06. Recommendation Systems

What are recommendation systems?

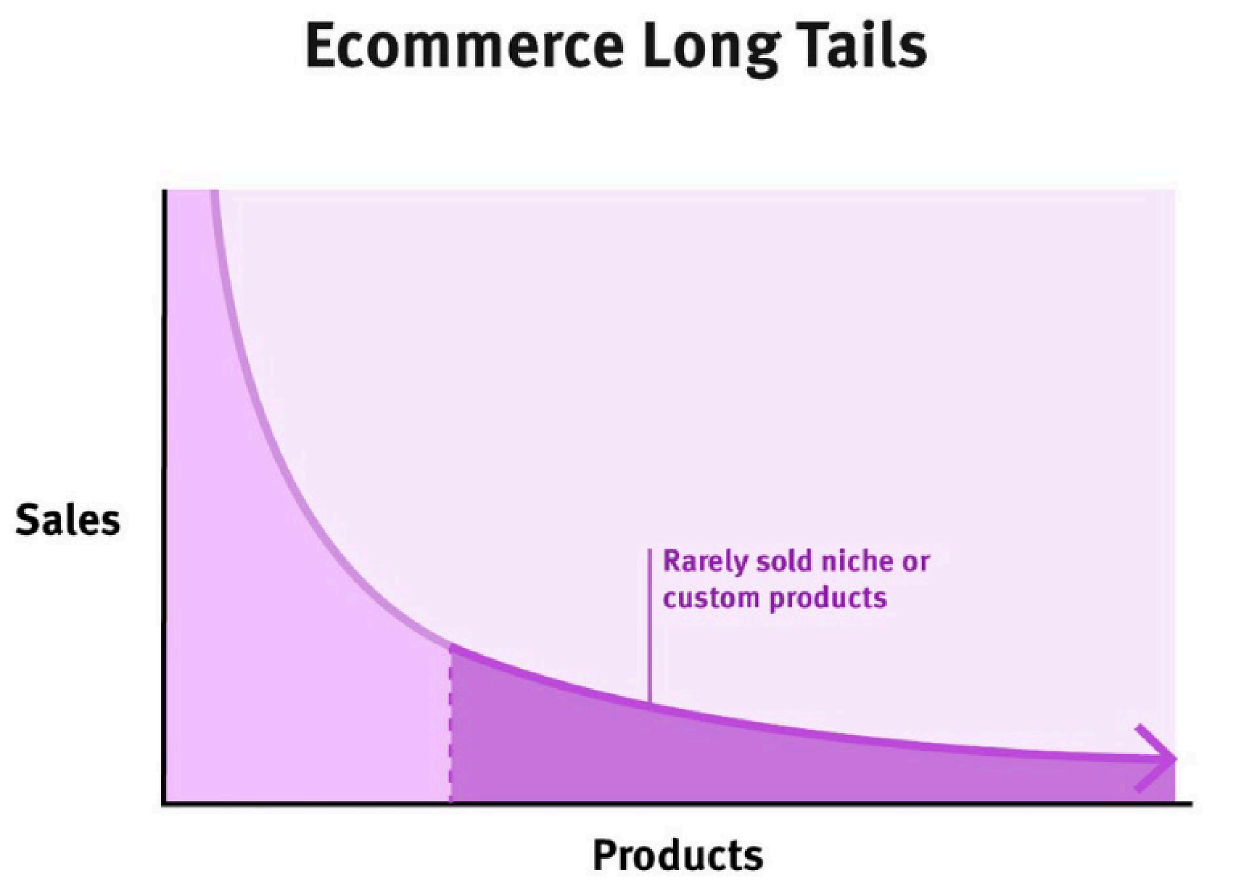
Purpose

In traditional media, broadcasting time is limited. The commercial income is dependent on the number of viewers consuming the media, so there was a focus on playing popular media.

The contrast with modern media is that what is broadcast depends from consumer to consumer. The purpose of recommendation systems is **to give everyone a personalized experience**, which retains their attention, and ensures income.

Problem: Long tail

20% of products account for 80% of the sales.



Other products might make you happier, but they don't get recommended.

Examples

Spotify

20-30 years ago, everyone had the same music taste, but now everyone is different. That is because of recommendation systems.

Netflix

> *The ones that made recommender systems popular.*

Created a challenge (Netflix challenge) that made them popular.

- First occasion of a large group of developers creating a successful recommendation system.

Social Media

In the early days of Facebook, the only content you could see was what was posted by your friends. Now, you see a bunch of other stuff that's optimized to keep your attention.

Tiktok has the best recommendation system in this field.

Dangers

1. The filter bubble is made specifically for you.
2. After a while, you always get the same recommendations.

Type of input data

> What does it need to be good?

Items data

> *I have a movie, what are the actors in it, what is the director and budget*

User data

> *Who are you? What is your gender, place of birth?*

Interaction

> *Person X watched the movie. Did they like it?*

Combine

All of these aspects combined produce recommendation systems.

Algorithm types

Non-Persanolized

> *Top 10 movies of the month

Persanolized

Knowledge-Based Filtering

> *If you are from Chine, you will like a Chinese movie*

Content-Based Filtering

> *If you like this movie, you will like this one, cause it ahs the same actors*

Collaborative Filtering

> *He liked this movie, so you will too*

- User-Based
- Item-Based
- Matrix Factorization

Context-Aware Filtering

> *It's Christmas, so you'll like this Christmas movie*

Hybrid

> *Combination of everything else*

- Deep Learning
- Factorization Machines
- etc.

Item-Content Matrix

On one dimension, there are different movies (**items**), and on the other dimension are the **properties** (genres).

User-Rating Matrix

X - items, y - users, each value is the user's rating

Explicit information

A rating that the user actually gives.

Large Rating scale

> Score of 5 stars

Gives a lot of information, but asks for a lot of user effort.

Small Rating scale

> Thumbs up, or thumbs down

Less information, but a lot easier to enter.

Even rating scale

You are forced to express your opinion.

Odd rating scale

More ratings, but they are less strong, because most people are neutral towards things.

User Bias

People tend to give really high ratings on booking.com. People have seen the owner, and don't want to be mean towards them.

Implicit information

Information that you get without explicitly asking the user.

> *Did the user watch the whole movie? Did he fast-forward?*

Implicit information is more important than explicit information

This is because explicit information is highly dependent on the person's mood.

Non-Persanolized Recommenders

1. Top Popular Item

When you have a matrix, you choose the item that has been rated the most.
It doesn't say anything about the quality of the item, just about the popularity of it.

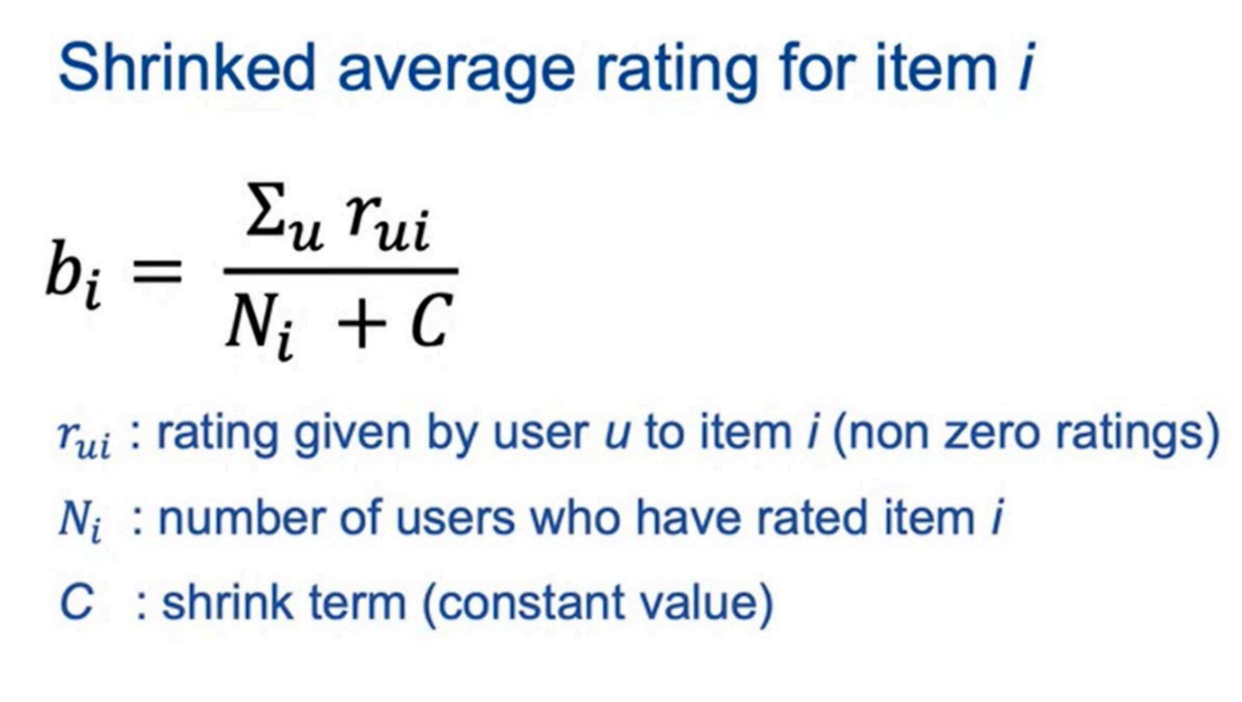
2. Best Rated Item

Picks the item with the largest average rating.

× **Problem:**

Rarely rated items can be recommended, if their rating is large.

✓ **Solution: Shrink Term**



Shrinked average rating for item *i*

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

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)

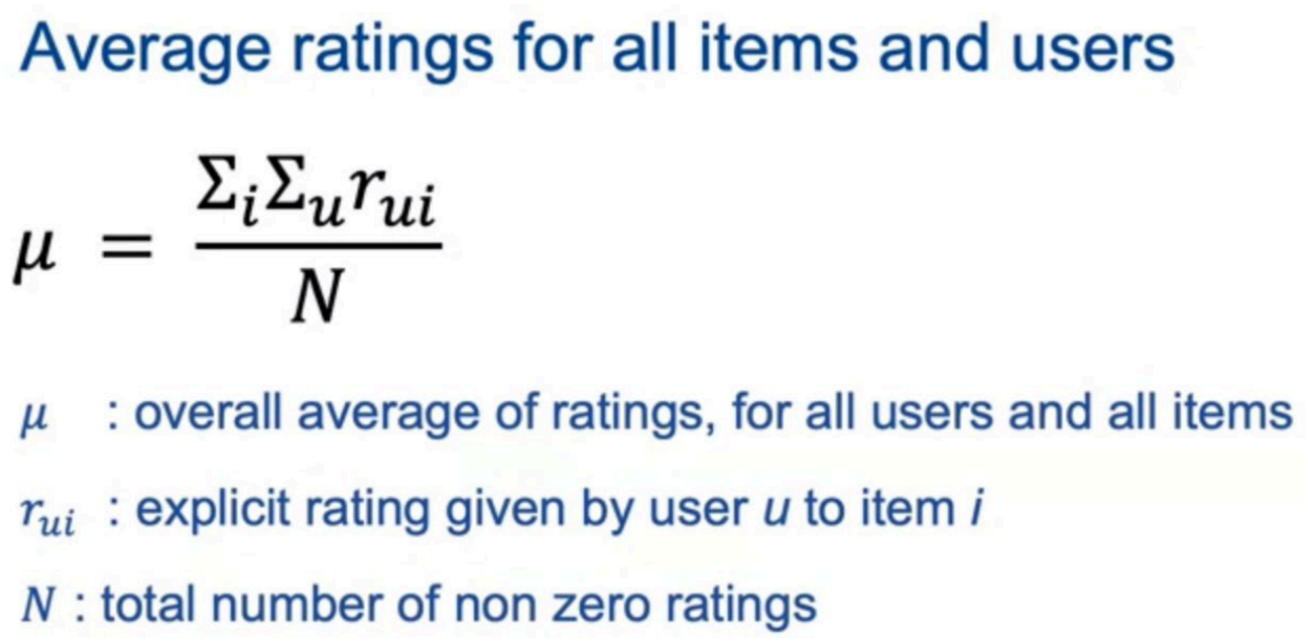
Preprocessing: Account for Bias

People will give a different score based on a very personal opinion.

Item Bias: Users think very differently about movies now than they did 20 years ago. (e.g. Matrix, Inception)

User Bias: Users can rate a movie differently based on nostalgia (e.g. Harry Potter).

Step 1: Average Ratings



Average ratings for all items and users

$$\mu = \frac{\Sigma_i \Sigma_u r_{ui}}{N}$$

μ : overall average of ratings, for all users and all items

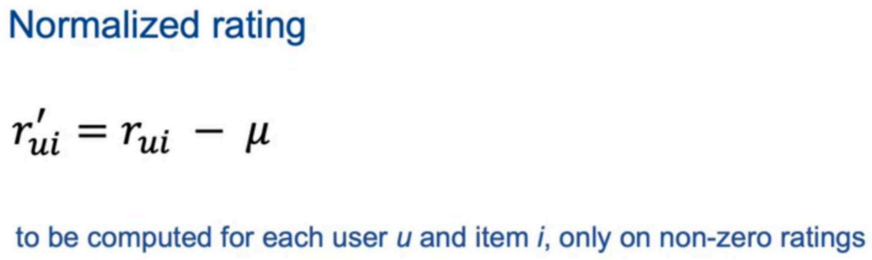
r_{ui} : explicit rating given by user *u* to item *i*

N : total number of non zero ratings

Step 2 and 3: Item Bias

1. Normalize the rating by subtracting the average rating from it.
2. Divide the sum of these normalized ratings by the amount of times they were rated.

Step 2

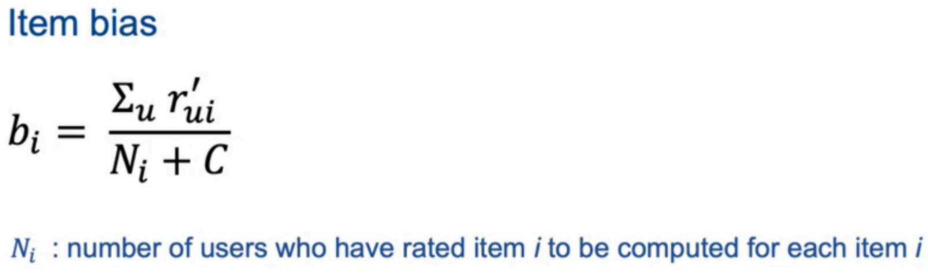


Normalized rating

$$r'_{ui} = r_{ui} - \mu$$

to be computed for each user *u* and item *i*, only on non-zero ratings

Step 3



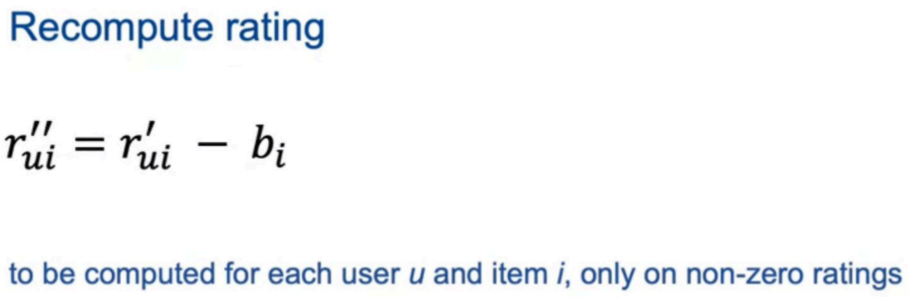
Item bias

$$b_i = \frac{\Sigma_u r'_{ui}}{N_i + C}$$

N_i : number of users who have rated item *i* to be computed for each item *i*

Step 4 and 5: User Bias

Step 4:

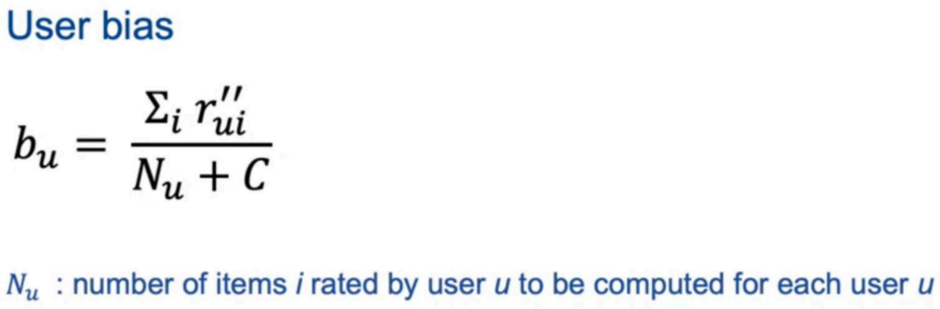


Recompute rating

$$r''_{ui} = r'_{ui} - b_i$$

to be computed for each user *u* and item *i*, only on non-zero ratings

Step 5:



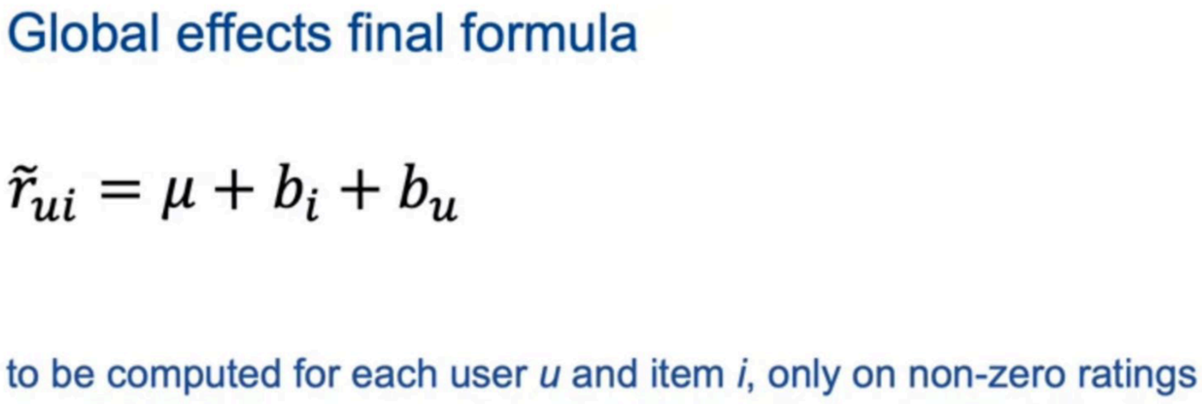
User bias

$$b_u = \frac{\sum_i r''_{ui}}{N_u + C}$$

N_u : number of items *i* rated by user *u* to be computed for each user *u*

Step 6: Final Formula

All of this is done to take into account the bias.



Global effects final formula

$$\tilde{r}_{ui} = \mu + b_i + b_u$$

to be computed for each user *u* and item *i*, only on non-zero ratings

Evaluation

On-line Evaluation

Direct user feedback: Ask the user what they think about the recommendation.

Work with 2 groups, one with the recommendation system, one without it, then ask for their opinion

Off-line Evaluation

Defining task:

Dataset: You have a lot of unknown ratings in the matrix. You also have relevant data (movies that the person is positive about) and non-relevant data (movies that the person is negative about).

Dataset Partitioning

What you do is you is you train a model on the dataset to find similar movies, then you use the relevant data to pick out the movies the person likes, and use the model to recommend similar movies.

You try to split up the dataset in 3 different parts. The X part, which

Content-Based Filtering

The idea

Compare items based on their attributes.

Consider Paolo, he has seen Star Wars, which is similar to Indiana Jones, so you'll recommend that to him.

Item Content Matrix

You have a matrix of y =items and x = attributes

It's a matrix of all 1 and 0. 1-item has that attribute.

You use the dot product between 2 items to see how similar the items are to each-other.

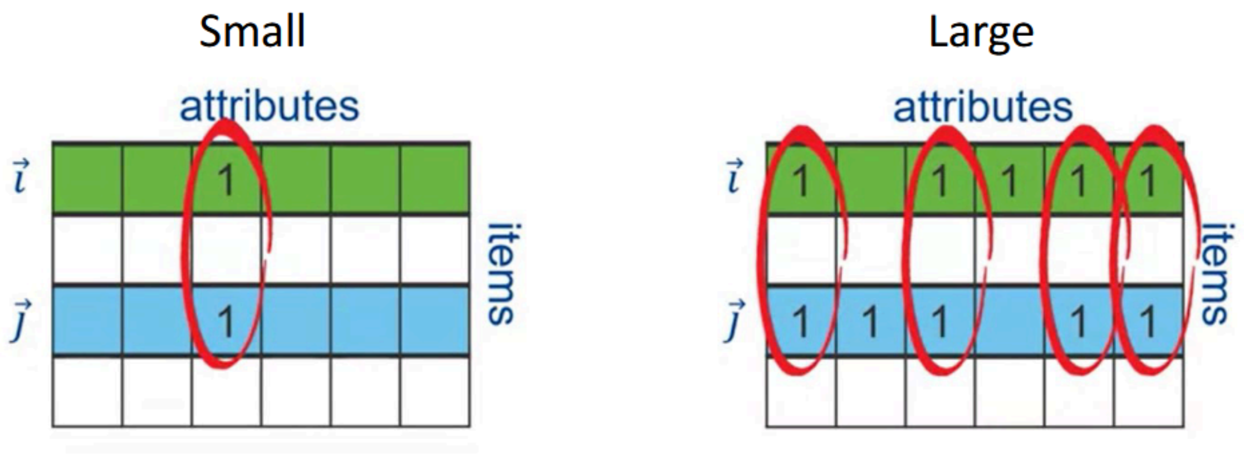
Cosine Similarity: Normalization of the dot product.

Support

The number of non-zero elements in the vector (item).

If the support is small, two items could be "similar", but in reality, there just hisn't enough data.

Support: # of nonzero elements in the vector



$$s_{ij} = \frac{1}{\sqrt{1\cdot 1}} = 1 \qquad s_{ij} = \frac{4}{\sqrt{5\cdot 5}} = 0.8$$

We want the dot product of larger support values be larger than that of small support values.

✓ **Solution: Shrink term**

Shrinking

- Reduce similarity to take into account only most similar with large support
- C: shrink term

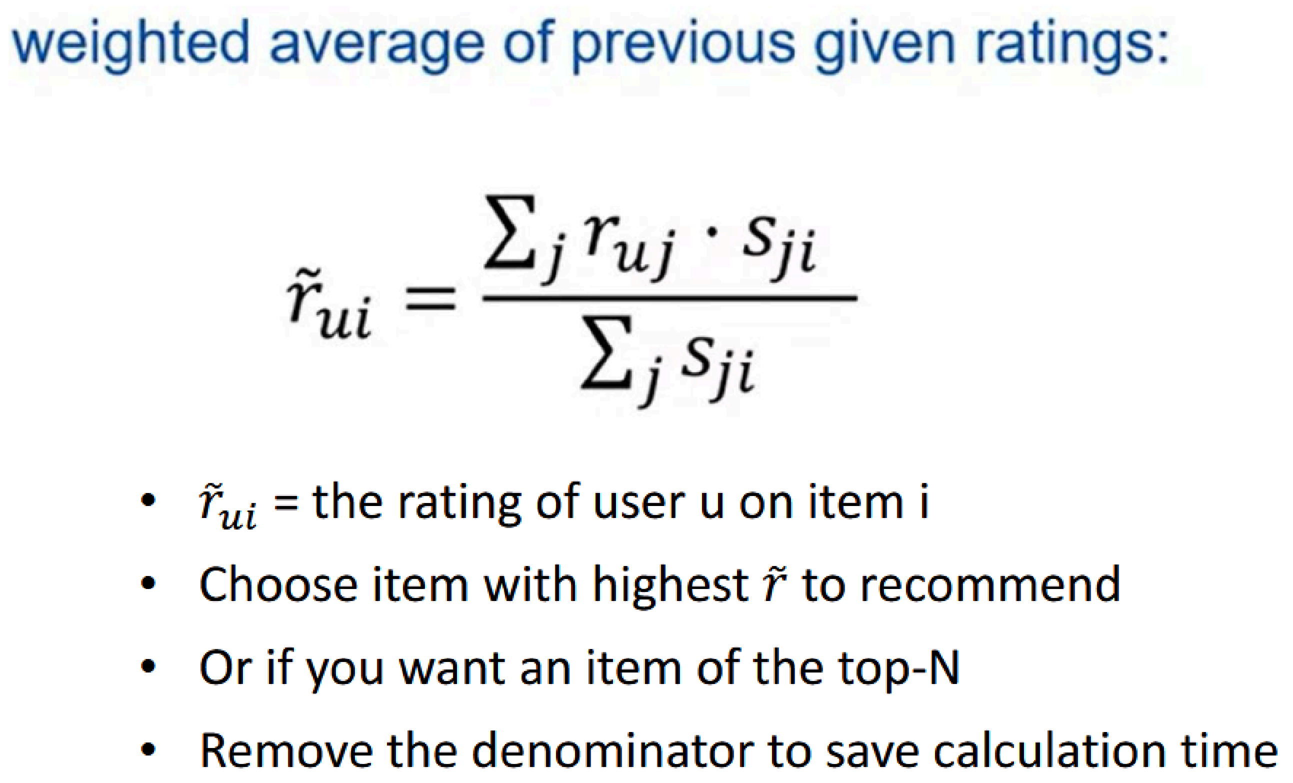
$$s_{ij} = \frac{\vec{i}\cdot\vec{j}}{|\vec{i}|_2\cdot|\vec{j}|_2 + \textcircled{C}}$$

Similarity Matrix

Contains the similarity values (shinking values) of pairs of items.
X - item 1
Y - item 2

Estimating a rating

Will I like Inception? -> the sum of all my ratings times the similarity between them and Inception, divided by the sum of similarities.



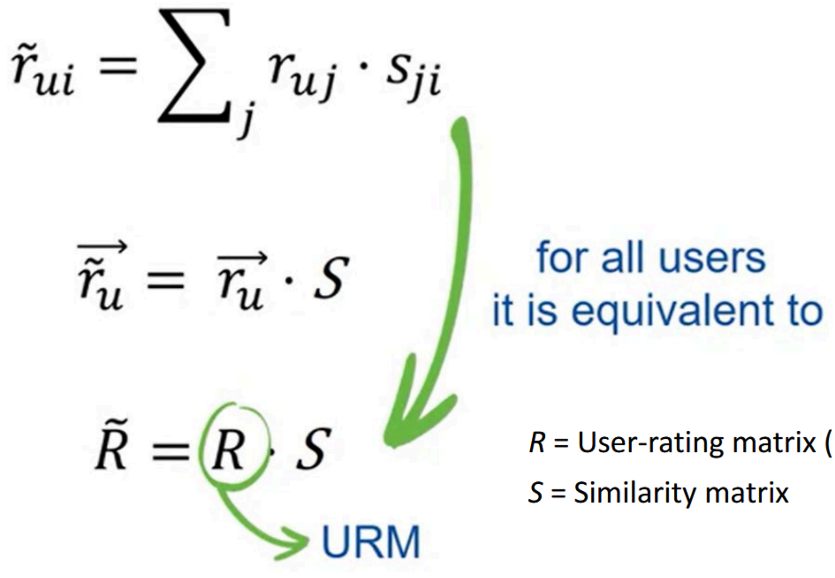
weighted average of previous given ratings:

$$\tilde{r}_{ui} = \frac{\sum_j r_{uj} \cdot s_{ji}}{\sum_j s_{ji}}$$

- $\tilde{r}_{ui}$ = the rating of user u on item i
- Choose item with highest $\tilde{r}$ to recommend
- Or if you want an item of the top-N
- Remove the denominator to save calculation time

Matrix Notation

We can say that all my ratings on movies will be all my ratings, times the similarity matrix.



$$\tilde{r}_{ui} = \sum_j r_{uj} \cdot s_{ji}$$

$$\vec{\tilde{r}}_u = \vec{r}_u \cdot S$$

$$\tilde{R} = R \cdot S$$

R = User-rating matrix (rating per user-item combination)

S = Similarity matrix

Improvement: K-Nearest Neighbours

The similarity matrix is huge, and it contains a lot of very similar, small values. They're not that nice, becuase they take up a lot of computation time, and the result is often 0 (or close).

KNN states that instead of using all the values, you just use the K largest values.

For every row, you take the K largest values, and all the rest become 0.

Problem: Finding optimal K

Just test

Improvement: IF-IDF

ICM improvements: Non-Binary attributes

E.g. Back to the future is a fusion of genres, western and science fiction, so instead of the attributes being (1, 1), let them be (1.2, 0.2) for science fiction and western respectively.

ICM improvements: Attribute Weights

Give a large weight to the title, description and actors.

ICM Improvements: TF-IDF

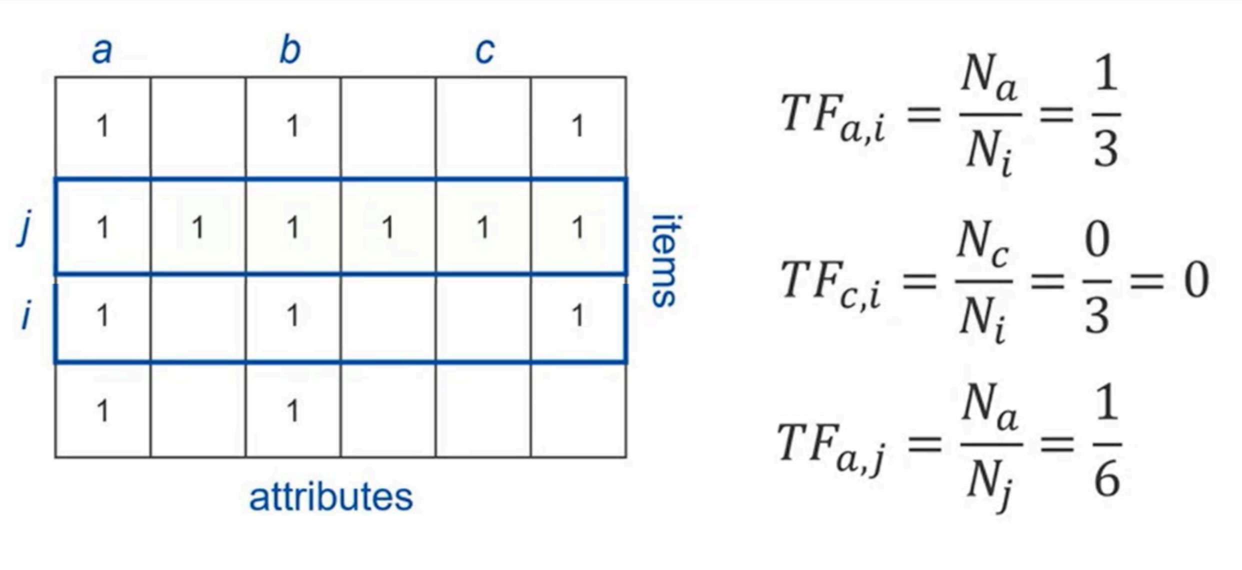
TF - Term Frequency
IDF - Inverse Document Frequency

Term Frequency

Take all the appearances of attribute A in and item i, and divide it by the number of attributes in the item i.

A term frequency is higher if the attribute appears more in an item

Example



	a		b		c	
	1		1			1
j	1	1	1	1	1	1
i	1		1			1
	1		1			

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

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

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

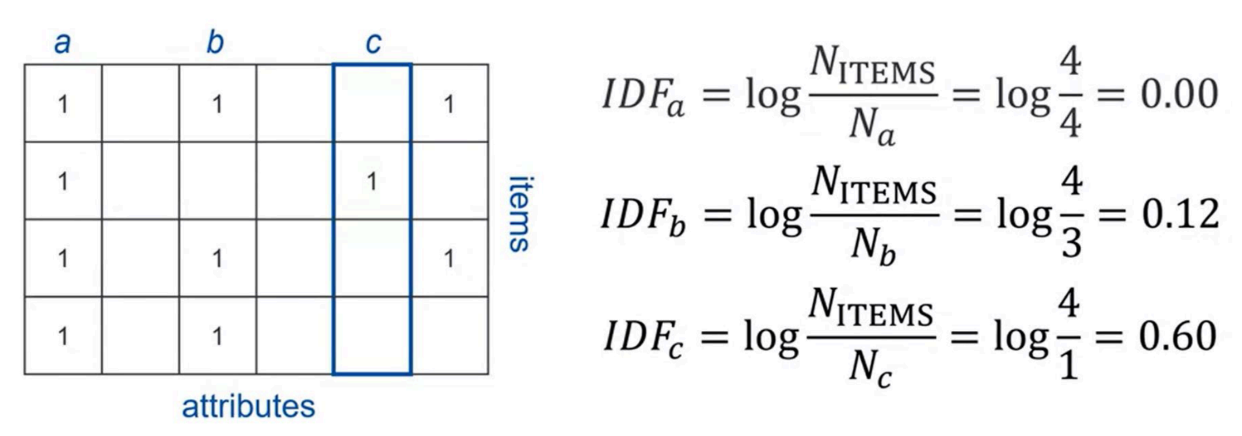
Between a and j, the value is 1/6

Inverse Document Grequency

totla number of items divided by the number of items with attribute a.

It's higher if an attribute appears more among all the items.

Example



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

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

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

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

For attribute a, the number of items are 4.
For attribute b, the number of items is 3.

Put it together

It will have higher value for terms that are frequent in one item, but less frequent among all items.

If you have 3 horror movies, and 10 000 movies, you know that if an attribute is 1 for horror, and 1 for horror in another movie, you'll know that that attribute is very important, because it's super uncommon.

How do you apply Content based filtering?

You apply TF-IDF on the Item-Content matrix.

You construct the similarity matrix, by applying cosine similarity.

Instead of using the whole similarity matrix, apply K-Nearest Neighbors.

The good, the bad and the ugly

✓ **Pros**

- You don't need much data to recommend.
- It is very explainable.
- Easy to recommend niche items.

✗ **Cons**

- You need to calculate the attributes.
- **You only get recommendations for something you've already seen**
  - It tends to get boring after a while, because you'll never see anything truly new.

Collaborative Filtering

Overview

Ratings that a user hasn't given to an item can be inferred based on the correlation of rating among other users:

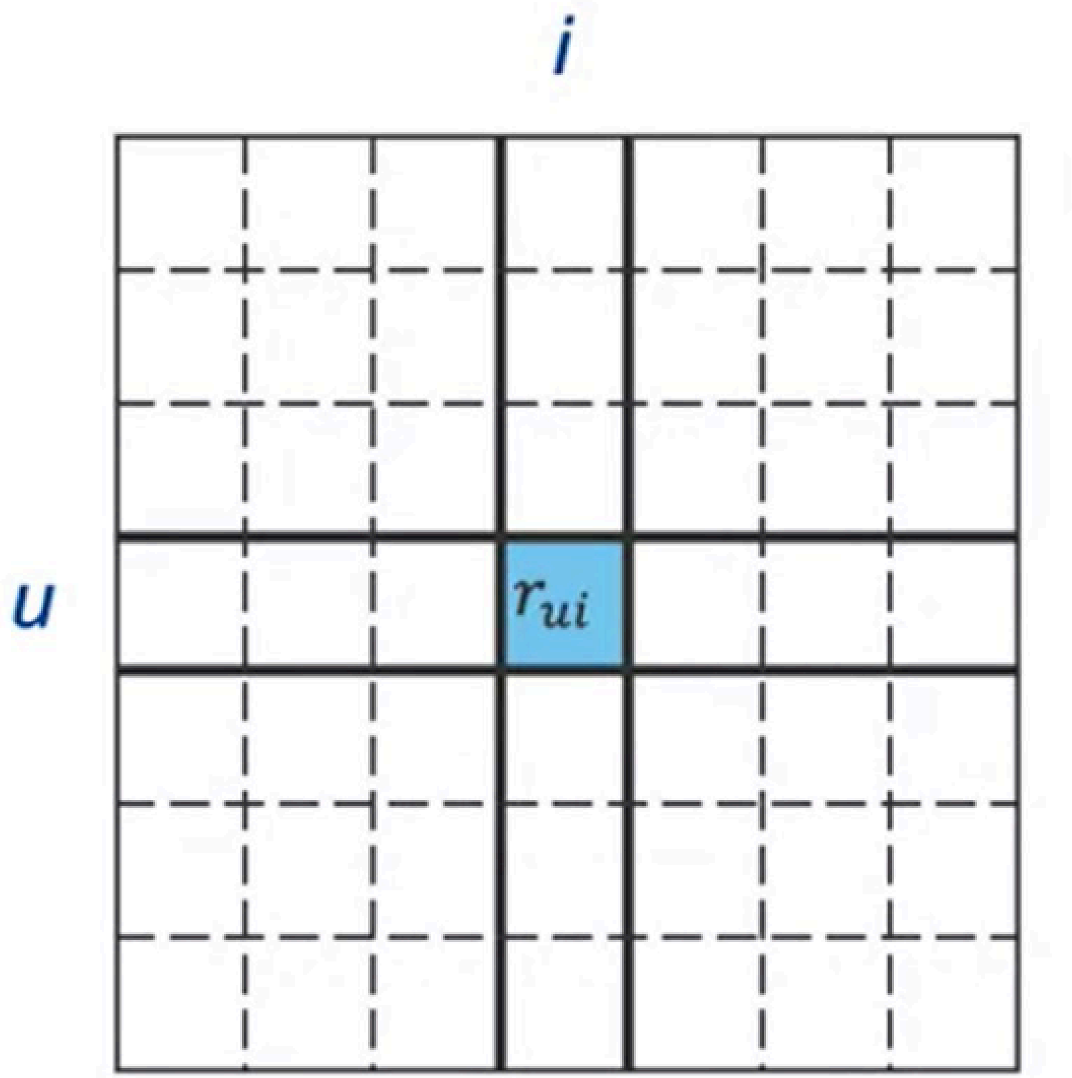
- There is a user that likes the same kind of movies that I watch.
- So, to find out what I might like, I check what he likes.

User Rating Matrix

Matrix containing past interactions between users and items.

Either the score is Implicit

- Either User-Based (slide 91)
- Or Item-Based (slide 95)
  , or Explicit
- Either User-Based (slide 93)
- Or Item-Based (slide 96)



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

Remember the similarity matrix

Instead of the item content matrix, we will use the User Rating Matrix, and we're gonna do exactly the same thing on it.

If two users give similar ratings to several items, we assume that they share the same opinion.

Similarity Matrix

The rating that I give on Inception will be the sum of ratings you give to inception, times the similarity between me and you, divided by the sum of similarity between me and you.

You is plural.

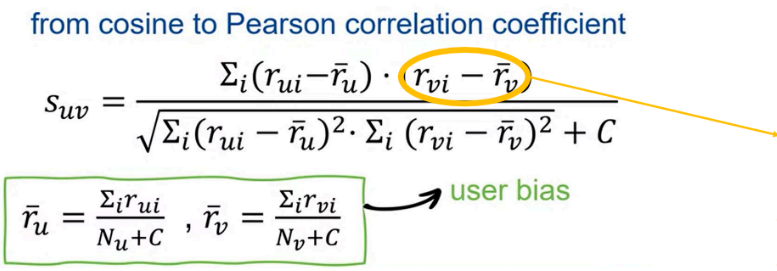
User-Based Filtering: General Case

We need to correct for user bias, so use Pierson Correlation

Imagine the case where we rate between 1 and 5 and 0 means no rating

Different users will give different average ratings so correct for user bias

➢ Calculate similarity in a slightly different way



from cosine to Pearson correlation coefficient

$$s_{uv} = \frac{\Sigma_i (r_{ui}-\bar{r}_u)\cdot(r_{vi}-\bar{r}_v)}{\sqrt{\Sigma_i (r_{ui}-\bar{r}_u)^2\cdot \Sigma_i\,(r_{vi}-\bar{r}_v)^2} + C}$$

$$\boxed{\bar{r}_u = \frac{\Sigma_i r_{ui}}{N_u+C}\ ,\ \bar{r}_v = \frac{\Sigma_i r_{vi}}{N_v+C}}$$

user bias

How much user v likes item I with respect to his/her average opinion

+ Likes item more than average

- Likes item less than average

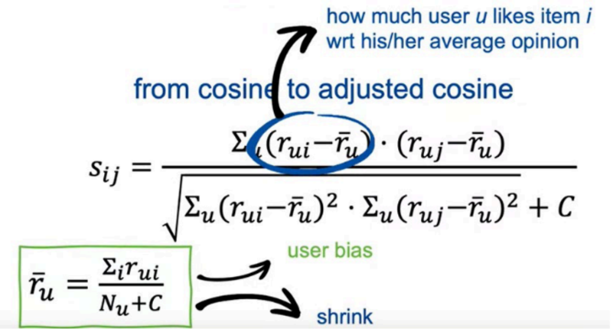
Item-Based filtering when ratings are 0 or 1

If you're doing Item-Based filtering, you can compare also the items, not just the users.

It's just another method to construct the similarity matrix.

Item-Based filtering: General Case

Step 1:



how much user *u* likes item *i* wrt his/her average opinion

from cosine to adjusted cosine

$$s_{ij} = \frac{\Sigma_u (r_{ui}-\bar{r}_u)\cdot(r_{uj}-\bar{r}_u)}{\sqrt{\Sigma_u (r_{ui}-\bar{r}_u)^2\cdot \Sigma_u (r_{uj}-\bar{r}_u)^2} + C}$$

$$\boxed{\bar{r}_u = \frac{\Sigma_i r_{ui}}{N_u+C}}$$

user bias

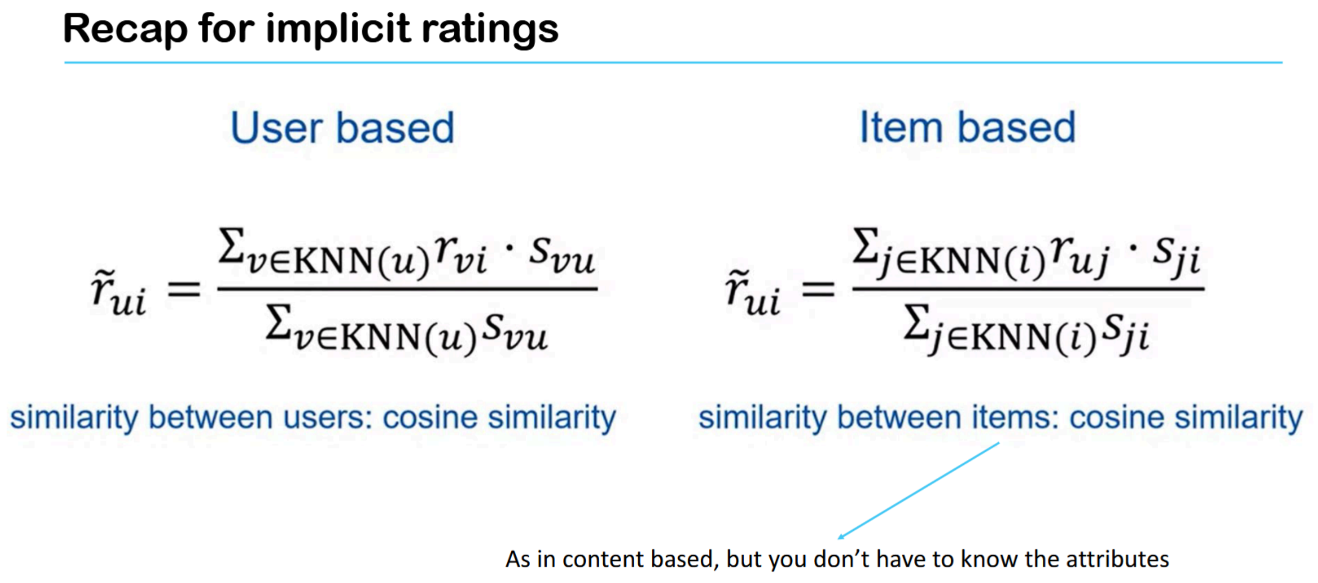
shrink

'Adjusted cosine'

Step 2:

$$\tilde{r}_{ui} = \frac{\sum_{j\in KNN(i)} (r_{uj} - \overline{r_u}).s_{ji}}{\sum_{j\in KNN(i)} s_{ji}} + \overline{r_u}$$

Recap



Recap for implicit ratings

User based

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

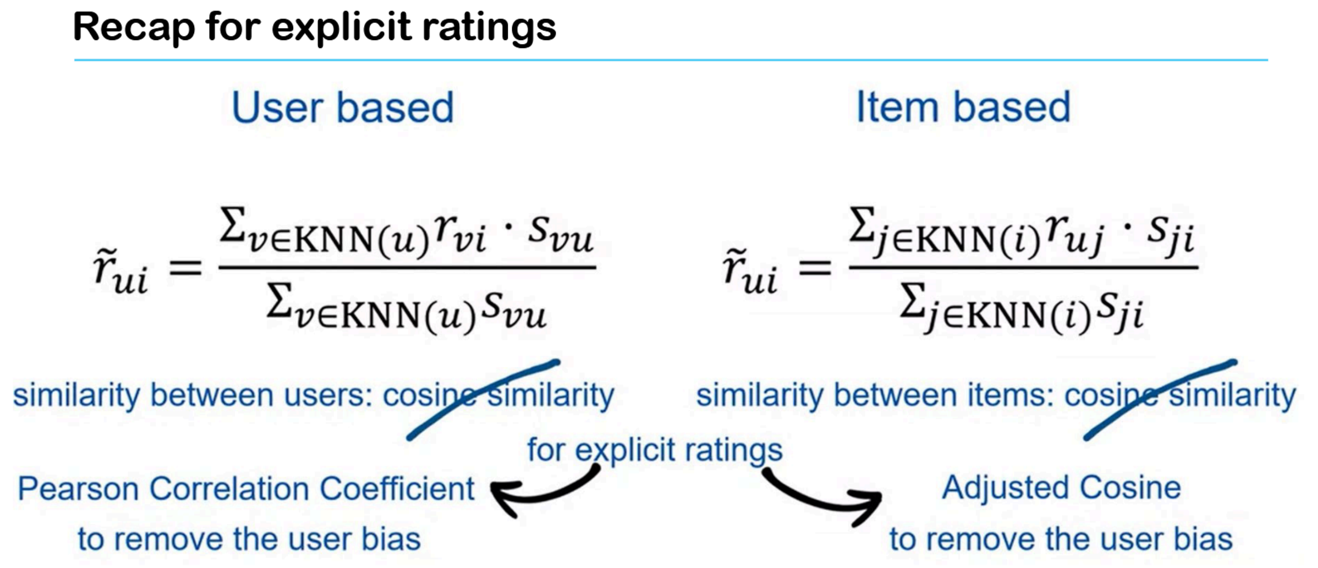
similarity between users: cosine similarity

Item based

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

similarity between items: cosine similarity

As in content based, but you don’t have to know the attributes



Recap for explicit ratings

User based

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

similarity between users: ~~cosine similarity~~

Pearson Correlation Coefficient to remove the user bias

Item based

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

similarity between items: ~~cosine similarity~~

Adjusted Cosine to remove the user bias

for explicit ratings

Demo contains explanations of these filtering methods.

Model Based vs Memory Based systems

> *How quickly do I get recommendations after signing up?*

Model Based

> AKA Item-Based

When a new user is added, the siimilarity matrix does noto change the models significantly.

In other words, if I introduce a new user, nothing in the similarity matrix will change. If I get a few movies that the user has watched, then I can quickly give good recommendations.

Memory Based

When a new user is added, they have to watch a lot more movies before I can give good recommendations.

Closing Remarks

The Netflix Challenge

Open competition for the best recommendation system.
Publicly available dataset made by Netflix.

Ran for 3 years, until a team (actually a combination of teams) finally reached a threshold.

Other Algorithms

We only scratched the surface of the algorithms used.

There are also DL-Based models. Less explainable, more complex, probably better.

Hybrid - show a combination of the following:

1. Most popular products (TOP 10)
2. Historical products
3. Frequently Bought Together (If you buy a smartphone, you'll buy a case for it)
4. This session + matrix factorization
5. Deep Learning

It's an uphill battle - Amazon has worked on this for 10 years, and they're still not finished.

Extra info

- this class
- extra info about DL based recommendation systems