Lab for *Recommender System*

The Lab includes two parts:

1. The first part is the **practical exercises** for the learning topic.
2. **After** the Lab demonstrator **demonstrates** the Practical Exercises, **if** the students **have finished** the practical exercises, they can use the rest of the time as Help session for the current assignment and ask Questions:
    - Please note that the tutors are not allowed to assess your current assignment solution, and can not help you to develop your solution.
    - They will help your assignment in other ways, e.g. guide you to the right knowledge (e.g. the corresponding lecture slides or tutorial material), clarify questions you have about assignment specifications, and python programming problems, etc.
    - Please also ask any questions you have about the **course materials**, including **lecture slides** and **Lab materials**.

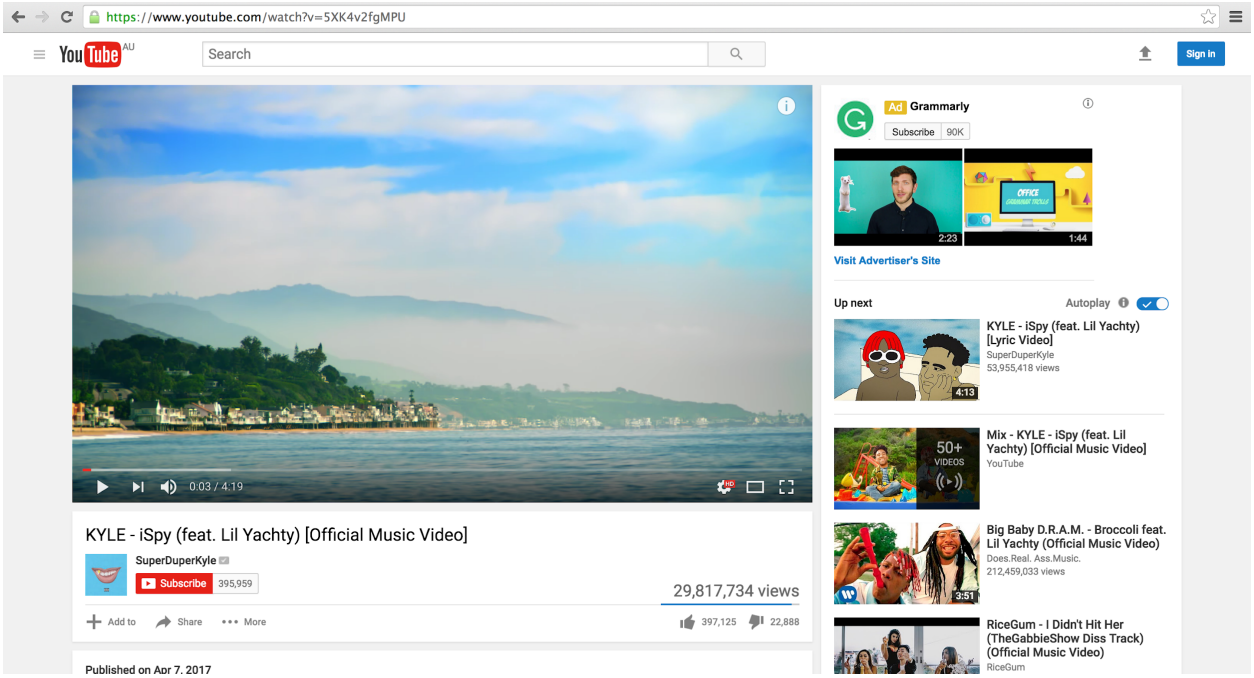
Practical exercise 1: Exploring A Real Recommender System

This week, let's first explore how Youtube makes recommendations for new users.

1. Launch Chrome, and create a new "incognito" window. (Or any other Web browser, but make sure that it is without logging into your youtube account, as we would like to investigate how Youtube recommends videos to new users.)
2. Go to youtube (https://www.youtube.com/)
3. Look at the homepage. Can you find:
    a. "**Trending**"
    b. "**Music:** **Hot Music, Top Genres**"
    c. "**Sports:** **Top Stories, Trending**"
    d. "**Gaming:** **Top live games, trending videos**"

    Please answer/discuss:
    - What is the meaning of each?
      Answer: For a, b, c, and d, "trending" means the current popular videos in each category, and "top" means the overall popular video in each category.
    - Why were the popular Music/Entertainment/Gaming-related videos recommended to you?
      Answer: This is because when you access Youtube from 'incognito' mode, there is no personal historical data to use for personalised recommendations. Then, these popularity-based recommendations were presented.

Please select one of the videos that you are interested in, then carefully check the videos listed on the right side of the page (as follows):



- Do you like the videos on the right side?
  Answer: Please check whether they are relevant and whether you like them or not. Then, try to discuss whether these recommendations are good enough or not.
- Can you tell why youtube recommend those videos to you?
  Answer: These recommendations are predicted by trying to understand what is your preference from those videos you just explored.
- What is the relationship between the video you selected, and these videos on the right?
  Answer: Youtube thinks the videos on the right matches your preference that they have extracted from what you have selected.
- Do you think Youtube is doing a good job here?
  Anser: Please discuss why.

Practical exercise 2: *The Long Tail in User-Item Rating Matrix*

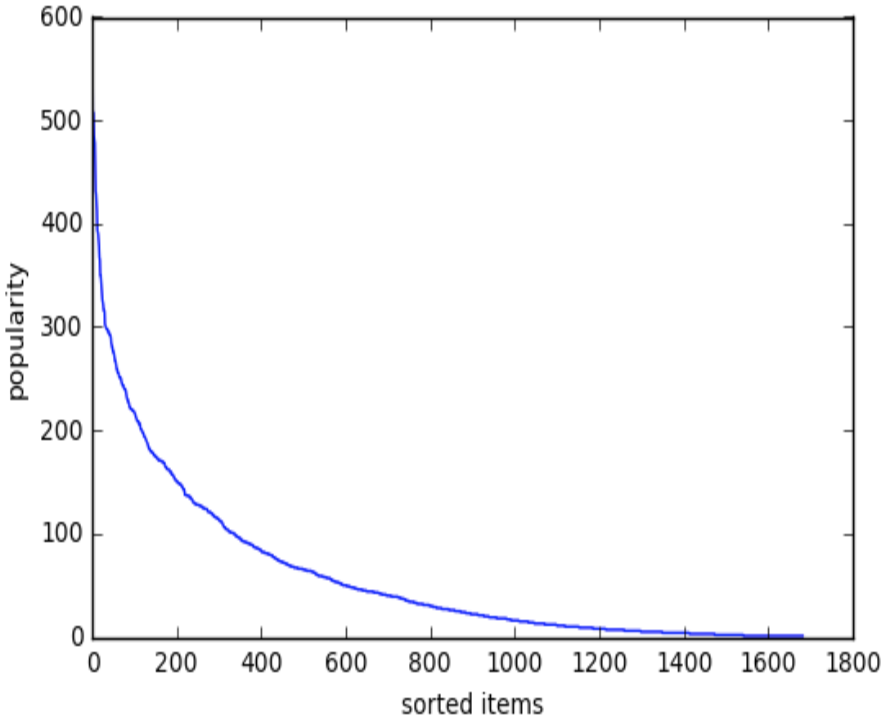
Next, we investigate the long tail effect in the user-item rating matrix. Please download the MovieLens dataset (ml-100k.zip) from Canvas to your current working directory. Then, decompress it.

```
In [1]: import numpy as np
In [2]: import pandas as pd
In [3]: cd ml-100k/
/Users/yongli/A-Teaching/PDS/code/recsys/ml-100k

In [4]: names = ['user_id', 'item_id', 'rating', 'timestamp']
In [5]: df = pd.read_csv('u.data', sep='\t', names=names)
In [6]: df.head()
Out [6]:
```

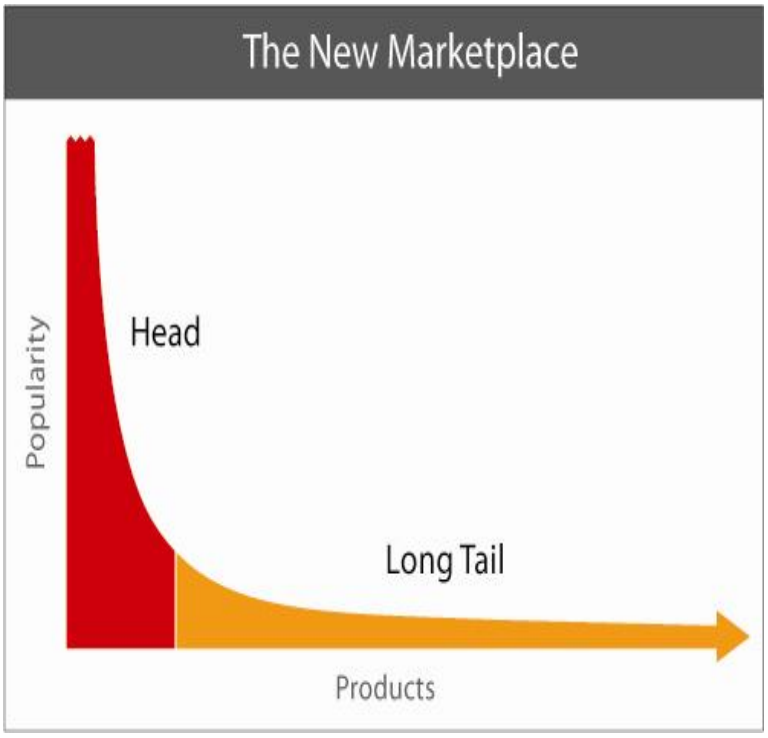
	user_id	item_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

```
In [7]: n_users = df.user_id.unique().shape[0]
In [8]: n_items = df.item_id.unique().shape[0]

In [9]: print(str(n_users) + ' users')
Out [9]: 943 users

In [10]: print(str(n_items) + ' items')
Out [10]: 1682 items

In [11]: ratingsNum = np.zeros((n_users, n_items))
In [12]: for row in df.itertuples():
             ratingsNum[row[1]-1, row[2]-1] = 1
In [13]: print(ratingsNum)
Out [13]:
[[ 1.  1.  1. ...,  0.  0.  0.]
 [ 1.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 ...,
 [ 1.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
```

```
 [ 0.  1.  0. ...,  0.  0.  0.]]

In [14]: itemRateNumCurrent = ratingsNum.sum(axis=0)
In [15]: itemRateNumCurrent.sort()
In [16]: import matplotlib.pyplot as plt
In [17]: plt.plot(itemRateNumCurrent[::-1])
In [18]: plt.xlabel('sorted items') # adds label to x axis
In [19]: plt.ylabel('popularity') # adds label to y axis
In [20]: plt.show()
```



Compare this long tail effect with the following one, and explain what this long tail theory means in the MovieLens data set?
(*Please revisit page 9 of Week 10's Lecture slides.*)



http://www.longtail.com/about.html

Practical exercise 3: *TopPop and MovieAvg*

```
In [21]: ratings = np.zeros((n_users, n_items))
In [22]: for row in df.itertuples():
             ratings[row[1]-1, row[2]-1] = row[3]
         print(ratings)

In [23]: ratingsNum = ratingsNum.sum(axis=0)
In [24]: itemRateSum = ratings.sum(axis=0)
In [25]: itemRateAvg = itemRateSum/ratingsNum
In [26]: print(itemRateAvg)
Out [26]:
[ 3.87831858  3.20610687  3.03333333 ...,  2.          3.          3.        ]

In [27]: i_cols = ['movie id', 'movie title' ,'release date','video release
date', 'IMDb URL', 'unknown', 'Action', 'Adventure', 'Animation',
'Children\'s', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller',
'War', 'Western']
In [28]: items = pd.read_csv('u.item', sep='|', names=i_cols,
encoding='latin-1')
```

```
In [29]: items.head()
Out [29]:
```

	movie id	movie title	release date	video release date	IMDb URL	unknown	Action	Adventure	Animation	Children's	...	Fantasy	Film-Noir	Horror	Mus
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	...	0	0	0	0
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0	...	0	0	0	0
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0	0	0	...	0	0	0	0
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0	0	0	...	0	0	0	0
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0	0	0	...	0	0	0	0

5 rows × 24 columns

```
In [30]: top_n = 5
In [31]: activeUser = 0
In [32]: mask_activeUser = ratings[activeUser, :] > 0
In [33]: itemRateNumCurrent = ratingsNum.copy()
In [34]: itemRateNumCurrent[mask_activeUser] = 0
In [35]: itemSortInd = itemRateNumCurrent.argsort()
In [36]: print('movie ID' + '\t movie title')
In [37]: print(items['movie title'][itemSortInd[:-1-top_n:-1]])
Out [37]:
movie ID      movie title
293               Liar Liar (1997)
285    English Patient, The (1996)
287                  Scream (1996)
299           Air Force One (1997)
312                 Titanic (1997)
Name: movie title, dtype: object

In [38]: mask_activeUser = ratings[activeUser, :] > 0
In [39]: itemRateAvgCurrent = itemRateAvg.copy()
In [40]: itemRateAvgCurrent[mask_activeUser] = 0
In [41]: itemSortInd = itemRateAvgCurrent.argsort()
In [42]: print('movie ID' + '\t movie title')
In [43]: print(items['movie title'][itemSortInd[:-1-top_n:-1]])
Out [43]:
movie ID      movie title
1535                                Aiqing wansui (1994)
1652    Entertaining Angels: The Dorothy Day Story (1996)
1200           Marlene Dietrich: Shadow and Light (1996)
```

```
1598                         Someone Else's America (1995)
1121                        They Made Me a Criminal (1939)
Name: movie title, dtype: object
```

After you build the above *TopPop* and *MovieAvg* recommender system, Please think/discuss:

1. Given the same active user, compare the difference between the recommendation list (`Out [37]` and `Out [43]`) from these two techniques.
2. Change the active user ID in `line [31]` to a different user, then try to generate the recommendation list for this user. Then, think which one is better.