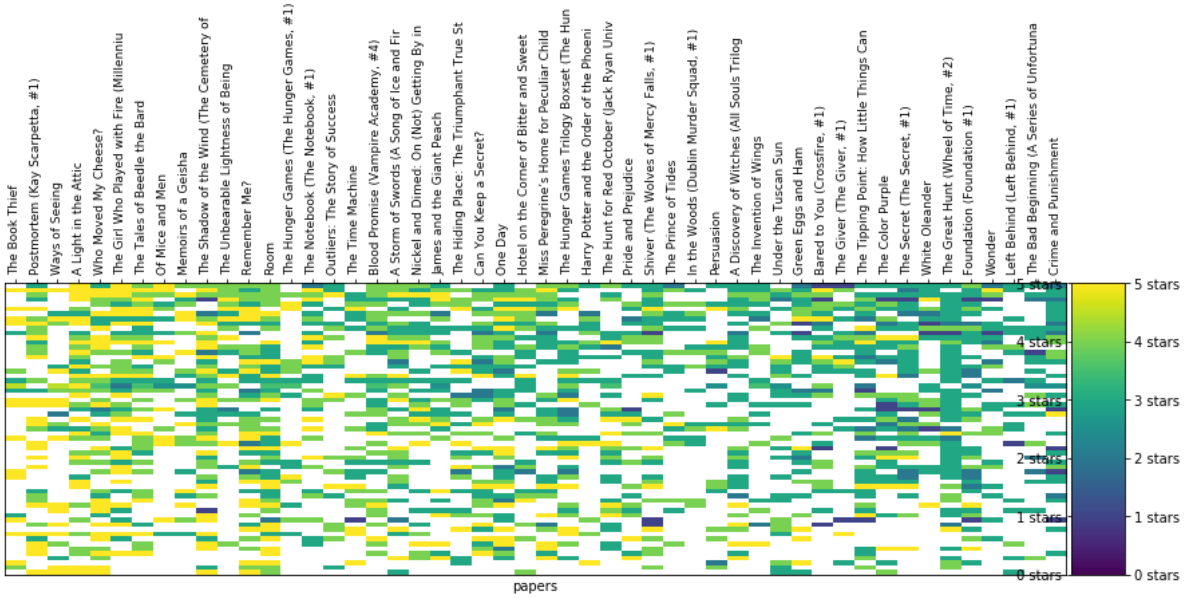

```
cluster # 10
# of users in cluster: 244. # of users in plot: 70
```



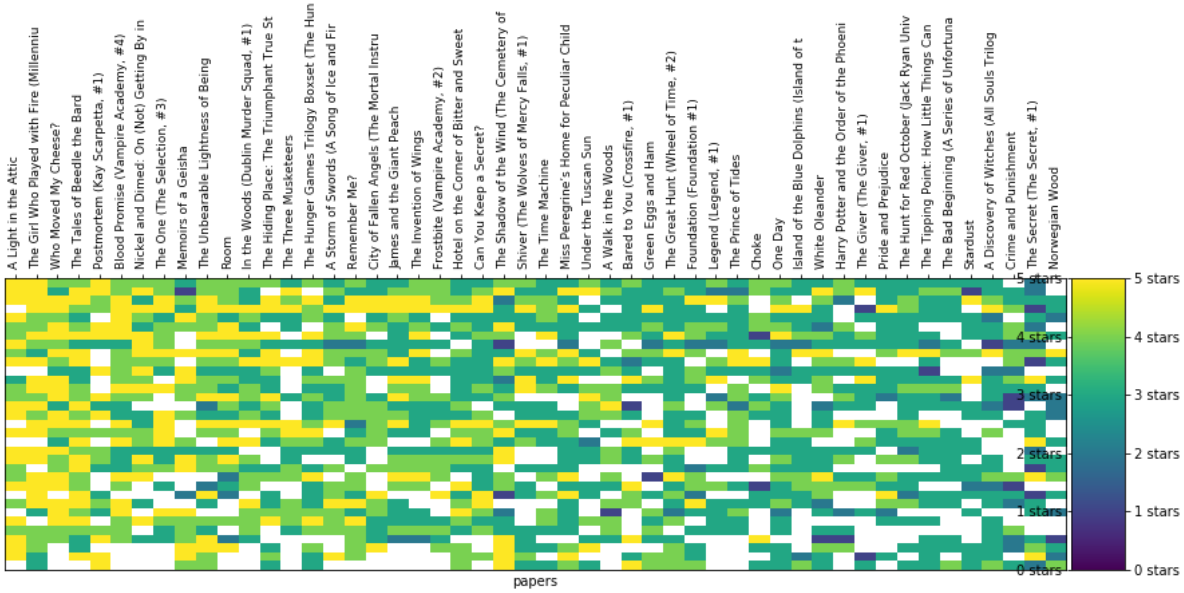
cluster # 5
# of users in cluster: 29. # of users in plot: 29



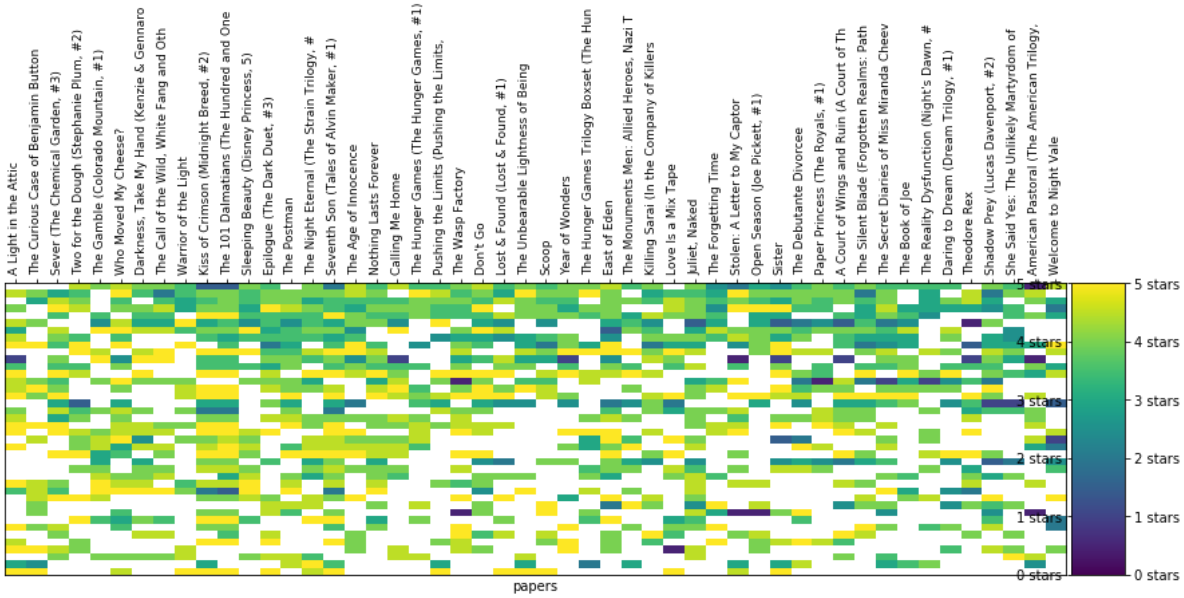
cluster # 18
# of users in cluster: 61. # of users in plot: 61



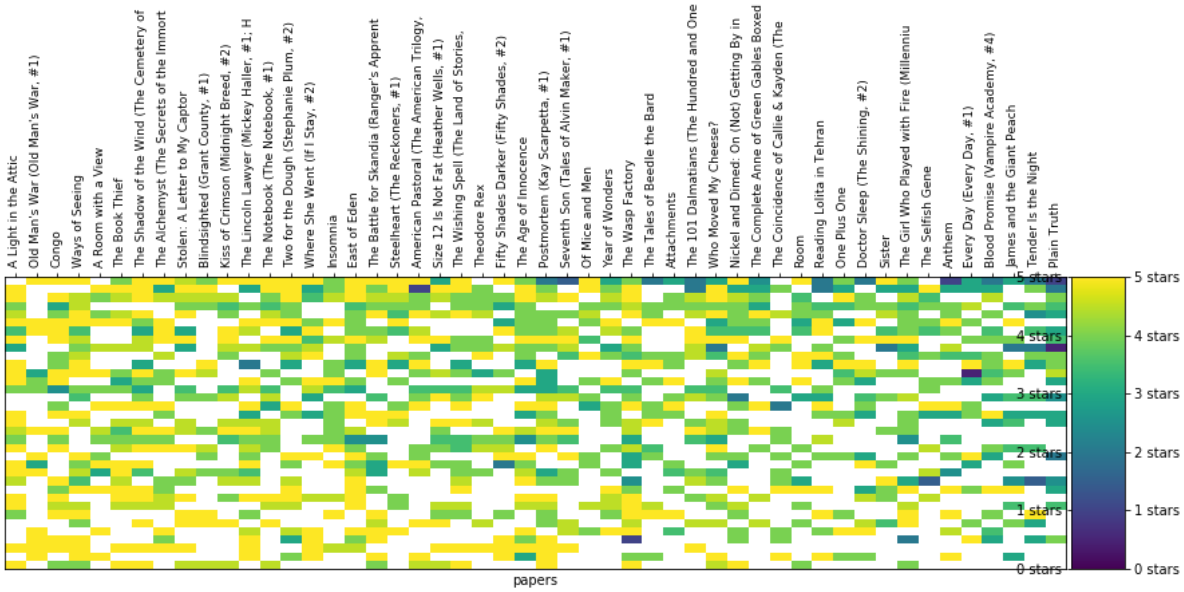
cluster # 6
# of users in cluster: 33. # of users in plot: 33



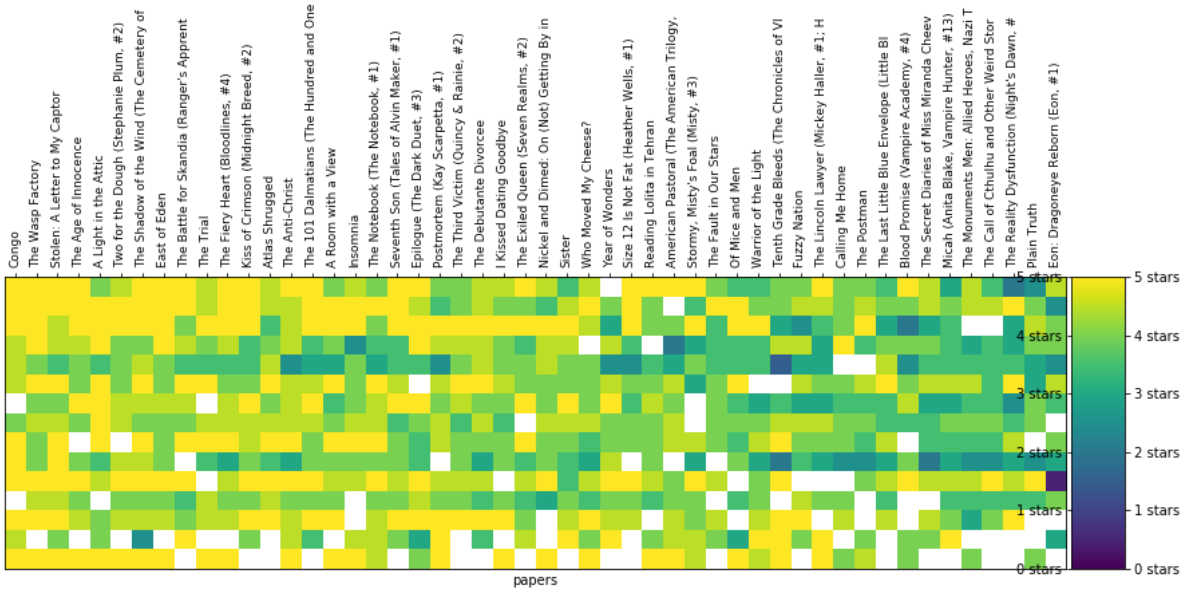
cluster # 16
# of users in cluster: 40. # of users in plot: 40



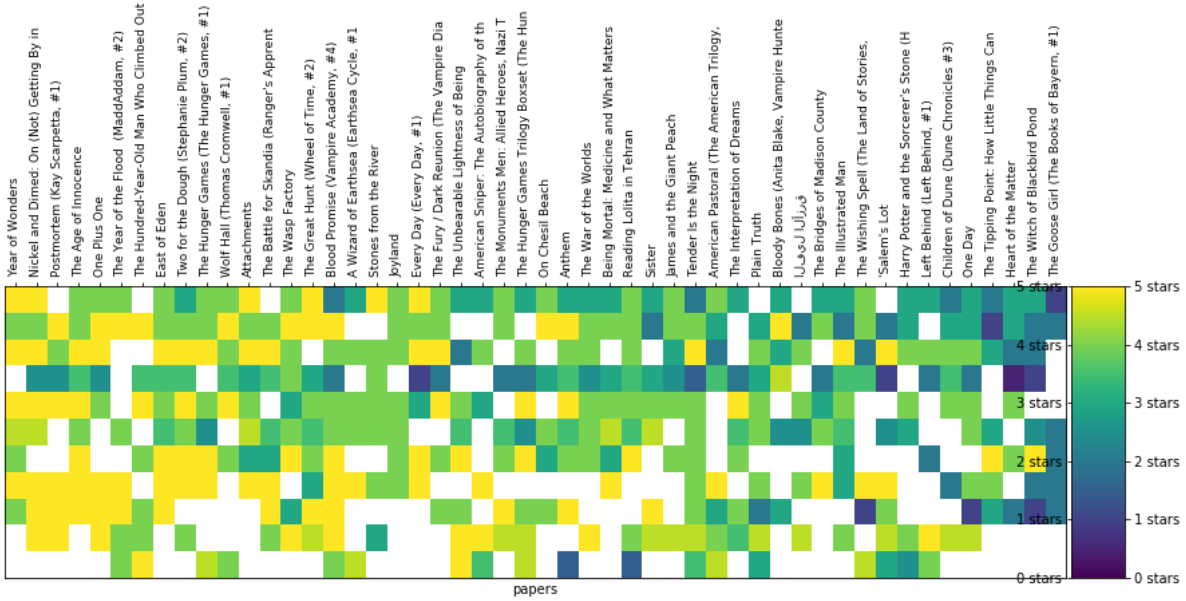
cluster # 19
# of users in cluster: 35. # of users in plot: 35



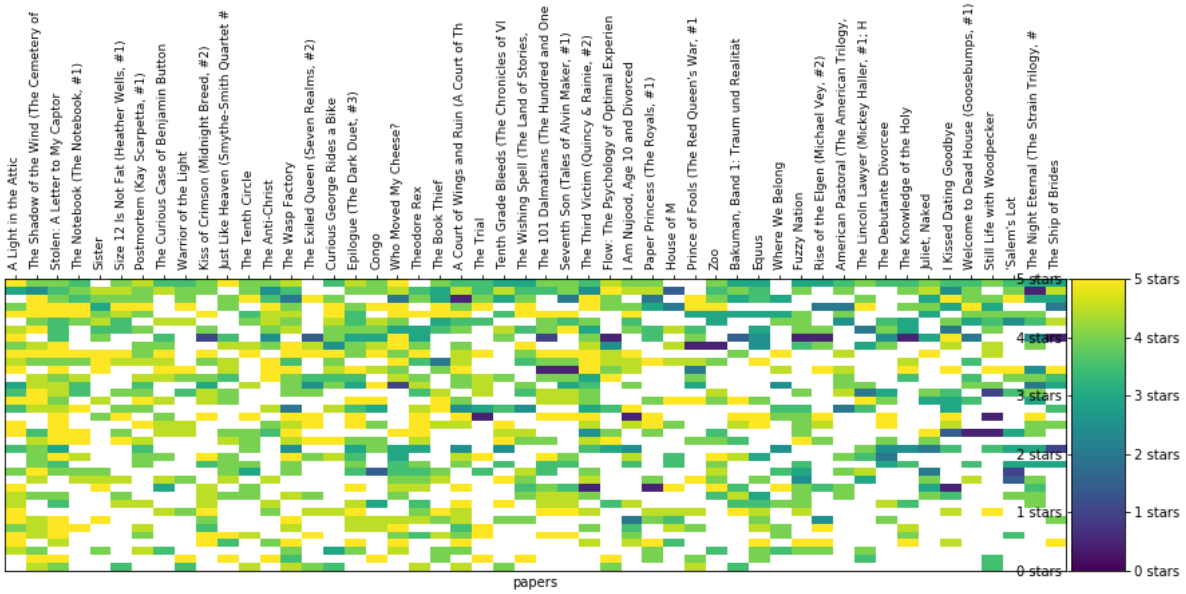
cluster # 2
# of users in cluster: 15. # of users in plot: 15



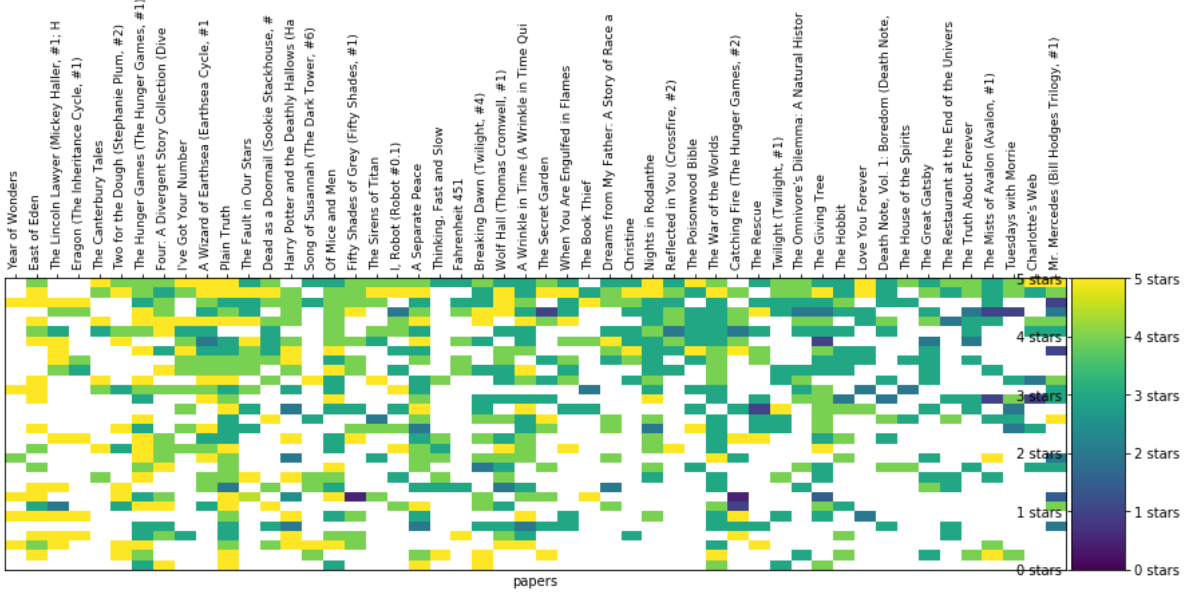

```
cluster # 3
# of users in cluster: 11. # of users in plot: 11
```



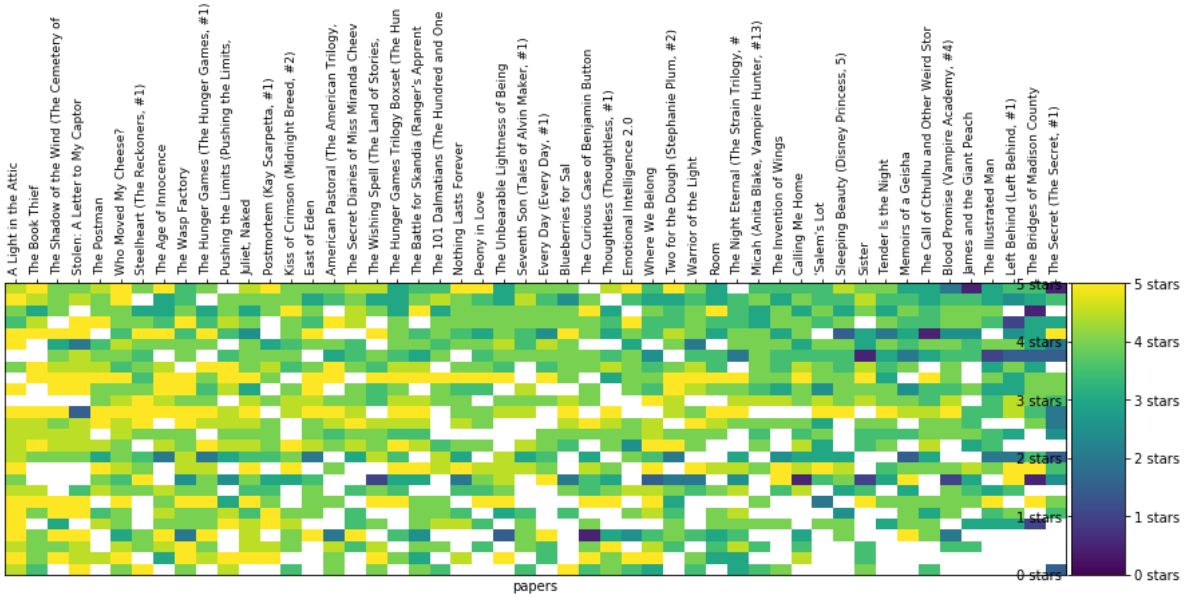

```
cluster # 1
# of users in cluster: 37. # of users in plot: 37
```



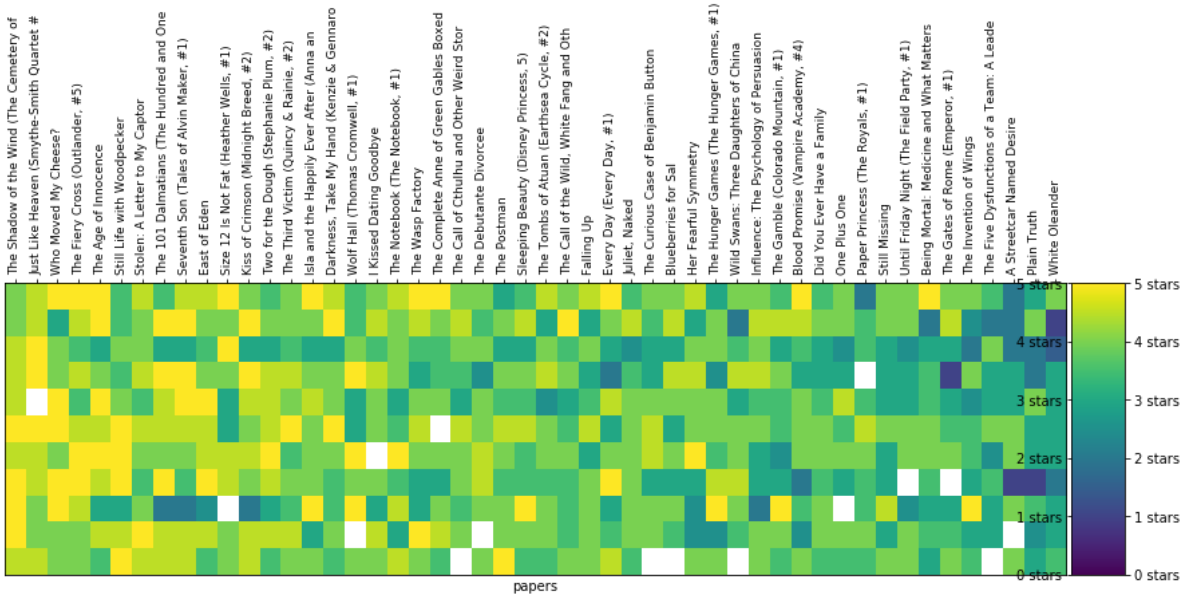
cluster # 8
# of users in cluster: 30. # of users in plot: 30



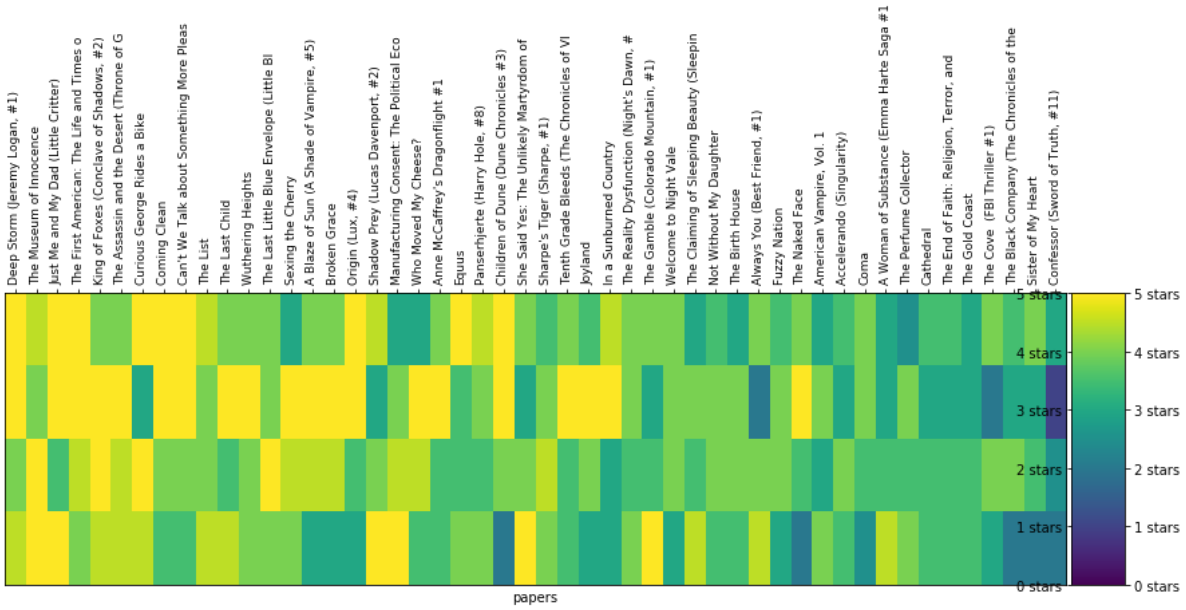
cluster # 7
# of users in cluster: 26. # of users in plot: 26



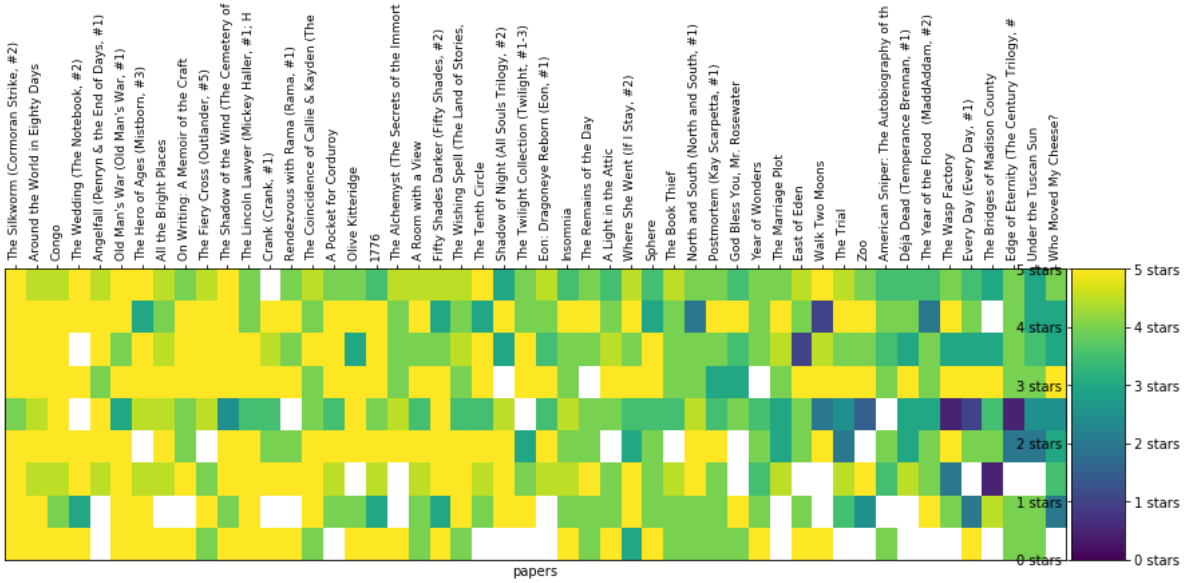
cluster # 12
# of users in cluster: 11. # of users in plot: 11



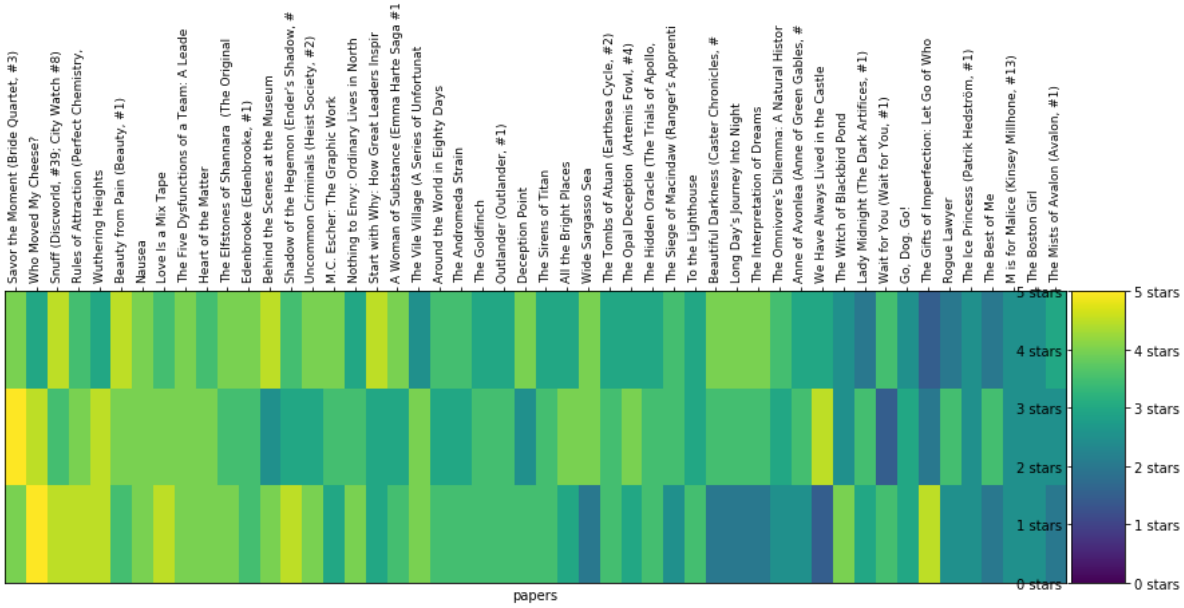
cluster # 13
# of users in cluster: 4. # of users in plot: 4



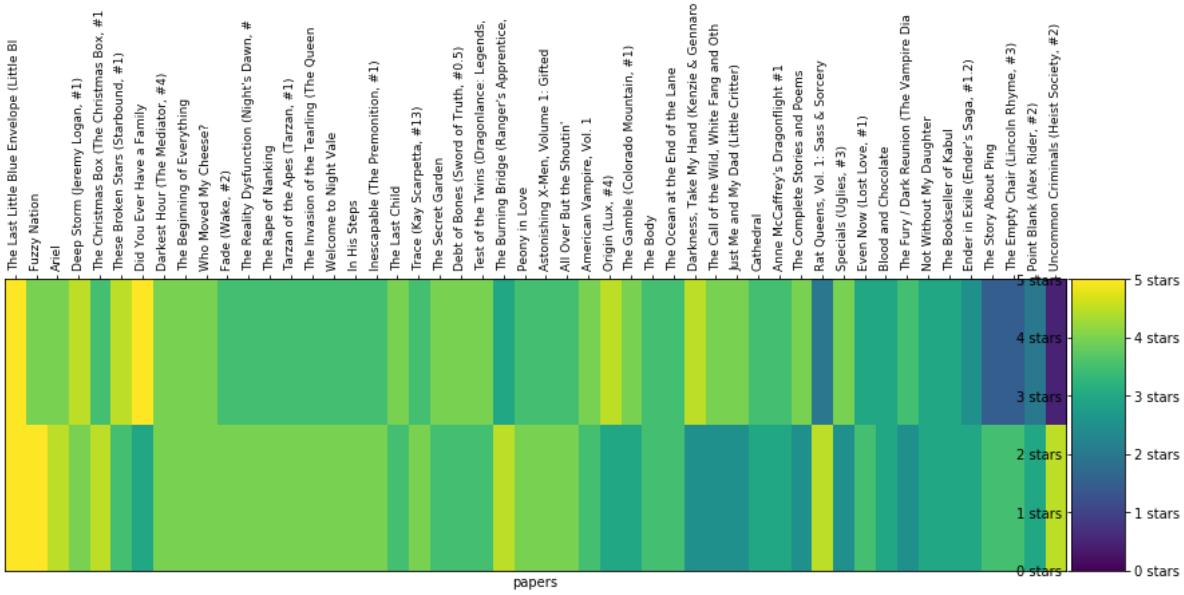
cluster # 14
# of users in cluster: 9. # of users in plot: 9



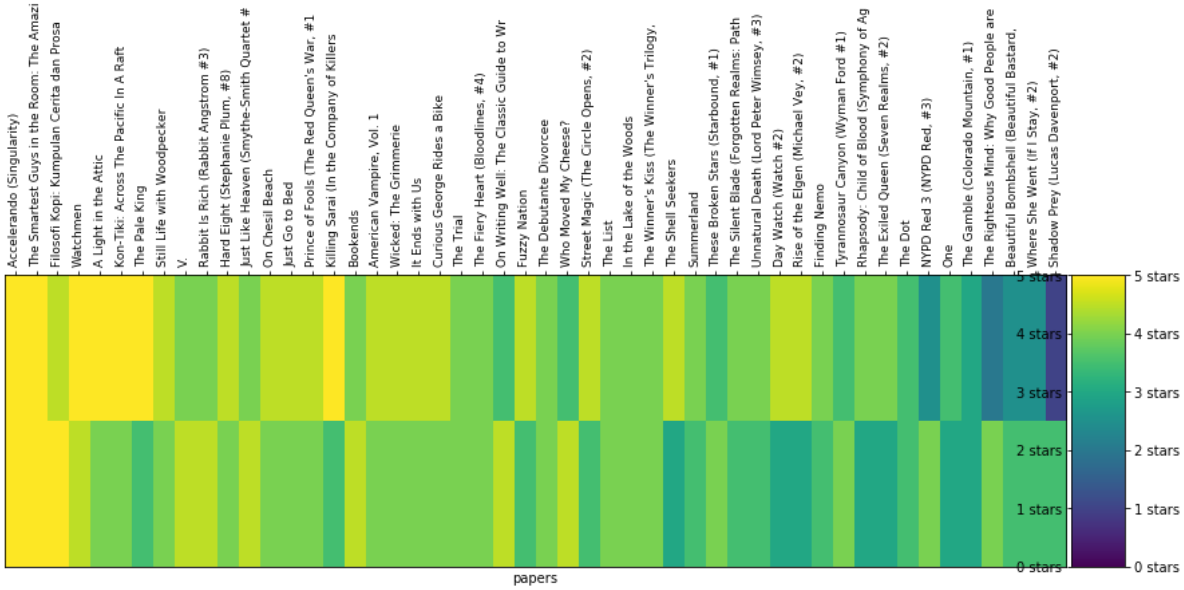
cluster # 15
# of users in cluster: 3. # of users in plot: 3



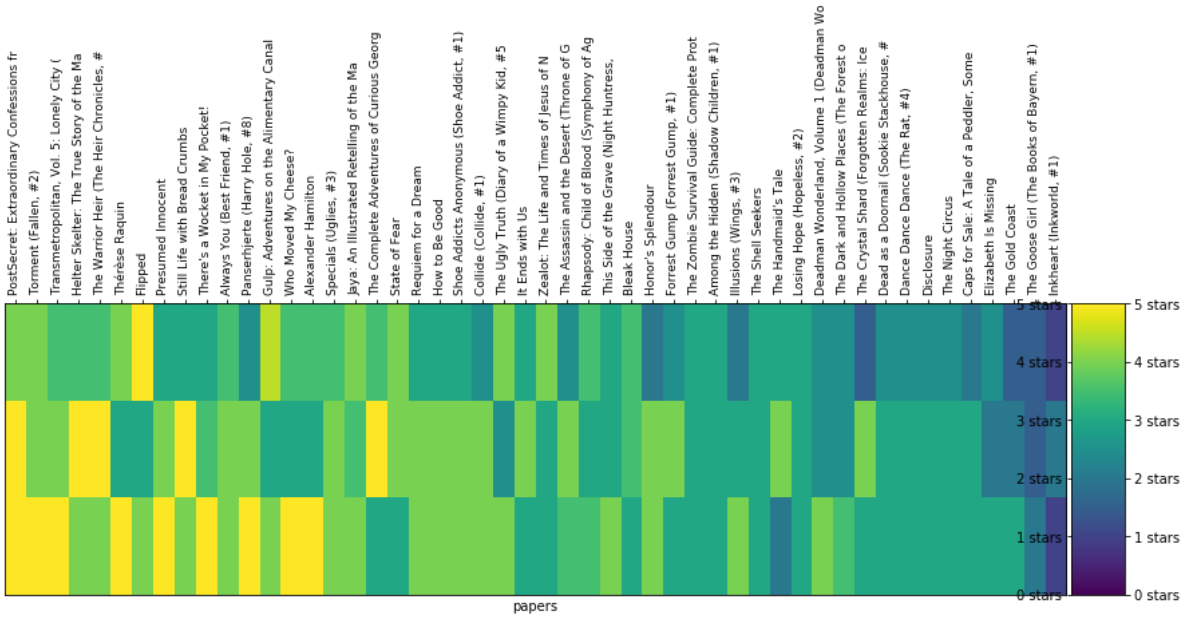
cluster # 0
# of users in cluster: 2. # of users in plot: 2



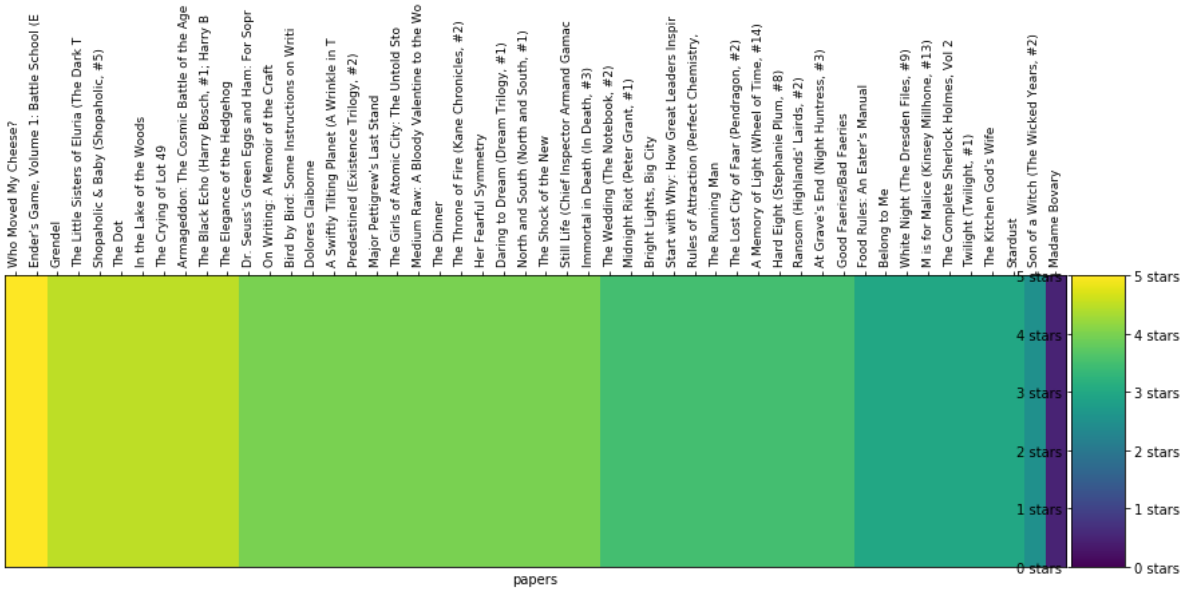
cluster # 17
# of users in cluster: 2. # of users in plot: 2




```
cluster # 11
# of users in cluster: 3. # of users in plot: 3
```




```
cluster # 9
# of users in cluster: 1. # of users in plot: 1
```



In [21]:
```
paper_name = "Ariel"
```

In [22]:
```
cl=0
ocl=0
while cl<n :
    n_users = 70
    n_papers = 50
    cluster = clustered[clustered.group == cl].drop(['index', 'group'], axis=1
)
    cluster = sort_by_rating_density(cluster, n_papers, n_users)
    lantren=cluster.columns.str[:100]
    for j in range(len(lantren)) :
        if lantren[j] == paper_name :
            print("THE CLUSTERS ARE NOT MUTUALLY EXCLUSIVE ONE PAPER CAN BE IN
DIFFERENT CLUSTERS :)....\n ")
            print("THE CLUSTER ID THAT CONTAINS THE paper NAME IS\n")
            ocl=cl
            k=j
            cl=n
            j=len(lantren)
            break
        else :
            continue
    cl=cl+1
print(ocl,"cluster \n")
print("THE INDEX OF THE paper IN THE CLUSTER #",ocl,"IS\n",k)
```

```
THE CLUSTERS ARE NOT MUTUALLY EXCLUSIVE ONE PAPER CAN BE IN DIFFERENT CLUSTER
S :)....

THE CLUSTER ID THAT CONTAINS THE paper NAME IS

0 cluster

THE INDEX OF THE paper IN THE CLUSTER # 0 IS
 24
```

In [23]:
```
cluster = clustered[clustered.group == ocl].drop(['index', 'group'], axis=1)
cluster = sort_by_rating_density(cluster, n_papers, n_users)
tclus=cluster.columns.str[:100]
print("the papers average rating by the users in the cluster ",ocl,"=",cluster
[paper_name].mean())
print("the paper name in the cluster",ocl,"at the index",k,"is  [**[",tclus[k
],"]**]")
```

```
the papers average rating by the users in the cluster  0 = 4.25
the paper name in the cluster 0 at the index 24 is  [**[ Ariel ]**]
```

In [24]:
```
print("you may also like to cite\n ")
for i in range(k-10,k+10) :
    if i<len(tclus) and i!=k :
        print(tclus[i],"[",cluster[tclus[i]].mean(),"]")
print("\nThe top rated papers by all the users who are similar to you\n")
print(cluster.mean().head(25))
```

you may also like to cite

The Last Child [3.75]
The Story About Ping [2.5]
Trace (Kay Scarpetta, #13) [3.75]
The Fury / Dark Reunion (The Vampire Diaries, #3-4) [3.0]
The Christmas Box (The Christmas Box, #1) [4.0]
Rat Queens, Vol. 1: Sass & Sorcery [3.25]
Debt of Bones (Sword of Truth, #0.5) [3.75]
These Broken Stars (Starbound, #1) [4.0]
Even Now (Lost Love, #1) [3.25]
The Bookseller of Kabul [3.0]
The Call of the Wild, White Fang and Other Stories [3.25]
Blood and Chocolate [3.0]
Astonishing X-Men, Volume 1: Gifted [3.75]
Specials (Uglies, #3) [3.25]
The Empty Chair (Lincoln Rhyme, #3) [2.5]
American Vampire, Vol. 1 [3.75]
Darkest Hour (The Mediator, #4) [4.0]
The Beginning of Everything [4.0]
Uncommon Criminals (Heist Society, #2) [2.5]

The top rated papers by all the users who are similar to you

Who Moved My Cheese?	4.00
Point Blank (Alex Rider, #2)	2.50
The Body	3.50
The Rape of Nanking	3.75
The Ocean at the End of the Lane	3.50
Just Me and My Dad (Little Critter)	3.25
The Last Little Blue Envelope (Little Blue Envelope, #2)	5.00
Cathedral	3.25
Deep Storm (Jeremy Logan, #1)	4.25
The Invasion of the Tearling (The Queen of the Tearling, #2)	3.75
The Complete Stories and Poems	3.25
Welcome to Night Vale	3.75
In His Steps	3.75
Inescapable (The Premonition, #1)	3.75
The Last Child	3.75
The Story About Ping	2.50
Trace (Kay Scarpetta, #13)	3.75
The Fury / Dark Reunion (The Vampire Diaries, #3-4)	3.00
The Christmas Box (The Christmas Box, #1)	4.00
Rat Queens, Vol. 1: Sass & Sorcery	3.25
Debt of Bones (Sword of Truth, #0.5)	3.75
These Broken Stars (Starbound, #1)	4.00
Even Now (Lost Love, #1)	3.25
The Bookseller of Kabul	3.00
Ariel	4.25

dtype: float64

In [25]:
```
#when we want to give suggestions based on the userid.......
#its simple cluster the user with others and then find the cluster containing
 the userid
#then extract the papers in the cluster
user_id = 6
ccl=0
oocl=0
while ccl<n :
    n_users = 70
    n_papers = 50
    cluster = clustered[clustered.group == ccl].drop(['index', 'group'], axis=
1)
    indo=cluster.iloc[:70,:0]
    z=len(indo)
    for j in range(z) :
        if indo.index[j] == user_id :
            oocl=ccl
            kit=j
            ccl=n
            j=len(indo)
            print("THE USER_ID IS",user_id)
            break
        else :
            continue
    ccl=ccl+1
print("THE CLUSTER ID THAT CONTAINS THE USER_ID IS=",oocl,"th cluster\n")
print("THE INDEX OF THE USER IN THE CLUSTER #",oocl,"IS=",kit,"\n")
```

```
THE USER_ID IS 6
THE CLUSTER ID THAT CONTAINS THE USER_ID IS= 16 th cluster

THE INDEX OF THE USER IN THE CLUSTER # 16 IS= 0
```

In [26]:
```
cluster = clustered[clustered.group == oocl].drop(['index', 'group'], axis=1)
indo=cluster.iloc[:70,:0]
print(indo.index[1])
cluster = sort_by_rating_density(cluster, n_papers, n_users)
ttclus=cluster.columns.str[:100]
print("you may also like to watch\n ")
for i in range(kit-8,kit+8) :
    if i<len(ttclus) and i!=kit :
        print(ttclus[i],"[",cluster[ttclus[i]].mean(),"]")
print("\nThe top rated papers by all the users who are similar to you\n")
print(cluster.mean().head(15))
```

```
9
you may also like to watch

Warrior of the Light [ 4.055555555555555 ]
The Hunger Games (The Hunger Games, #1) [ 3.9722222222222223 ]
The Hunger Games Trilogy Boxset (The Hunger Games, #1-3) [ 3.888888888888889
]
Open Season (Joe Pickett, #1) [ 3.75 ]
The Curious Case of Benjamin Button [ 4.25 ]
Pushing the Limits (Pushing the Limits, #1) [ 3.9705882352941178 ]
Daring to Dream (Dream Trilogy, #1) [ 3.5588235294117645 ]
The Reality Dysfunction (Night's Dawn, #1) [ 3.588235294117647 ]
Seventh Son (Tales of Alvin Maker, #1) [ 4.014285714285714 ]
Kiss of Crimson (Midnight Breed, #2) [ 4.03030303030303 ]
Juliet, Naked [ 3.8225806451612905 ]
East of Eden [ 3.8793103448275863 ]
Who Moved My Cheese? [ 4.089285714285714 ]
The Wasp Factory [ 3.9642857142857144 ]
The Secret Diaries of Miss Miranda Cheever (Bevelstoke, #1) [ 3.6666666666666
665 ]

The top rated papers by all the users who are similar to you

The 101 Dalmatians (The Hundred and One Dalmatians, #1)
4.028571
Seventh Son (Tales of Alvin Maker, #1)
4.014286
Kiss of Crimson (Midnight Breed, #2)
4.030303
Juliet, Naked
3.822581
East of Eden
3.879310
Who Moved My Cheese?
4.089286
The Wasp Factory
3.964286
The Secret Diaries of Miss Miranda Cheever (Bevelstoke, #1)
3.666667
Don't Go
3.962963
Nothing Lasts Forever
4.000000
The Postman
4.018519
The Night Eternal (The Strain Trilogy, #3)
4.018519
Sever (The Chemical Garden, #3)
4.153846
Epilogue (The Dark Duet, #3)
4.019231
The Silent Blade (Forgotten Realms: Paths of Darkness, #1; Legend of Drizzt,
#11)    3.680000
dtype: float64
```

In [ ]:

RESULTS AND DISSCUSSION

Here from the above we can see that the recommendations are based on both userid and user searches. Apart from the present system we are able to generate the recommendations based on the user to which cluster he belongs to.

From the results we can see that the results are appropriate.

If the dataset we are gonna pick is small and then the results would be more appropriate.

CONCLUSION

Here from the above we can see that the recommendations are based on both userid and user searches. Apart from the present system we are able to generate the recommendations based on the user to which cluster he belongs to. The goal of the most recommendation system is to predict the buyer's interest and recommends the books accordingly. This book recommendation has consider that the user ratings of the books to cluster the similar user. And from we can say that we are able to give appropriate output.

And if the data is not appropriate then the output will vary accordingly

In our project we will be able to examine each and every cluster by looking the heat maps genarated. And the enhancement can be done in this by adding some sentimental analysis.

So that the system could generate appropriate results.

REFERENCES

[1] Thede, L.,Marshall, V.A.,Rick W.:An Economic Answer to Unsolicited Communication. EC'04. (2004).

[2] SHARDANAND, U. AND MAES, P. 1995. Social information filtering: algorithms for automating "word of mouth". In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'95). ACM Press/Addison-Wesley Publishing Co., New York, NY, pp. 210–217.

[3] Resnick, P.,and Hal, R. V., 1997. Recommender Systems,Communications of the ACM, 40, 3, pp. 56-58.

[4] FOLTZ, P. W. AND DUMAIS, S. T. 1992.Personalized information delivery: an analysis of information filtering methods. Comm. ACM 35(12), pp. 51–60.

[5] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl. J., 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews, Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work, Chapel Hill, NC, pp.175-186.

[6] SARWAR, B., KARYPIS, G., KONSTAN, J., AND REIDL, J. 2001. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web (WWW'01). ACM, New York, NY, pp.285–295.

[7] J. Han, M. Kamber, Data Mining: Concepts and Techniques, The Morgan Kaufmann Series, 2001.

[8] Agrawal, R., Imielinski, T., Swami, A. N. "Mining association rules between sets of items in large databases". In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 207216, 1993.