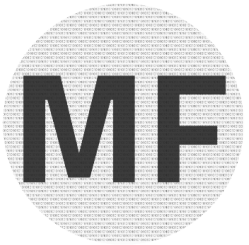
http://bit.ly/2nHFCWY

Introduction to Recommendation Systems and Collaborative Filtering

Manuel Ignacio Franco Galeano

maigfrga@gmail.com



February 1, 2018

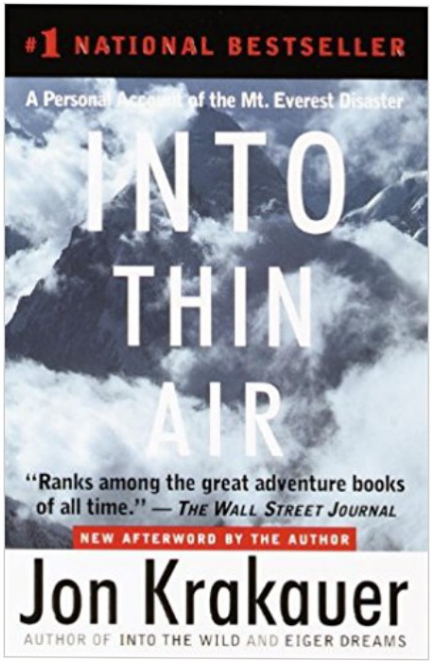
https://goo.gl/forms/6EGgxNQzC8d2elT12

A Simple Collective Intelligence Game

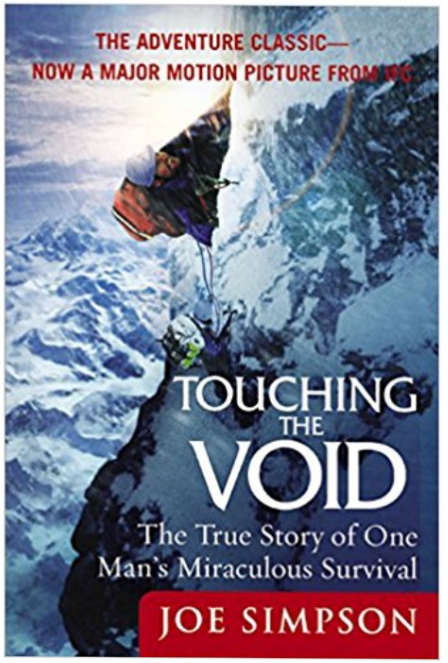
How Many Candies Are Inside The Jar?

https://goo.gl/forms/6EGgxNQzC8d2elT12

The Story Of Two Books



Into Thin Air
http://amzn.to/2rDaj4i



Touching the Void
http://amzn.to/2GgX3Wm

https://goo.gl/forms/6EGgxNQzC8d2elT12

Touching the Void(1988)

- The True Story of One Man's Miraculous Survival.
- Modest Success.

Into Thin Air (2008)

- A Personal Account of the Mt. Everest Disaster.
- Immediate Success.

https://goo.gl/forms/6EGgxNQzC8d2elT12

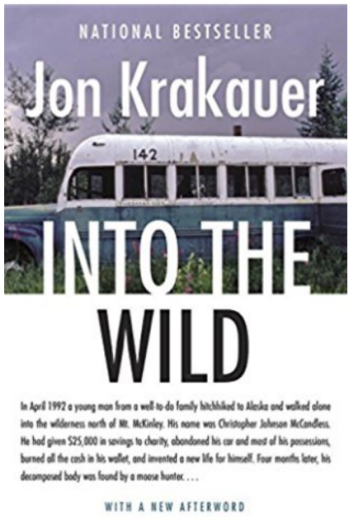
Touching the Void Unexpected Sales Increase

- Book scale up to the New York Times Best Seller List for 14 weeks.
- Why a book suddenly increase its sales after 10 years of modest success?

https://goo.gl/forms/6EGgxNQzC8d2elT12

The Power Of Recommendations

Customers who bought this item also bought

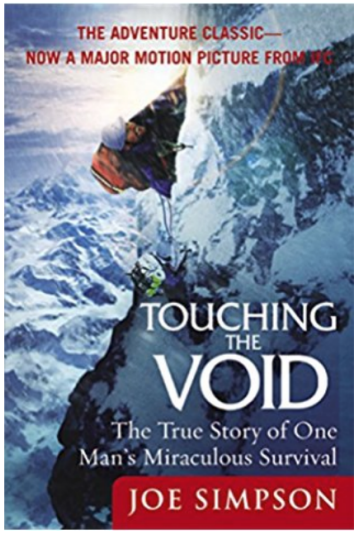


Into the Wild

› Jon Krakauer

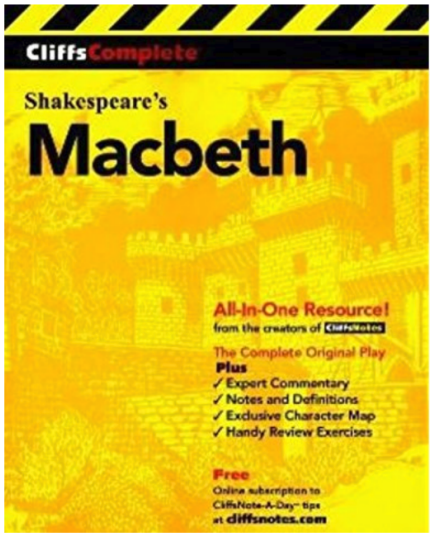
3,030

Paperback



Touching the Void: The True Story of One Man's Miraculous Survival

› Joe Simpson



CliffsComplete Macbeth

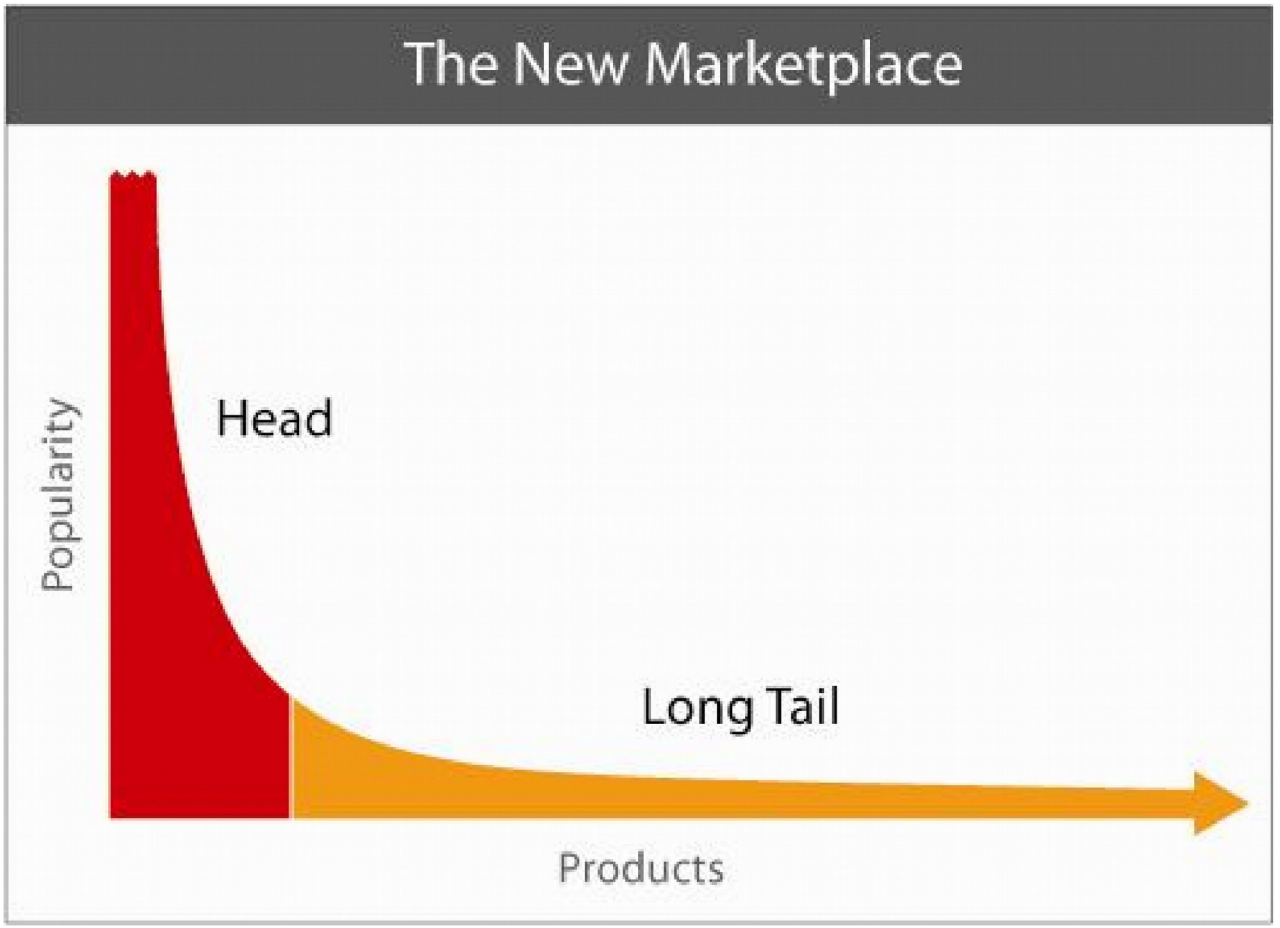
› William Shakespeare

13

Paperback

https://goo.gl/forms/6EGgxNQzC8d2elT12

The Long Tail Economy



https://goo.gl/forms/6EGgxNQzC8d2elT12

The Long Tail Economy

T-shirt
http://amzn.to/2Eb8aym

Nirvana T-shirt
http://amzn.to/2FX1ODw

https://goo.gl/forms/6EGgxNQzC8d2elT12

Recommendation Systems Help People To Access Goods And Services In The Long Tail

https://goo.gl/forms/6EGgxNQzC8d2elT12

Types of Recommendation Systems (Among Others)

- Non Personalized Recommendation Systems.
- Personalized Recommendation Systems.
- Content Based Recommendation Systems.
- Case Based Recommendation Systems.

https://goo.gl/forms/6EGgxNQzC8d2elT12

Non Personalized Recommendation Systems

- Personal user preferences are ignored.
- Easy to implement.
- Data is easy to collect.
- Generic recommendations (no customized to specific user needs).

https://goo.gl/forms/6EGgxNQzC8d2elT12

Personalized Recommendation Systems

- Individual preferences are used in order to generate recommendations
- Better Accuracy
- Privacy concerns as user data collection is required

https://goo.gl/forms/6EGgxNQzC8d2elT12

Personalized Recommendation Systems

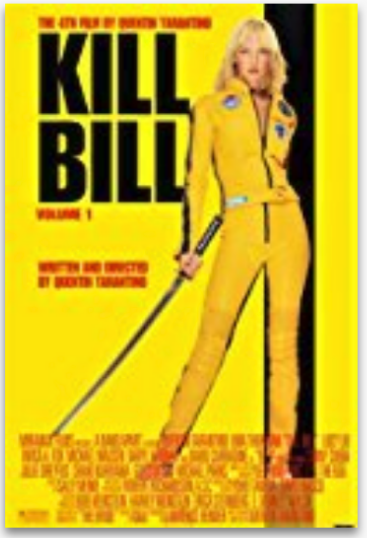
https://goo.gl/forms/6EGgxNQzC8d2elT12

Personalized Recommendation Systems

People who liked this also liked...

Learn more


◄ Prev 6 Next 6 ►



Add to Watchlist

Next »

Kill Bill: Vol. 1 (2003)

R Action | Crime | Thriller

8.1/10

The Bride wakens from a four-year coma. The child she carried in her womb is gone. Now she must wreak vengeance on the team of assassins who betrayed her - a team she was once part of.

Director: Quentin Tarantino
Stars: Uma Thurman, David Carradi…

https://goo.gl/forms/6EGgxNQzC8d2elT12

Content Based Recommendation Systems

	Headline
1	Open source software in the government has saved millions of dollars to the tax payers.
2	Increase of budget for public education has boost the productivity of the country.
3	World cup has been cancelled as people decide to focus on what is really important.

Table 1.1: Similarity Between Headlines in Content Based Recommendation Systems

Table 1.1 shows 3 examples of news, two of them Are similar because they are about economics, the third one is different because is about sports.

https://goo.gl/forms/6EGgxNQzC8d2elT12

Collaborative Filtering

Collaborative filtering is a technique used in recommendation systems for two purposes:

- Generate a list of similar items.
- Generate predictions for specific items.

https://goo.gl/forms/6EGgxNQzC8d2elT12

User-Based Collaborative Filtering

Collaborative Filtering Based On User Preferences

https://goo.gl/forms/6EGgxNQzC8d2elT12

User-Based Collaborative Filtering

- Build user-item matrix.
- Compute similarity among users by using statistical correlations.
- Build a subset of similar users.
- Generate a prediction or a list of items.

https://goo.gl/forms/6EGgxNQzC8d2elT12

User-Item Matrix

- A data structure used to store all ratings for all users

	Item 1	Item 2	Item 3	Item 4	..:.	Item n
User 1	3	2			...	3
User 2	1		4	2	...	3
...	...	...	...	...	...	...
User N	5	3		1	...	4

Table 2.1: User-Item Matrix Example

https://goo.gl/forms/6EGgxNQzC8d2elT12

Matrix Sparsity

- Not all users have a rating for every item.
- Majority of items in the long tail are not rated for every user.
- A matrix with this structure is known as sparse matrix.
- In this type of matrix, almost all items are null or 0.

https://goo.gl/forms/6EGgxNQzC8d2elT12

Perform operations on very sparse matrix is computational expensive, therefore, special data structures suitable to deal with this kind of data are required.

https://goo.gl/forms/6EGgxNQzC8d2elT12

Matrix Density

$$density = totalRatings/totalPosibleRatings$$

Density closer to 0 for a given matrix means that the matrix is very sparse.

https://goo.gl/forms/6EGgxNQzC8d2elT12

User Similarity Computation

Social information filtering techniques compute similarities between users in order to generate recommendations.

https://goo.gl/forms/6EGgxNQzC8d2elT12

Mean Square Differences

$$sim(u_i, u_j) = \sum_{\forall item_k \in corated(u_i, u_j)} \frac{(rating(u_i, item_k) - rating(u_j, item_k))^2}{|corated(u_i, u_j)|} \quad (2.2)$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Mean Square Differences

```
import math
import statistics


def msd(x, y):
    """
    Mean square difference
    https://en.wikipedia.org/wiki/Mean_squared_displacement
    """
    assert len(x) == len(y)
    assert len(x) > 0

    items = [
        (x[0] - x[1]) ** 2 for x in zip(x, y)
    ]
    return sum(items) / len(items)
```

https://goo.gl/forms/6EGgxNQzC8d2elT12

Cosine Similarity

$$sim(u_i, u_j) = cos(u_i, u_j) = \frac{\sum_{\forall item_k \in corated(u_i, u_j)} r(u_j, item_k) \times r(u_j, item_k)}{\sqrt{\sum_{\forall item_k \in u_i} r(u_i, item_k)^2} \times \sqrt{\sum_{\forall item_k \in u_j} r(u_j, item_k)^2}} \quad (2.3)$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Cosine Similarity

```
def cosine(x, y):
    """
    cosine similarity
    https://en.wikipedia.org/wiki/Cosine_similarity
    """
    assert len(x) == len(y)
    above = []
    below_x = []
    below_y = []

    for i in zip(x, y):
        above.append(i[0] * i[1])
        below_x.append(i[0] ** 2)
        below_y.append(i[1] ** 2)

    below = math.sqrt(sum(below_x)) * math.sqrt(sum(below_y))

    if below == 0:
        return 0
    else:
        return sum(above) / below
```

https://goo.gl/forms/6EGgxNQzC8d2elT12

Euclidean Distance

$$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} \tag{2.4}$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Euclidean Distance

```
def euclidean(x, y):
    """
    euclidean simiarity
    https://en.wikipedia.org/wiki/Euclidean_distance
    """
    assert len(y) == len(y)

    return math.sqrt(sum([
        (i[0] - i[1]) ** 2 for i in zip(x, y)
    ]))
```

https://goo.gl/forms/6EGgxNQzC8d2elT12

Pearson Correlation

	Item 1	Item 2	Item 3	Item 4	..:.	Item n
User 1	2	5	3	1	...	3
User 2	1	4.5	4	2	...	3

Table 2.2: Users with Similar Rating Behaviour

This measure is able to detect linear dependence (correlation) between two variables X and Y.
Therefore, it can be used to asses similarity between two users that rate items in a similar way.

$$sim(u_i, u_j) = \frac{\sum_{\forall item_k \in corated(u_i,u_j)} (r(u_i, item_k) - \overline{r}(u_i)) \times (r(u_j, item_k) - \overline{r}(u_j))}{\sqrt{\sum_{\forall item_k \in corated(u_i,u_j)} (r(u_i, item_k) - \overline{r}(u_i))^2} \times \sqrt{\sum_{\forall item_k \in corated(u_i,u_j)} (r(u_j, item_k) - \overline{r}(u_j))^2}} \quad (2.5)$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Pearson Correlation

	Item 1	Item 2	Item 3	Item 4	..:.	Item n
User 1	2	5	3	1	...	3
User 2	1	4.5	4	2	...	3

Table 2.2: Users with Similar Rating Behaviour

This measure is able to detect linear dependence (correlation) between two variables X and Y.
Therefore, it can be used to asses similarity between two users that rate items in a similar way.

$$sim(u_i, u_j) = \frac{\sum_{\forall item_k \in corated(u_i,u_j)} (r(u_i, item_k) - \overline{r}(u_i)) \times (r(u_j, item_k) - \overline{r}(u_j))}{\sqrt{\sum_{\forall item_k \in corated(u_i,u_j)} (r(u_i, item_k) - \overline{r}(u_i))^2} \times \sqrt{\sum_{\forall item_k \in corated(u_i,u_j)} (r(u_j, item_k) - \overline{r}(u_j))^2}} \quad (2.5)$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Pearson Correlation

```
def pearson(x, y):
    """
    https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
    """
    assert len(y) == len(y)

    avg_x = statistics.mean(x)
    avg_y = statistics.mean(y)

    above = []
    below_x = []
    below_y = []

    for i in zip(x, y):
        above.append((i[0] - avg_x) * (i[1] - avg_y))
        below_x.append((i[0] - avg_x) ** 2)
        below_y.append((i[1] - avg_y) ** 2)
    below = math.sqrt(sum(below_x)) * math.sqrt(sum(below_y))

    if below == 0:
        return 0
    else:
        return sum(above) / below
```

https://goo.gl/forms/6EGgxNQzC8d2elT12

Pearson Correlation

Returns a value between -1 and 1. A value of -1 means perfect negative correlation, i.e, two users are completely different. A value of 1 means perfect correlation, therefore,two users are very similar. A value of 0 means no correlation at all.

https://goo.gl/forms/6EGgxNQzC8d2elT12

Neighborhood Formation

- A neighborhood is a subset of similar users.
- A common approach used to build a neighbourhood is to use K Nearest-neighbor (KNN) algorithm.
- The size of K must be chosen by experimentation and it may vary depending of the domain.
- A small neighbourhood size may result in bad accuracy for users that do not have close neighbors, whereas a big size may lead reduced accuracy by ignoring the influence of closest neighbors

https://goo.gl/forms/6EGgxNQzC8d2elT12

Making Predictions

Mean-Rating Prediction

Relies in the wisdom of the crowd to predict a rating for a given
pair (user, item)

$$Prediction(user_i, item_k) = \frac{\sum_{r \in ratings(item_k)} r}{|ratings(item_k)|} \quad (2.6)$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Weighted Average Approach

Computes a weighted average of the users ratings for the item.

$$Prediction(user_i, item_k) = \frac{\sum_{r \in ratings(item_k)} sim(u_i, u_j) \times r}{|sim(u_i, u_j)|} \quad (2.7)$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Deviation from Mean Approach

Resnick's prediction technique. This equation normalizes all the ratings for a given user by subtracting the mean rating for all ratings.

$$prediction(u_i, item_k) = \overline{r}(u_i) + \frac{\sum_{u_j \in neighbourhood(u_j)} (r(u_j, item_k) - \overline{r}(u_j)) \times sim(u_i, u_j)}{\sum_{u_j \in neighbourhood(u_j)} |sim(u_i, u_j)|} \quad (2.8)$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Making Recommendations

A list of items is created by taking the union of all items that has been rated for other users in the neighborhood. The subset of items that have been rated for the active user should be excluded from the recommendation list. The remainder list may be ranked by rating frequency, mean rating across neighbors, predicted rating, etc.

https://goo.gl/forms/6EGgxNQzC8d2elT12

Making Recommendations

- Most Frequent Item: This mechanism returns a ranked list ordered by most frequent items across users in the same neighborhood.
- Linked Item: Ranks a list of items based mean of the ratings across neighbours.
- Predictor Recommendation: It uses a collaborative filtering predictor to rank a list of items based on predictions.

https://goo.gl/forms/6EGgxNQzC8d2elT12

Collaborative Filtering Advantages and Disadvantages

- Reflects user behavior by making predictions and recommendations based in user preferences.
- Prone to the cold start problem (lack of initial ratings).
- The neighbourhood formation is problematic for users that have no ratings.
- Sparsity may lead to escalability may lead to problems when the number of users is very high.

https://goo.gl/forms/6EGgxNQzC8d2elT12

MovieLens Dataset Analysis

- Movielens is a non-commercial, personalized movie recommendations dataset suitable for research and education.

Total Users	Total Movies (Items)	Total Tags	Matrix Density
671	9125	1296	0.0163

Table 3.1: Movielens Dataset Reduced Version Total Records

The density of the user-item matrix suggests that the matrix is very sparse.

https://goo.gl/forms/6EGgxNQzC8d2elT12

MovieLens Dataset Analysis

- Table 3.2 shows statistics about the number of ratings by users. Data suggest that even tough, every user has rated 149 movies in average, the reality is that the number of ratings by user is highly variable. This fact may explain the sparsity of the matrix.

Max Ratings	Min Ratings	Median Ratings	Mean Ratings	Standard Deviation Ratings
2391	20	71	149	231

Table 3.2: Ratings by User Statistics for MovieLens Dataset Reduced Version

https://goo.gl/forms/6EGgxNQzC8d2elT12

MovieLens Dataset Analysis

- Data suggest that users do not provide negative feedback as often as they provide positive ratings.

0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
1101	3326	1687	7271	4449	20064	10538	28750	7723	15095

Table 3.3: Number of Movies by Rating

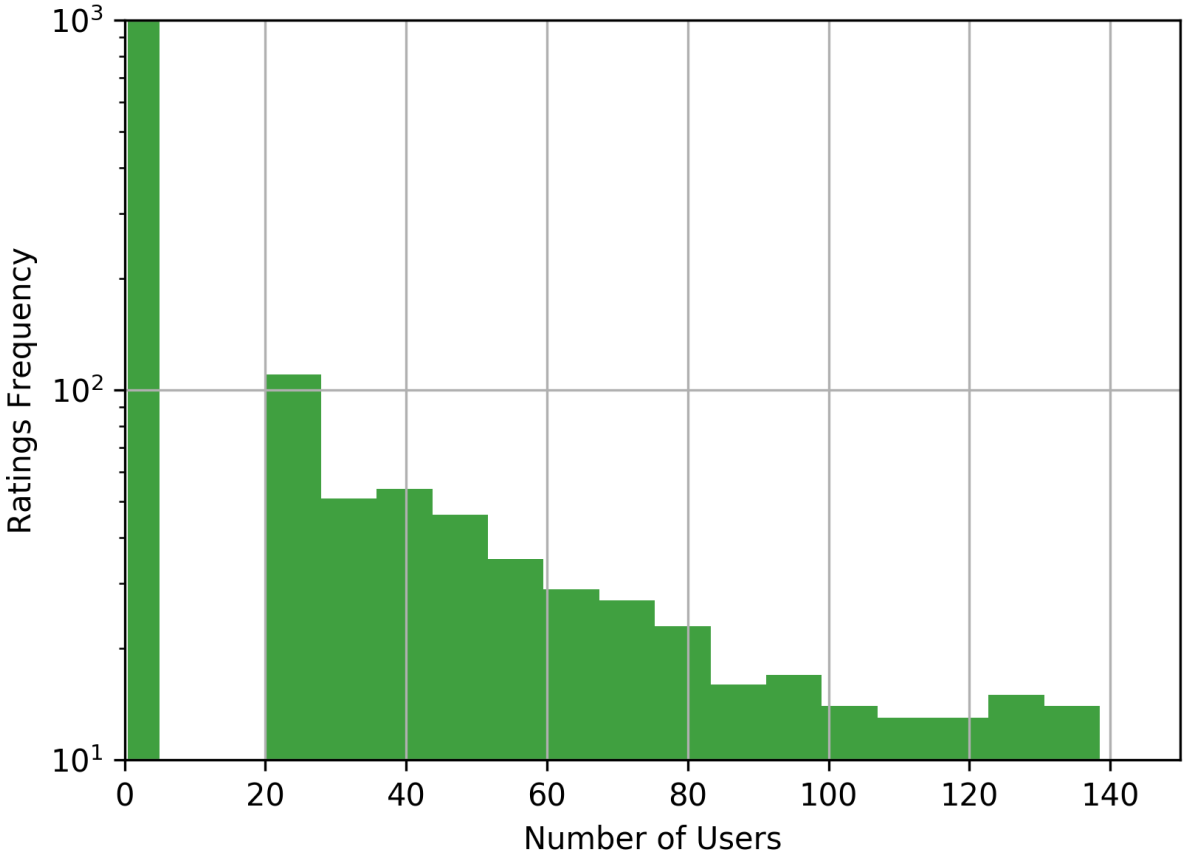
https://goo.gl/forms/6EGgxNQzC8d2elT12

MovieLens Dataset Analysis

- Distribution of ratings by user in the dataset.
- Only a tiny amount of users has rated more than 1,000 movies.
- The majority of the users have rated less 100 movies.
- The number of ratings by user is highly variable as the high standard deviation suggests.

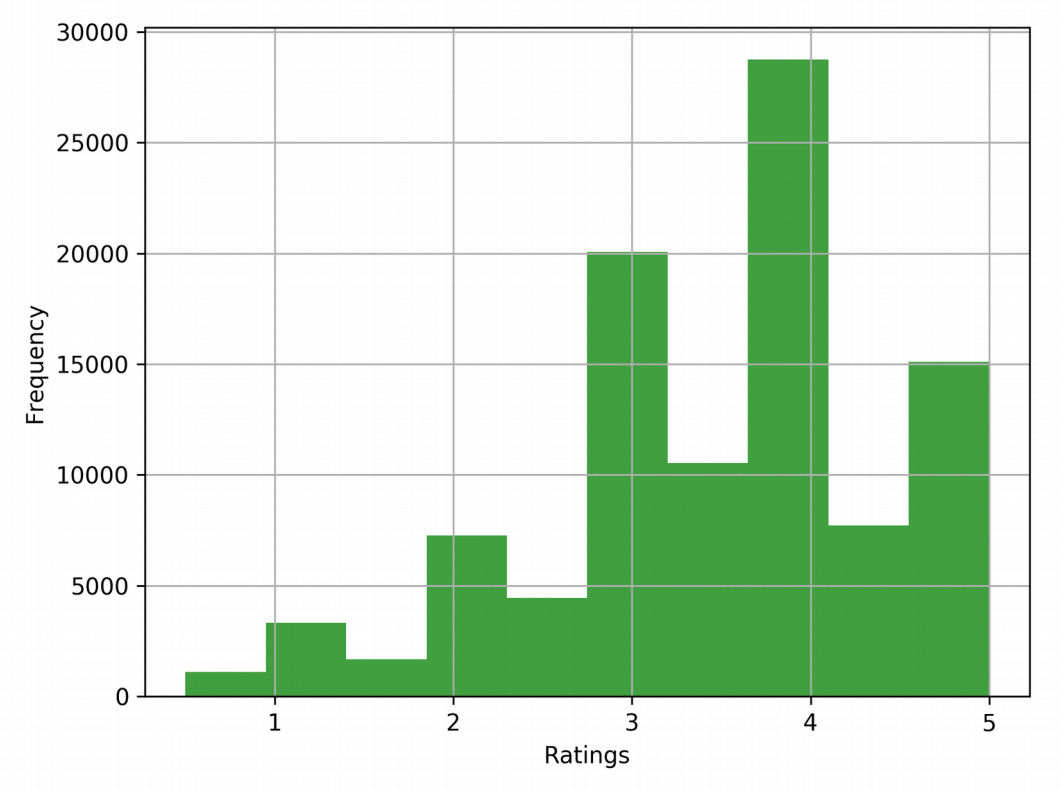
https://goo.gl/forms/6EGgxNQzC8d2elT12

MovieLens Dataset Analysis



https://goo.gl/forms/6EGgxNQzC8d2elT12

MovieLens Dataset Analysis



Most popular rating in the dataset is close to 4.0. Data suggest that people tend to rate in a positive way

https://goo.gl/forms/6EGgxNQzC8d2elT12

Recommendations And Collaborative Filtering Performance Analysis

https://goo.gl/forms/6EGgxNQzC8d2elT12

Evaluation Criteria For Predictors

- Mean Squared Error (MSE): This metric squares the differences between the predicted and the real rating. One of the goals of each one of the collaborative filtering techniques is to minimize this value as much as possible.

$$|\overline{MSE}| = \frac{\sum_{i=1}^{n} |p_i - r_i|^2}{N} \qquad (4.1)$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Evaluation Criteria For Predictors

- Root Mean Square Error (RMSE): This metric uses the same scale of the ratings.

$$|\overline{E}| = \sqrt{\frac{\sum_{i=1}^{n} |p_i - r_i|^2}{N}} \qquad (4.2)$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Evaluation Criteria For Recommenders

- Precision: This metric focuses on the exactness of the results, therefore, it measures the effectiveness of a recommendation.

$$precision = \frac{relevant\_documents \bigcap retrieved\_documents}{retrieved\_documents} \tag{4.3}$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Evaluation Criteria For Recommenders

- Recall: This metrics focuses in the completeness of the results. It represents the probability of a relevant item to be recommended.

$$recall = \frac{relevant\_documents \bigcap retrieved\_documents}{relevant\_documents} \quad (4.4)$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Evaluation Criteria For Recommenders

Precision and Recall, represent conflicting properties. F1 combine both metrics.

$$F_1 = \frac{2 \times Recall \times Presicion}{Recall + Presicion} \tag{4.5}$$

Coverage measures the percentage of users where a prediction or recommendation was possible

$$coverage = \frac{|ratings\_where\_predition\_was\_possible|}{|reviews|} \tag{4.6}$$

https://goo.gl/forms/6EGgxNQzC8d2elT12

Predictors Benchmark

Predictor	Similarity Metric	Neighbourhood Size	Coverage	RMSE	Total Execution Time (Minutes)
Mean Predictor	-	-	0.849	1.41	62
Collaborative Filtering	Pearson	100	0.729	1.55	609
Collaborative Filtering	MSD	100	0.193	3	703
Resnik Collaborative	Pearson	100	0.312	3.56	773

Table 4.2: Predictors Benchmark For Movielens Dataset

https://goo.gl/forms/6EGgxNQzC8d2elT12

Recommendation Benchmark

Recommendation	Neighbourhood size	Similarity	Precision	Recall	F1	Total Execution Time (Minutes)
Frequent Item Recommendation	100	Pearson	0.307	0.058	0.090	767
Linked Item Recommendation	10	Pearson	0.013	0.007	0.007	555
Linked Item Recommendation	100	Pearson	0.006	0.001	0.002	619
Mean Predictor Recommendation	10	Pearson	0.013	0.006	0.007	460
Collaborative Predictor Recommendation	10	Pearson	0.039	0.031	0.029	731.80
Collaborative Predictor Recommendation	100	Pearson	0.015	0.004	0.005	648
Resnik Predictor Recommendation	10	Pearson	0.064	0.063	0.053	486
Resnik Predictor Recommendation	100	Pearson	0.039	0.024	0.025	516

Table 4.3: Predictors Benchmark For Movielens Dataset

https://goo.gl/forms/6EGgxNQzC8d2elT12

nt-recommend: Experimentation Framework for Learning Recommender Systems

- http://bit.ly/2E3UHIH
- This project is a basic recommender system experimentation framework built in python3. It covers collaborative filtering initially, but it can be extended to support other recommendation techniques. It is not compatible with python2.
- This framework is only for educational proposes, we use basic native python data structures, we have implemented all the algorithms by ourselves with the only aim of learning, therefore, this project is not suitable for production environments.

https://goo.gl/forms/6EGgxNQzC8d2elT12

Conclusions

- Collaborative filtering models may have limited performance when the user-item matrix is very sparse.
- Wisdom of the crowd (average rating approach) is good enough for an initial prototype.
- Neighbourhood formation may be computational expensive (Not suitable for real time systems)

https://goo.gl/forms/6EGgxNQzC8d2elT12

Conclusions

- Hybrid approaches combining different recommendation techniques may improve recommendation and prediction performance.
- Performance assessment in production systems may be achieved by using A/B testing techniques.
- The experimentation framework that we use to implement the different techniques has been released as open source software and it is available for its usage in further researches.

https://goo.gl/forms/6EGgxNQzC8d2elT12

Thanks

https://goo.gl/forms/6EGgxNQzC8d2elT12