Practical Data Science: Recommender Systems

Dr. Yongli Ren

(yongli.ren@rmit.edu.au)

Computer Science & IT

School of Science

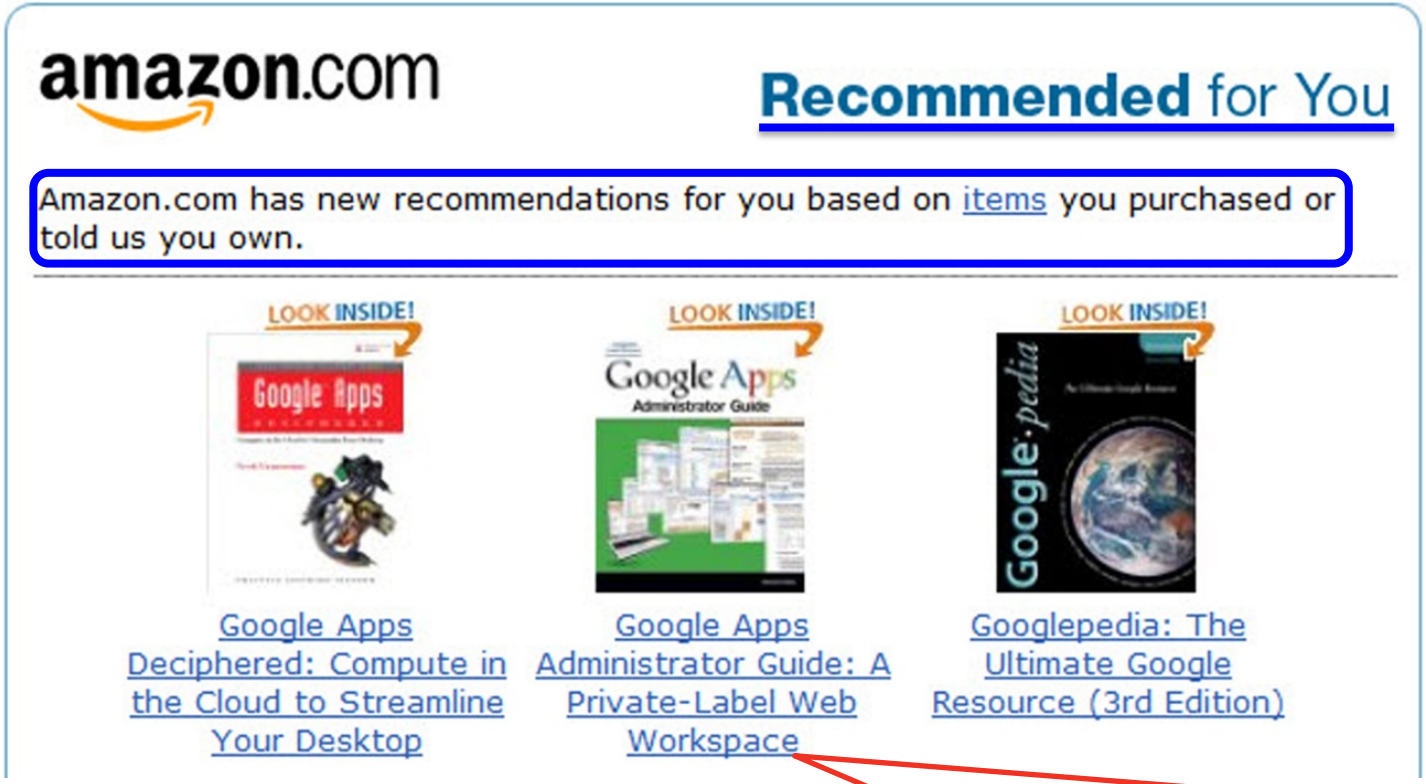
Outline

- Part 1: What is a Recommender System?
- Part 2: How to Do Recommendations?

Practical Data Science – COSC2670

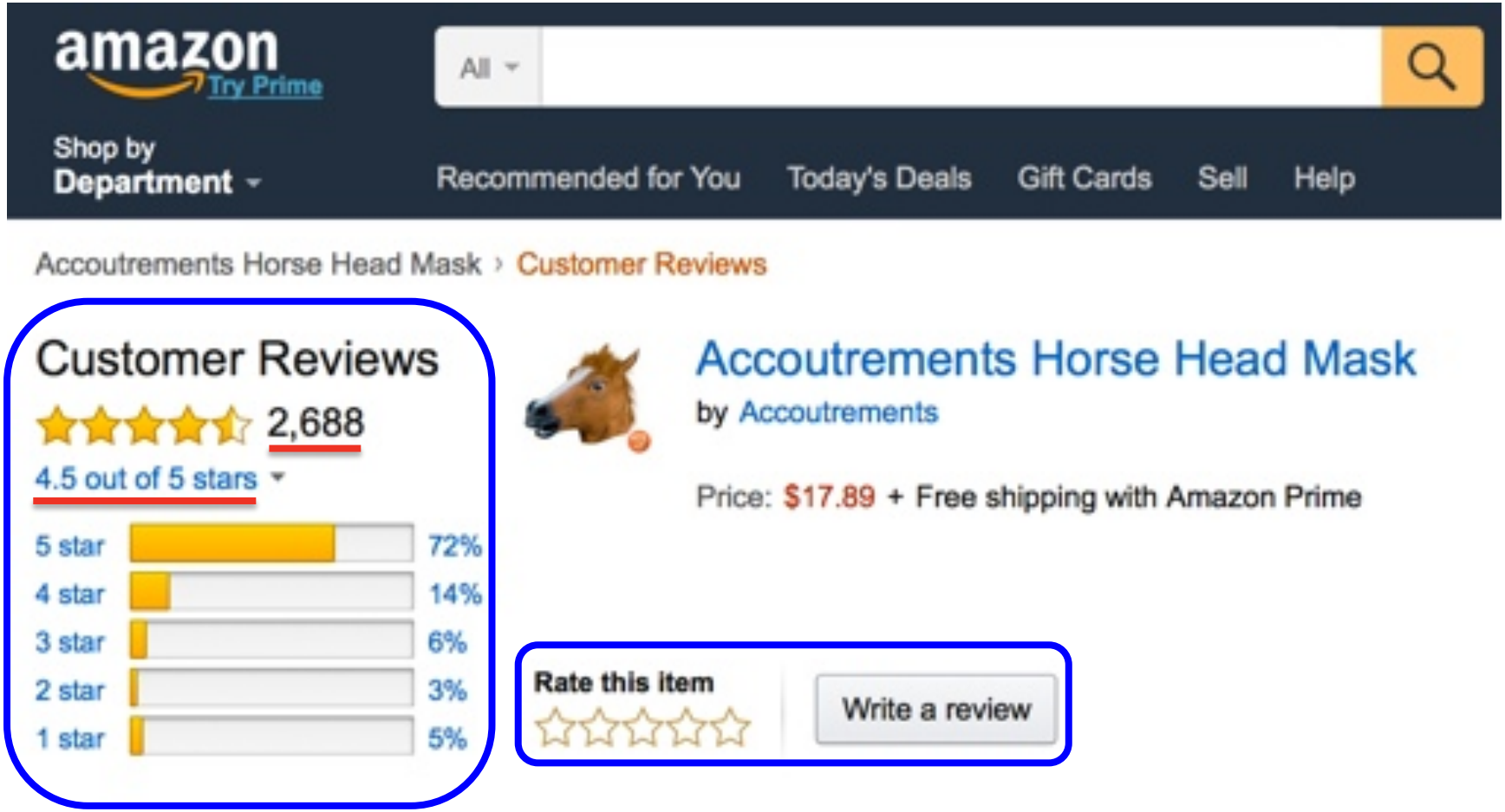
PART 1: WHAT IS A RECOMMENDER SYSTEM?

What is A Recommender System?
- Amazon Book Recommendation



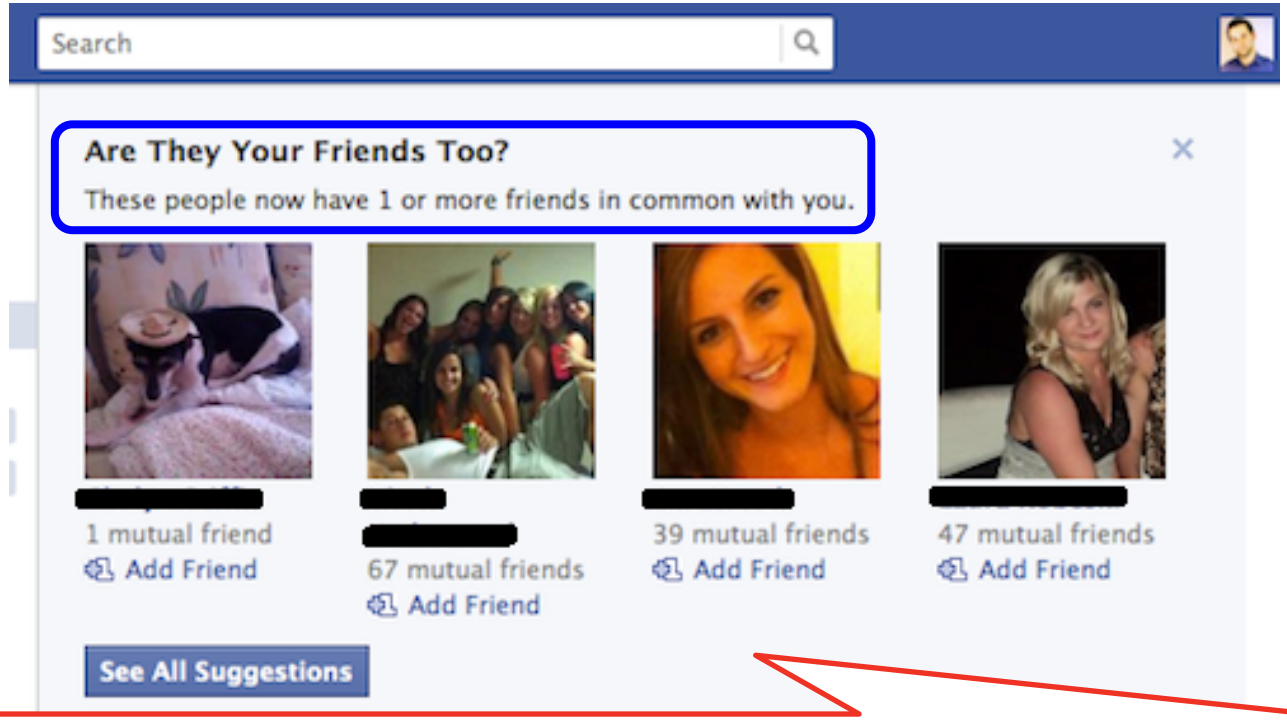
1. Amazon gets about a 35% increase in revenue due to its recommendation system.
2. Research has shown that sites that use predictive personalization experience 20-50% better revenue performance.
3. Amazon: with our *data science techniques*, the more you use us, the smarter we get.

What is A Recommender System?
- Amazon Customer Rating/Review



$5 * 0.72 + 4 * 0.14 + 3 * 0.06 + 2 * 0.03 + 1 * 0.05 \approx 4.5$

What is A Recommender System?
- Facebook Friend Recommendation



1. This kind of recommendation has slightly different goals than a product recommendation.
2. While a product recommendation directly increases the profit of the merchant by facilitating product sales, an increase in the number of social connections improves the experience of a user at a social network.
3. This, in turn, encourages the growth of the social network.
4. Social networks are heavily dependent on the growth of the network to increase their advertising revenues.

http://sproutsocial.com/insights/facebook-recommendations-2/
C.C. Aggarwal, Recommender Systems: The Textbook, Chapter 1, 2016

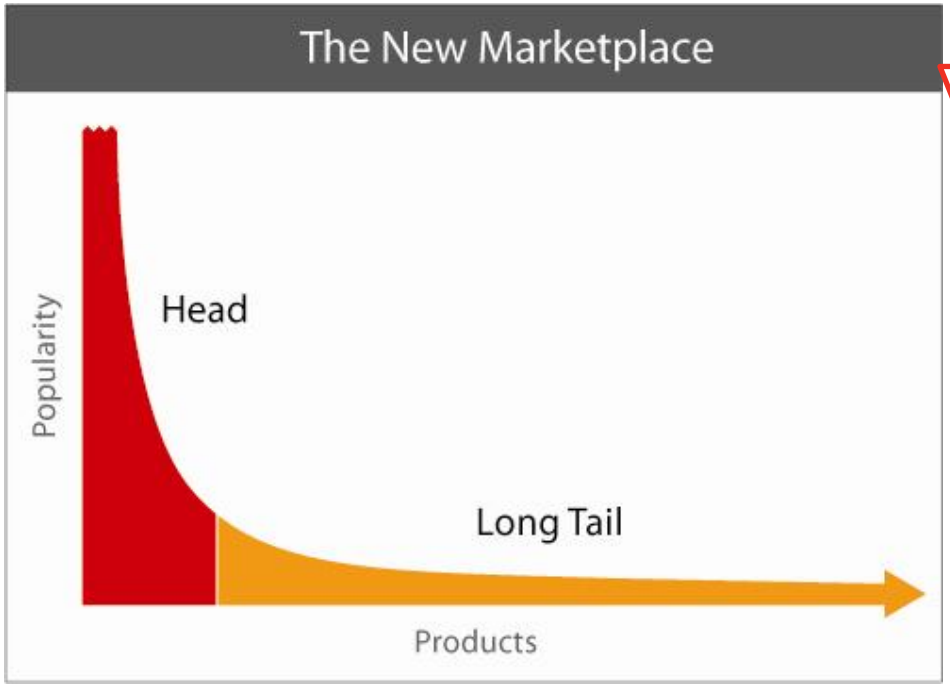
Netflix Prize
- US$1,000,000

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences:
Improvement over Netflix's own algorithm, called *Cinematch*, by 10%.

What is A Recommender System?

- It is a system
  - Attempting to recommend products,
    - which a user probably likes.
  - Recommendations are based on the user's preference profiles extracted from their consuming/rating history, including
    - The content of consumed items;
    - The explicit ratings given to those items.
  - The product could be:
    - Movies from IMDB/Netflix …
    - Videos from Youtube …
    - Books from Amazon …
    - Music…
    - ….

Why Recommendations?
- The Long Tail



The theory of the Long Tail is that:

Our culture and economy is increasingly shifting away from a focus on a relatively small number of "hits" (mainstream products and markets) at the head of the demand curve and toward a huge number of niches in the tail.

1. As the costs of production and distribution fall, especially online, there is now less need to lump products and consumers into one-size-fits-all containers.
2. In an era without the constraints of physical shelf space and other bottlenecks of distribution, narrowly-targeted goods and services can be as economically attractive as mainstream fare.
3. **Selling Less of More: products in low demand or that have a low sales volume can collectively build a better market share than their rivals.**

Why Recommendations?
- Purposes

- Information Overload:
  - is a term used to describe the difficulty of understanding an issue and effectively making decisions when one has too much information about that issue.
- To discover the large volume of long-tail items.
- Specifically, the purpose of recommendations:
  - For a particular user, recommend items he/she may be interested in.
    - Movie, book recommendations, …
  - For a particular item, identify potential users who may like it.
    - Personalised advertising, …
- Recommender System is an important topic in both
  - Academia / research:
    - It is still young, as it emerged in the 1990s.
  - Industry:
    - Very practical, especially for online retailers/organizations.

https://en.wikipedia.org/wiki/Information_overload

User Representation

- User-Item rating matrix.

Items. e.g. films

Users	Spider-Man	2012	The Godfather	The Social Network
Bob	3	2	4	5
Cindy	ø	3	5	4
Paul	2	3	4	ø
David	3	ø	4	1

rating

No rating

David did not watch '2012' ?

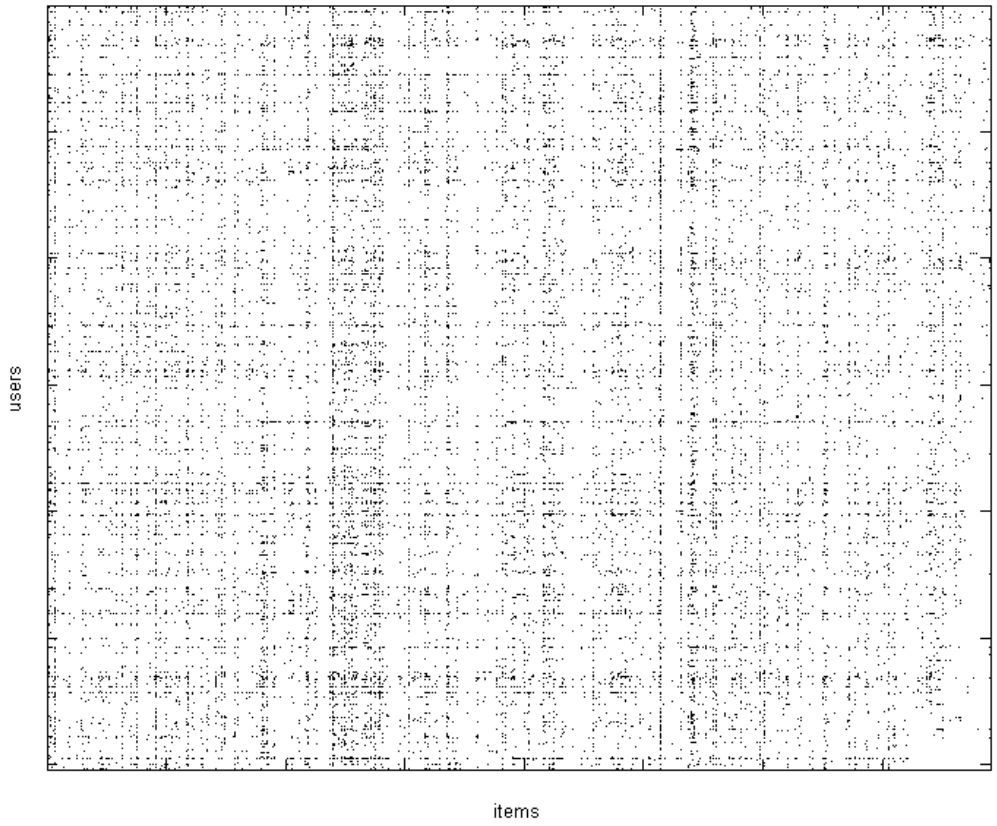
David does not like '2012' ?

- Users: a long vector with each cell value representing the users' rating on the corresponding item.
  - E.g. there are around 500,000 movies, so users in a movie recommender system is a vector of length 500,000.

Challenge: Data Sparsity

- Data Sparsity refers to
  - insufficient information on a user's rating history
- The corresponding user-item matrix
  - is extremely sparse
    (the density normally is around 1%)
  - This is mainly because users are only able to rate a small number of available items.
    - E.g. each user only watched/rated around 500 movies on average.

Challenge: Data Sparsity



MovieLens: 1M ratings from 6040 users on 3900 movies with a density 4.25%

Practical Data Science – COSC2670

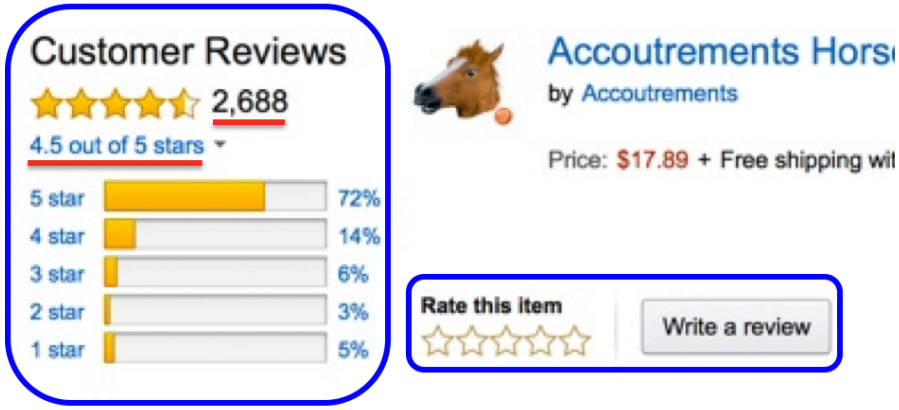
PART 2: HOW TO DO RECOMMENDATION?

How to Do Recommendations?

- Although peoples' tastes vary, they do follow patterns, which can be used to predict such likes and dislikes.
  - People tend to like things that are similar to other things they like.
  - People tend to like things that similar people like.
- Recommendation is all about
  - Estimating these patterns of taste, and
  - Discovering new and desirable items
    - That people didn't already know.

How to Do Recommendations?

- Non-personalized vs personalized
  - Non-personalized:
    - Recommended items are generated without considering a user's rating/consuming history.
      - TopPop (Top Popular): recommends items with the highest popularity (largest number of ratings).
      - MovieAvg (Movie Average): recommends top-N items with the highest average rating.
  - Personalized
    - Recommended items are generated based on a user's rating/consuming history.
      - Collaborative Filtering
      - Content-based Methods



How to Do Recommendations?
- Data From A Movie Recommender System

- MovieLens (100k)
  - This data set consists of:
    - 100,000 ratings (1-5) from 943 users on 1682 movies.
    - Each user has rated at least 20 movies.
    - Simple demographic info for the users (age, gender, occupation, zip)
- The data was collected
  - through the MovieLens web site (movielens.umn.edu)
  - during the seven-month period from September 19th, 1997 through April 22nd, 1998.
  - This data has been cleaned up
    - users who had less than 20 ratings or did not have complete demographic information were removed from this data set.

MovieLens
-Ratings and User Info

- u.data
  - The full u data set, 100000 ratings by 943 users on 1682 items.
  - Each user has rated at least 20 movies.
  - Users and items are numbered consecutively from 1.
  - The data is randomly ordered.
  - This is a tab separated list of
    - user id | item id | rating | timestamp.
    - The time stamps are unix seconds since 1/1/1970 UTC

- u.info
  - The number of users, items, and ratings in the u data set.

MovieLens
-Items (Movies)

- u.item
  - Information about the items (movies); this is a tab separated list of

    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 |
  - The last 19 fields are the genres, a 1 indicates the movie is of that genre, a 0 indicates it is not; movies can be in several genres at once.
  - The movie ids are the ones used in the u.data data set.

MovieLens
-Others

- u.genre
  - A list of the genres.

- u.user
  - Demographic information about the users;
  - this is a tab separated list of

    user id | age | gender | occupation | zip code
  - The user ids are the ones used in the u.data data set.

- u.occupation
  - A list of the occupations.

Recommendation with Python
- Non-personalized Methods

1. Load the data
   - u.data: ratings
   - u.item: movie information
2. Convert data into user-item rating matrix
3. Calculate the popularity of movies
4. Calculate the average rating of movies
5. Recommend the movies that the active user did not watch before, by using TopPop or MovieAvg.
6. Present the recommendations.

Recommendation with Python
- Load the Data

```
import numpy as np
import pandas as pd
```

Load packages

```
cd ml-100k/
```

Move into the data directory

```
/Users/yongli/A-Teaching/PDS/code/recsys/ml-100k
```

```
ls
```

List the files in the ml-100k folder

```
README          u.info          u1.test         u4.base         ua.test
allbut.pl*      u.item          u2.base         u4.test         ub.base
mku.sh*         u.occupation    u2.test         u5.base         ub.test
u.data          u.user          u3.base         u5.test
u.genre         u1.base         u3.test         ua.base
```

```
!head u.data
```

Print the first 10 lines in u.data:
user id | item id | rating | timestamp.

```
196     242     3       881250949
186     302     3       891717742
22      377     1       878887116
244     51      2       880606923
166     346     1       886397596
298     474     4       884182806
115     265     2       881171488
253     465     5       891628467
305     451     3       886324817
6       86      3       883603013
```

Recommendation with Python
- Load the Data

```
names = ['user_id', 'item_id', 'rating', 'timestamp']
df = pd.read_csv('u.data', sep='\t', names=names)
df.head()
```

	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

Load the rating data

Recommendation with Python - Load the Data

Load the item data

```
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']
items = pd.read_csv('u.item', sep='|', names=i_cols, encoding='latin-1')

items.head()
```

	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

Recommendation with Python
- Convert Data Into User-Item Rating Matrix

```
n_users = df.user_id.unique().shape[0]
n_items = df.item_id.unique().shape[0]
print(str(n_users) + 'users')
print(str(n_items) + 'items')
```

```
943 users
1682 items
```

Check the size of the data

```
ratings = np.zeros((n_users, n_items))
ratingsNum = np.zeros((n_users, n_items))
```

Construct the user-item rating matrix: ratings

Construct the user-item binary 'rating' matrix: ratingsNum

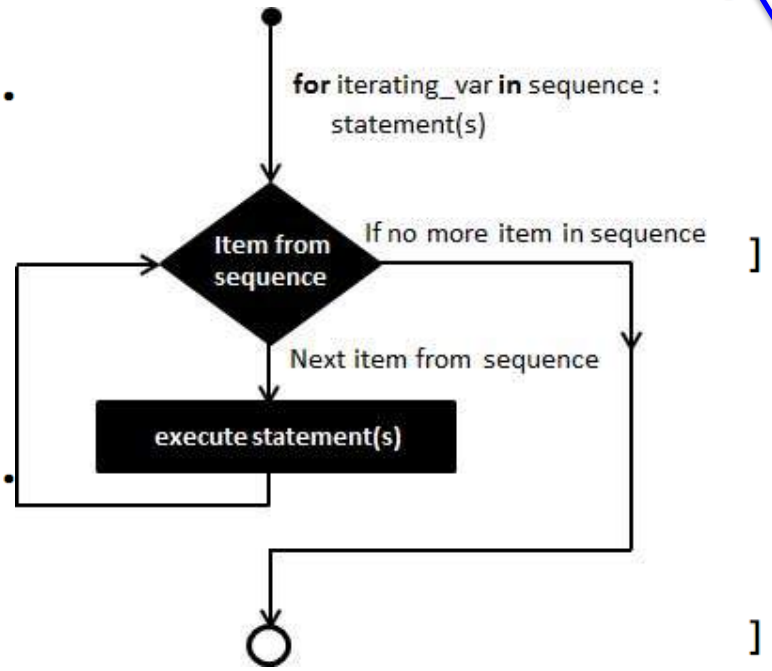
Recommendation with Python
- Convert Data Into User-Item Rating Matrix

```
for row in df.itertuples():
    ratings[row[1]-1, row[2]-1] = row[3]
    ratingsNum[row[1]-1, row[2]-1] = 1
print ratings
print ratingsNum
```

Note the indentation!

	user_id	item_id	rating	timestamp
0	196	242	3	881250949

```
[[ 5.  3.  4. ...,  0.  0.  0.]
 [ 4.  0.  0. ...,  0.  0.  0.]
 [
 ..
 [
 [
 [                                ]
[[
 [
 [
 ..
 [
 [
 [                                ]
```



- **for** is the name of the command
- An index variable is used to hold each of the different values of a sequence
- The word **in**
- A function the generates a sequence: the index variable will be the name for one value in the sequence, each time through the loop
- A colon (“**:**”)
- And a block (the indented lines of code)

Recommendation with Python
- Calculate Movie's Popularity/Average Rating

Calculate the popularity: the number of 'rated' times

```
itemRateNum = ratingsNum.sum(axis=0)
itemRateSum = ratings.sum(axis=0)
```

Calculate the total ratings obtained by each item

```
print(itemRateNum)
print(itemRateSum)
```

```
[ 452.  131.   90. ...,    1.    1.    1.]
[ 1753.   420.   273. ...,     2.     3.     3.]
```

```
itemRateAvg = itemRateSum / itemRateNum
print(itemRateAvg)
```

Calculate the average rating for each item

```
[ 3.87831858  3.20610687  3.03333333 ...,  2.          3.          3.        ]
```

Recommendation with Python - TopPop

Specify the number of recommended items
Specify the ID of the active user

```
top_n = 5
activeUser = 0
mask_activeUser = ratings[activeUser, :] > 0
itemRateNumCurrent = itemRateNum.copy()
itemRateNumCurrent[mask_activeUser] = 0
itemSortInd = itemRateNumCurrent.argsort()
print('movie ID' + '\t movie title')
print(items['movie title'][itemSortInd[list(range(len(itemSortInd) - 1, len(itemSortInd) - top_n - 1, -1))]])
```

Mask the 'rated' items for the active user

Make a copy of itemRateNum

Exclude rated items from recommendation by setting their popularity to 0

Sort Movies by their popularity

```
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
```

Recommend the top-N ranked items

As itemRateNumCurrent is sorted in ascending order, the top *n* ranked items are obtain by using the last *n* indexes: from *len(itemSortInd) – 1* to *len(itemSortInd)-top_n-1*

Sorted itemRateNumCurrent
itemSortInd

```
[   0    0    0 ..., 478 481 485]
[   0 172 173 ..., 287 285 293]
```

len(itemSortInd)

len(itemSortInd) – 1

Recommendation with Python - MovieAvg

Specify the number of recommended items
Specify the ID of the active user

```
top_n = 5
activeUser = 0
mask_activeUser = ratings[activeUser, :] > 0
itemRateAvgCurrent = itemRateAvg.copy()
itemRateAvgCurrent[mask_activeUser] = 0
itemSortInd = itemRateAvgCurrent.argsort()
print('movie ID' + '\t movie title')
print(items['movie title'][itemSortInd[list(range(len(itemSortInd) - 1, len(itemSortInd) - top_n - 1, -1))]])
```

Mask the 'rated' items for the active user

Make a copy of itemRateAvg

Exclude rated items from recommendation by setting their average rating to 0

Sort Movies by their average rating

```
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
```

Recommend the top-N ranked items

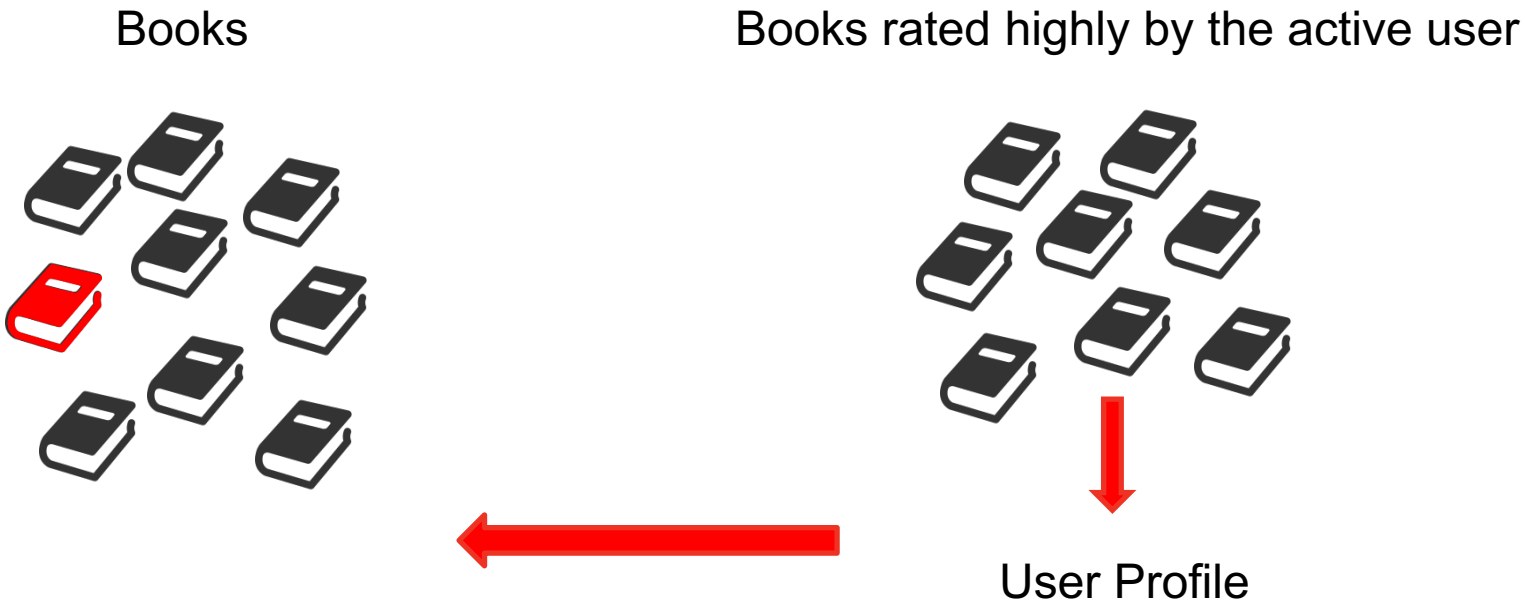
As itemRateAvgCurent is sorted in ascending order, the top n ranked items are obtain by using the last n indexes: from *len(itemSortInd) – 1* to *len(itemSortInd)-top_n-1*

How to Do Recommendations? - Personalized

- Personalized
  - According to the information used, personalized method are categorized as:
    - Content-based methods
      - Estimate the rating on an unknown item based on the content of that item and the contents of other rated items.
    - Collaborative Filtering
      - Predict the rating on an unknown item based on the ratings on that item by other similar users.
    - Hybrid Methods
      - Make rating prediction based on both item content and ratings.

Content-based Methods

- For the active user, content-based method
  - estimate the rating on an unknown item based on the content of that item and the contents of other rated items.



Content-based Methods

- The term “content” refers to the descriptions of the item.
- For example, consider a situation where
  - John has rated the movie Terminator highly.
  - The item description of Terminator contains similar genre keywords as other science fiction movies, such as Alien and Predator.
  - In such cases, these movies can be recommended to John.

Content-based Methods

- New User Problem
  - Refers to the difficulty of making recommendations to a new user who has no rating history
  - This means no item content can be used to produce recommendations.
- Limitation in item content
  - Extracting content from some items is much more difficult than extracting from text documents,
    - E.g. multi-media resources (songs, videos, pictures) …

References and Further Reading

- Xiaoyuan Su and Taghi M. Khoshgoftaar, “A Survey of Collaborative Filtering Techniques,” Advances in Artificial Intelligence, vol. 2009, Article ID 421425, 19 pages, 2009. doi:10.1155/2009/421425
- Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. on Knowl. and Data Eng. 17, 6 (June 2005), 734-749. DOI: https://doi.org/10.1109/TKDE.2005.99
- Yongli Ren, Gang Li, Wanlei Zhou: A survey of recommendation techniques based on offline data processing. Concurrency and Computation: Practice and Experience 27(15): 3915-3942 (2015)

Data
Science

Thanks!