Data Science & Machine Learning

Eman Raslan

Course Schedule

Week #	Day1	Day2
1	24-Nov-2023	
2	1-Dec-2023	2-Dec-2023
3	8-Dec-2023	9-Dec-2023
4	15-Dec-2023	16-Dec-2023
5	22-Dec-2023	23-Dec-2023
6	29-Dec-2023	30-Dec-2023
7	5-Jan-2023	6-Jan-2023
8	12-Jan-2023	13-Jan-2023
	19-Jan-2023	
Project		

Course Agenda

Statistics & Linear Algebra Basics

Python for Data Science

Data Exploration & Preparation

Machine Learning

Machine Learning

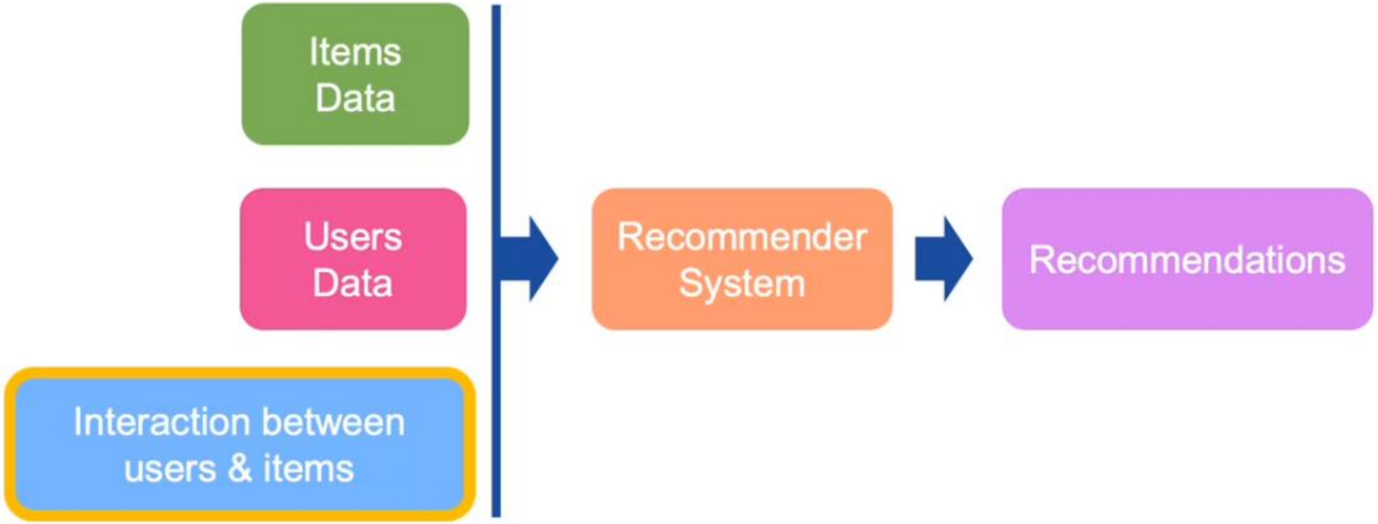
Recommender Systems

Recommender Systems

- Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user.
- The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read.
- “Item” is the general term used to denote what the system recommends to users.
- A RS normally focuses on a specific type of item.

Recommender Systems

- Amazon keeps track of items you look at to give you tailor made suggestions about other similar items that might interest you.
- Spotify analyzes genres, offers other user's playlist and more to give you suggestions.
- In the same way, Netflix suggests you to watch next.



Recommender Systems Function

- Increase the number of items sold.
- Sell more diverse items.
- Increase the user satisfaction.
- Increase user loyality.
- Better understand what the user wants.

Recommendation systems help you surface the things your users love

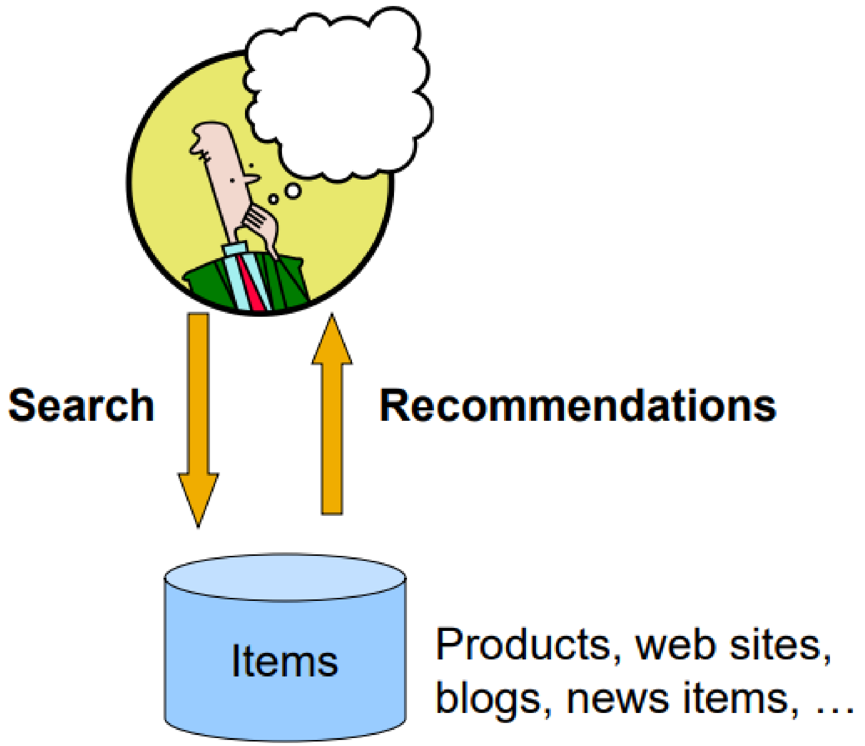
From online shopping through restaurants to video apps, recommenders account for:

- **40%** of app installs on Google Play
- **60%** of watch time on YouTube
- **35%** of purchase on Amazon*
- **75%** of movie watches on Netflix*

*Source: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

Data and Knowledge Sources

- RSs are information processing systems that actively gather various kinds of data in order to build their recommendations.
- Data is primarily about the items to suggest and the users who will receive these recommendation



Data and Knowledge Sources

- **Items**. Items are the objects that are recommended.
- The value of an item may be positive if the item is useful for the user, or negative if the item is not appropriate and the user made a wrong decision when selecting it.
- RSs, according to their core technology, can use a range of properties and features of the items.
- For example in a movie recommender system, the genre (such as comedy, thriller, etc.), as well as the director, and actors can be used to describe a movie and to learn how the utility of an item depends on its features.

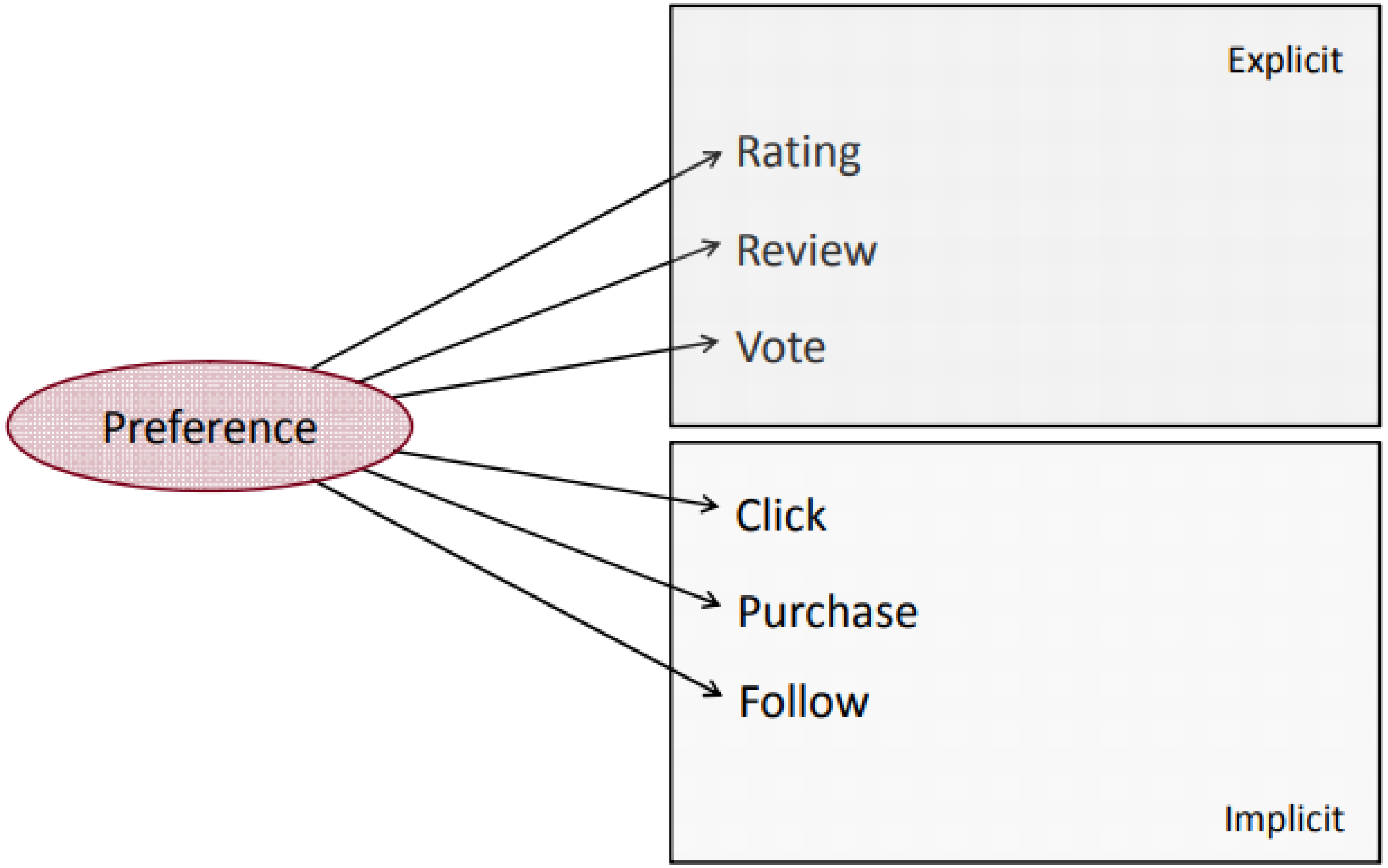
Users

- **Users** of a RS, may have very diverse goals and characteristics. In order to personalize the recommendations and the human-computer interaction, RSs exploit a range of information about the users.
- This information can be structured in various ways and again the selection of what information to model depends on the recommendation technique.
- For instance, in collaborative filtering, users are modeled as a simple list containing the ratings provided by the user for some items.
- In a demographic RS, sociodemographic attributes such as age, gender, profession, and education, are used.
- User data is said to constitute the user model.

Preference/ Transactions

- We generically refer to a transaction as a recorded interaction between a user and the RS.
- Transactions are log-like data that store important information generated during the human-computer interaction and which are useful for the recommendation generation algorithm that the system is using.
- For instance, a transaction log may contain a reference to the item selected by the user and a description of the context (e.g., the user goal/query) for that particular recommendation.
- If available, that transaction may also include an explicit feedback the user has provided, such as the rating for the selected item.

Inferring Preferences



explicitly Ratings

Just ask the users what they think!

- **Ratings** are the most popular form of transaction data that a RS collects.
- In the explicit collection of ratings, the user is asked to provide her opinion about an item on a rating scale.
- Ratings can take on a variety of forms:
  - Numerical ratings such as the 1-5 stars provided in the book recommender associated with Amazon.com.
  - Ordinal ratings, such as “strongly agree, agree, neutral, disagree, strongly disagree” where the user is asked to select the term that best indicates her opinion regarding an item (usually via questionnaire).
  - Binary ratings that model choices in which the user is simply asked to decide if a certain item is good or bad.

Difficulties with Ratings

- Are ratings reliable and accurate?
- Do user preferences change?
- What does a rating mean?

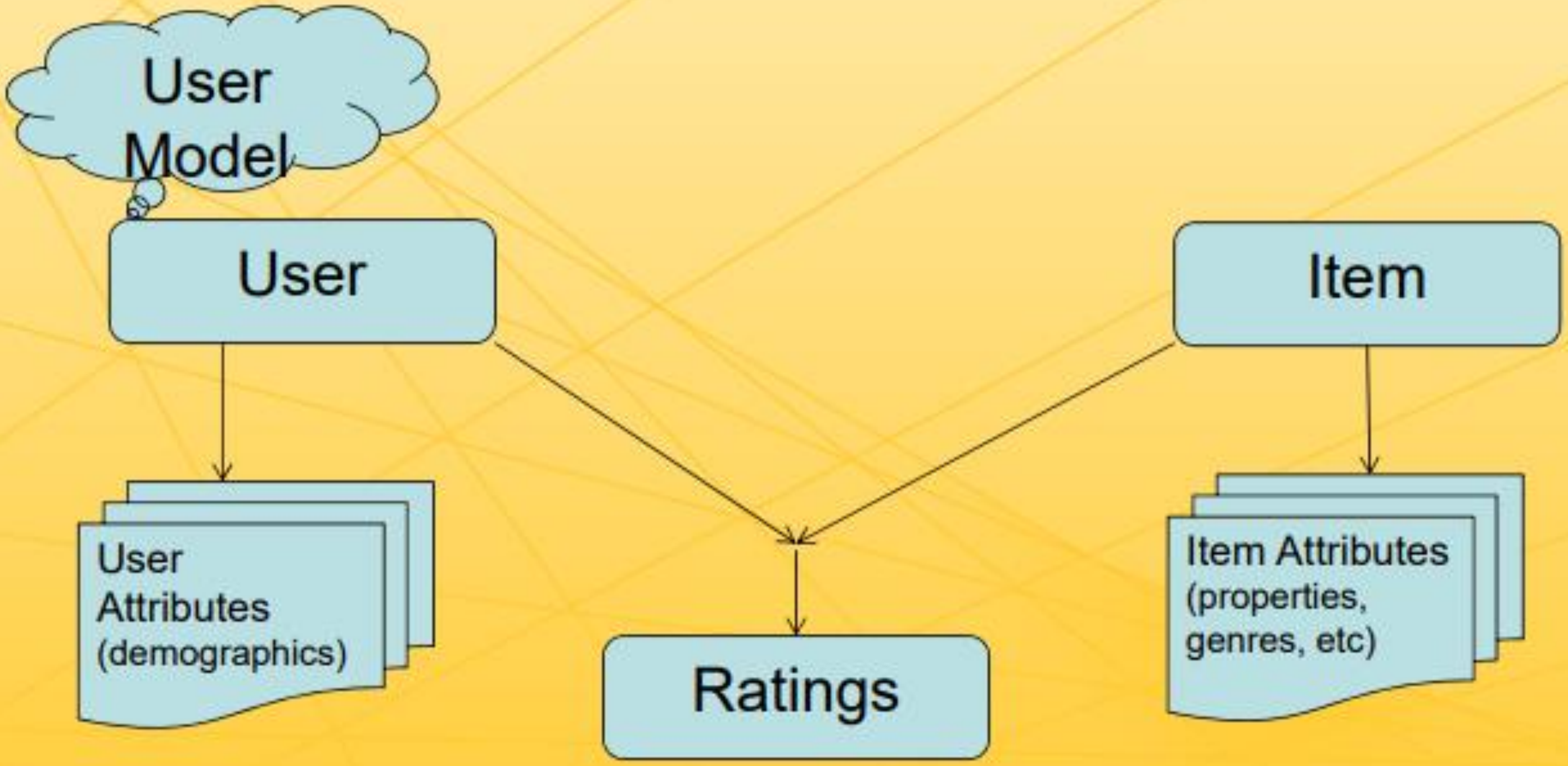
implicit Ratings

Their actions say a lot!

- In transactions collecting **implicit** ratings, the system aims to infer the users opinion based on the user's actions.
- For example, if a user enters the keyword "Yoga" at Amazon.com she will be provided with a long list of books. In return, the user may **click** on a certain book on the list in order to receive additional information. At At this point, the system may infer that the user is somewhat interested in that book.

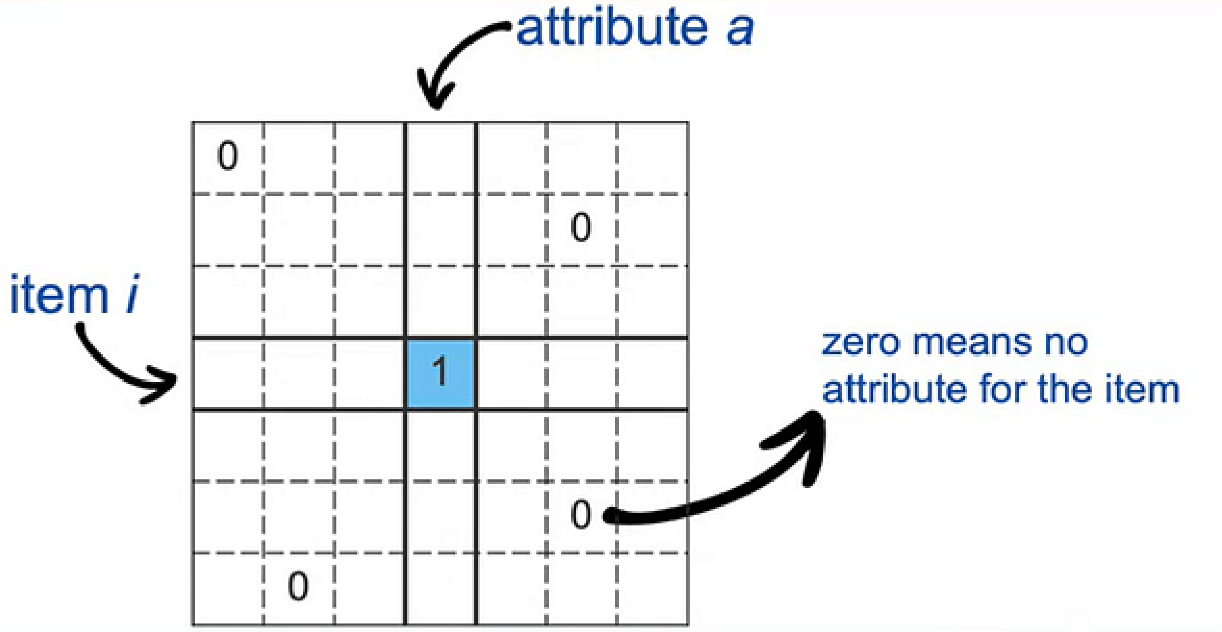
implicit Ratings

- What does the action mean?
  - –Purchase: they might still hate it
  - –Don't click: expect bad, or didn't see
- How to scale/represent actions?


User
Model
User
Item
User
Attributes
(demographics)
Ratings
Item Attributes
(properties,
genres, etc)

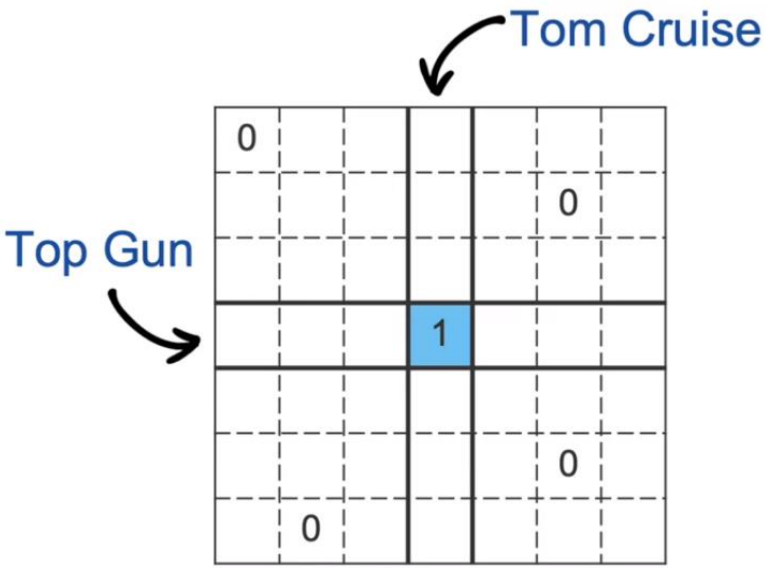
Item Context Matrix

- Item context matrix or ICR is a mathematical way to define input to a recommender system as a list of items and their attributes.
- Rows in the item context matrix represent items and columns represent the attributes.



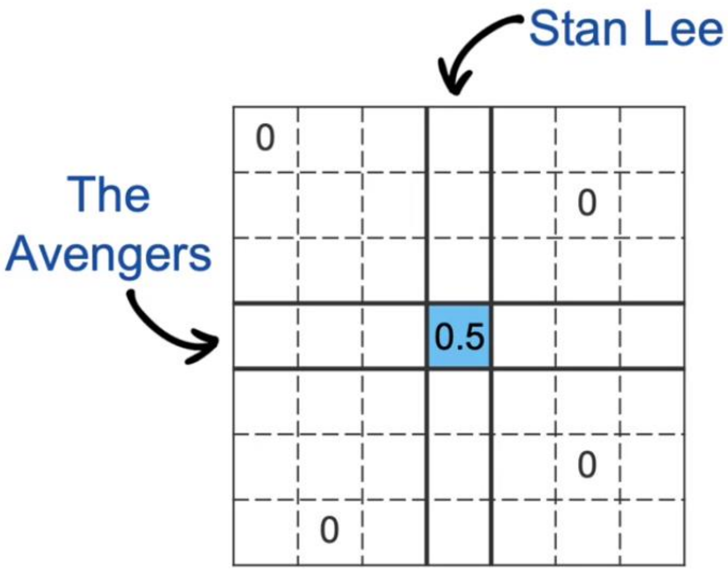
Item Context Matrix

- In the simplest form the values in the item context matrix are in binary format, either one or zero.
- If an item contains a specific attribute, the corresponding value in the matrix will be set to one or zero.
- In this example the ICM represents Tom Cruise as an attribute for the movie Top Gun.



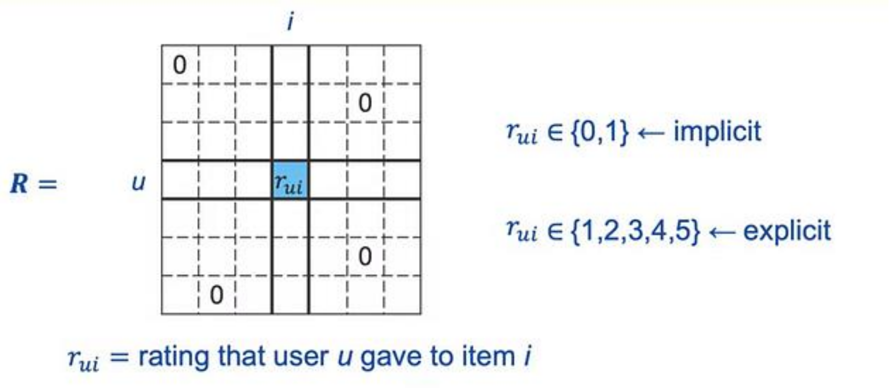
Item Context Matrix

- In a more useful scenario, each number in the item context matrix represents how important an attribute is to characterize an item and can assume a positive value.
- For instance, Stan Lee made a cameo appearance in the movie, The Avengers so the corresponding value in ICM should be set to lower than the value we use to describe leading actors.



User Rating Matrix

- One of the most important inputs to our recommender systems is the user interaction matrix that is the past interactions between the users and items.
- These interactions can be mathematically described as a user-rating matrix.
- Numbers in URM represent ratings, either implicit or explicit.
- The Rows in the user-rating matrix represent the users, the columns represent the items.
- If we have no information about the opinion of the user on an item, the corresponding value will be set to zero.

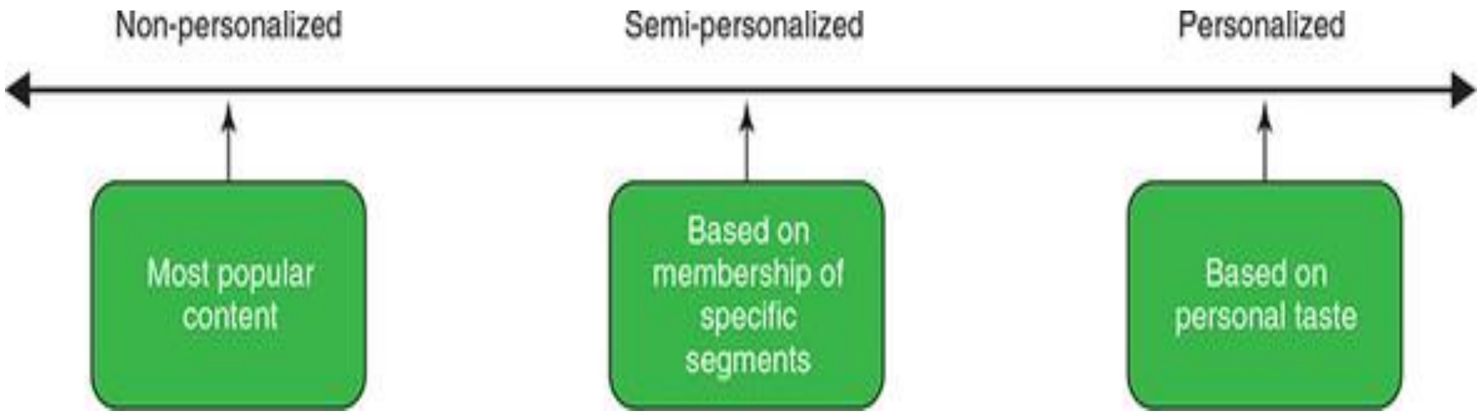


User Rating Matrix

- Implicit ratings typically have only zero or one possible value. This is because we can only look at the behavior of a user to understand why he liked something or not. We can mark the interaction as one if we think he is interested in it, zero in case we think he is not.
- In Explicit ratings we can ask the user to rate an item, for instance in one to five rating scale and the value in URM describes the rating. Zero can indicate the fact that we have no information on that item for that user.
- The goal of any recommender system is to predict the missing values in the URM.

Recommendation Algorithms

- Non-Personalized Summary Statistics
- Content-Based Filtering
- Collaborative Filtering

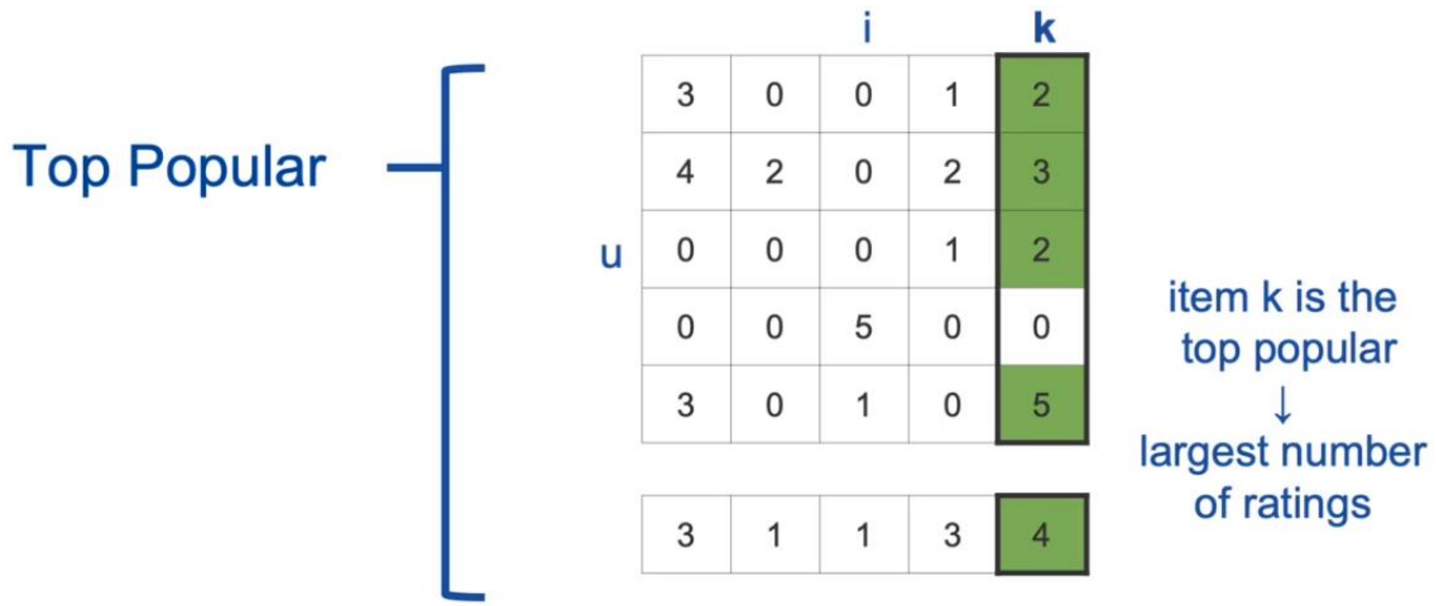


Non-personalized Recommenders

- Non-personalized techniques recommend all the users the same list of items.

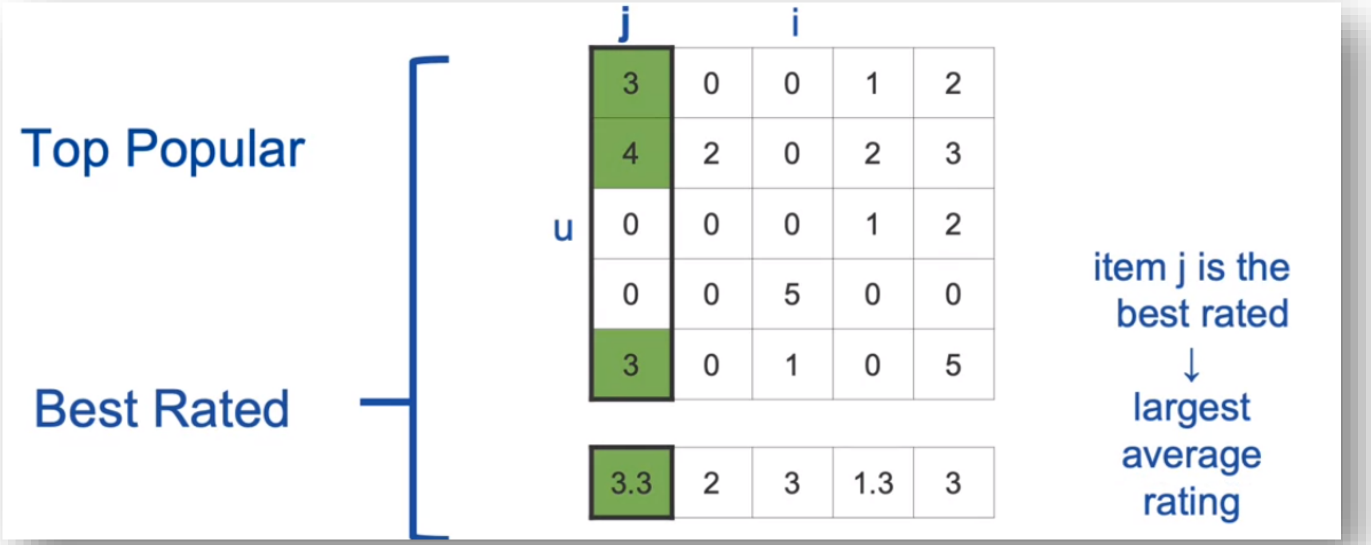
Non-personalized Recommenders

- An intuitive type of this category of recommender systems are **top popular** recommendations.
- We take the URM matrix and **count** the number of non-zero ratings for each item. We will see which items have been rated for the greatest number of times and in this way we see which items are most popular.
- The popularity of an item is computed by using its rating **without** taking into account the opinion of the users but just the number of users by which the item has been judged.



Non-personalized Recommenders

- Another non-personalized technique is based on best rated items. In order to compute the best rated items, we take the URM, extract the average rating per item and identify the items with the **largest** average rating.



Average rating for item *i*

$$b_i = \frac{\Sigma_u r_{ui}}{N_i}$$

r_{ui} : rating given by user *u* to item *i* (non zero ratings)

N_i : number of users who rated item *i*

Non-personalized Recommenders

- It puts on the same page the items rated by hundreds of users or the items rated by a single user.

Average rating for item

$$b_i = \frac{5+4+3}{3} = 4 \qquad b_j = \frac{5}{1} = 5$$

i		j		
5	0	0	1	2
4	2	0	2	3
0	0	0	1	2
0	0	5	0	0
3	0	0	0	5

Non-personalized Recommenders

- To correct this bias and give statistical significance, we take the same formula and add a C term to the denominator.
- The C term is called the shrink term. It is a constant value chosen depending on the properties of the URM.

Shrinked average rating for item *i*

$$b_i = \frac{\sum_u r_{ui}}{N_i + C}$$

r_{ui} : rating given by user *u* to item *i* (non zero ratings)

N_i : number of users who have rated item *i*

C : shrink term (constant value)

Non-personalized Recommenders

Average rating for item

$$b_i = \frac{5+4+3}{3} = 4 \qquad b_j = \frac{5}{1} = 5$$

Shrinked average rating for item (C=1)

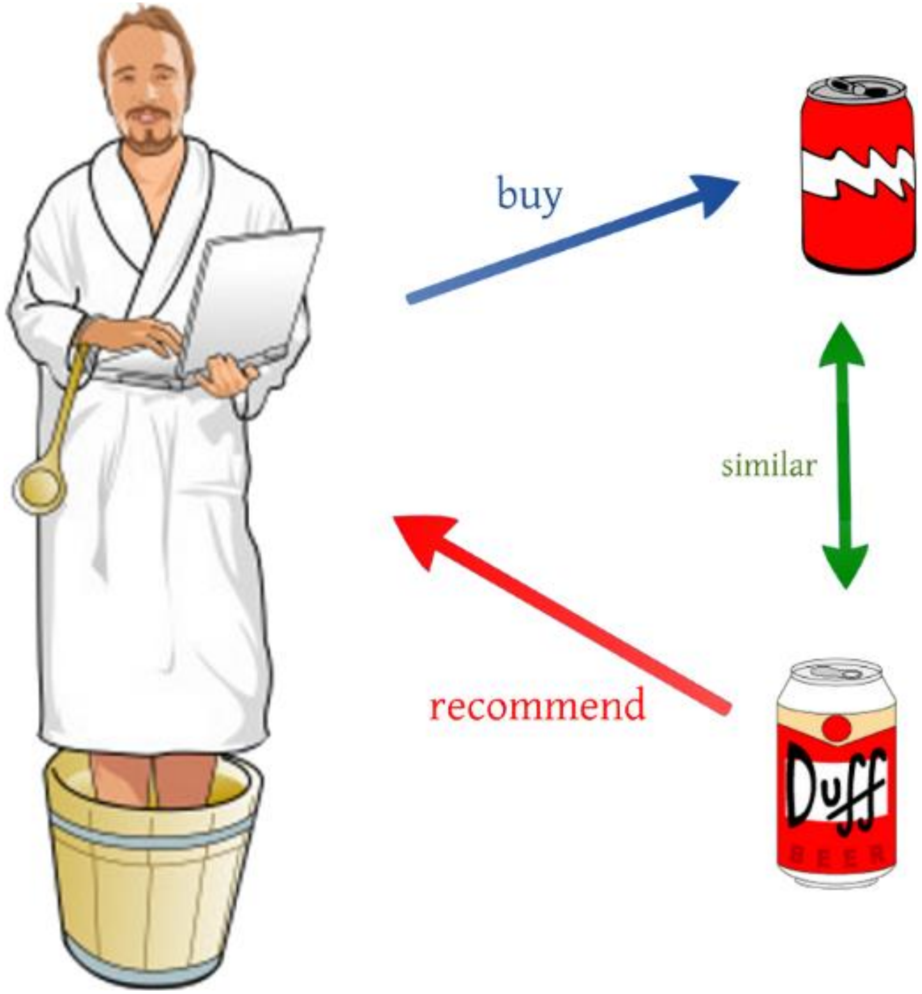
$$b_i = \frac{5+4+3}{3+1} = 3 \qquad b_j = \frac{5}{1+1} = 2.5$$

i		j		
5	0	0	1	2
4	2	0	2	3
0	0	0	1	2
0	0	5	0	0
3	0	0	0	5

Recommender Systems

Type	Definition	Example
content-based filtering	Uses similarity between items to recommend items similar to what the user likes.	If user A watches two cute cat videos, then the system can recommend cute animal videos to that user.
collaborative filtering	Uses similarities between queries and items simultaneously to provide recommendations.	If user A is similar to user B, and user B likes video 1, then the system can recommend video 1 to user A (even if user A hasn't seen any videos similar to video 1).

Content-based Filtering



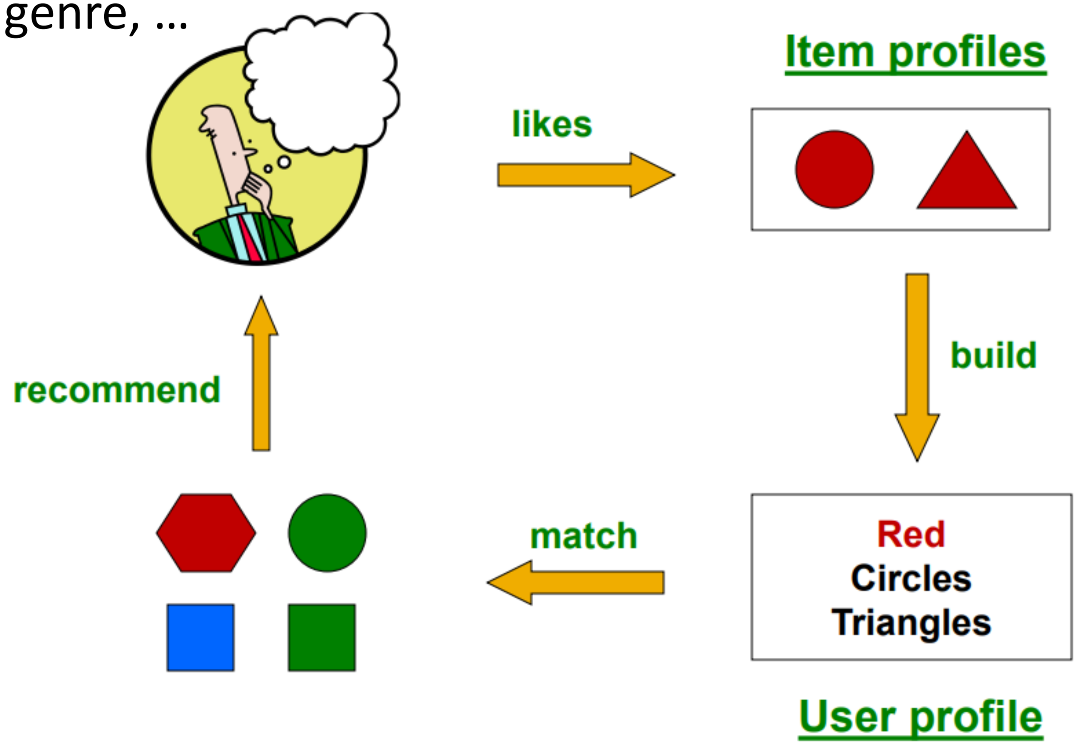
People who liked this also liked these as well

Content-based Filtering

Recommend items to customer x similar to previous items rated highly by x

Example:

- Movie recommendations
  - Recommend movies with same actor(s), director, genre, …
- Websites, blogs, news
  - Recommend other sites with "similar" content



Formal Model

- X = set of Users
- S = set of Items
- Utility function u: $X \times S \to R$
  - R = set of ratings
  - R is a totally ordered set
    - e.g., 0-5 stars, real number in [0,1]

Utlilty Matrix

Extrapolate **unknown ratings** from the **known** ones

	Avatar	LOTR	Matrix	Pirates
Alice	1	?	0.2	?
Bob	?	0.5	?	0.3
Carol	0.2	?	1	?
David	?	?	?	0.4

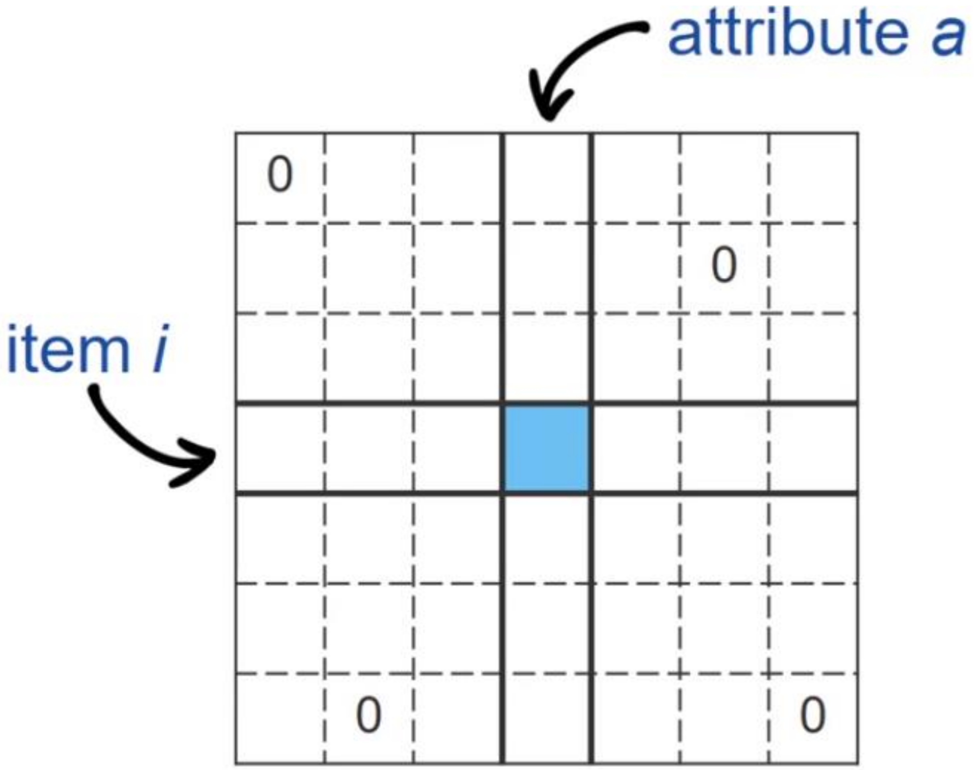
Item Profile

- For each item, create an item profile
- Profile is a set (vector) of features
  - Movies: author, title, actor, director,…
  - Text: Set of "important" words in document
  - Images, videos: metadata and tags
  - People: set of friends
- How to pick important text features?
  - Usual heuristic from text mining is **TF-IDF** (Term frequency * Inverse Doc Frequency)

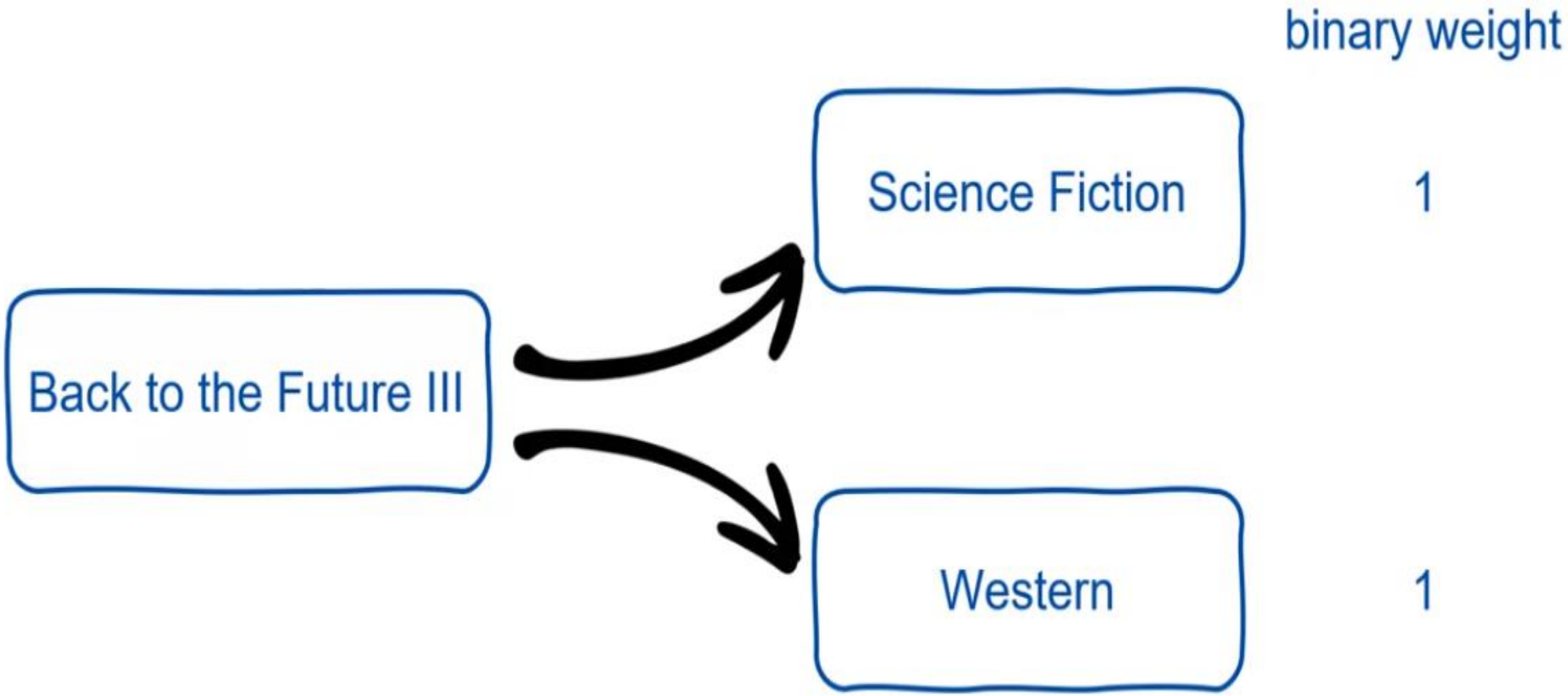
Item Profile

An element of the ICM is:

- 1 if the item has that attribute
- 0 otherwise



Non-binary attributes



Non-binary attributes

movieID	title	description	year	actors	director	cost
1	The Avengers	…	2012	Robert Downey JR, Scarlett Johansson, Chris Hemsworth, …	Joss Whedon, Anthony and Joe Russo	220M
2	Snow White and the Huntsman	…	2012	Kristen Stewart, Chris Hemsworth, Charlize Theron, …	Rupert Sanders	170M
3	The Avengers: Endgame	…	2019	Robert Downey JR, Scarlett Johansson, Chris Hemsworth, …	Anthony and Joe Russo	356M

Have these features the same importance?

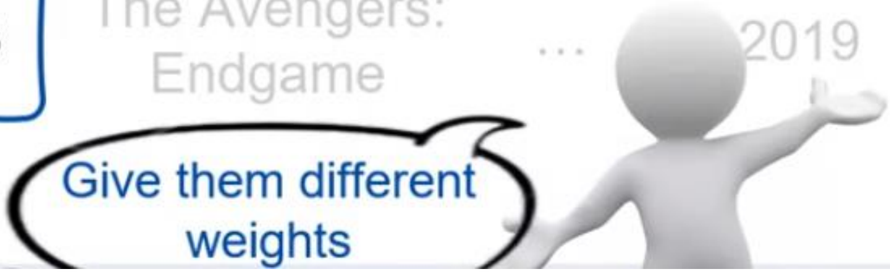
Non-binary attributes

movieID	title	description	year	actors	director	cost
1	The Avengers	…	2012	Robert Downey JR, Scarlett Johansson, Chris Hemsworth, …	Joss Whedon, Anthony and Joe Russo	220M
2	Snow White and the Huntsman	…	2012	Kristen Stewart, Chris Hemsworth, Charlize Theron, …	Rupert Sanders	170M
3	The Avengers: Endgam	…	2019	Robert Downey JR, Scarlett Johansson, Chris Hemsworth, …	Anthony and Joe Russo	356M

Movie 1 and 3: are the year of production and the cost essential attributes?

Non-binary attributes

movieID	title	description	year	actors	director	cost
	0.8	1.5	0.5	1.5	0.8	0.5
1	The Avengers	…	2012	Robert Downey JR, Scarlett Johansson, Chris Hemsworth, …	Joss Whedon, Anthony and Joe Russo	220M
2	Snow White and the Huntsman	…	2012	Kristen Stewart, Chris Hemsworth, Charlize Theron, …	Rupert Sanders	170M
3	The Avengers: Endgame	…	2019	Robert Downey JR, Scarlett Johansson, Chris Hemsworth, …	Anthony and Joe Russo	356M



TF-IDF

- **Term frequency - inverse document frequency** scales word occurrences by the inverse of their frequencies in the entire dataset instead of building the occurrence matrix on counts alone.
- TF-IDF assumes a document is just a "**bag of words**"

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

TF-IDF

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

title	(Crime, Romance)	(Comedy, Film-Noir)	(Comedy, Sci-Fi)	(Horror, Mystery, Thriller)	(Drama, Sci-Fi)
War Stories (1995)	0.0	0.0	0.000000	0.0	0.0
True Crime (1995)	0.0	0.0	0.000000	0.0	0.0
Color of Money, The (1986)	0.0	0.0	0.000000	0.0	0.0
My Cousin Vinny (1992)	0.0	0.0	0.000000	0.0	0.0
Hush (1998)	0.0	0.0	0.000000	0.0	0.0
Sour Grapes (1998)	0.0	0.0	0.000000	0.0	0.0
Roula (1995)	0.0	0.0	0.000000	0.0	0.0
Mars Attacks! (1996)	0.0	0.0	0.265326	0.0	0.0
Snow White and the Seven Dwarfs (1937)	0.0	0.0	0.000000	0.0	0.0
Faust (1994)	0.0	0.0	0.000000	0.0	0.0

IDF: Inverse Document Frequency

	a		b		c		
	1		1			1	
j	1	1	1	1	1	1	items
	1		1			1	
	1		1				

attributes

$$TF_{a,i} = \frac{N_a}{N_i} = \frac{1}{3}$$

$$TF_{c,i} = \frac{N_c}{N_i} = \frac{0}{3} = 0$$

$$TF_{a,j} = \frac{N_a}{N_j} = \frac{1}{6}$$

IDF: Inverse Document Frequency

a		b		c		
1		1			1	items
1				1		
1		1			1	
1		1				

attributes

$$IDF_a = \log \frac{N_{\text{ITEMS}}}{N_a} = \log \frac{4}{4} = 0.00$$

IDF: Inverse Document Frequency

a		b		c	
1		1			1
1				1	
1		1			1
1		1			

attributes

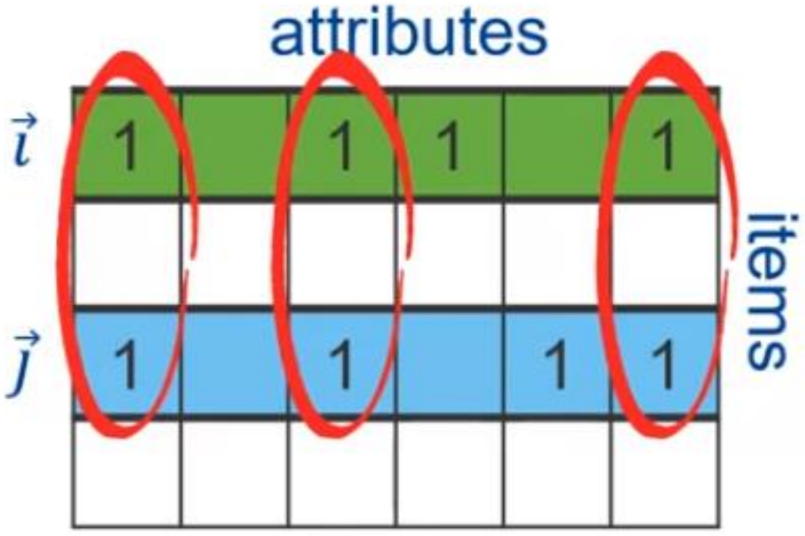
items

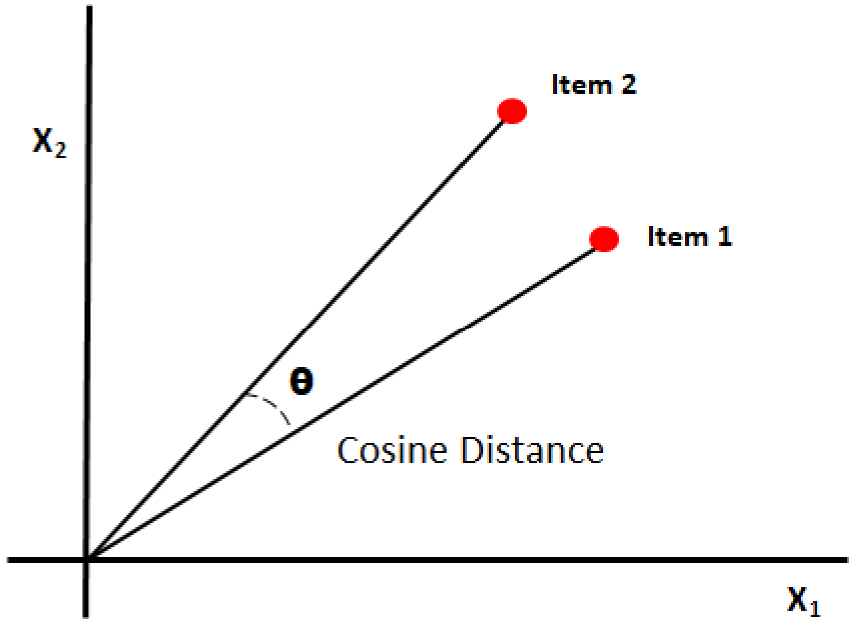
$$IDF_a = \log \frac{N_{\text{ITEMS}}}{N_a} = \log \frac{4}{4} = 0.00$$

$$IDF_b = \log \frac{N_{\text{ITEMS}}}{N_b} = \log \frac{4}{3} = 0.12$$

$$IDF_c = \log \frac{N_{\text{ITEMS}}}{N_c} = \log \frac{4}{1} = 0.60$$

Computing Predictions …
Using Dot Product as a Similarity Measure





many attributes in common → the two items are very similar

Content-based Filtering Advantages

- No need for data on other users
  - The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.
- Able to recommend to users with unique tastes
- Able to recommend new & unpopular items
  - The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.
- Able to provide explanations
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended.

Content-based Filtering Disadvantages

- Finding the appropriate features is hard
  - E.g., images, movies, music
- Recommendations for new users
  - How to build a user profile?
- Overspecialization
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users.
- Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.

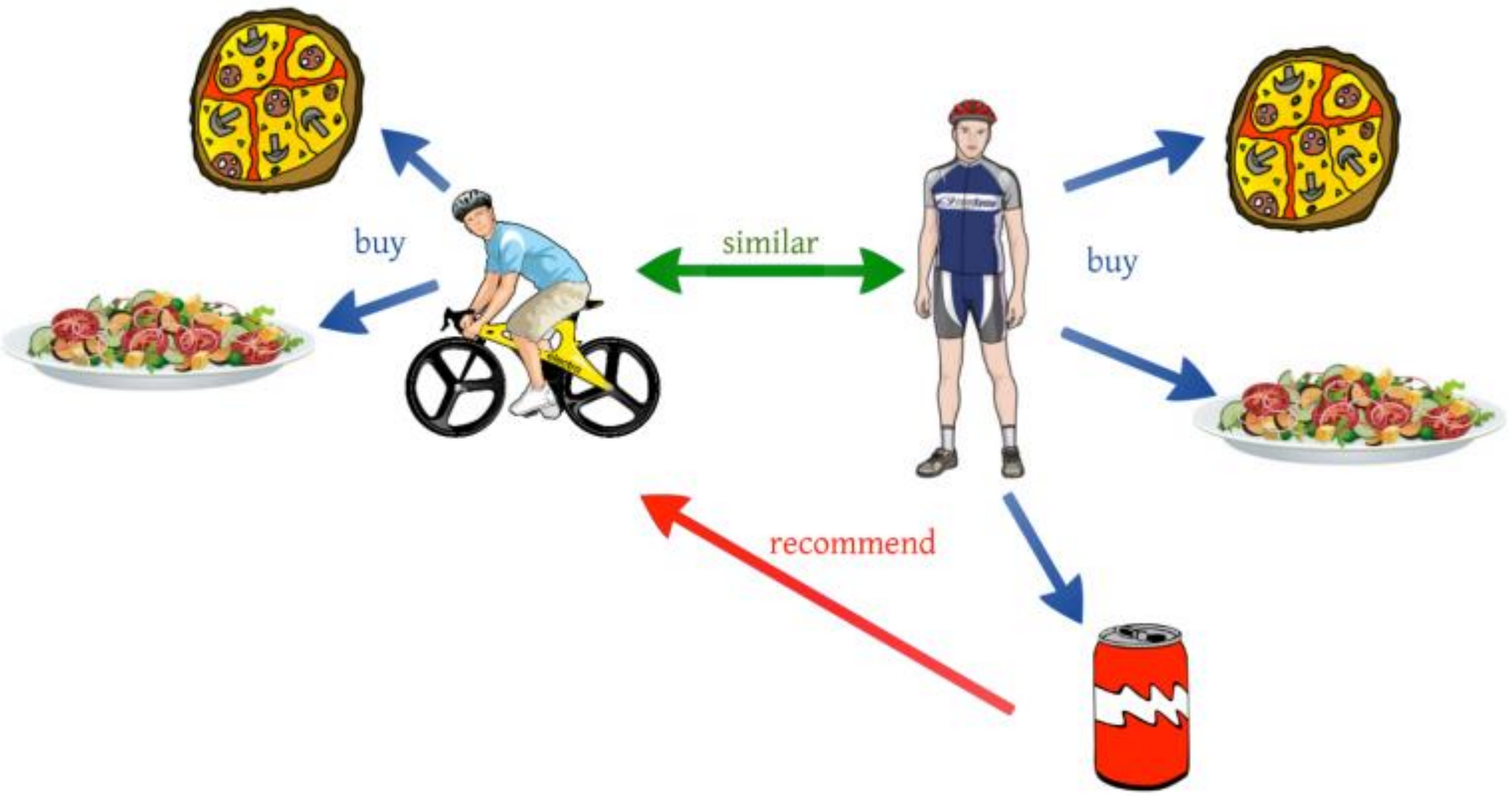
- lab

Collaborative Filtering

- To address some of the limitations of content-based filtering, collaborative filtering uses similarities between users and items simultaneously to provide recommendations.
- This allows for serendipitous recommendations; that is, collaborative filtering models can recommend an item to user A based on the interests of a similar user B.
- Furthermore, the embeddings can be learned automatically, without relying on hand-engineering of features.

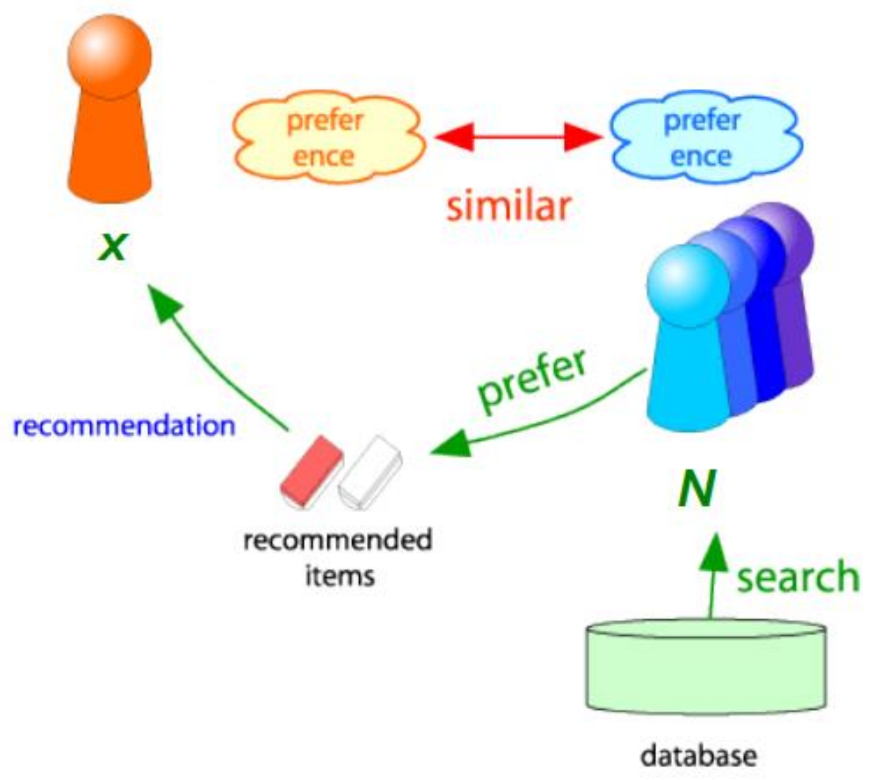
Collaborative Filtering

People with similar taste to you like the thing you like.



Collaborative Filtering

- Find set N of other users whose ratings are "similar" to x's ratings
- Estimate x's ratings based on ratings of users in N



A Movie Recommendation Example

- Consider a movie recommendation system in which the training data consists of a feedback matrix in which:
  - Each row represents a user.
  - Each column represents an item (a movie).
- The feedback about movies falls into one of two categories:
  - **Explicit**— users specify how much they liked a particular movie by providing a numerical rating.
  - **Implicit**— if a user watches a movie, the system infers that the user is interested.
- To simplify, we will assume that the feedback matrix is binary; that is, a value of 1 indicates interest in the movie.
- When a user visits the homepage, the system should recommend movies based on both:
  - similarity to movies the user has liked in the past
  - movies that similar users liked

Collaborative Filtering

- User-user collaborative filtering
- Item-Item collaborative filtering

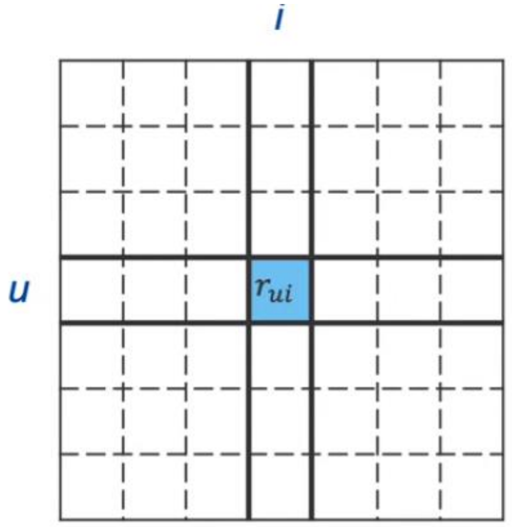
User-user Vs Item-Item collaborative filtering

User based

$$\tilde{r}_{ui} = \frac{\sum_{v \in \mathrm{KNN}(u)} r_{vi} \cdot s_{vu}}{\sum_{v \in \mathrm{KNN}(u)} s_{vu}}$$

Item based

$$\tilde{r}_{ui} = \frac{\sum_{j \in \mathrm{KNN}(i)} r_{uj} \cdot s_{ji}}{\sum_{j \in \mathrm{KNN}(i)} s_{ji}}$$



r_{ui} = rating that user *u* gave to item *i*

User-user Vs Item-Item collaborative filtering

User based	Item based
$\tilde{r}_{ui} = \frac{\sum_{v \in \mathrm{KNN}(u)} r_{vi} \cdot s_{vu}}{\sum_{v \in \mathrm{KNN}(u)} s_{vu}}$	$\tilde{r}_{ui} = \frac{\sum_{j \in \mathrm{KNN}(i)} r_{uj} \cdot s_{ji}}{\sum_{j \in \mathrm{KNN}(i)} s_{ji}}$

User based

1. Find similar users based on interactions with common items.
2. Identify the items rated high by similar users but have not been exposed to the active user of interest.
3. Calculate the weighted average score for each item.
4. Rank items based on the score and pick the top n items to recommend.

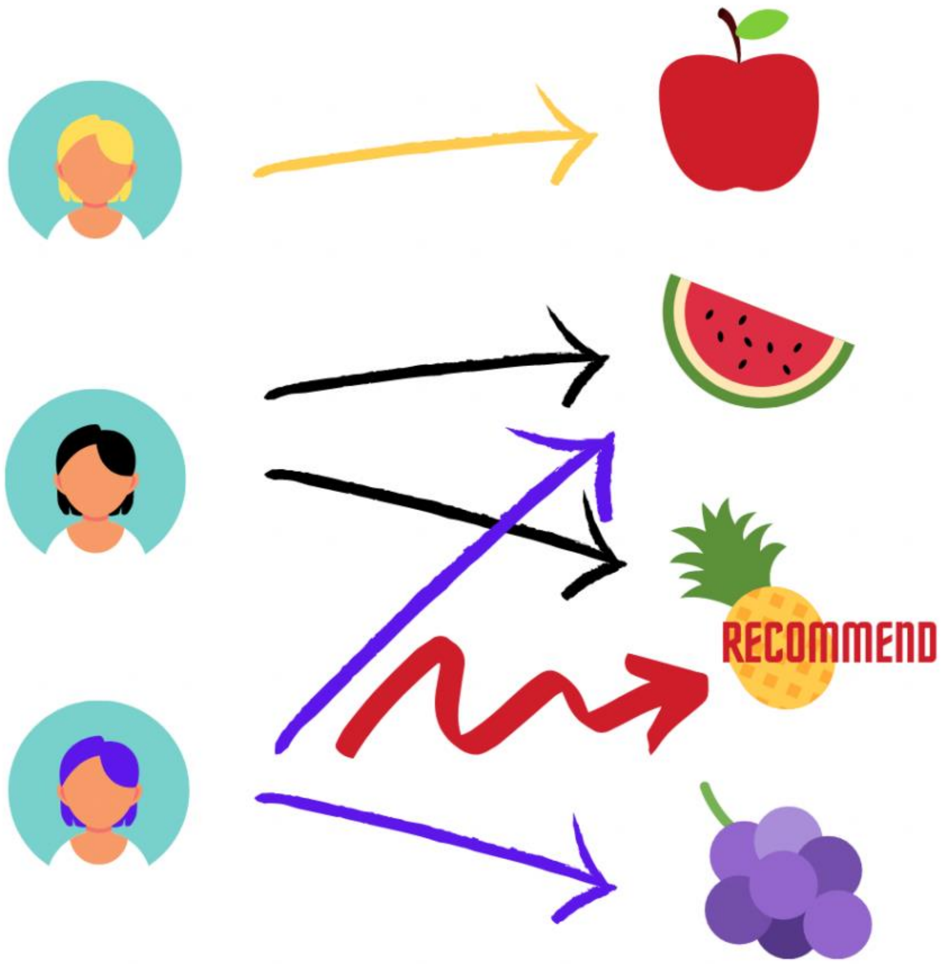
Item based

1. Calculate item similarity scores based on all the user ratings.
2. Identify the top n items that are most similar to the item of interest.
3. Calculate the weighted average score for the most similar items by the user.
4. Rank items based on the score and pick top n items to recommend.

User-user collaborative filtering

User-based collaborative filtering

- Ms. Blond likes apples. Ms. Black likes watermelon and pineapple. Ms. Purple likes watermelon and grape.
- Because Ms. Black and Ms. Purple like the same fruit, watermelon, they are similar users.
- Since Ms. Black likes pineapple and Ms. Purple has not been exposed to pineapple yet, the recommendation system recommends pineapple to Ms. purple.



User-based collaborative filtering algorithm steps:

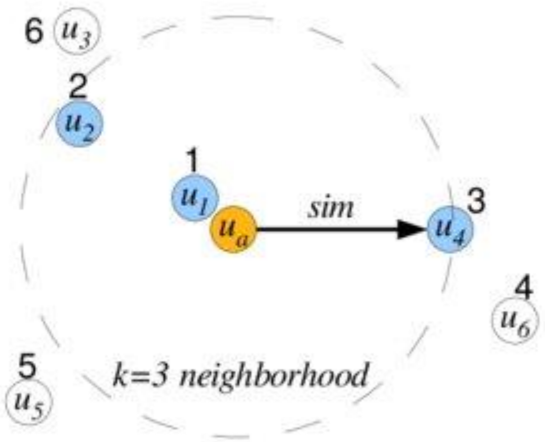
1. Find similar users based on interactions with common items.
2. Identify the items rated high by similar users but have not been exposed to the active user of interest.
3. Calculate the weighted average score for each item.
4. Rank items based on the score and pick the top n items to recommend.

R	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	?	4.0	4.0	2.0	1.0	2.0	?	?
u_2	3.0	?	?	?	5.0	1.0	?	?
u_3	3.0	?	?	3.0	2.0	2.0	?	3.0
u_4	4.0	?	?	2.0	1.0	1.0	2.0	4.0
u_5	1.0	1.0	?	?	?	?	?	1.0
u_6	?	1.0	?	?	1.0	1.0	?	1.0
u_a	?	?	4.0	3.0	?	1.0	?	5.0
$\hat{r}_a$	3.5	4.0			2.3		2.0	

(a)

S_a	u_a
u_1	0.3
u_2	1.0
u_3	0.2
u_4	0.3
u_5	0.1
u_6	0.1

(b)



(c)

User based

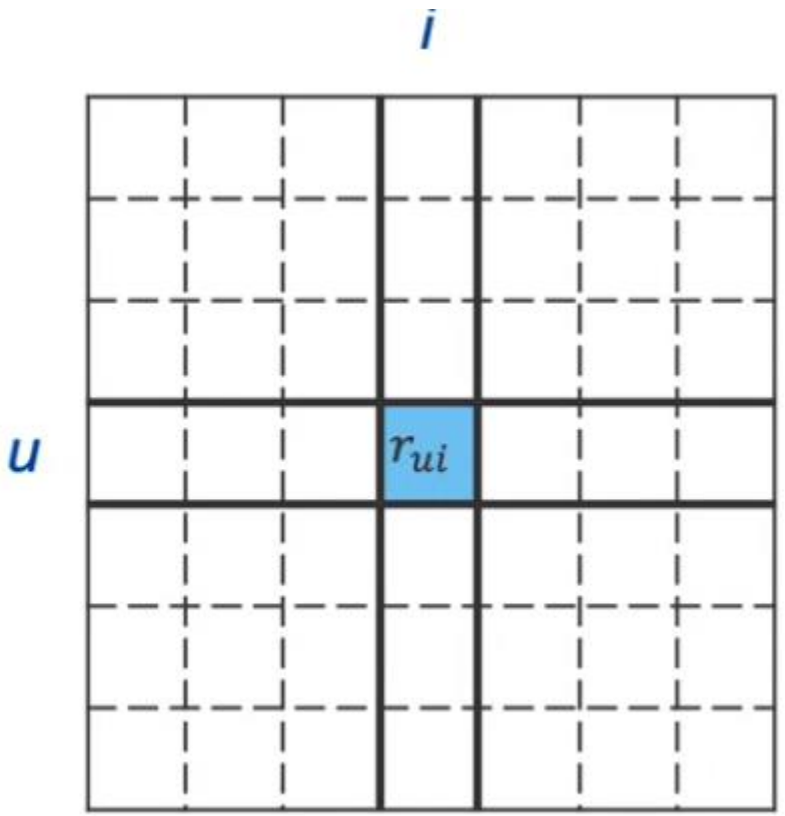
$$\tilde{r}_{ui} = \frac{\Sigma_{v \in \mathrm{KNN}(u)} r_{vi} \cdot s_{vu}}{\Sigma_{v \in \mathrm{KNN}(u)} s_{vu}}$$

User Rating Matrix

Input to CF algorithms:

matrix containing past interactions between users and items

- for explicit ratings, examples of values can be from 1 to 5 (stars)
- for implicit ratings, 1 if the user has interacted with the item, 0 otherwise



r_{ui} = rating that user *u* gave to item *i*

How to find "similar" Users

- Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings
- **Jaccard similarity measure**
  - **Problem:** Ignores the value of the rating

r_x, r_y as sets:
$r_x = \{1, 4, 5\}$
$r_y = \{1, 3, 4\}$

- **Cosine similarity measure**
  - sim($\boldsymbol{x}$, $\boldsymbol{y}$) = cos($\boldsymbol{r_x}$, $\boldsymbol{r_y}$) = $\frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
  - **Problem:** Treats missing ratings as "negative"

r_x, r_y as points:
$r_x = \{1, 0, 0, 1, 3\}$
$r_y = \{1, 0, 2, 2, 0\}$

How to find “similar” Users

- **Pearson correlation coefficient**
  - S_{xy} = items rated by both users ***x*** and ***y***

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

$\overline{r_x}, \overline{r_y}$ … avg. rating of **x**, **y**

How to find "similar" Users

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Consider users $\boldsymbol{x}$ and $\boldsymbol{y}$ with rating vectors $\boldsymbol{r_x}$ and $\boldsymbol{r_y}$
- We need a similarity metric sim($\boldsymbol{x}$, $\boldsymbol{y}$)

Jaccard Similarity

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- sim(A,B) = $| r_A \cap r_B | / | r_A \cup r_B |$
- sim(A,B) = 1/5; sim(A,C) = 2/4
  - sim(A,B) < sim(A,C)

Ignores the value of the rating

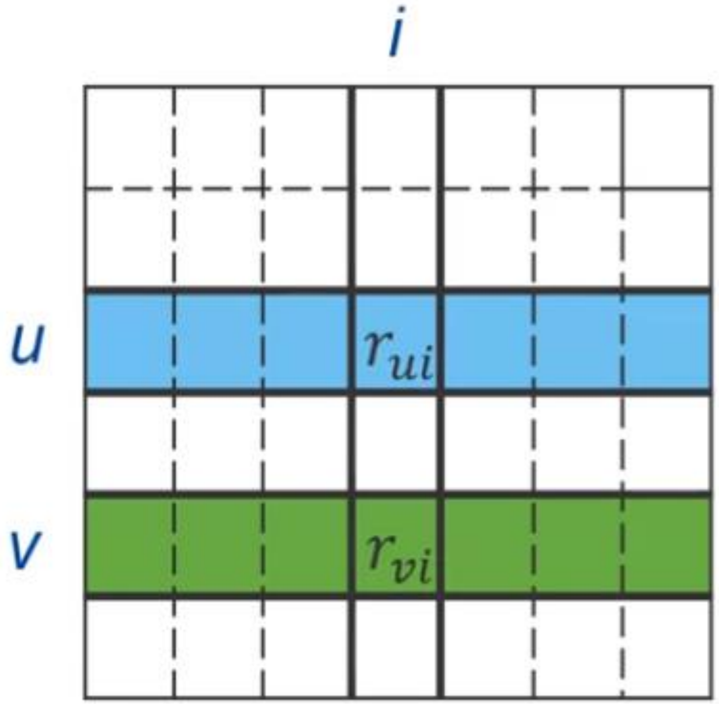
Cosine Similarity

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Treat unknown values as **zeros**

- sim(A,B) = cos(r_A, r_B)
- sim(A,B) = 0.38, sim(A,C) = 0.32
  - sim(A,B) > sim(A,C), but not by much

User Similarity: Implicit Ratings



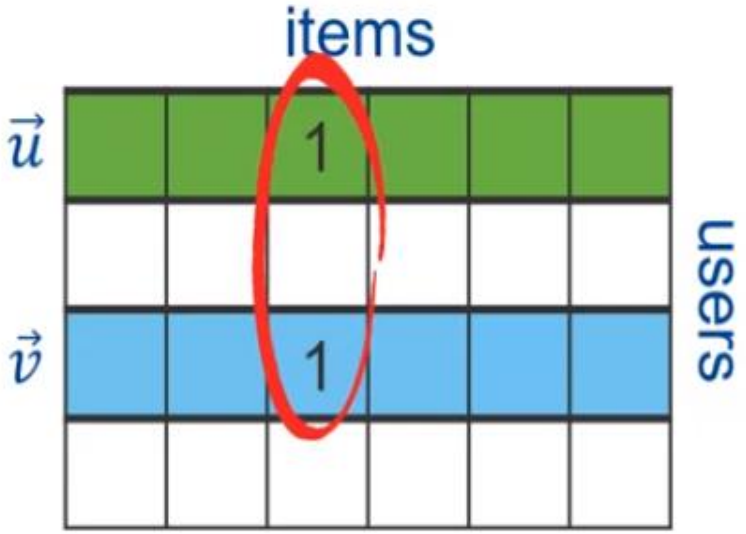
Cosine similarity:

$$s_{uv} = \frac{\Sigma_i r_{ui} \cdot r_{vi}}{\sqrt{\Sigma_i r_{ui}^2 \cdot \Sigma_i r_{vi}^2}} = \frac{\vec{r}_u \cdot \vec{r}_v}{|\vec{r}_u|_2 \cdot |\vec{r}_v|_2}$$

User Similarity: Implicit Ratings

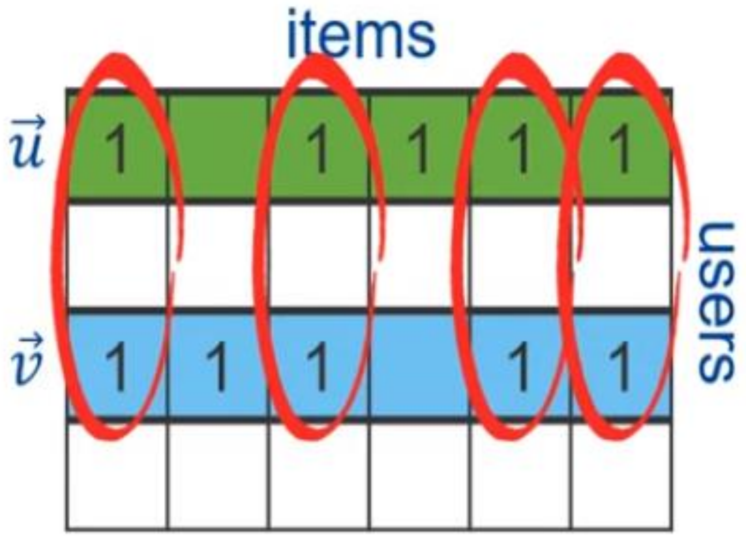
$$s_{uv} = \# < \mathrm{u}, \mathrm{v} > = \Sigma_i r_{ui} \cdot r_{vi} = \vec{u} \cdot \vec{v} \qquad s_{uv} = 4$$

Support: examples



small support

$$s_{uv} = \frac{1}{\sqrt{1 \cdot 1}} = 1$$



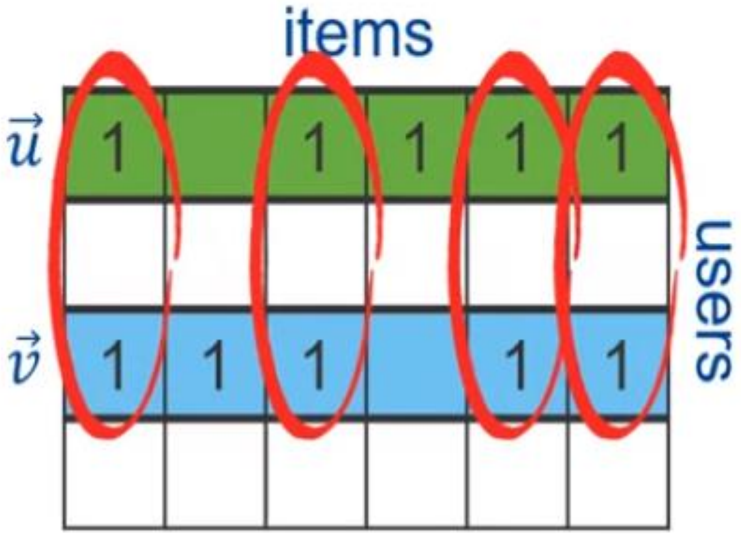
large support

$$s_{uv} = \frac{4}{\sqrt{5 \cdot 5}} = 0.8$$

Support: examples

small support

$$s_{uv} = \frac{1}{\sqrt{1 \cdot 1}} = 1$$



large support

$$s_{uv} = \frac{4}{\sqrt{5 \cdot 5}} = 0.8$$

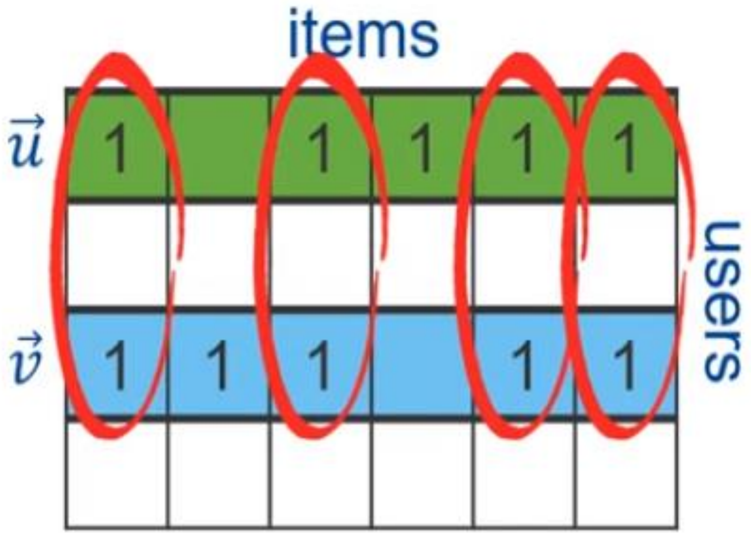
Shrinking

trust similarities only if several items share the same ratings:

add Shrink Term *C*

$$s_{uv} = \frac{\vec{r}_u \cdot \vec{r}_v}{|\vec{r}_u|_2 \cdot |\vec{r}_v|_2 + C} = \frac{\# < u, v >}{\sqrt{\# < u > \# < \# v >} + C}$$

Shrinking



example: C = 3

$$s_{uv} = \frac{1}{\sqrt{1 \cdot 1} + 3} = 0.25$$

$$s_{uv} = \frac{4}{\sqrt{5 \cdot 5} + 3} = 0.5$$

Similarity matrix

$$s_{uv} = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}|_2 \cdot |\vec{v}|_2 + C}$$

	user v			
user u	-	0.3	0.15	0.2
	0.3	-	0.6	0.43
	0.15	0.6	-	0.98
	0.2	0.43	0.98	-

Pearson Correlation

- Normalize ratings by subtracting row mean

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

Pearson Correlation

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	−7/3		
B	1/3	1/3	−2/3				
C				−5/3	1/3	4/3	
D		0					0

- sim(A,B) = cos(r_A, r_B) = 0.09; sim(A,C) = -0.56
  - sim(A,B) > sim(A,C)
- Captures intuition better
  - Missing ratings treated as "average"
  - Handles "tough raters" and "easy raters"

From similarity metric to recommendations

- Let r_x be the vector of user x's ratings
- Let N be the set of k users most similar to x who have rated item i
- Prediction for item i of user x:
  - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
  - $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

Shorthand:
$s_{xy} = sim(x, y)$

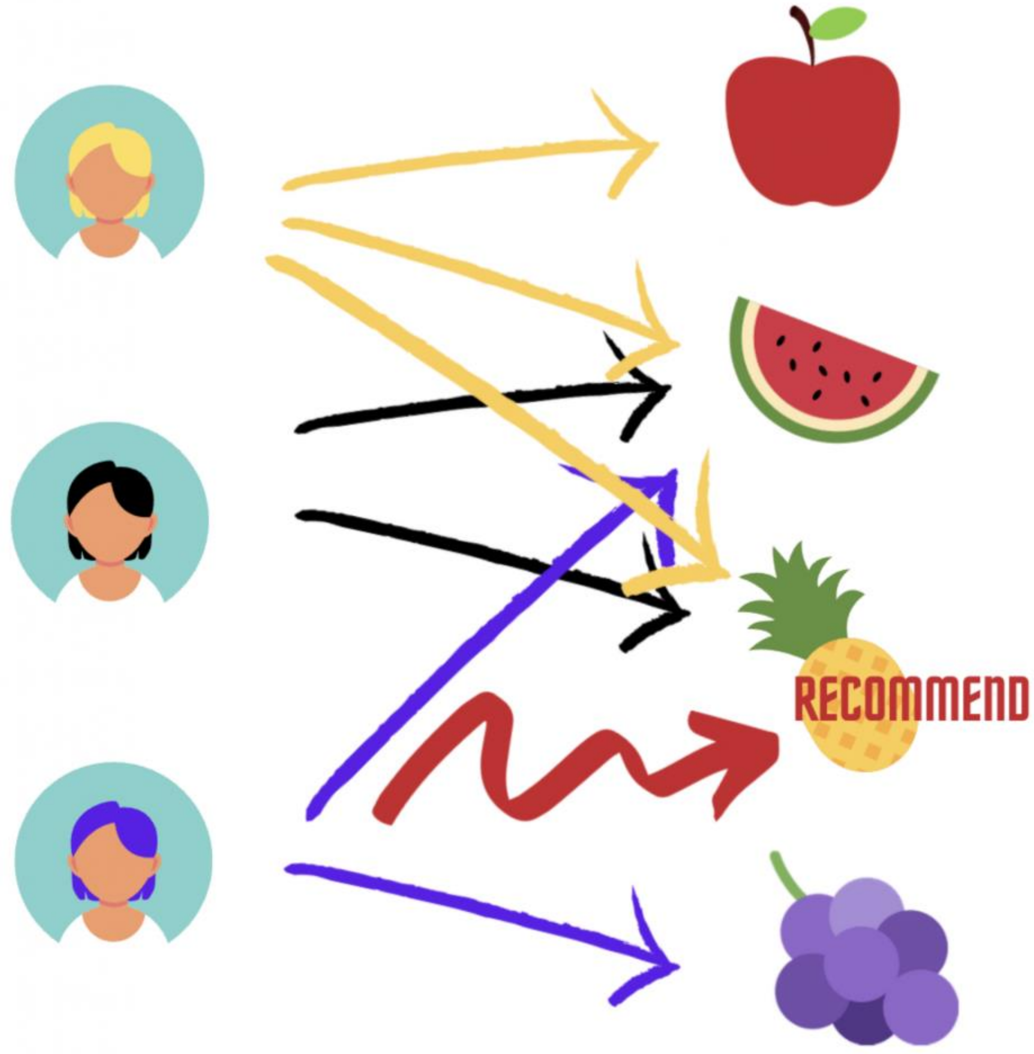
Item-Item collaborative filtering

Item-item collaborative filtering

- Item-based collaborative filtering makes recommendations based on user-product interactions in the past.
- The assumption behind the algorithm is that users like similar products and dislike similar products, so they give similar ratings to similar products.

Item-item collaborative filtering

- Ms. Blond likes apples, watermelons, and pineapples. Ms. Black likes watermelons and pineapples. Ms. Purple likes watermelons and grapes.
- Because watermelons and pineapples are liked by the same persons, they are considered similar items.
- Since Ms. Purple likes watermelons and Ms. Purple has not been exposed to pineapples yet, the recommendation system recommends pineapples to Ms. purple.



Item-item collaborative filtering

- For item i, find other similar items
- Estimate rating for item i based on ratings for similar items
- Can use same similarity metrics and prediction functions as in user-user model

$$\boxed{r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}}$$

s_{ij}… similarity of items i and j
r_{xj}…rating of user u on item j
$N(i;x)$… set items rated by x similar to i

Item-based collaborative filtering algorithm steps

1. Calculate item similarity scores based on all the user ratings.
2. Identify the top n items that are most similar to the item of interest.
3. Calculate the weighted average score for the most similar items by the user.
4. Rank items based on the score and pick top n items to recommend.

Item-item collaborative filtering

- **Item to Item Similarity:** The first step is to build the model by finding similarity between all the item pairs. The similarity between item pairs can be found in different ways.
- **Prediction Computation:** The second stage involves executing a recommendation system. It uses the items (already rated by the user) that are most similar to the missing item to generate rating. We hence try to generate predictions based on the ratings of similar products. We compute this using a formula which computes rating for a particular item using weighted sum of the ratings of the other similar products.

$$\boxed{r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}}$$

s_{ij}… similarity of items ***i*** and ***j***
r_{xj}…rating of user ***u*** on item ***j***
N(i;x)… set items rated by ***x*** similar to ***i***

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- unknown rating

- rating between 1 to 5

Item-Item CF (|N|=2)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- estimate rating of movie **1** by user **5**

Item-Item CF (|N|=2)

The first step is to build the model by finding similarity between all the item pairs.

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	..
3	2	4		1	2		3		4	3	5		?
4		2	4		5			4			2		..
5			4	3	4	2					2	5	..
6	1		3		3			2			4		

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Item-Item CF (|N|=2)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use Pearson correlation as similarity:

1) Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = 3.6$

row 1: [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]

2) Compute cosine similarities between rows

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

Compute similarity weights:

$s_{1,3}=0.41$, $s_{1,6}=0.59$

Item-Item CF (|N|=2)

Calculate the weighted average score for the most similar items by the user.

users

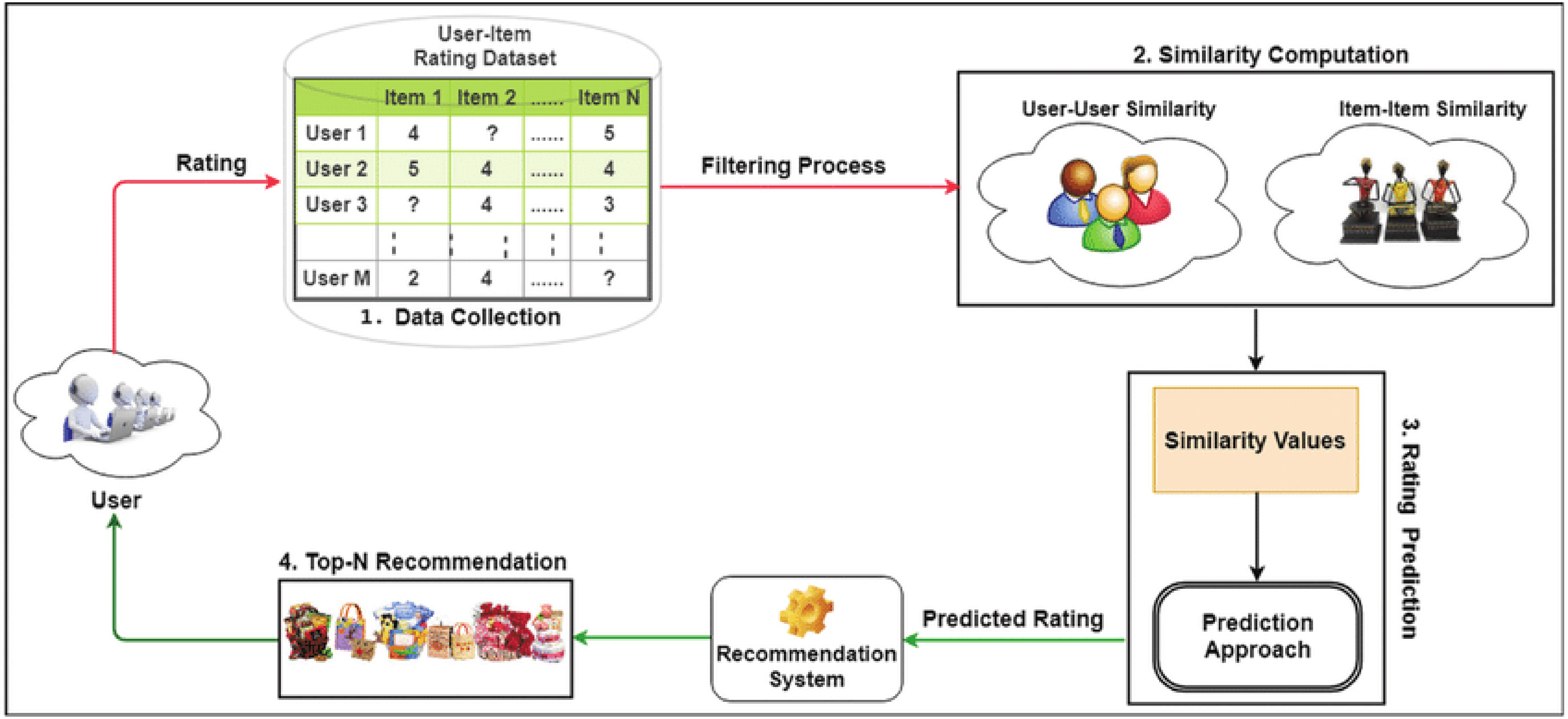
movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		**2.6**	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Predict by taking weighted average:

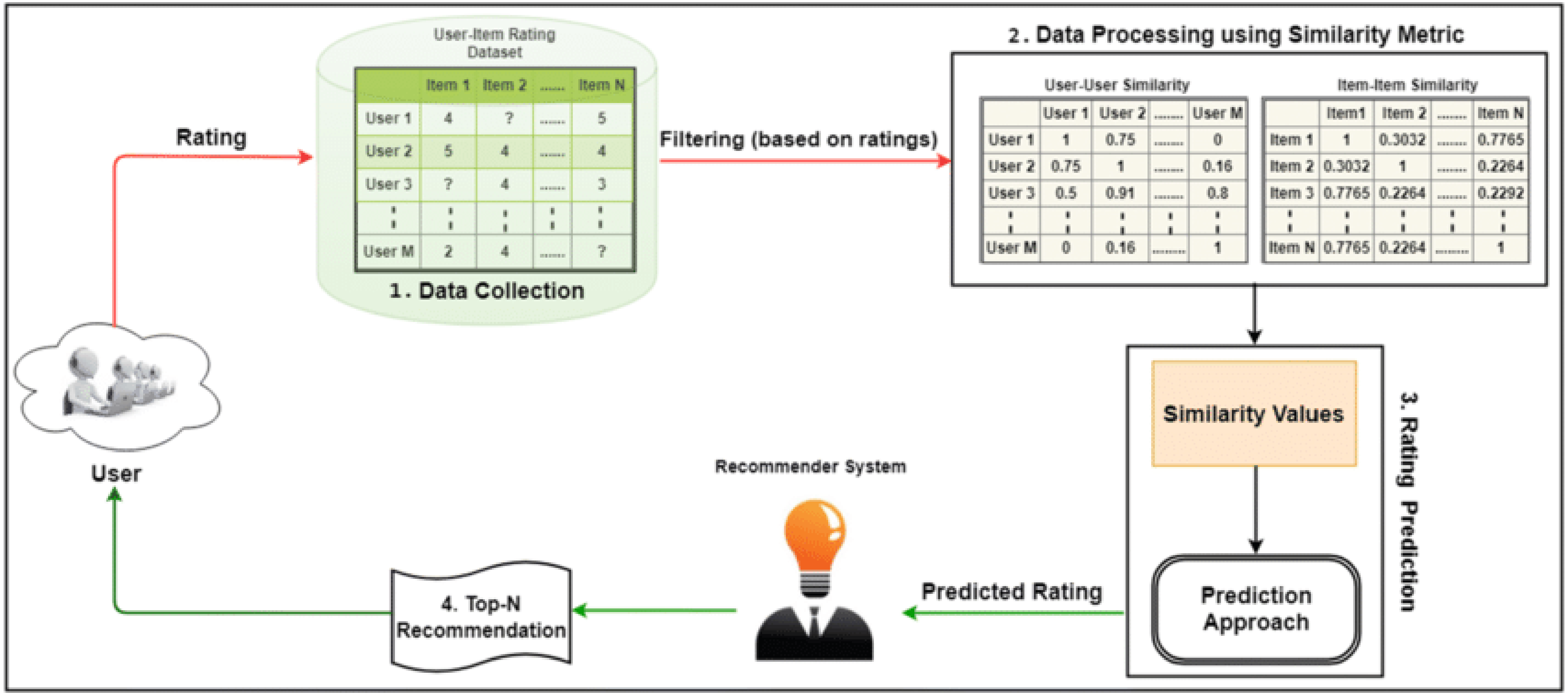
$r_{1.5}$ = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

Collaborative Filtering



Collaborative Filtering



User-User Vs Item-Item collaborative filtering

- In practice, item-item often works better than user-user

Why?

Items are simpler, users have multiple tastes

(People are more complex than Items)

Pros/Cons of Collaborative Filtering

+ Works for any kind of item
  - No feature selection needed

- Cold Start:
  - Need enough users in the system to find a match

- Sparsity:
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items

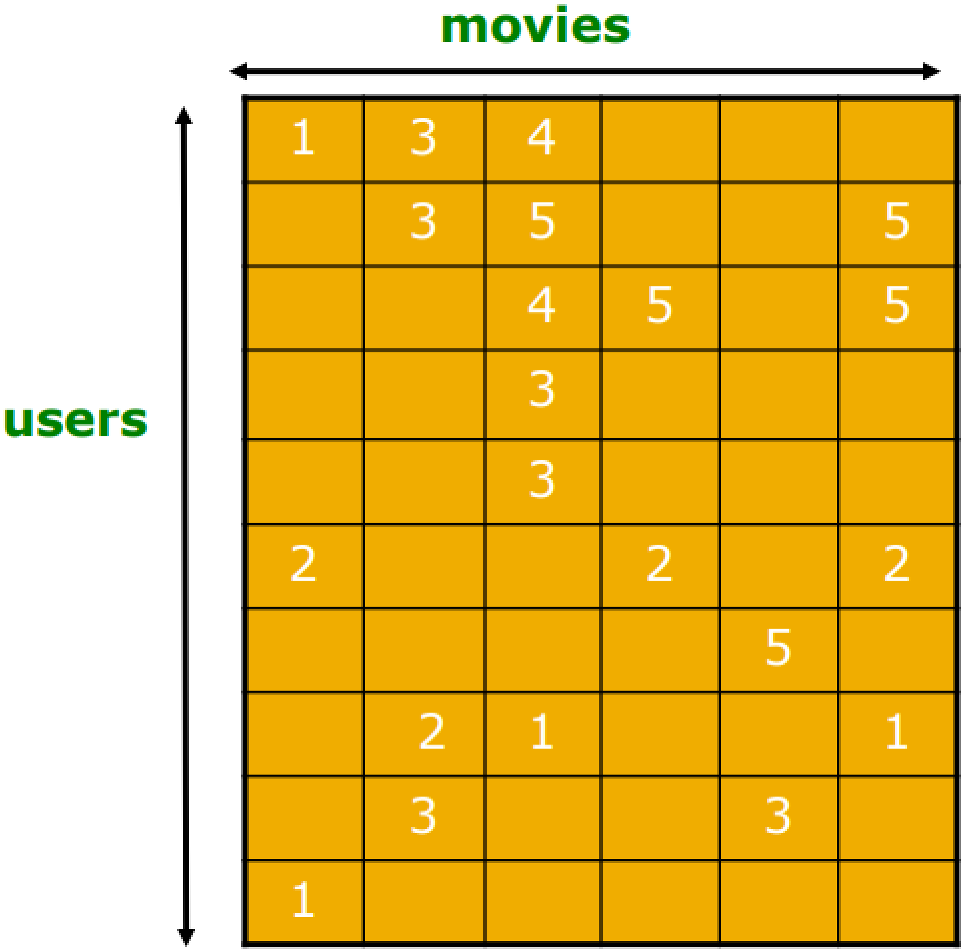
- First rater:
  - Cannot recommend an item that has not been previously rated

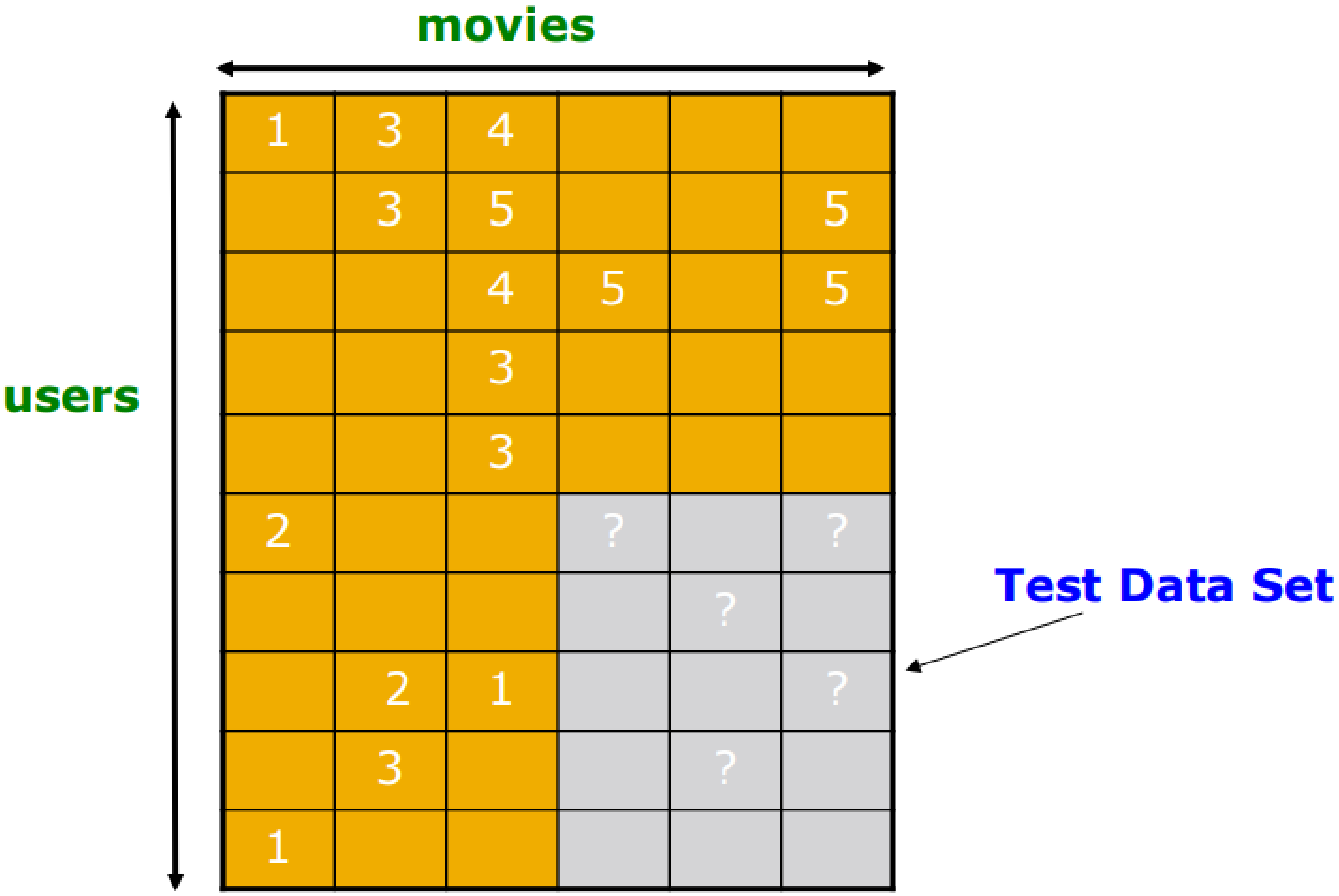
- Popularity bias:
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items

Hybrid Methods

- Implement two or more different recommenders and combine predictions
- Add content-based methods to collaborative filtering
  - Item profiles for new item problem
  - Demographics to deal with new user problem

Evaluation





Evaluation

- Compare predictions against withheld known ratings (test set *T*)

- Root-mean-square error (RMSE)

$$\sqrt{\frac{\sum_{(x,i)\in T}(r_{xi}-r^{*}_{xi})^2}{N}}$$

where $N = |T|$

r_{xi} is the predicted rating

r_{xi}^{*} is the actual rating