VIT®

Vellore Institute of Technology

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

BOOK RECOMMENDATION BASED ON K-MEANS CLUSTERING

By

REDG NO: 18BIT0126

NAME: ROHITH REDDY

SLOT: E1+TE1

PROJECT ID- 01

SUBMITTED TO:

SATHIYAMOORTHY. E

ABSTRACT

Now a days many of us use Netflix, Hotstar and Prime that means we are much reliable on these Online services rather than the offline or cable operators. And also the e-books are available in like kindle etc….

After the evolution of the e-book system we purchase book online and we get a copy and we read it.so here we are proposing an algorithm to identify the related books to your previous reads and to recommend the best books out of them.

Here in our project we consider a huge dataset of the books names and their ratings after that we cluster the dataset by the method of k-means. Clustering is always getting the similar type of things into one cluster. Here the clustering is done based on the books that are read by the user.

Whenever the user searches a book by the clusters we can find the similar books in the same cluster we can display them as the recommendation. i.e.. here we go for classification after the clustering also known as clustered classification.

INTRODUCTION

We currently live in an era of information. We are surrounded by a plethora of data in the form of reviews, blogs, papers and comments on various websites. The number of people around the world who use the internet has witnessed an increase of approximately 40% since 1995 and reached a count of 3.2 billion. The increased information flow has opened more avenues, but it has also led to added confusion for the user. Amidst this huge amount of data, the task of making certain

decisions becomes difficult. It is rightly said that one should make an informed decision, but too much information can also hinder the decision-making process. Thus, in order to save a user from this confusion and make the experience of surfing the internet a pleasurable one, recommender systems were introduced. Francesco Ricci, LiorRokach and BrachaShapira define the recommender systems as software tools that make relevant suggestions to a user. Depending upon the user profile and the product profile, which are formed using various techniques and algorithms, suggestions are made. More than 32% of consumers rate a product online, over 33% writes reviews and nearly 88% trust online ratings and reviews. So here we are proposing a recommender system that clusters the books based on the ratings and then produce recommendations.

LITERATURE SURVEY

Over the years, recommender systems have been studied widely and are divided into different categories according to the approach being used. The categories are collaborative filtering (CF), content based and context based.

Types of recommendation systems

Collaboration filtering

Collaborative filtering (CF) uses the numerical reviews given by the user and is mainly based upon the historical data of the user available to the system. The historical data available helps to build the user profile and the data available about the item is used to make the item profile. Both the user profile and the item profile are used to make a recommendation system. The Netflix Competition has given much

popularity to collaborative filtering. Collaborative filtering is considered the most basic and the easiest method to find recommendations and make predictions regarding the sales of a product. It does have some disadvantages which has led to the development of new methods and techniques.

Content Based Recommender System

Content based systems focus on the features of the products and aim at creating a user profile depending on the previous reviews and also a profile of the item in accordance with the features it provides and the reviews it has received. It is observed that reviews usually contain product feature and user opinion in pairs. It is observed that users reviews contain a feature of the product followed by his/her opinion about the product. Content based recommendation systems help overcome sparsity problem that is faced in collaborative filtering based recommendation system.

Context Based Recommender System

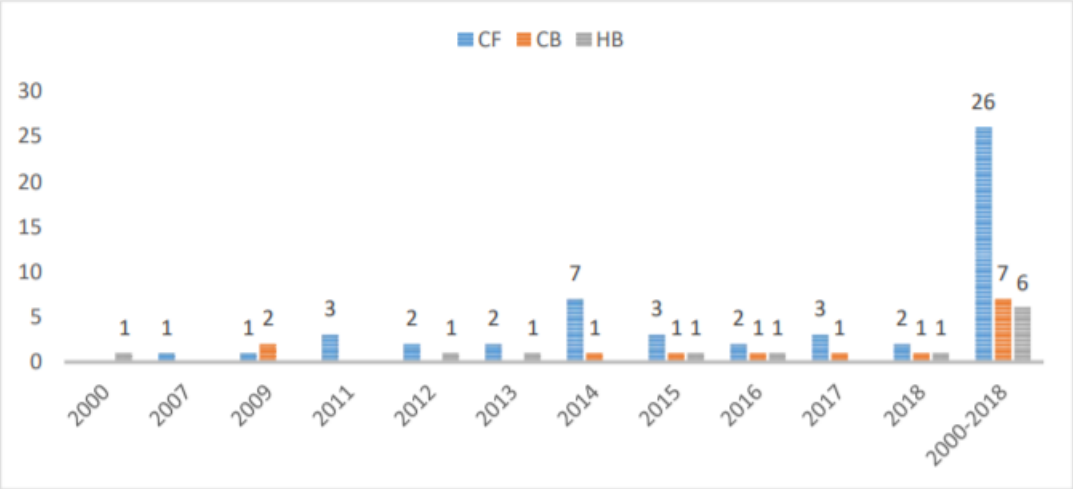
Extending the user/item convention to the circumstances of the user to incorporate the contextual information is what is achieved in context-based recommender systems. This helps to abandon the cumbersome process of making the user fill a huge number of personal details.

Hybrid-Based Filtering:

For the sake of improving recommendation effect, i.e. more personalizing and more accurate, different recommendation methods are combined and employed, forming a hybrid model. HF combines the advantages of both collaborative, content-based filtering and other

methods, which can have an integrated outcome and avoid their individual limitations at the same time.

Classification of methods used in book recommendation:



CF- COLLABORATIVE FILTERING

CB- CONTENT BASED

HB- HYBRID

In Park et al. (2012), the authors presented a review and classification of different approaches of recommender systems, grouping them based on their application fields and the types of data mining techniques that were used (Park et al., 2012). The authors identified 164 articles on recommender systems, which are published from 2001 to 2009 conducted from top 125 journals of the MIS Journal Rankings. Using the classification framework, the recommender systems were classified into eight categories of recommendation fields (e.g. shopping, book, movie, and others), and eight categories into data mining techniques (e.g. association rule, clustering, decision tree, k-nearest, neural network, link analysis, regression and other heuristic methods).

COLLABRATIVE FILTERING METHODS:

Method	Description
Unified relevance model	It is a probabilistic item-to-user relevance framework which uses Parzen-window method for density estimation. This approach reduces data sparsity problem.
Hybrid CF model	It introduces effective recommender system using sequential mixture CF and joint mixture CF. It also implements advanced Bayes belief networks.
Fuzzy Association Rules and Multilevel Similarity (FARAMS)	It uses fuzzy association rule mining to extend the existing techniques. FARAMS achieved the task of generating more qualitative predictions.
Flexible mixture model (FMM)	Simultaneous creation of user and item clusters. It introduces preference nodes to study a dramatic variation of the rating among users with similar tastes.
Maximum entropy approach	Clustering of items based on user access path in order to reduce the apriori probability. This helps in addressing sparsity and dimensionality reduction.

CONTENT BASED RECOMMENDER SYSTEMS METHODS:

Method	Description
Content-Boosted Collaborative Filtering	It gives an approach to combine content and collaboration to enhance existing user data and to give better performance than a pure content based predictor.
FAB Technique	An adaptive recommendation service for collection and selection of web pages. It makes the system more personalized and combines the benefits of content analysis with the shared user interests.
Bayesian hierarchical model(BHM)	Proposes a faster technique to gather a huge number of individual user profiles even if feedbacks available are less. It uses various parameters of BHM for optimization of joint data likelihood.

REFERENCES

[1] Ramakrishnan, G., Saicharan, V., Chandrasekaran, K., Rathnamma, M. V., & Ramana, V. V. (2020). Collaborative Filtering for Book Recommendation System. In Soft Computing for Problem Solving (pp. 325-338). Springer, Singapore.

[2] Anoop, A., & Ubale, N. A. (2020, August). Cloud Based Collaborative Filtering Algorithm for Library Book Recommendation System. In 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT) (pp. 695-703). IEEE.

[3] Kommineni, M., Alekhya, P., Vyshnavi, T. M., Aparna, V., Swetha, K., & Mounika, V. (2020, January). Machine Learning based Efficient Recommendation System for Book Selection using User based Collaborative Filtering Algorithm. In 2020 Fourth International Conference on Inventive Systems and Control (ICISC) (pp. 66-71). IEEE.

[4] Davagdorj, K., Park, K. H., & Ryu, K. H. (2020). A Collaborative Filtering Recommendation System for Rating Prediction. In Advances in Intelligent Information Hiding and Multimedia Signal Processing (pp. 265-271). Springer, Singapore.

[5] Mccrea, J., Zada, N., King, R., Li, J., Connolly, C., Lester, P., & Kennedy, C. (2020). U.S. Patent No. 10,805,102. Washington, DC: U.S. Patent and Trademark Office.

[6] Yuchen, X. (2020). Intelligent ordering recommendation system based on microservice. Academic Journal of Computing & Information Science, 3(3).

[7] Cruz, A. F. T., & Coronel, A. D. (2020). Towards Developing a Content-Based Recommendation System for Classical Music. In Information Science and Applications (pp. 451-462). Springer, Singapore.

[8] Selvi, C., Vaishnavi, N. G., Nataraja, B., & Arsaath, M. Effective Sensitive Recommendation for Online Book with User Relationship Similarity.

[9] Sowmiya, S. A., & Hamsagayathri, P. A Collaborative Approach for Course Recommendation System. In Advances in Smart Grid Technology (pp. 527-536). Springer, Singapore.

[10] Voggu, S. V. S., Champawat, Y. S., Kothari, S., & Tripathy, B. K. (2020). Recommendation System Using Community Identification. In International Conference on Innovative Computing and Communications (pp. 125-132). Springer, Singapore.

BOOK RECOMMENDATION BASED ON K-MEANS CLUSTERING

BY 18BIT0126- N. Rohith Reddy

SUBMITTED TO: Sathiyamoorthy. E

SLOT: E1+TE1

ABSTRACT:

Now a days many of us use Netflix, Hotstar and Prime that means we are much reliable on these Online services rather than the offline or cable operators. And also the e-books are available in like kindle etc…. After the evolution of the e-book system we purchase book online and we get a copy and we read it.so here we are proposing an algorithm to identify the related books to your previous reads and to recommend the best books out of them.

Here in our project we consider a huge dataset of the books names and their ratings after that we cluster the dataset by the method of k-means. Clustering is always getting the similar type of things into one cluster. Here the clustering is done based on the books that are read by the user. Whenever the user searches a book by the clusters we can find the similar books in the same cluster we can display them as the recommendation. i.e.. here we go for collabrative filtering with clustering.

METHOD THAT WE ARE GOING TO USE:

1. we consider a dataset
2. After dataset selection we have to preprocess the data if there is a noise in my case i have no null or missing values
3. after preprocessing here we convert the data set into sparse matrix to increase processing speed and memory comsumption
4. we use the silhouette score to find the optimal k value
5. now by using optimal k we cluster the data
6. after clustering when we search for a book we go through the clusters and find the book presence
7. based on the book presence we will be going to display reccomendations based on the cluster id to which it belongs.
8. here we also generate the recommendations based on the user id too.

DRAWBACKS OF K-MEANS:

1. here the main draw back is to find the optimal k manually (here we overcome that with the help of Silhouette score
2. data that k means can handle is less(so here we select the data we want to process by slicing the dataset)

In [1]:
```
#here pandas can be used to work with dataframes and some mathematical operati
ons on them
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
import itertools
from scipy import sparse
import scipy as sc
from sklearn import metrics
# here we import the kmeans which help us to cluster the books
from sklearn.cluster import KMeans
#here the silhouette_score helps us to effectively define the clusters
from sklearn.metrics import silhouette_samples, silhouette_score
#With this the output of plotting commands is displayed inline within frontend
s like the Jupyter notebook
%matplotlib inline
#here we read the books dataset from the given directory
books = pd.read_csv(r'C:\Users\asus\Desktop\books.csv')
books.head(10)
```

Out[1]:

	bookId	title
0	2767052	The Hunger Games (The Hunger Games, #1)
1	3	Harry Potter and the Sorcerer's Stone (Harry P...
2	41865	Twilight (Twilight, #1)
3	2657	To Kill a Mockingbird
4	4671	The Great Gatsby
5	11870085	The Fault in Our Stars
6	5907	The Hobbit
7	5107	The Catcher in the Rye
8	960	Angels & Demons (Robert Langdon, #1)
9	1885	Pride and Prejudice

In [2]:
```
#here we read the ratings dataset from the given directory
ratings = pd.read_csv(r'C:\Users\asus\Desktop\ratings.csv')
#here we print the top most entries in the dataset
ratings.head(10)
```

Out[2]:

	userId	bookId	rating
0	314	1	5
1	439	1	3
2	588	1	5
3	1169	1	4
4	1185	1	4
5	2077	1	4
6	2487	1	4
7	2900	1	5
8	3662	1	4
9	3922	1	5

In [3]:
```
#here we get the null entries in the dataset
books.isnull().sum()
```

Out[3]:
```
bookId    0
title     0
dtype: int64
```

In [4]:
```
#here we get the null entries in the dataset
ratings.isnull().sum()
```

Out[4]:
```
userId    0
bookId    0
rating    0
dtype: int64
```

In [5]:
```
#info gives the information about data types and entries
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 2 columns):
bookId    10000 non-null int64
title     10000 non-null object
dtypes: int64(1), object(1)
memory usage: 156.4+ KB
```

In [6]:
```
#info gives the information about data types and entries
ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 981756 entries, 0 to 981755
Data columns (total 3 columns):
userId    981756 non-null int64
bookId    981756 non-null int64
rating    981756 non-null int64
dtypes: int64(3)
memory usage: 22.5 MB
```

In [7]:
```
def get_most_rated_books(user_book_ratings, max_number_of_books):
    # 1- Count =appending the no of users for each book
    user_book_ratings = user_book_ratings.append(user_book_ratings.count(),ign
ore_index=True)
    # 2- sorting the books ratings
    #.drop() function in Pandas be used to delete rows from a DataFrame, with
 the axis set to 0
    user_book_ratings_sorted = user_book_ratings.sort_values(len(user_book_rat
ings)-1, axis=1, ascending=False)
    user_book_ratings_sorted = user_book_ratings_sorted.drop(user_book_ratings
_sorted.tail(1).index)
    # 3- slice getting the required no of books from the array
    most_rated_books = user_book_ratings_sorted.iloc[:, :max_number_of_books]
    return most_rated_books

def sort_by_rating_density(user_book_ratings, n_books, n_users):
    #calling the functions defined above
    most_rated_books = get_most_rated_books(user_book_ratings, n_books)
    most_rated_books = get_users_who_rate_the_most(most_rated_books, n_users)
    return most_rated_books
```

In [8]:
```
def get_users_who_rate_the_most(most_rated_books, max_number_of_books):
    # Get most voting users
    # 1- Count= count the users who rated most of the books
    #converting into a one-dimensional labeled array to store any type of data
    most_rated_books['counts'] = pd.Series(most_rated_books.count(axis=1))
    # 2- Sort=sorting the users by user ratings values(counts)
    most_rated_books_users = most_rated_books.sort_values('counts', ascending=
False)
    # 3- Slice=selecting the max_users from the array(selecting the max_books
 for operation)
    most_rated_books_users_selection = most_rated_books_users.iloc[:max_number
_of_books, :]
    #.drop() function in Pandas be used to delete rows from a DataFrame, with
 the axis set to 0
    most_rated_books_users_selection = most_rated_books_users_selection.drop([
'counts'], axis=1)
    return most_rated_books_users_selection
```

In [9]:
```
def draw_book_clusters(clustered, max_users, max_books):
    for cluster_id in clustered.group.unique():
        # To improve visibility, we're showing at most max_users users and max
_books  per cluster.
        d = clustered[clustered.group == cluster_id].drop(['index', 'group'],a
xis=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 books by rating density books rated most are top of clust
er
        d = sort_by_rating_density(d, max_books, 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_books into d
        #.iloc[] is primarily integer position based (from 0 to length-1 of th
e axis),but may also be used with a boolean array.
        d = d.iloc[:max_users, :max_books]
        n_users_in_plot = d.shape[0]
        # We're selecting to show all clusters to have some restriction we can
have like 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), '#o
f 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]
            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('books')
            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',le
ft='off',labelbottom='off',labelleft='off')
            plt.show()
```

In [10]:
```
# Merge the two tables then pivot so we have Users X books dataframe
ratings_title = pd.merge(ratings, books[['bookId', 'title']], on='bookId' )
# pd.pivot_table()= create a spreadsheet-style pivot table as a DataFrame.
user_book_ratings = pd.pivot_table(ratings_title, index='userId', columns= 'title', values='rating')
# Print he number of dimensions and a subset of the dataset
print('dataset dimensions: ', user_book_ratings.shape, '\n\nSubset example:')
#printing the subset of the user_book_ratings
user_book_ratings.iloc[:6, :10]
```

```
dataset dimensions:  (28906, 812)

Subset example:
```

Out[10]:

title	'Salem's Lot	'Tis (Frank McCourt, #2)	1421: The Year China Discovered America	1776	1984	A Bend in the River	A Bend in the Road	A Brief History of Time	A Briefer History of Time	A Case of Need
userId										
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [11]:
```
n_books = 30
n_users = 18
#sort_by_rating_density function call to get the most_rated_books_users_select
ion data
most_rated_books_users_selection = sort_by_rating_density(user_book_ratings,n_
books, n_users)
# Print the result
print('dataset dimensions: ', most_rated_books_users_selection.shape)
#printing the top most in the most_rated_books_users_selection.head()
most_rated_books_users_selection.head()
```

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

Out[11]:

title	Perfume: The Story of a Murderer	Plum Lovin' (Stephanie Plum, #12.5)	Pearls of Lutra (Redwall, #9)	The Thorn Birds	Persuasion	The Testament	Play It as It Lays	The Terror	Pompeii	H C D
14415	5.0	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	
8746	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	
4241	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	
6876	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	
11101	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	

5 rows × 30 columns

In [12]:
```
#this below line was from above cell
user_book_ratings = pd.pivot_table(ratings_title, index='userId', columns='tit
le', values='rating')
total=1000
#getting the thousand books that are rated most by the users
most_rated_books_1k = get_most_rated_books(user_book_ratings, total)
#creating the sparse matrix for most_book_ratings
#sparse matrices to store data that contains a large number of zero-valued ele
ments
#can both save a significant amount of memory and speed up the processing of t
hat data
#csr matrix (compresed row storage) has three (1-d)array with non zero values,
extent of rows,column indices
sparse_ratings = csr_matrix(pd.SparseDataFrame(most_rated_books_1k).to_coo())
sparse_ratings
```

```
C:\Users\asus\anaconda3\lib\site-packages\ipykernel_launcher.py:7: FutureWarn
ing: 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\asus\anaconda3\lib\site-packages\pandas\core\frame.py:3471: FutureWa
rning: 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)
```

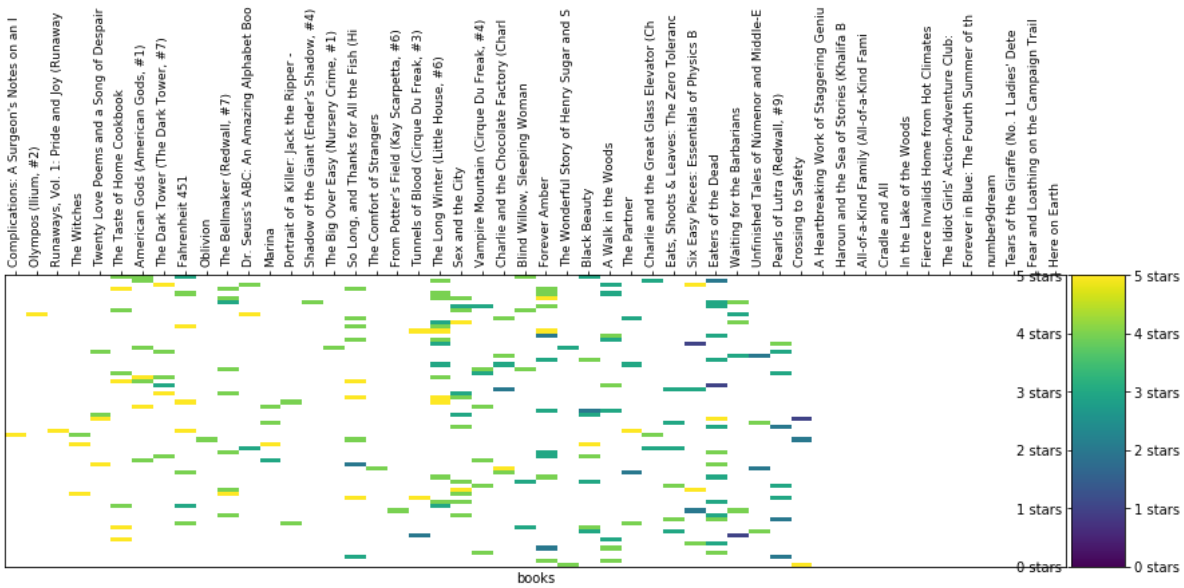
Out[12]:
```
<28906x812 sparse matrix of type '<class 'numpy.float64'>'
        with 79531 stored elements in Compressed Sparse Row format>
```

In [13]:
```
X=sparse_ratings
#finding the silhouette score for each k value to get the best and optimal k f
or k-means
def clustering_errors(k, data):
    #fitting the data
    kmeans = KMeans(n_clusters=k).fit(data)
    #prediction or clustering the data
    predictions = kmeans.predict(data)
    #finding the score(silhouette)
    silhouette_avg = silhouette_score(data, predictions)
    #printing the scores with k values
    print(k,"->",silhouette_avg)
for k in range(2,30,3):
    clustering_errors(k,X)
#where the value is equal to zero i.e the score we consider it as the good and
optimal k value for clustering
```

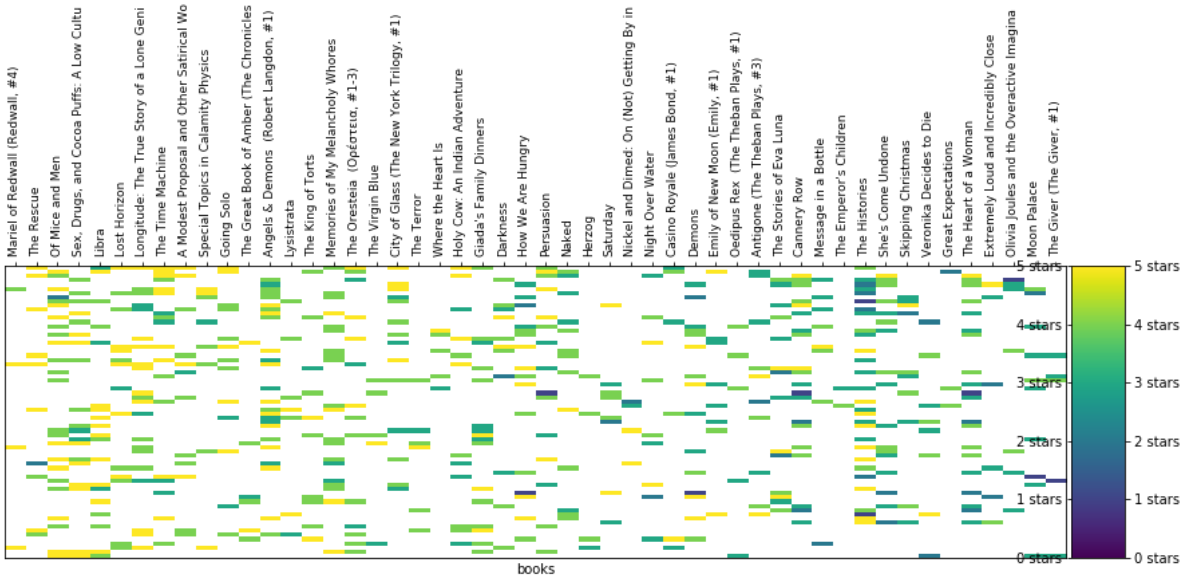
```
2 -> 0.5528707888942672
5 -> -0.005336708399299322
8 -> 0.08411415858812094
11 -> -0.03944539498581238
14 -> -0.03849628195924156
17 -> -0.04870365889244765
20 -> 0.0011278214122548349
23 -> -0.04117999507346132
26 -> -0.10278898266537848
29 -> -0.002257139140302551
```

In [14]:
```
#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)
#here we give the input that the no of m books and users ratings that are to b
e displayed in the heatmap
max_users = 70
max_books = 50
#concatination along the axis of the data pd.concat
#arthematic align of rows and columns in the dataset pd.dataframe
#here we are combing the cluster group into the data set most_rated_books
clustered = pd.concat([most_rated_books_1k.reset_index(), pd.DataFrame({'grou
p':predictions})], axis=1)
#drawing the clusters with max_users and max_books
draw_book_clusters(clustered, max_users, max_books)
```

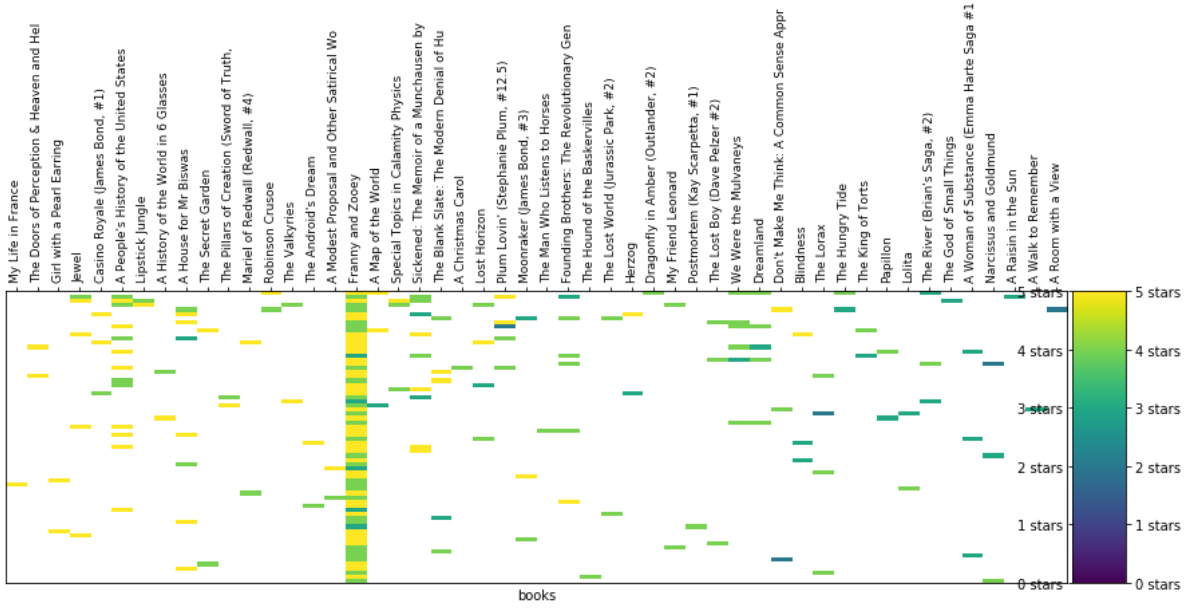
```
cluster # 1
# of users in cluster: 25391. #of users in plot: 70
```



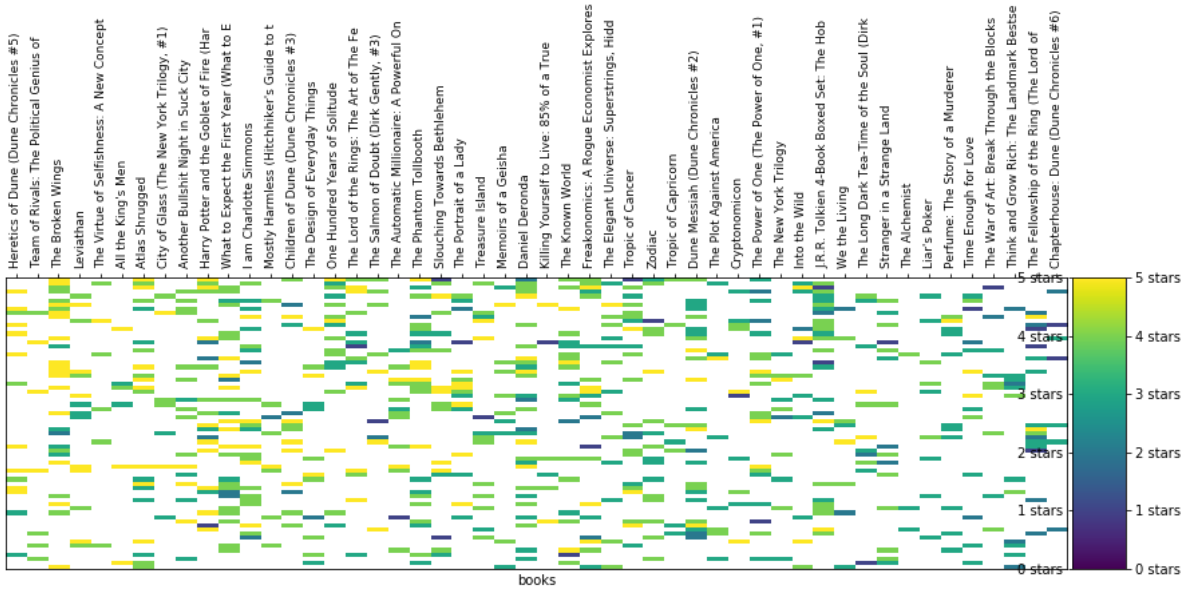

```
cluster # 18
# of users in cluster: 565. #of users in plot: 70
```



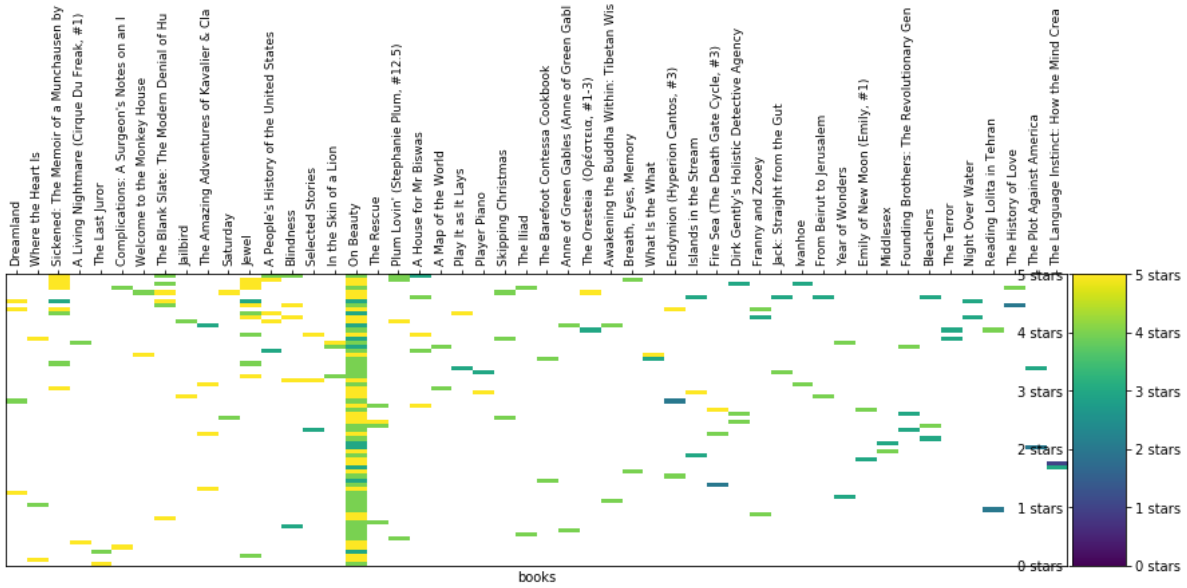

```
cluster # 19
# of users in cluster: 96. #of users in plot: 70
```



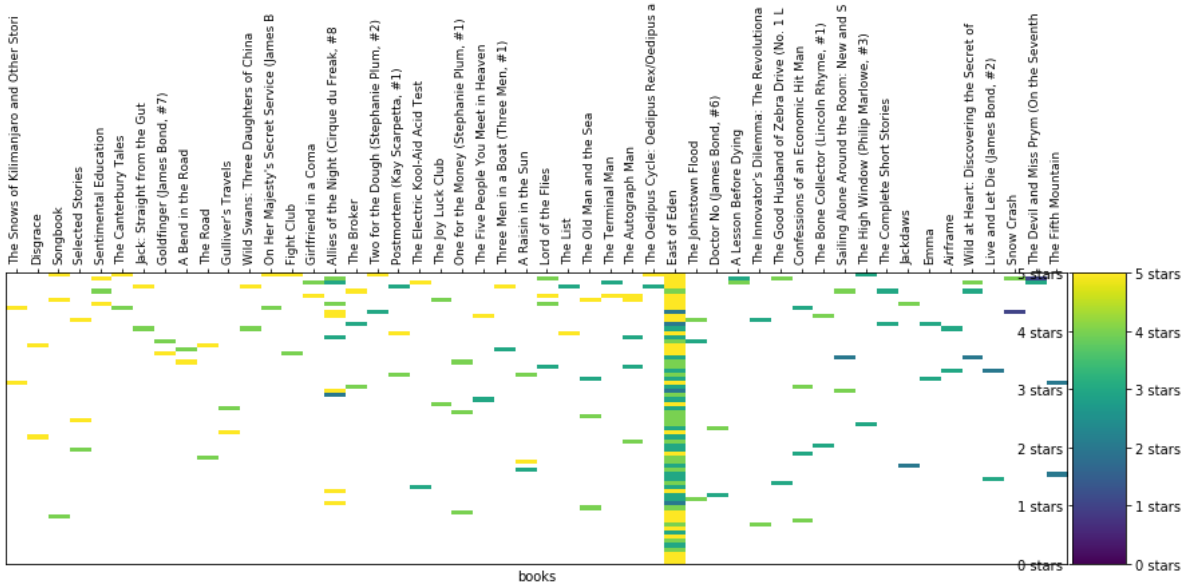
cluster # 11
# of users in cluster: 391. #of users in plot: 70



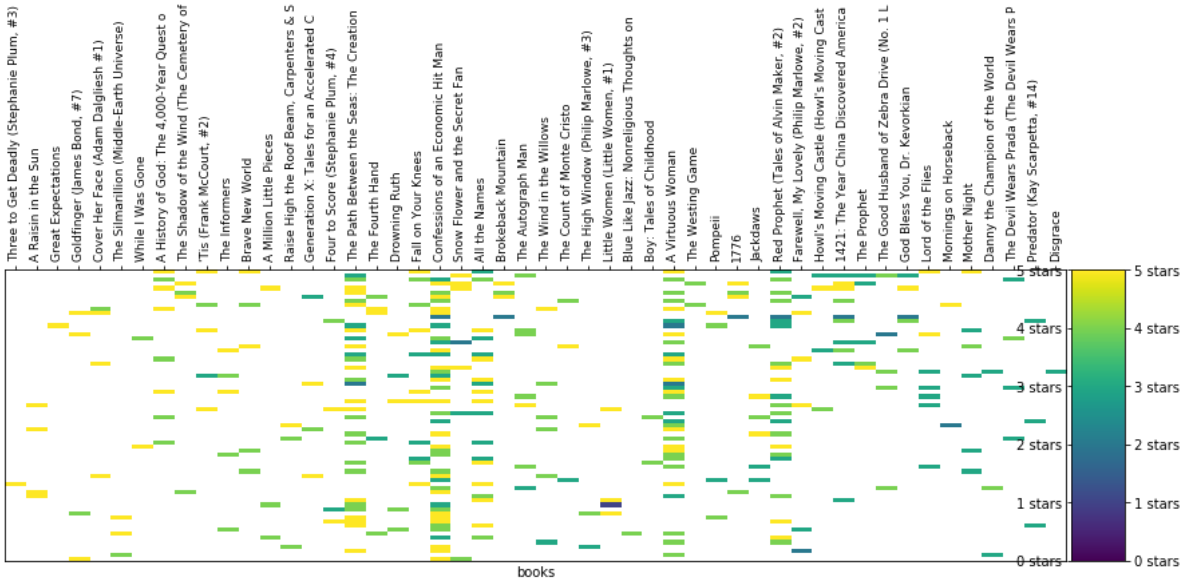
cluster # 9
# of users in cluster: 95. #of users in plot: 70



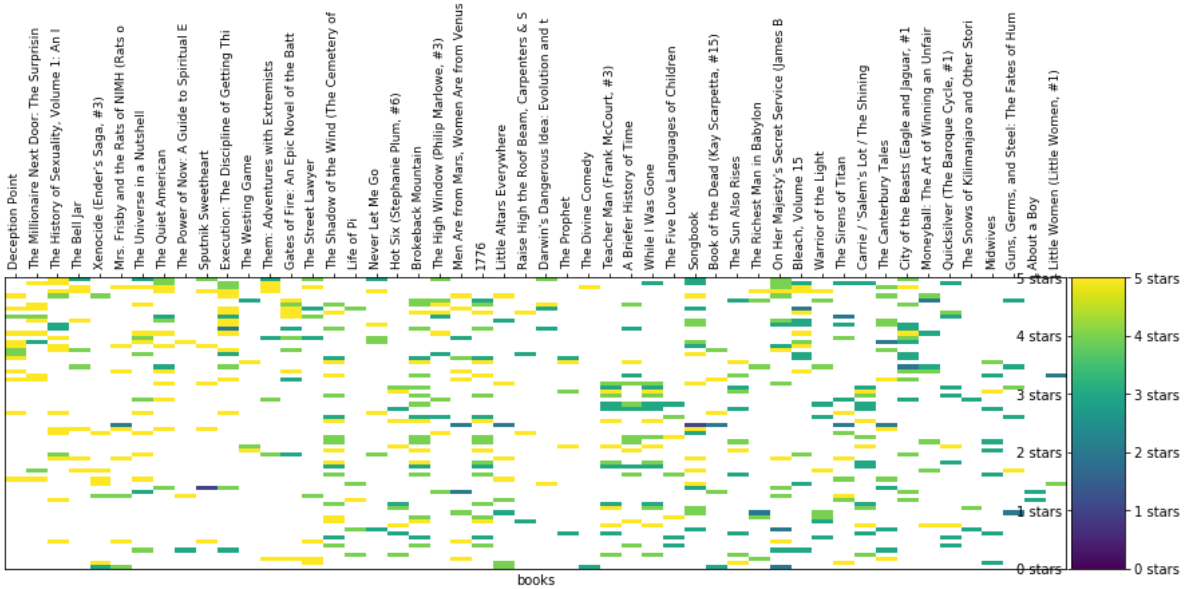

```
cluster # 10
# of users in cluster: 87. #of users in plot: 70
```



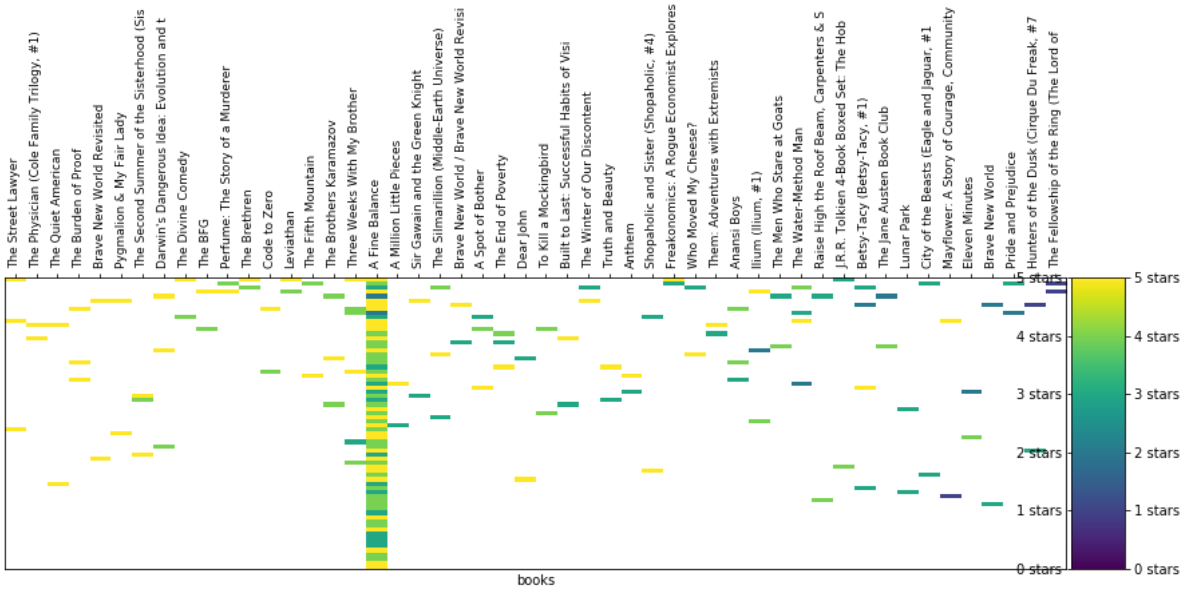

```
cluster # 0
# of users in cluster: 172. #of users in plot: 70
```



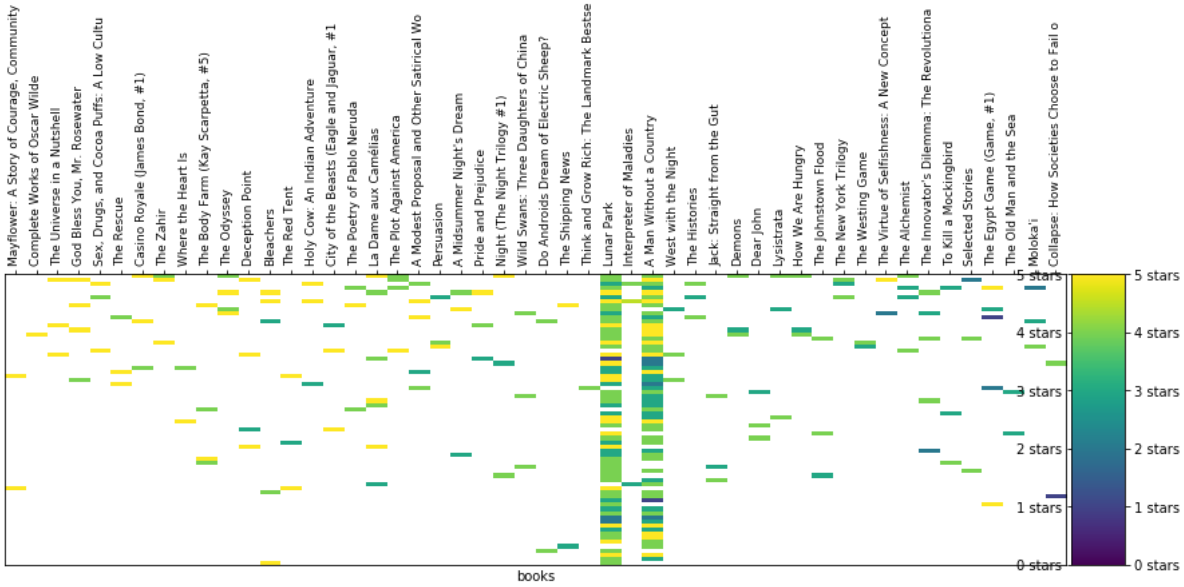
cluster # 16
# of users in cluster: 491. #of users in plot: 70



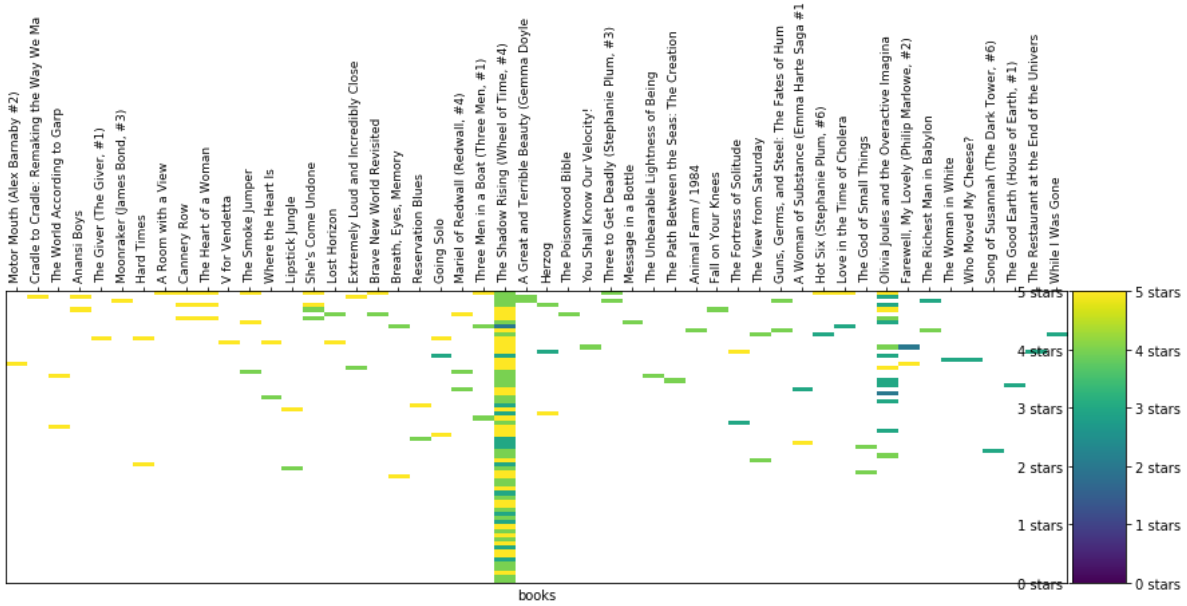
cluster # 17
# of users in cluster: 93. #of users in plot: 70



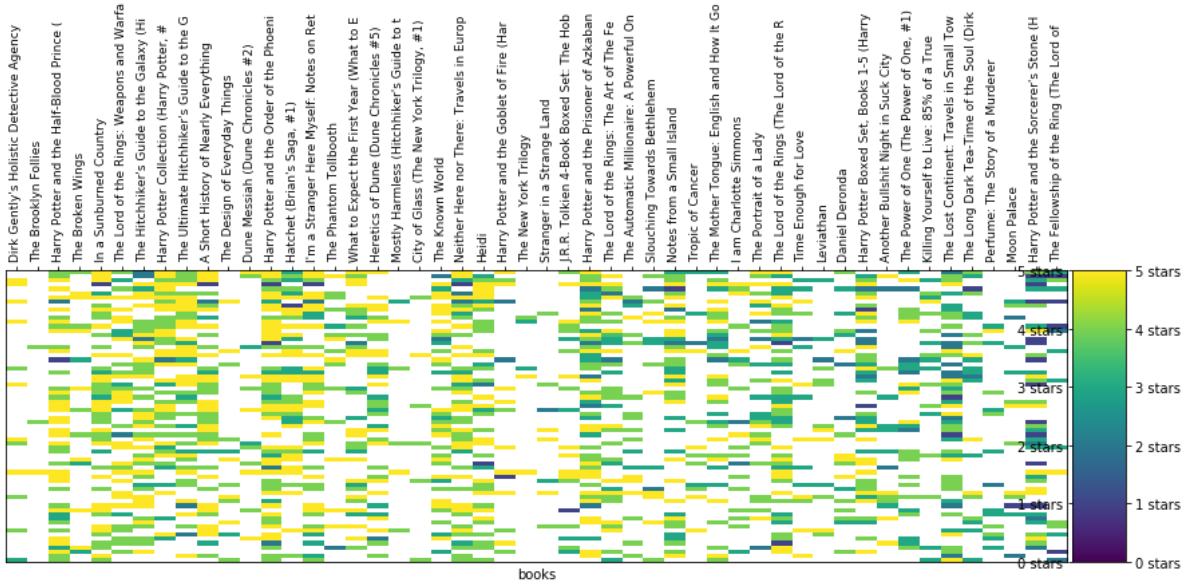

```
cluster # 13
# of users in cluster: 100. #of users in plot: 70
```



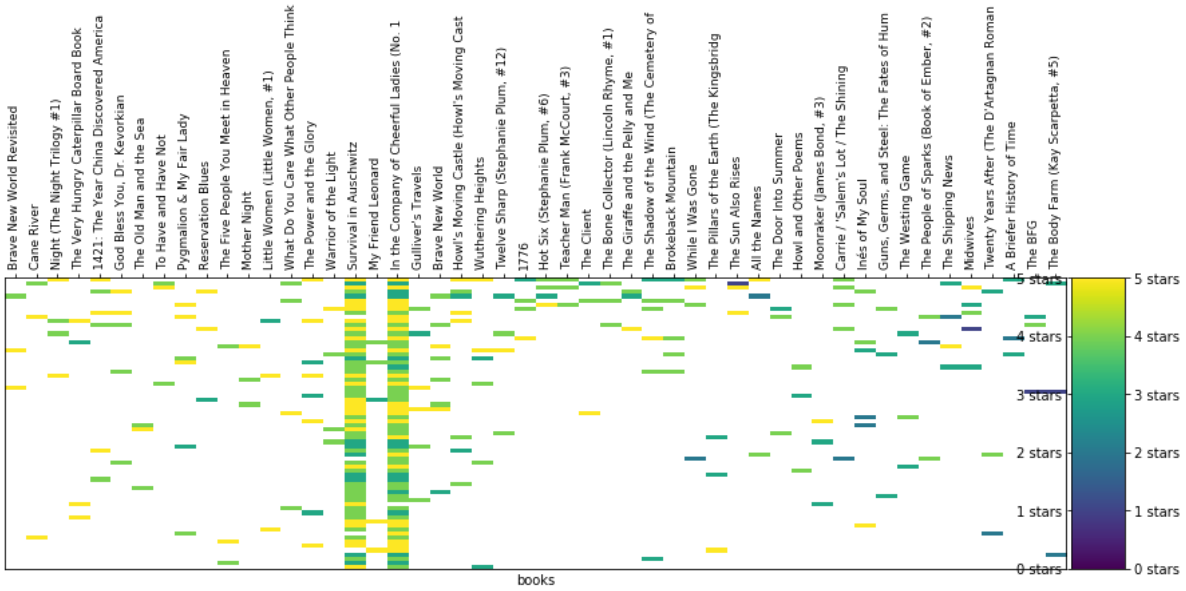

```
cluster # 12
# of users in cluster: 92. #of users in plot: 70
```



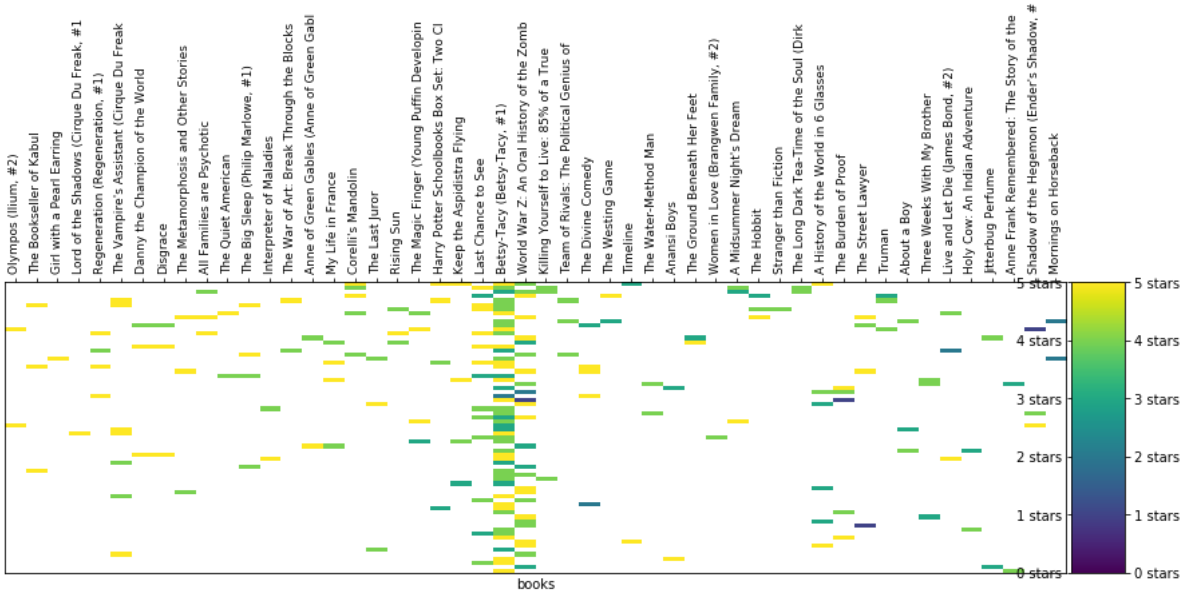
cluster # 15
# of users in cluster: 168. #of users in plot: 70



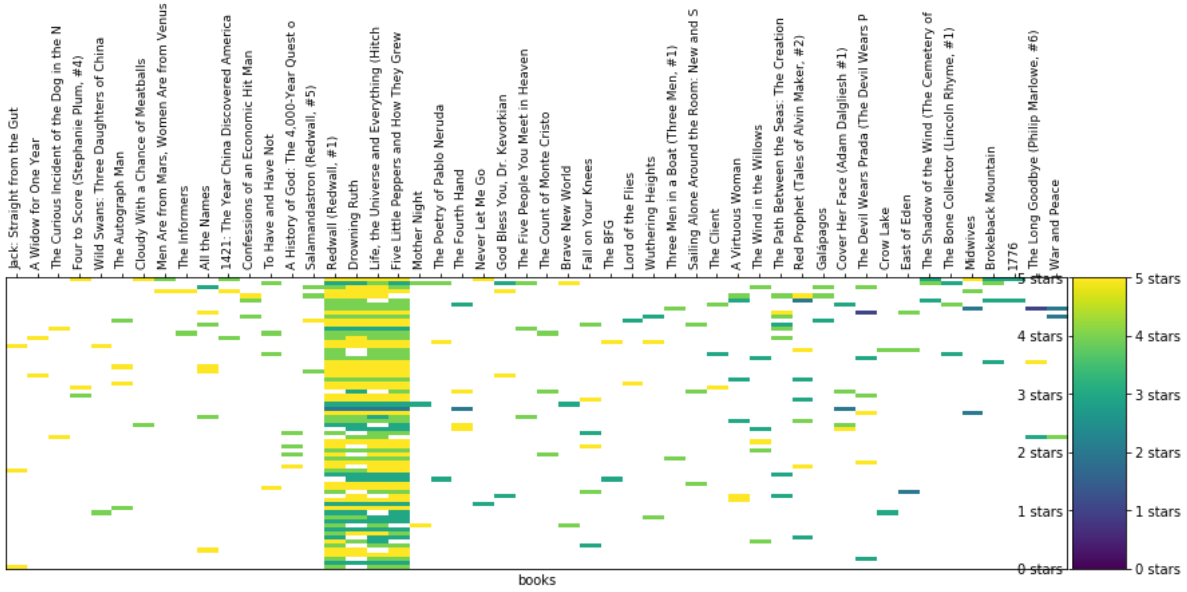
cluster # 5
# of users in cluster: 92. #of users in plot: 70



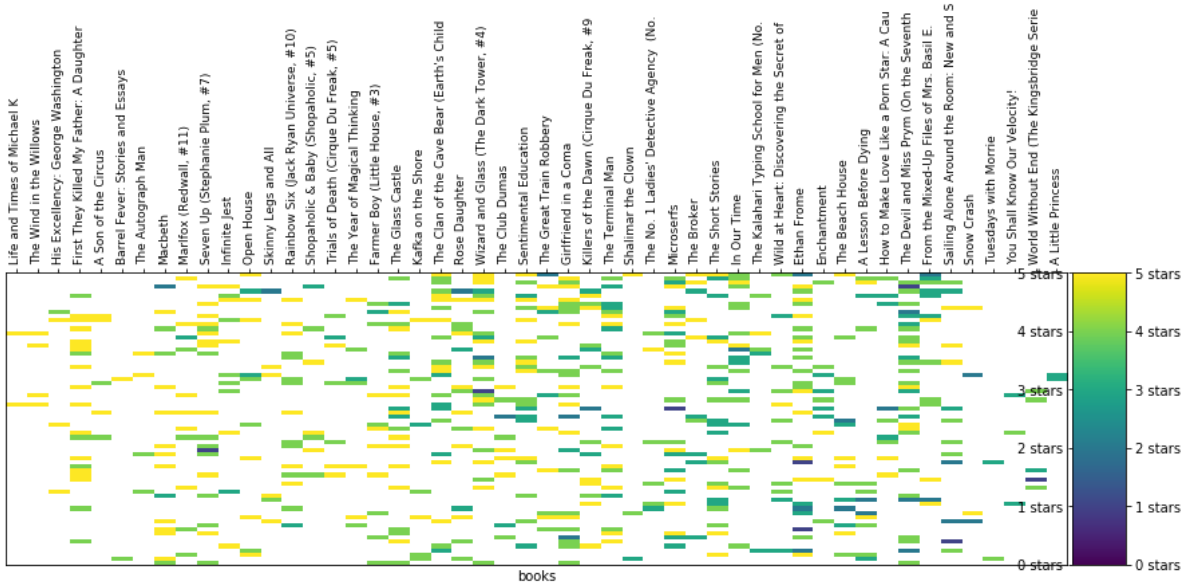
cluster # 7
# of users in cluster: 171. #of users in plot: 70



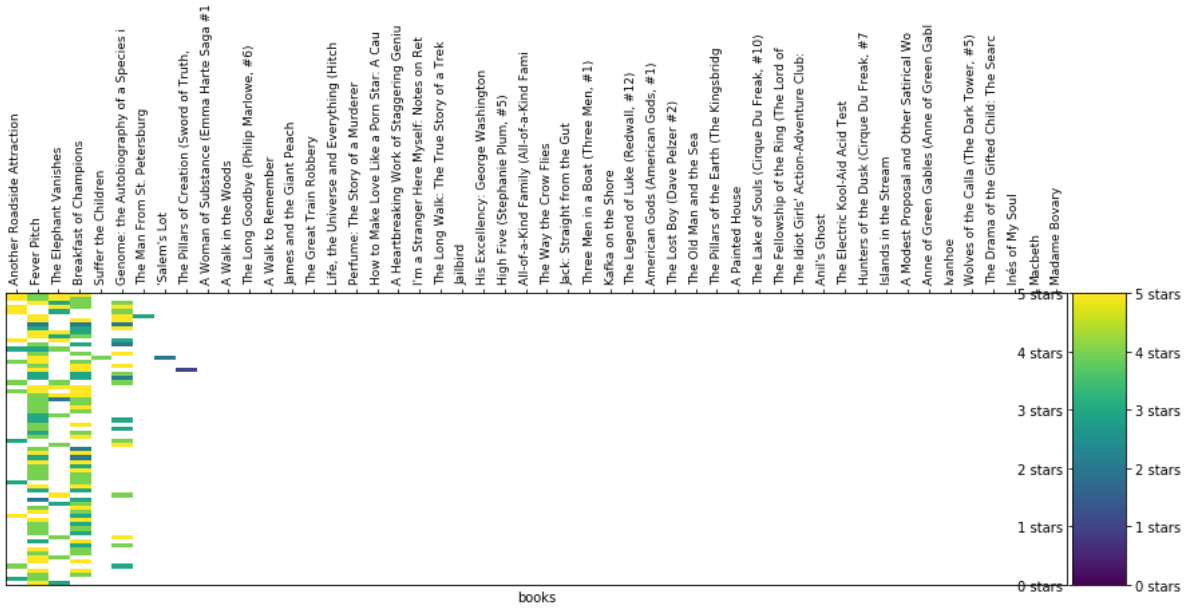
cluster # 3
# of users in cluster: 94. #of users in plot: 70



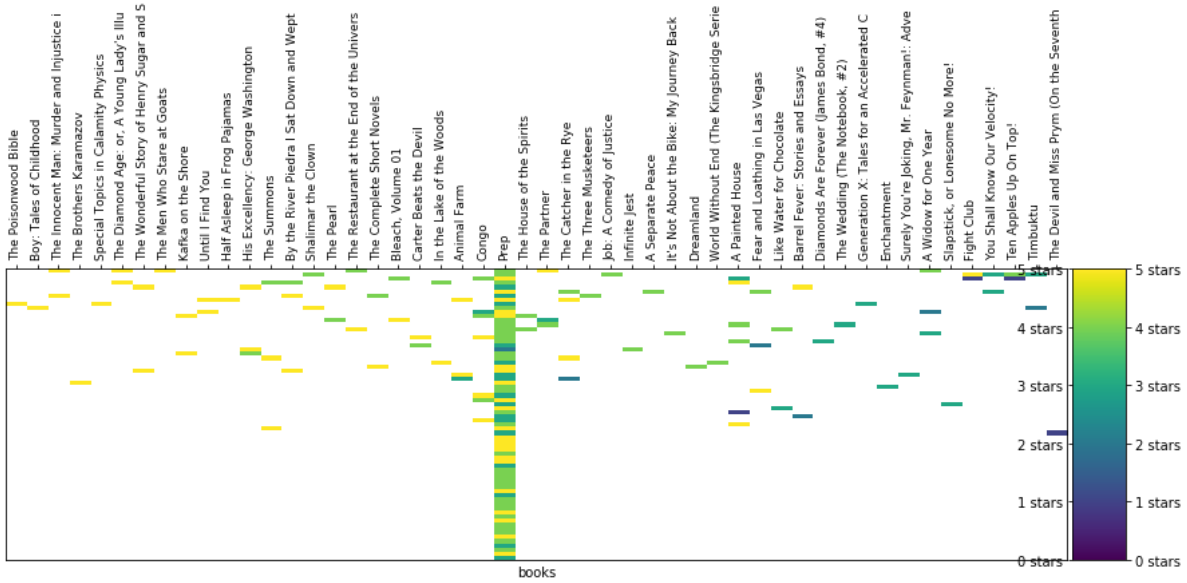
cluster # 6
# of users in cluster: 477. #of users in plot: 70



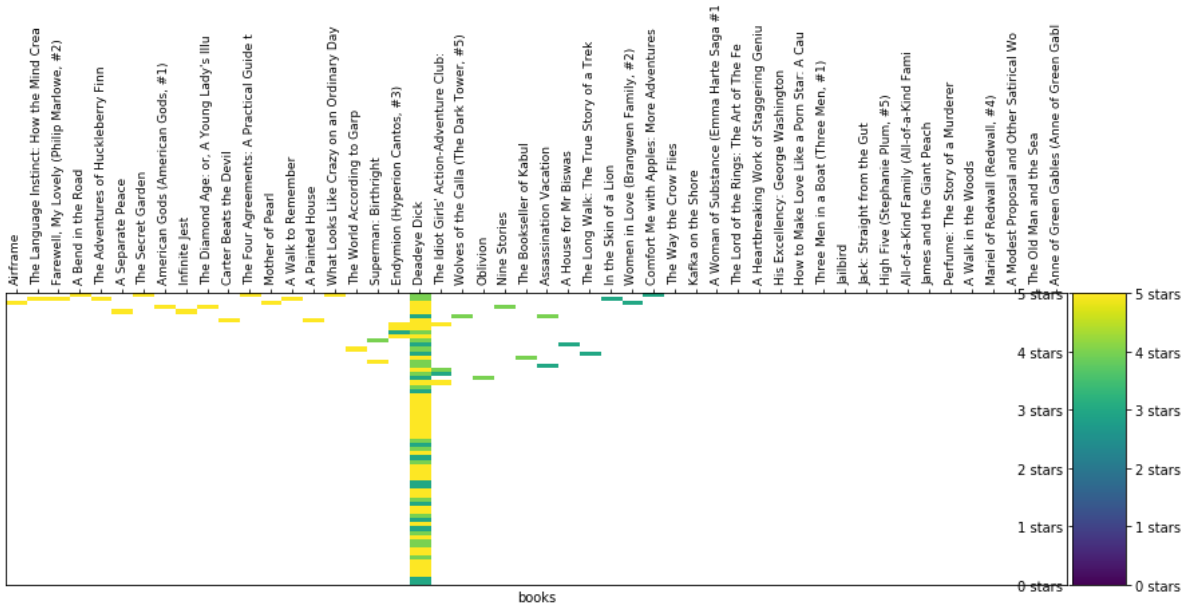
cluster # 4
# of users in cluster: 165. #of users in plot: 70



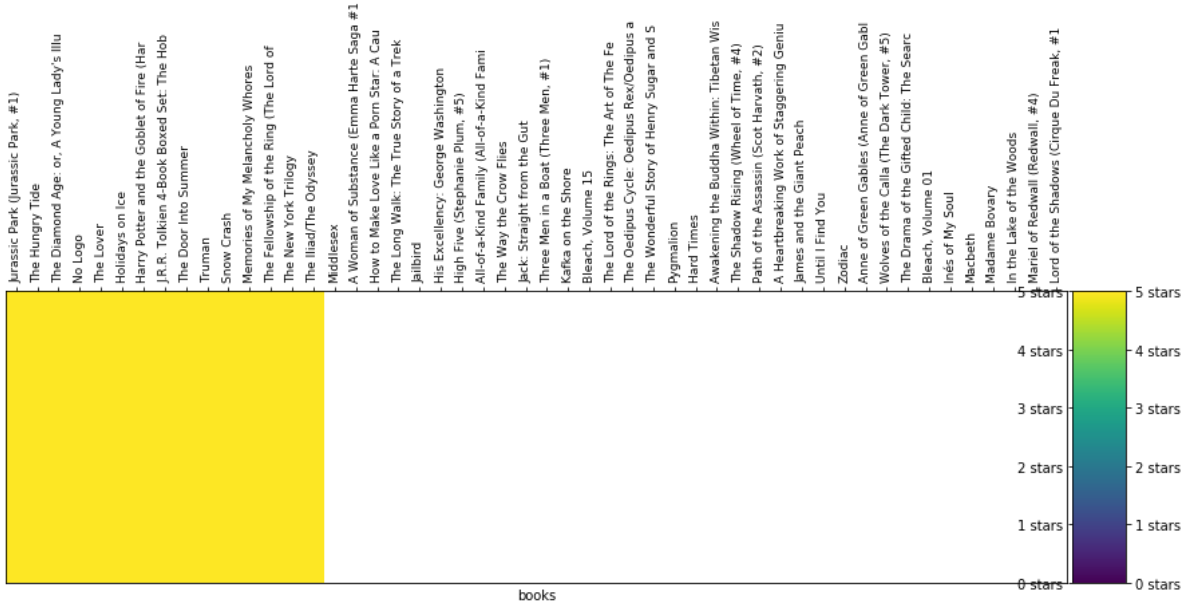
cluster # 8
# of users in cluster: 84. #of users in plot: 70



cluster # 14
# of users in cluster: 81. #of users in plot: 70




```
cluster # 2
# of users in cluster: 1. #of users in plot: 1
```



In [15]:
```
#input the book name that we want to serach
book_name = "Truman"
#cl is used for iterating throught he clusters
#ocl is used to store the cluster id
cl=0
ocl=0
print("THE CLUSTERS ARE NOT MUTUALLY EXCLUSIVE ONE book CAN BE IN DIFFERENT CL
USTERS :)....\n ")
while cl<n :
    n_users = 70
    n_books = 50
    cluster = clustered[clustered.group == cl].drop(['index', 'group'], axis=1
)
    cluster = sort_by_rating_density(cluster, n_books, n_users)
    lantren=cluster.columns.str[:100]
    for j in range(len(lantren)) :
        #lantern is used to store the cluster group that we are going to itera
te
        if lantren[j] == book_name :
            print("THE CLUSTER ID THAT CONTAINS THE book NAME IS\n")
            ocl=cl
            k=j
            print(ocl,"cluster \n")
            print("THE INDEX OF THE book IN THE CLUSTER #",ocl,"IS\n",k)
        else :
            continue
    cl=cl+1
# here from the above statement we will get what are all the occurences in the
cluster does the movie have.
#from the occurances we choose the cluster that has the movie with the lower i
ndex
```

```
THE CLUSTERS ARE NOT MUTUALLY EXCLUSIVE ONE book CAN BE IN DIFFERENT CLUSTERS
:)....

THE CLUSTER ID THAT CONTAINS THE book NAME IS

2 cluster

THE INDEX OF THE book IN THE CLUSTER # 2 IS
 5
THE CLUSTER ID THAT CONTAINS THE book NAME IS

7 cluster

THE INDEX OF THE book IN THE CLUSTER # 7 IS
 21
```

In [16]:
```
ocl=2 #cluster id
k=5   #index in the above cluster
cluster = clustered[clustered.group == ocl].drop(['index', 'group'], axis=1)
cluster = sort_by_rating_density(cluster, n_books, n_users)
tclus=cluster.columns.str[:100]
print("the books average rating by the users in the cluster ",ocl,"=",cluster[
book_name].mean())
print("the book name in the cluster",ocl,"at the index",k,"is [**[",tclus[k],
"]**]")
```

```
the books average rating by the users in the cluster  2 = 5.0
the book name in the cluster 2 at the index 5 is [**[ Truman ]**]
```

In [17]:
```
print("you may also like to read\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 books by all the users who are similar to you\n")
print(cluster.mean().head(25))
```

you may also like to read

Macbeth [nan]
Madame Bovary [nan]
In the Lake of the Woods [nan]
Mariel of Redwall (Redwall, #4) [nan]
Lord of the Shadows (Cirque Du Freak, #11) [nan]
The Iliad/The Odyssey [5.0]
The New York Trilogy [5.0]
The Fellowship of the Ring (The Lord of the Rings, #1) [5.0]
Memories of My Melancholy Whores [5.0]
Snow Crash [5.0]
The Door Into Summer [5.0]
J.R.R. Tolkien 4-Book Boxed Set: The Hobbit and The Lord of the Rings [5.0]
Harry Potter and the Goblet of Fire (Harry Potter, #4) [5.0]
Holidays on Ice [5.0]
The Lover [5.0]
No Logo [5.0]
The Diamond Age: or, A Young Lady's Illustrated Primer [5.0]
The Hungry Tide [5.0]
Jurassic Park (Jurassic Park, #1) [5.0]

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

The Iliad/The Odyssey	5.0
The New York Trilogy	5.0
The Fellowship of the Ring (The Lord of the Rings, #1)	5.0
Memories of My Melancholy Whores	5.0
Snow Crash	5.0
Truman	5.0
The Door Into Summer	5.0
J.R.R. Tolkien 4-Book Boxed Set: The Hobbit and The Lord of the Rings	5.0
Harry Potter and the Goblet of Fire (Harry Potter, #4)	5.0
Holidays on Ice	5.0
The Lover	5.0
No Logo	5.0
The Diamond Age: or, A Young Lady's Illustrated Primer	5.0
The Hungry Tide	5.0
Jurassic Park (Jurassic Park, #1)	5.0
Middlesex	NaN
A Woman of Substance (Emma Harte Saga #1)	NaN
How to Make Love Like a Porn Star: A Cautionary Tale	NaN
The Long Walk: The True Story of a Trek to Freedom	NaN
Jailbird	NaN
His Excellency: George Washington	NaN
High Five (Stephanie Plum, #5)	NaN
All-of-a-Kind Family (All-of-a-Kind Family, #1)	NaN
The Way the Crow Flies	NaN
Jack: Straight from the Gut	NaN

dtype: float64

In [18]:
```
#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 books in the cluster
user_id = 56
ccl=0
oocl=0
while ccl<n :
    n_users = 70
    n_books = 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 56
THE CLUSTER ID THAT CONTAINS THE USER_ID IS= 1 th cluster

THE INDEX OF THE USER IN THE CLUSTER # 1 IS= 50
```

In [19]:
```
cluster = clustered[clustered.group == oocl].drop(['index', 'group'], axis=1)
indo=cluster.iloc[:70,:0]
cluster = sort_by_rating_density(cluster, n_books, n_users)
ttclus=cluster.columns.str[:100]
print("you may also like to Read\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 books by all the users who are similar to you\n")
print(cluster.mean().head(15))
```

```
you may also like to Read

Here on Earth [ nan ]
Tunnels of Blood (Cirque Du Freak, #3) [ 4.0 ]
Blind Willow, Sleeping Woman [ 3.6 ]
The Witches [ 4.666666666666667 ]
Six Easy Pieces: Essentials of Physics By Its Most Brilliant Teacher [ 3.3333
333333333335 ]
From Potter's Field (Kay Scarpetta, #6) [ 4.0 ]
The Wonderful Story of Henry Sugar and Six More [ 3.5 ]
A Walk in the Woods [ 3.4444444444444446 ]

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

Complications: A Surgeon's Notes on an Imperfect Science
5.000000
The Bellmaker (Redwall, #7)
4.000000
Dr. Seuss's ABC: An Amazing Alphabet Book! (Bright and Early Board Books)
4.000000
A Heartbreaking Work of Staggering Genius
NaN
The Long Winter (Little House, #6)
3.933333
Eaters of the Dead
3.187500
Haroun and the Sea of Stories (Khalifa Brothers, #1)
NaN
Vampire Mountain (Cirque Du Freak, #4)
3.714286
Fahrenheit 451
4.125000
Twenty Love Poems and a Song of Despair
4.500000
Marina
4.000000
Portrait of a Killer: Jack the Ripper - Case Closed
4.000000
Shadow of the Giant (Ender's Shadow, #4)
4.000000
Sex and the City
3.923077
American Gods (American Gods, #1)
4.285714
dtype: float64
```