SINGAPORE INSTITUTE OF TECHNOLOGY

# Introduction to Recommender Systems and Collaborative Filtering

**Lucas Vinh Tran**

16 March 2023

# About Me

- Principal Scientist / Director of Personalization at JPMorgan Chase; Previous at Apple
- Lead Personalization and Recommendation team
- Before:
  - Ph.D. in Computer Science (Personalized and Group Recommender Systems)

# Our Aims

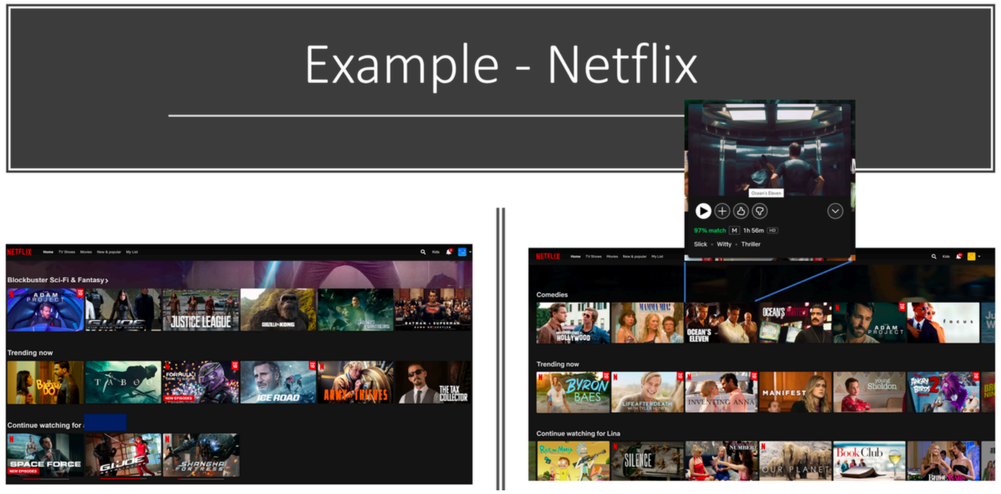
- It is not possible to cover every as aspect of the recommender systems in deep within 3 lectures
- We will try to cover a board spectrum of recommender systems, from traditional algorithms such as collaborative filtering, matrix factorization, deep learning, … to recommender systems in industry

# Lectures

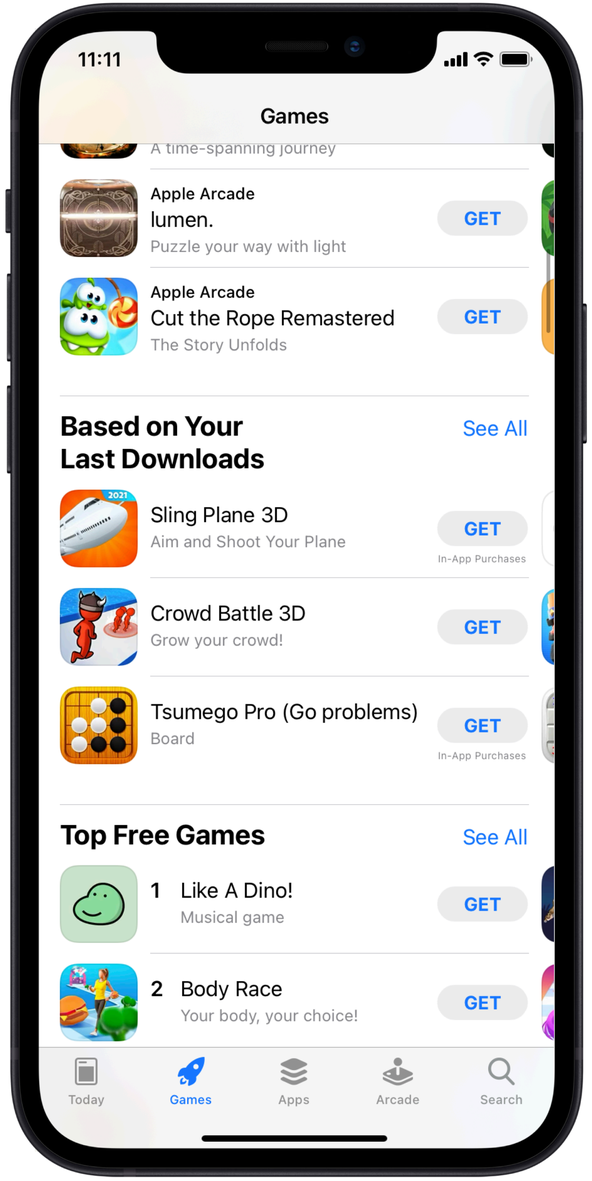
- Week 11: Introduction to Recommender Systems and Collaborative Filtering
- Week 12: Recommender Systems before Deep Learning era
  - Lab: Focus on topics in Week 11
- Week 13: Modern Recommender Systems in Theory and Practice
  - Lab: Focus on topics in Week 12 and 13

# Introduction

# What is a Recommender System?







# Why recommender systems?

- Netflix: 2/3 of the watched movies are recommended
- Google News: recommendations generate 38% more clickthrough
- Amazon: 35% sales achieved are from recommendations
- …


*"Can I tell you about a few items that aren't on the menu?"*

CartoonCollections.com

# Why recommender systems?

Bring values to **customers**

- Find things that are more relevant
- Narrow down the set of choices
- Explore and discover more interesting spaces
- …

# Why recommender systems?

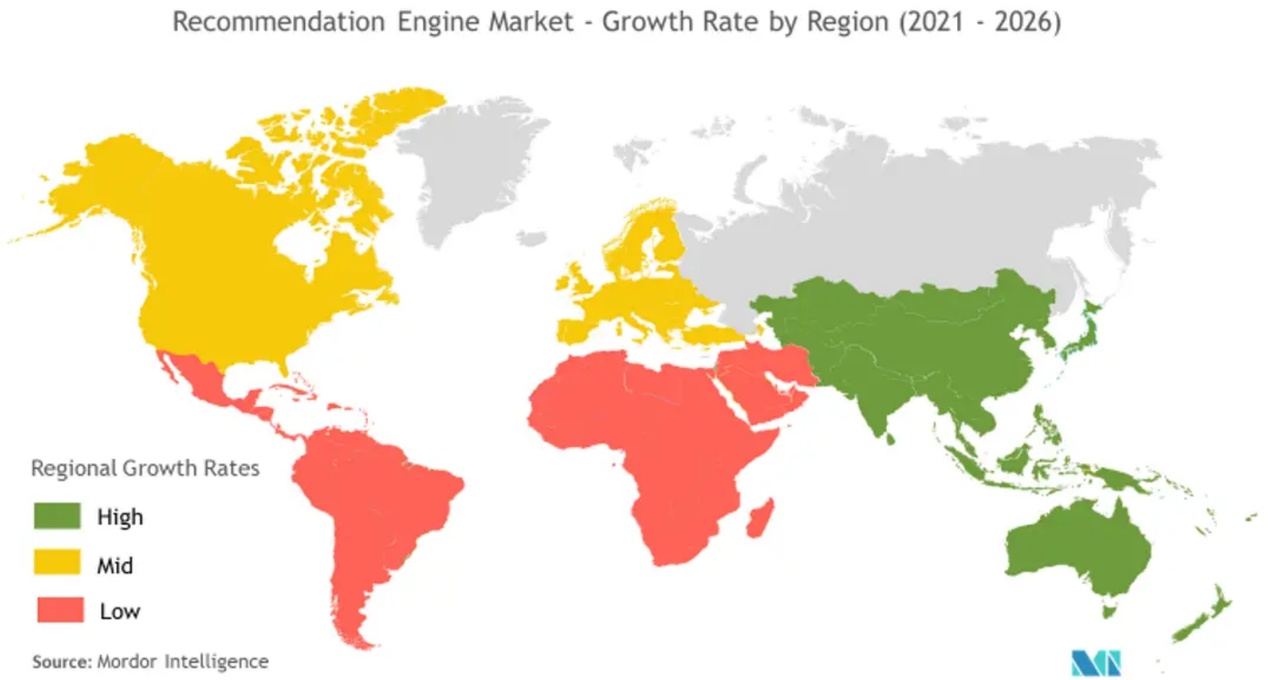
Bring values to **providers**

- Improve profit by providing relevant and accurate recommendations
- Increase customer retention (e.g., customer loyalty)
- Guide consuming behaviours
- …

# Everything is Recommendation!

- The Recommendation Engine market was valued at USD 2.12 billion in 2020, and it is expected to reach USD 15.13 billion by 2026

| | |
|---|---|
| Deployment Mode | On-Premise<br>Cloud |
| Types | Collaborative Filtering<br>Content-Based Filtering<br>Hybrid Recommendation Systems<br>Other Types |
| End-user Industry | IT and Telecommunication<br>BFSI<br>Retail<br>Media and Entertainment<br>Healthcare<br>Other End-user Industries |
| Geography | North America<br>Europe<br>Asia Pacific<br>Latin America<br>Middle East and Africa |

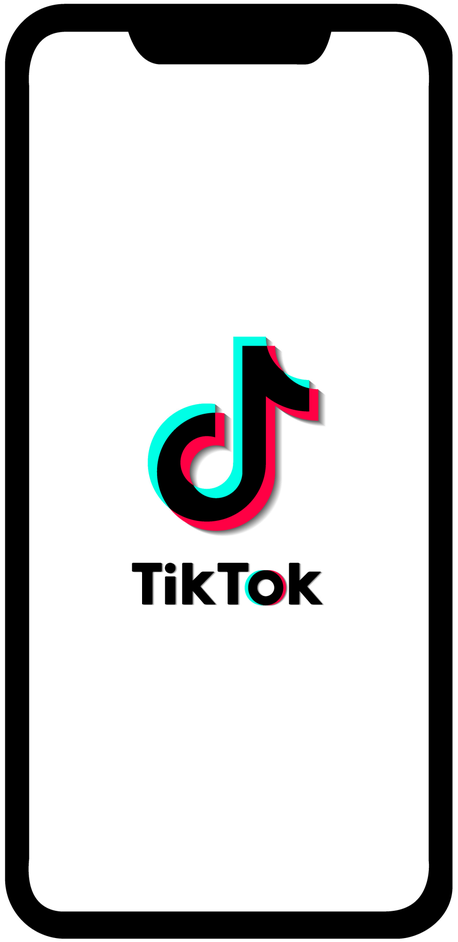



Source: Mordor Intelligence

# Example

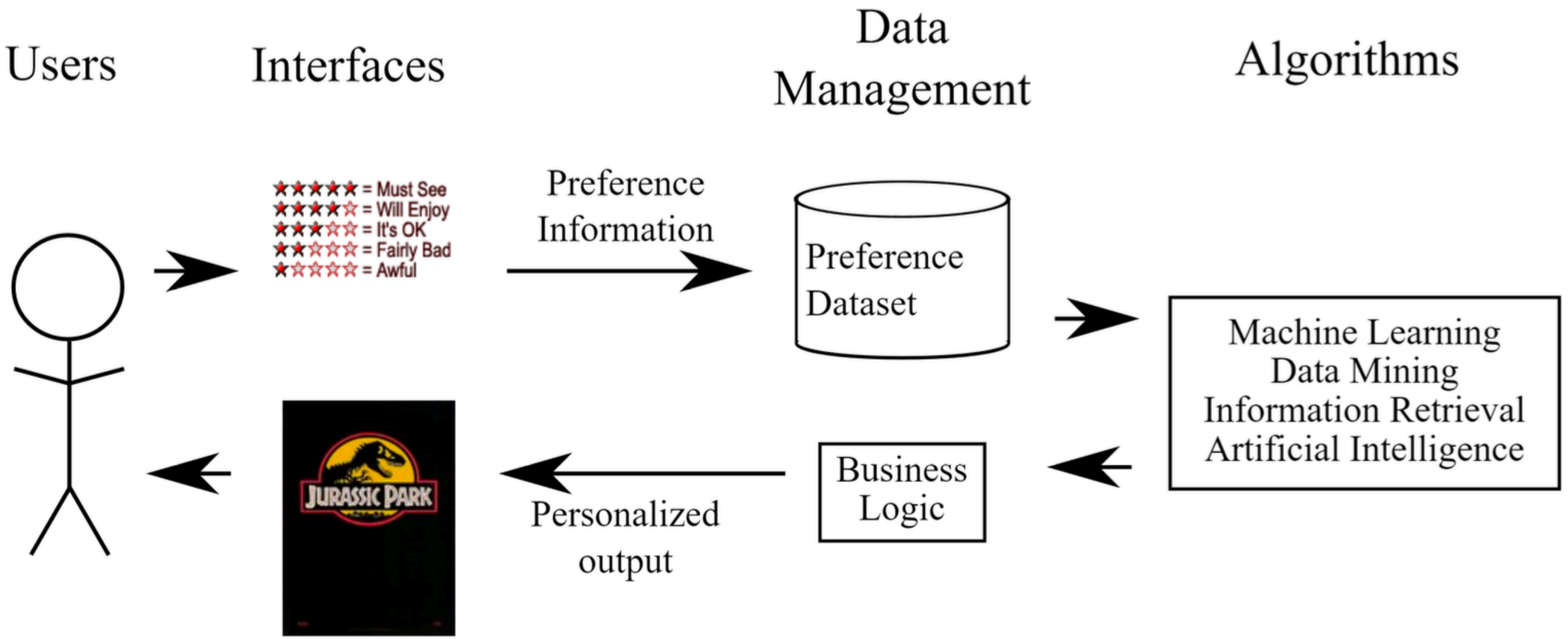
According to TikTok: “The system recommends content by ranking videos based on a combination of factors — starting from interests you express as a new user and adjusting for things you indicate you’re not interested in, too.”

- User Activity: likes, shares, comments, …
- Content: video, audio, hashtags, effects, …
- Auxiliary information: gender, country, language, …
- Social information: follower/followee, tags, linked to other accounts, …
- Contextual information: location, geographic, time, …



# Pipeline of RS

- Learn a utility function that predicts a user's preference towards an item




Source: https://www-users.cs.umn.edu/ kluve018/what-is-a-recommender.html

# Data Source of RS

- Users: structured, semi-structured, unstructured
- Items: structured, semi-structured, unstructured
- Interactions: ratings, like/dislike
- Contexts: side information
- Knowledge: domain experts
- …

# Formulation of RS

**"RS are software agents that elicit the interests and preferences of individual consumers and make recommendations accordingly. "**

**"They have the potential to support and improve the quality of the decisions consumers make while searching for and selecting products online."**

NETFLIX

amazon

ebay


**Goal:**

Learn a ***utility function*** that ***predicts*** a user's preference towards an item

- ❑ **Inputs:**
  - *User model* (e.g. implicit/explicit feedbacks, preferences, demographics, social connections, situational context)
  - *Items* (with or without description of item characteristics)
- ❑ **Outputs:**
  - Predicted preference scores (Used for ranking)

Source: UNSW

# Challenges of RS

- Scalability
- Sparsity/Density
- Cold-start
- Biases and Fairness
- Privacy and Security
- …

# Collaborative Filtering

# Collaborative Filtering

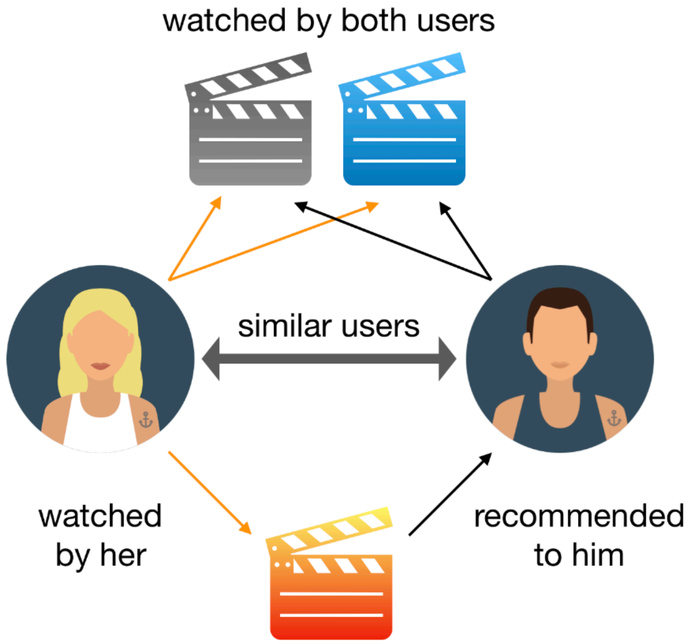
Basic Idea

- “Wisdom of the crowd” — look at the ratings of like-minded users to recommend items
- Analyze the *known* preferences of a group of users to make predictions of the unknown preferences for other users.

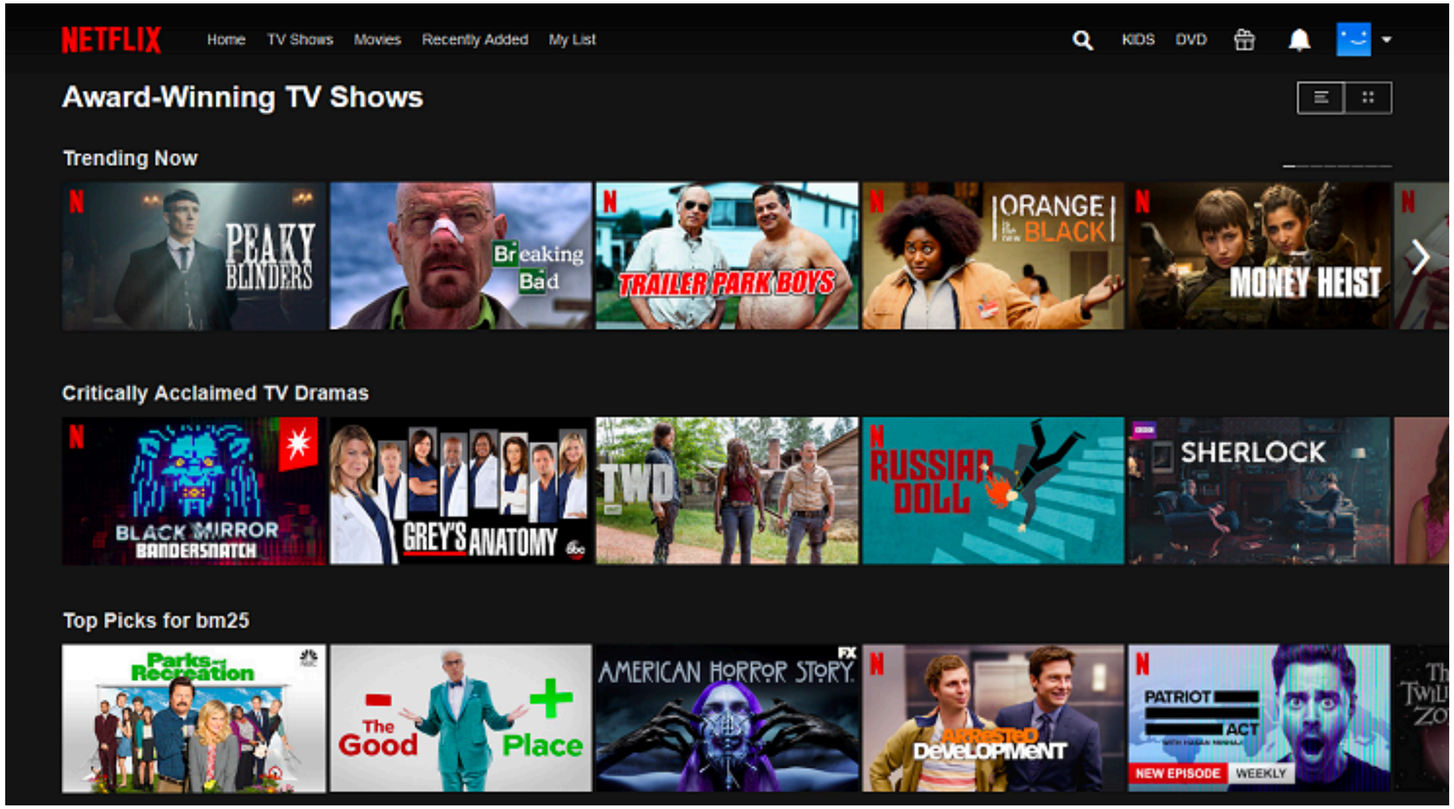
# Collaborative Filtering

Basic Assumptions

- Users give ratings to a group of items
- Users who had similar tastes in the past, will have similar tastes in the future



# Collaborative Filtering




| | Breaking Bad | American Horror Story | Money Heist | Sherlock | Grey's Anatomy |
|---|---|---|---|---|---|
| Lucas | 5 | 3 | 4 | 4 | ? |
| User 1 | 3 | 1 | 2 | 3 | 3 |
| User 2 | 4 | 3 | 4 | 3 | 5 |
| User 3 | 3 | 3 | 1 | 5 | 4 |
| User 4 | 1 | 5 | 5 | 2 | 1 |

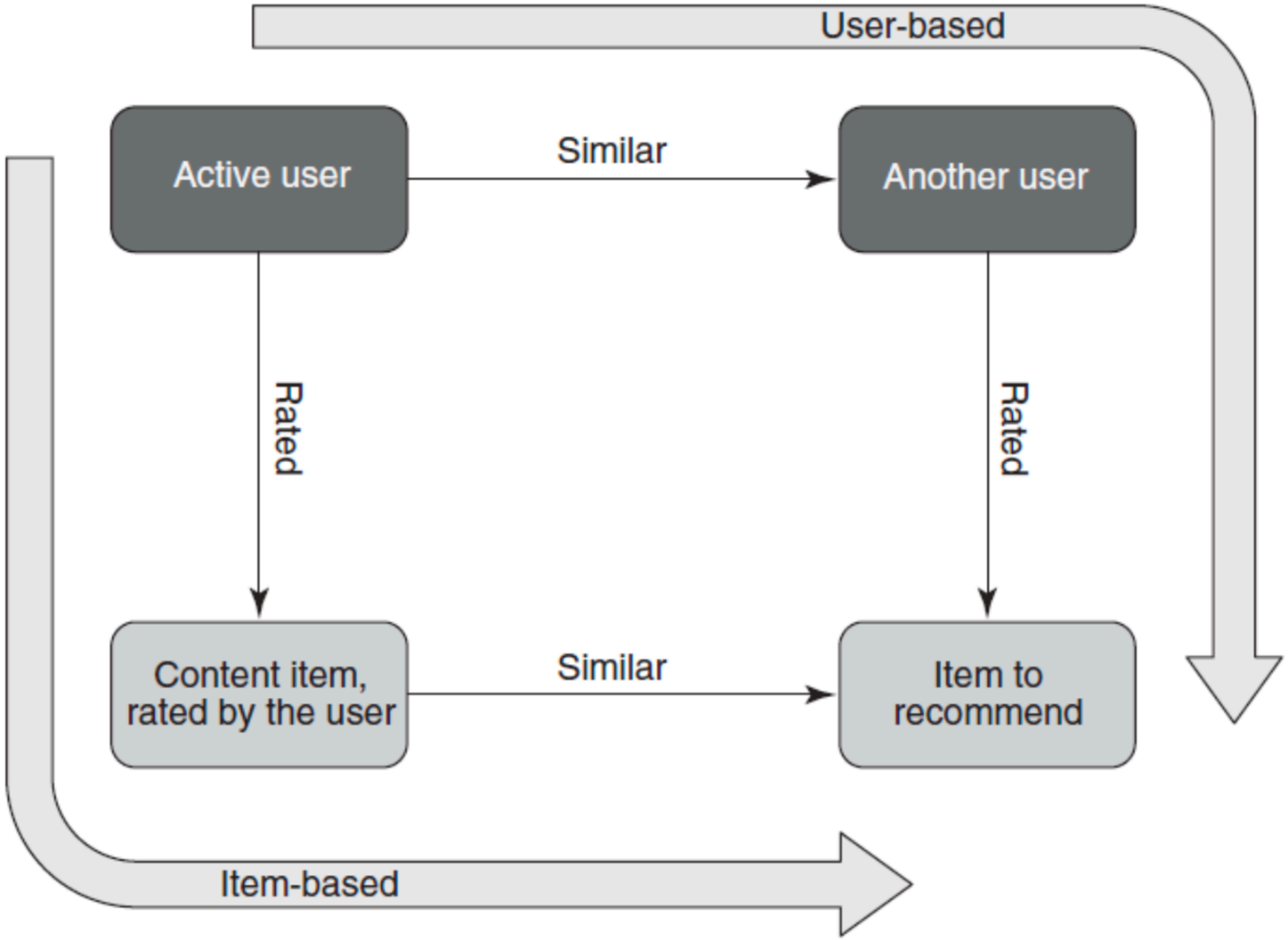
# Collaborative Filtering

- Neighbourhood-based: Compute similarities between users or items (i.e., “memory” of the system) that are successively exploited to produce recommendations
  - User-based: finding similar users
  - Item-based: finding similar items
- Model-based: Estimate or learn a model, then apply this model to make rating predictions

# Neighbourhood-based CF

# Neighbourhood-based CF

## User-based vs. Item-based



# User-based CF

- If users had similar tastes in the past, they will have similar tastes in the future
- User preferences retain stable and consistent over time

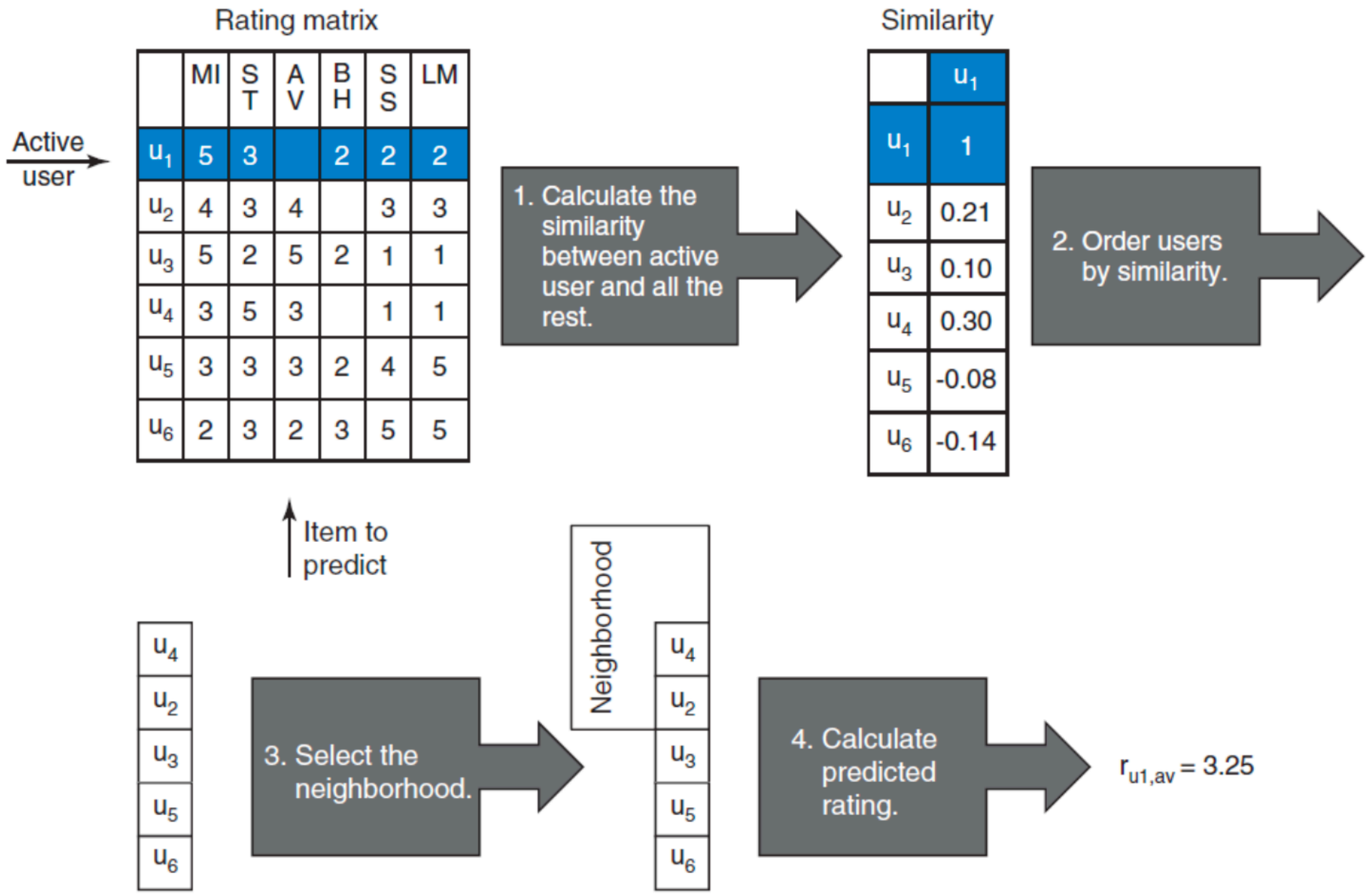
| | Grey's Anatomy | American Horror Story | Money Heist | Sherlock | Breaking Bad |
|---|---|---|---|---|---|
| **Lucas** | 5 | 3 | 4 | 4 | **?** |
| **User 1** | 3 | 1 | 2 | 3 | 3 |
| **User 2** | 4 | 3 | 4 | 3 | 5 |
| **User 3** | 3 | 3 | 1 | 5 | 4 |
| **User 4** | 1 | 5 | 5 | 2 | 1 |

# User-based CF

Steps:

1. Determine the similarity of users and a give user (i.e., target user)
2. Aggregate the ratings of similar users to predict target user's ratings
3. Decide how many similar users' opinions should be considered

# User-based CF




Source: Practical Recommender Systems. Kim Falk

# User-based CF

## Step 1: determine the similarity

- Jaccard Distance: $sim(u_i, u_j) = \frac{U_i \cap U_j}{U_i \cup U_j}$
- Cosine Distance: $sim(u_i, u_j) = \frac{\sum_{i=1}^{n} U_i U_j}{\sqrt{\sum_{i=1}^{n} U_i^2}\sqrt{\sum_{i=1}^{n} U_j^2}}$
- Pearson Correlation Coefficient:

$$sim(u_i, u_j) = \frac{\sum_{p \in P}(r_{i,p} - \overline{r_i})(r_{j,p} - \overline{r_j})}{\sqrt{\sum_{p \in P}(r_{i,p} - \overline{r_i})^2}\sqrt{\sum_{p \in P}(r_{j,p} - \overline{r_j})^2}}$$

Where $u_i$, $u_j$ is a pair of users; $U_i$, $U_j$ is item list rated by user $u_i$, $u_j$ respectively; $P$ is the set of items rated by users $u_i$ and $u_j$; $\bar{r}$ is the user's average ratings

# User-based CF

## Step 1: determine the similarity

| | Grey's Anatomy | American Horror Story | Money Heist | Sherlock | Breaking Bad |
|---|---|---|---|---|---|
| **Lucas** | 5 | 3 | 4 | 4 | **?** |
| **User 1** | 3 | 1 | 2 | 3 | 3 |
| **User 2** | 4 | 3 | 4 | 3 | 5 |
| **User 3** | 3 | 3 | 1 | 5 | 4 |
| **User 4** | 1 | 5 | 5 | 2 | 1 |

- Pearson Correlation Coefficient (PCC)

$$= \frac{\sum_{p \in P}(r_{Lucas,p} - \overline{r_{Lucas}})(r_{1,p} - \overline{r_1})}{\sqrt{\sum_{p \in P}(r_{Lucas,p} - \overline{r_{Lucas}})^2}\sqrt{\sum_{p \in P}(r_{1,p} - \overline{r_1})^2}} = \frac{(5-4)(3-2.25) + (3-4)(1-2.25) + (4-4)(2-2.25) + (4-4)(3-2.25)}{\sqrt{(5-4)^2 + (3-4)^2 + (4-4)^2 + (4-4)^2}\sqrt{(3-2.25)^2 + (1-2.25)^2 + (2-2.25)^2 + (3-2.25)^2}}$$

Where $\overline{r_{Lucas}} = 4$, and $\overline{r_1} = 2.25$ and PCC is $\in [-1,1]$

# User-based CF

## Step 1: determine the similarity

Experience:

- Generate user-user similarity matrix

| | Lucas | User 1 | User 2 | User 3 | User 4 |
|---|---|---|---|---|---|
| **Lucas** | 1 | | | | |
| **User 1** | X | 1 | | | |
| **User 2** | X | X | 1 | | |
| **User 3** | X | X | X | 1 | |
| **User 4** | X | X | X | X | 1 |

- Empirical analyses show that for user-based CF — and at least for the best studied recommendation domains — the PCC outperforms other measures of comparing users (e.g., Spearman's rank correlation coefficient)

# User-based CF

## Step 2: aggregate the ratings of similar users to predict

- Calculate if the neighbours' ratings for the unseen item (e.g., Breaking Bad) are higher or lower than their average
- Combine the rating differences — use the similarity as the weight
- Add/subtract the neighbours' bias from the target user's (e.g., Lucas's) average and use this as a prediction
  - Different types of aggregation functions:

$$r_{u,i} = k \sum_{u' \in N} sim(u, u') \times r_{u',i}$$

$$r_{u,i} = \overline{r_u} + \frac{\sum_{u' \in N} sim(u, u') \times (r_{u',i} - \overline{r_{u_i}})}{\sum_{u' \in N} |sim(u, u')|}$$

# User-based CF

## Step 3: which and how many neighbours for aggregation

- Only consider positively correlated neighbours (or set a threshold)
- Can be optimized based on dataset
- Experience: empirical studies show that aggregating between 50-200 neighbours could be good enough

# User-based CF

## Discussion

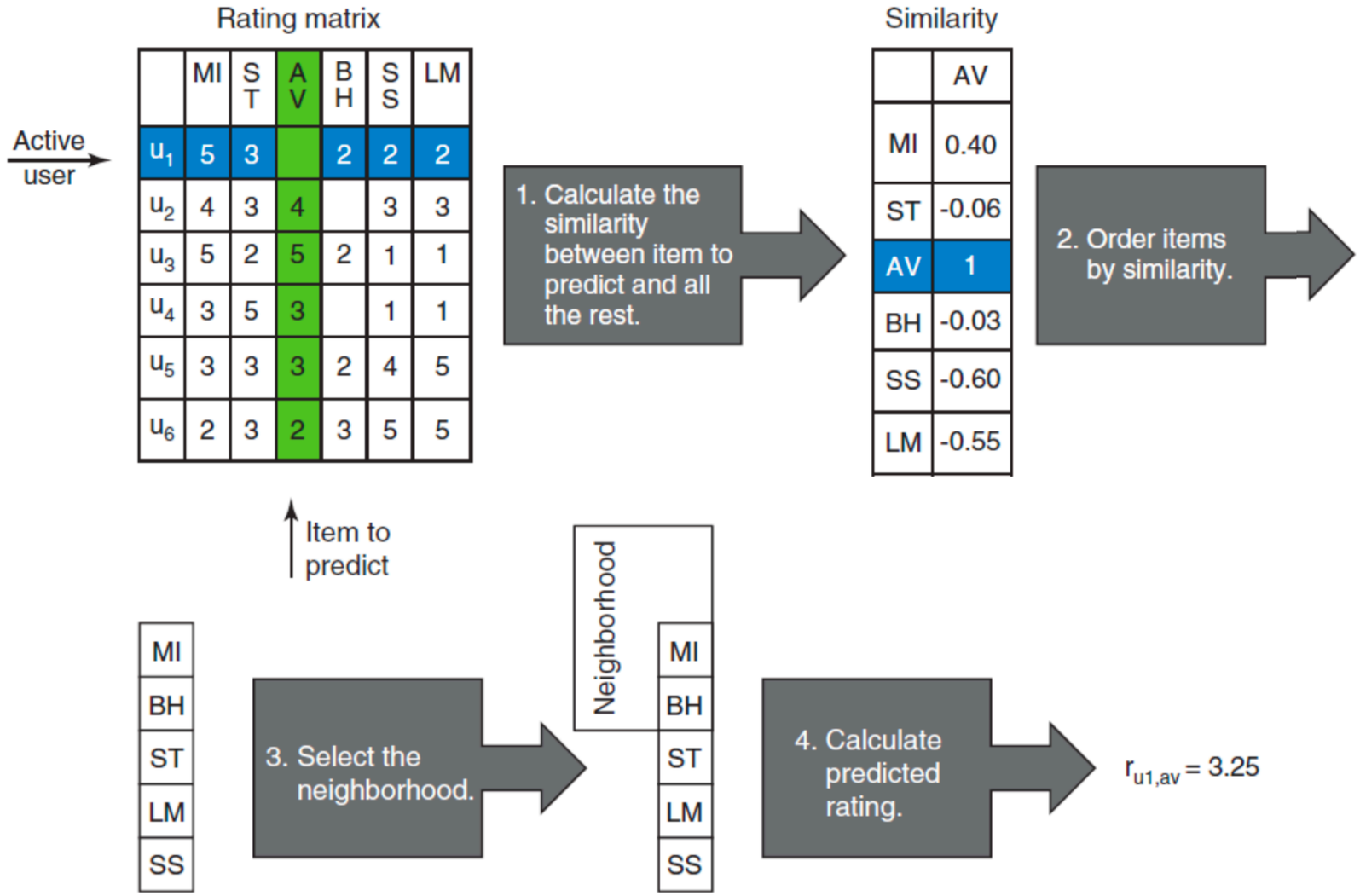
- Very simple scheme leading to quite accurate recommendations
- Possible Issues?

# Break

# Item-based CF

- Instead of calculating similarity between users, what if we use the similarity between items to make predictions
- Why?
  - The need to scan a vast number of potential neighbours makes it impossible to compute predictions in real-time
  - Item-based is more adaptable for offline pre-processing and thus allows for the computation of recommendations in real-time even for a very large rating matrix

# Item-based CF




Source: Practical Recommender Systems. Kim Falk

# Item-based CF

Steps:

1. Look into the items that have been rated and compute how similar they are to the target item

2. Select k most similar items

3. Compute prediction by taking weighted average on the target user's ratings on the most similar items

# Item-based CF

## Step 1: determine the similarity

| | Grey's Anatomy | American Horror Story | Money Heist | Sherlock | Breaking Bad |
|---|---|---|---|---|---|
| **Lucas** | 5 | 3 | 4 | 4 | **?** |
| **User 1** | 3 | 1 | 2 | 3 | 3 |
| **User 2** | 4 | 3 | 4 | 3 | 5 |
| **User 3** | 3 | 3 | 1 | 5 | 4 |
| **User 4** | 1 | 5 | 5 | 2 | 1 |

- We first compare the rating vectors of the other items and look for items that have ratings similar to target item
- Then, we take Lucas's ratings for these items to predict the rating for the target item (i.e., Breaking Bad)

# Item-based CF

## Step 1: determine the similarity

| | Grey's Anatomy | American Horror Story | Money Heist | Sherlock | Breaking Bad |
|---|---|---|---|---|---|
| Lucas | 5 | 3 | 4 | 4 | ? |
| User 1 | 3 | 1 | 2 | 3 | 3 |
| User 2 | 4 | 3 | 4 | 3 | 5 |
| User 3 | 3 | 3 | 1 | 5 | 4 |
| User 4 | 1 | 5 | 5 | 2 | 1 |

- Cosine similarity between Grey's Anatomy and Breaking Bad

$$= \frac{3 \times 3 + 5 \times 4 + 4 \times 3 + 1 \times 1}{\sqrt{3^2 + 5^2 + 4^2 + 1^2}\sqrt{3^2 + 4^2 + 3^2 + 1^2}} = 0.99$$

- Similar to user-based, we want to generate item-item similarity matrix

| | Grey's Anatomy | American Horror Story | Money Heist | Sherlock | Breaking Bad |
|---|---|---|---|---|---|
| Grey's Anatomy | 1 | | | | |
| American Horror Story | X | 1 | | | |
| Money Heist | X | X | 1 | | |
| Sherlock | X | X | X | 1 | |
| Breaking Bad | X | X | X | X | 1 |

# Item-based CF

## Step 2, 3: aggregate and predict

- After the similarities between the items are determined (i.e., the item-item similarity matrix is generated), we predict the rating of Lucas for Breaking Bad as:

$$r_{u,i} = \frac{\sum_{i' \in N} sim(i, i') \times r_{u,i'}}{\sum_{i' \in N} |sim(i, i'|} = \frac{\sum_{i' \in N} sim(BB, i') \times r_{L,i'}}{\sum_{i' \in N} |sim(BB, i'|}$$

Where $N$ is the set of neighbourhood items

# Item-based CF

## Discussion

- Bottleneck — similarity computation
- Time complexity: millions of users and items
- Possible Issues?

# Item-based CF

## Amazon.com Recommendations

- Generally, it's assumed that item similarity is stable, so item similarities can be calculated beforehand or offline
- Generate a data set where you can look up similar items for the current item, making it faster to calculate the predictions.

```
For each item in product catalog, I1
  For each customer C who purchased I1
    For each item I2 purchased by
       customer C
      Record that a customer purchased I1
       and I2
  For each item I2
    Compute the similarity between I1 and I2
```

Source: https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

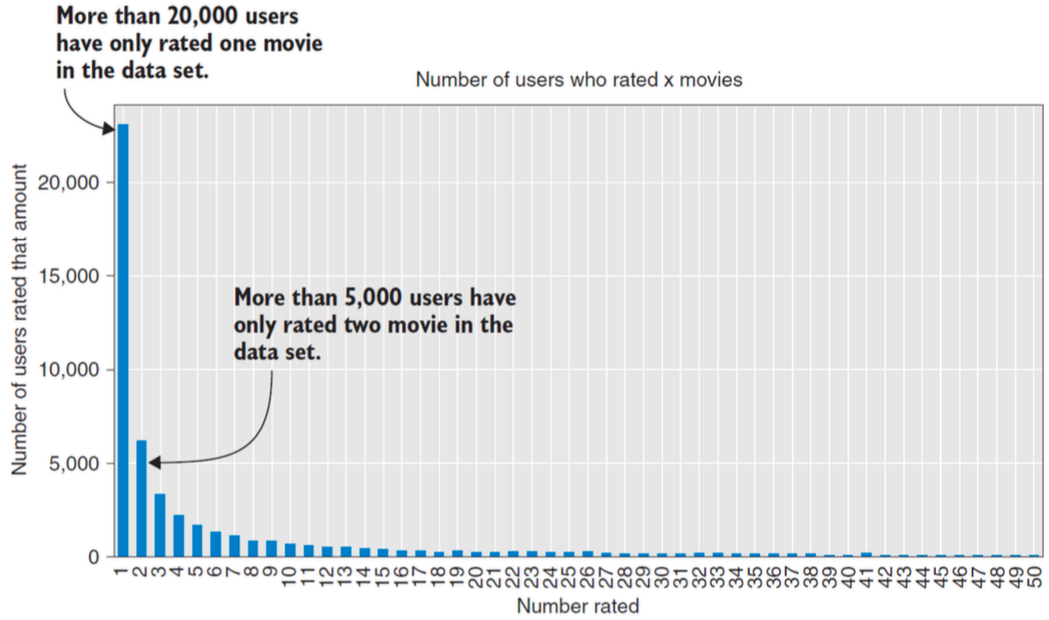
# Recap - Neighbourhood-based CF

# Recap

- The rating matrix is directly used to find neighbours / make predictions
- User-based CF: similarity between users is dynamic, pre-computing user neighbourhood can lead to poor prediction
- Item-based CF: similarity between items is static, as such pre-computing is possible
- Choice of similarity functions: Pearson Correlation Coefficient (PCC), Cosine, Adjusted Cosine, Euclidean, etc.

# Recap

- Some items that only be rated very rarely



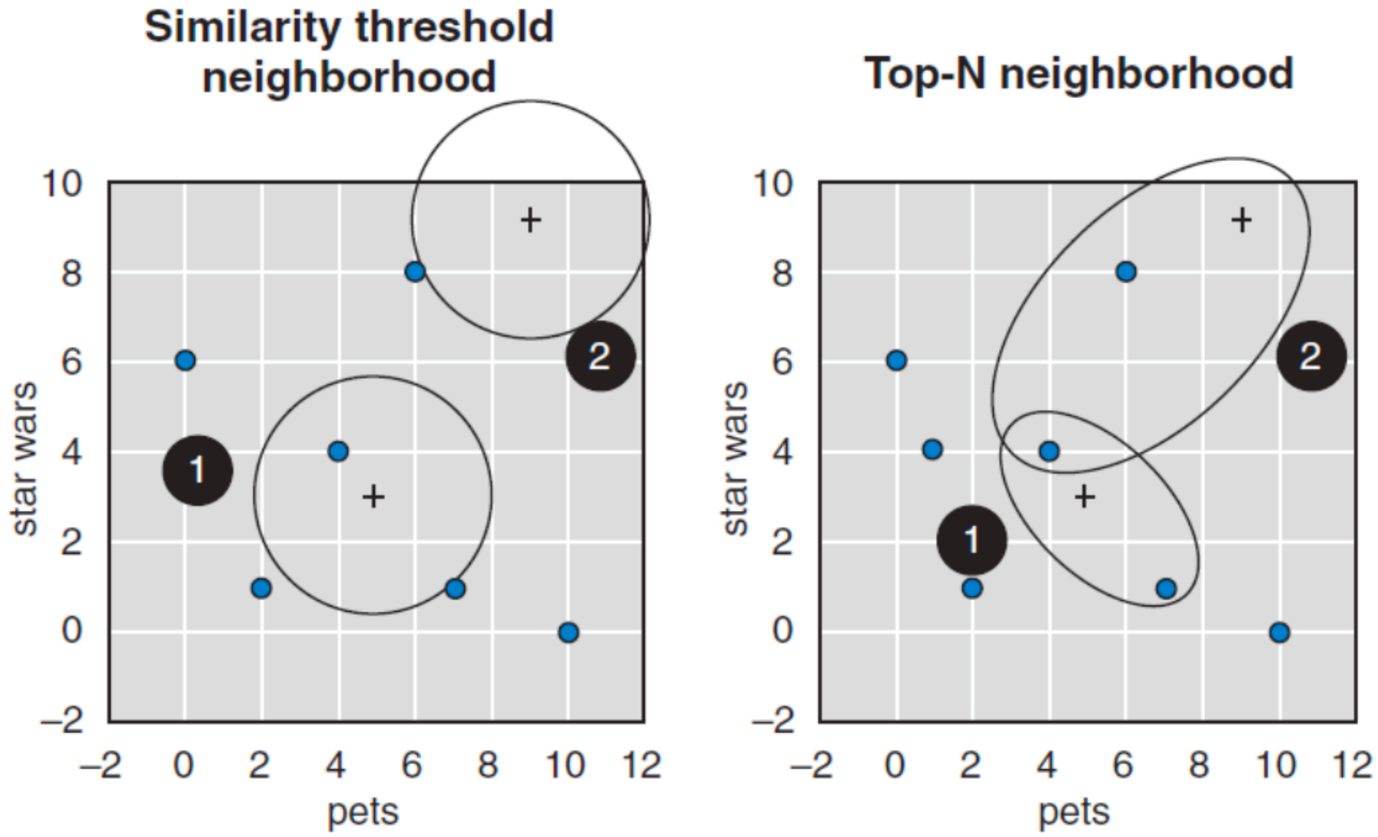

- If an item has only **one** rating, the average rating is equal to itself. Thus, its means is 0, which makes the similarity function undefined

- How about a user who rated **two** items?

# Recap

## Neighbourhood selection

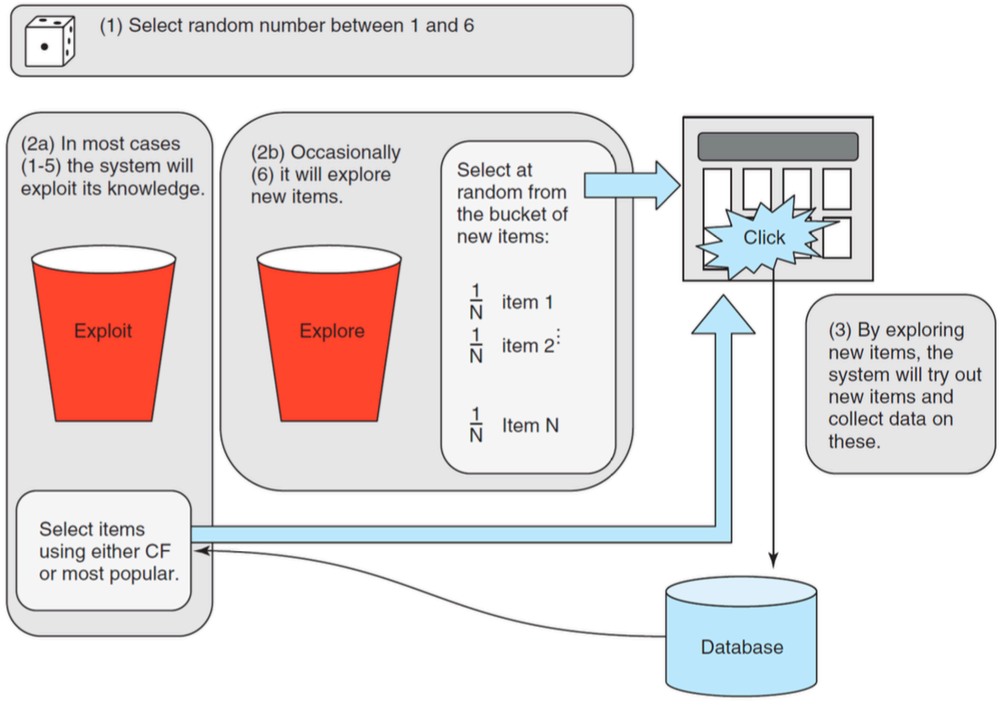
- Clustering, threshold, top-N



# Recap

## Cold-start

- Explore: show a new item, in which the items gets exposed and users hopefully interact with them
- Exploit: exploit the knowledge you have and return recommendations




Source: Practical Recommender Systems. Kim Falk

# Evaluation

# Evaluation

## In practice

- Different perspective/aspects
- Retrieval perspective
  - Reduce search costs and provide “correct” recommendations
  - Assumption: users already know what they want in advance
- Recommendation perspective
  - Serendipity: identify items from the Long Tail (i.e., users did not know)
  - Provide “appropriate” recommendations

# Evaluation

## In research

- Offline experiments: historical data, prediction accuracy, etc.
- Test with real users:
  - A/B tests
  - Sales increases/decreases, CTR (click-through rate)
- Simulator

# Evaluation

## What to evaluate

- Understand users' tastes by minimizing the prediction error
- Coverage: ensure the algorithm will recommend everything in your catalog and whether it can recommend something to all registered users
- Diversity: help users navigate a larger catalog than what is found on a single page
- Serendipity: give users 'sensation', so that all their visits aren't more of the same



Source: Beyond accuracy: evaluating recommender systems by coverage and serendipity. RecSys 2010.

# Evaluation

## Accuracy

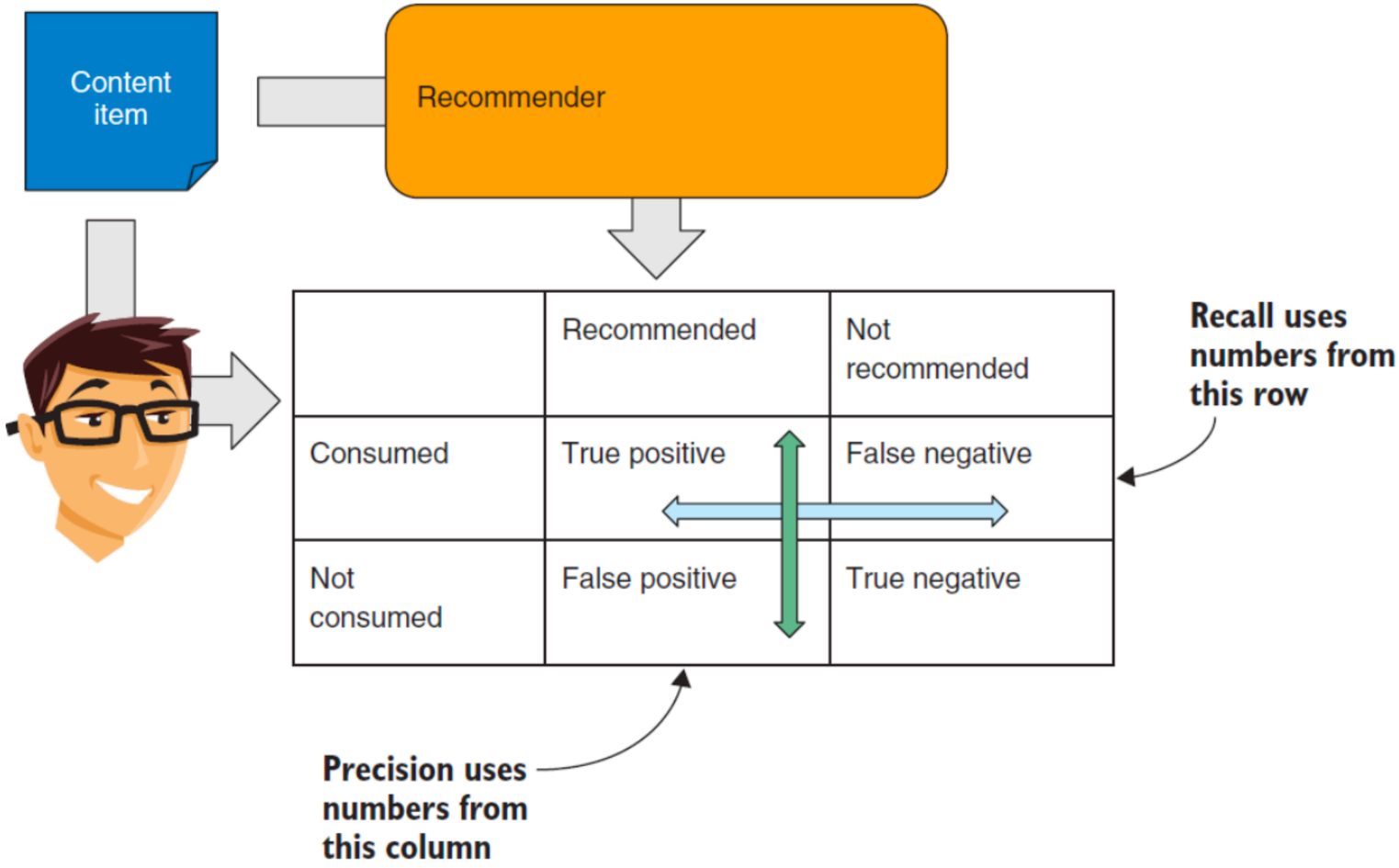
- Precision: a measure of exactness, determines the fraction of relevant items retrieved out of all items retrieved
  - E.g., the proportion of recommended movies that are actually good
- Recall: a measure of completeness, determines the fraction of relevant items retrieved out of all relevant items
  - E.g., the proportion of all good movies recommended

$$\text{Precision@K} = \frac{tp}{tp + fp} = \frac{|\text{appropriate movies recommended}|}{|\text{all recommendations}|}$$

$$\text{Recall@K} = \frac{tp}{tp + fn} = \frac{|\text{appropriate movies recommended}|}{|\text{all appropriate movies}|}$$

# Evaluation

## Accuracy




Source: UNSW

# Evaluation

## Error rate

- Mean Absolute Error (MAE): computes the deviation between predicted ratings and actual ratings
- Root Mean Square Error (RMSE): similar to MAE, with more emphasis on larger deviation

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |pred_i - r_i| \qquad \text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (pred_i - r_i)^2}$$

# Evaluation

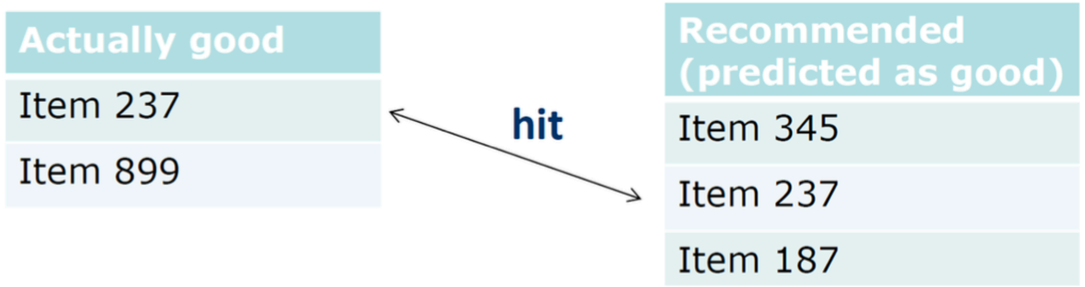
## Rank Score

- Extends recall and precision to take the positions of correct items in a ranked list into account
- This is particularly important in recommender systems because lower ranked items may be overlooked by users
  - Learning-to-rank: optimise models for such measures (e.g., AUC)

# Evaluation

## Rank Score

- Hit Ratio: generate top-K recommendations of a target user (i.e., masking rated items from the training set, then use the rest to train to produce top-K recommendations
  - Average reciprocal hit rate (ARHR)
- AUC: measures the likelihood that a random relevant item is ranked higher than a random irrelevant item

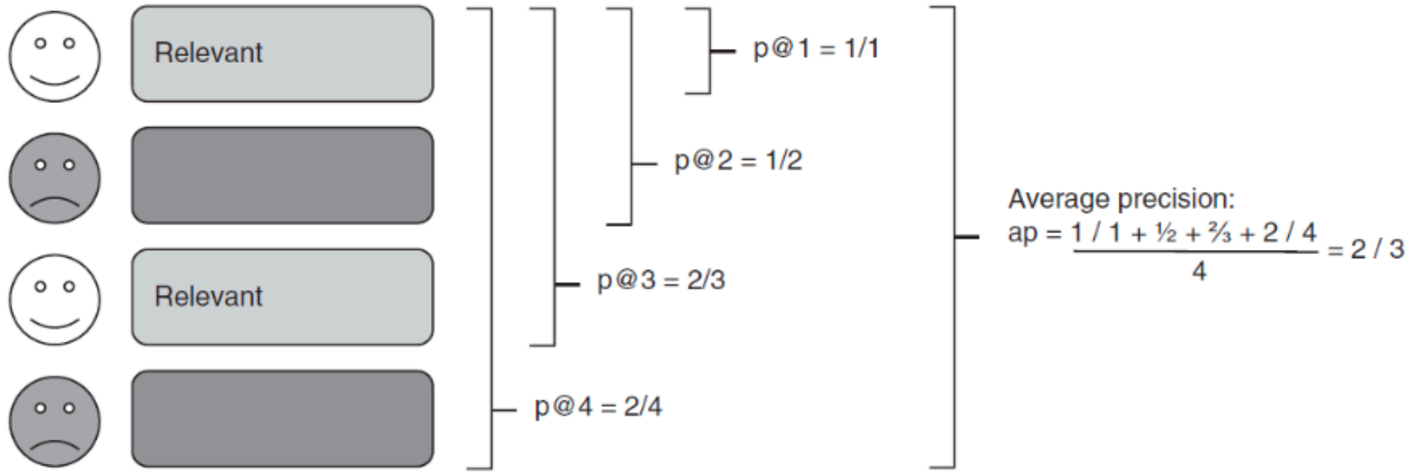


# Evaluation

## Rank Score

- Mean Average Precision (MAP): measure how good the rank is by running the precision from $1$ to $m$, where $m$ is $k$ items that are recommended

$$AP(U) = \frac{\sum_{k=1}^{m} Precision@k(u)}{m} \; ; \; MAP(U) = \frac{\sum_{u \in U} AP(u)}{|U|}$$



# Evaluation

## Rank Score

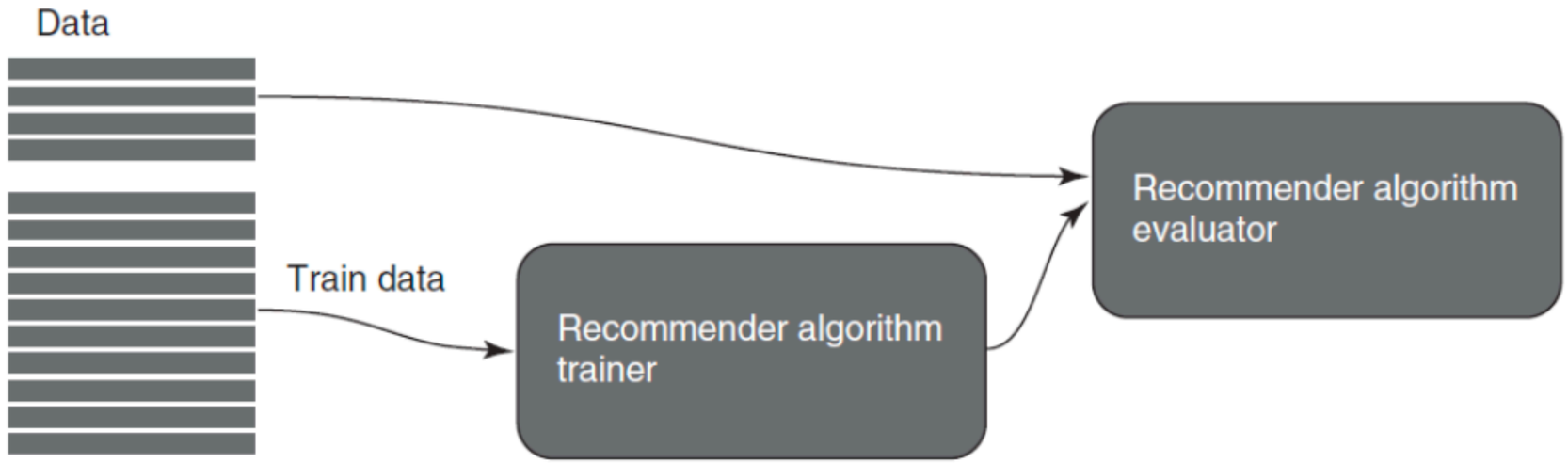
- DCG and NDCG: take into account the ‘position’ of the ranking list
- Hits at the beginning count more (more “gain”) and items of higher relevance are more important
- Given a rank position $pos$, and the graded relevance “rel” of an item $i$:

$$DCG = rel_1 + \sum_{i=1}^{pos} \frac{rel_i}{\ln(i)}$$

- Since some queries are harder than others and will likely produce lower DCG scores than easier queries, we normalize by scaling the results based on ideal results (i.e., $NDCG$)

# Evaluation

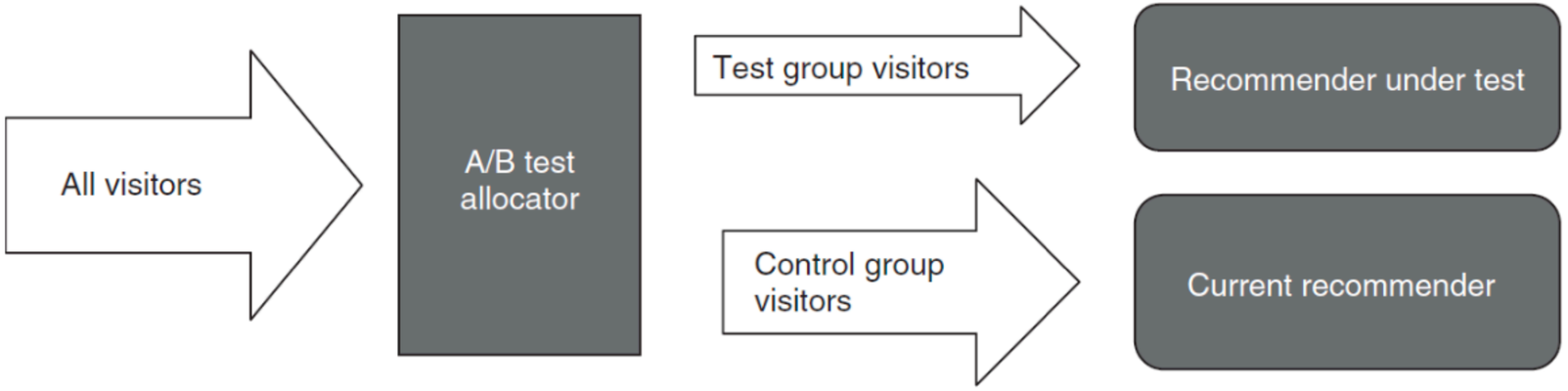
## Offline evaluation



# Evaluation

## A/B Testing

- Offline evaluation shows good results
- Conduct A/B test to understand if there's a significant difference between the current recommender and the new one
- In practice, we divert a small percentage of traffic to the new feature
  - Possible issue?



# Summary

# Summary

- Introduction to Recommender Systems
- Neighbourhood-based Collaborative Filtering
  - User-based
  - Item-based
- Evaluation

# Thank You!