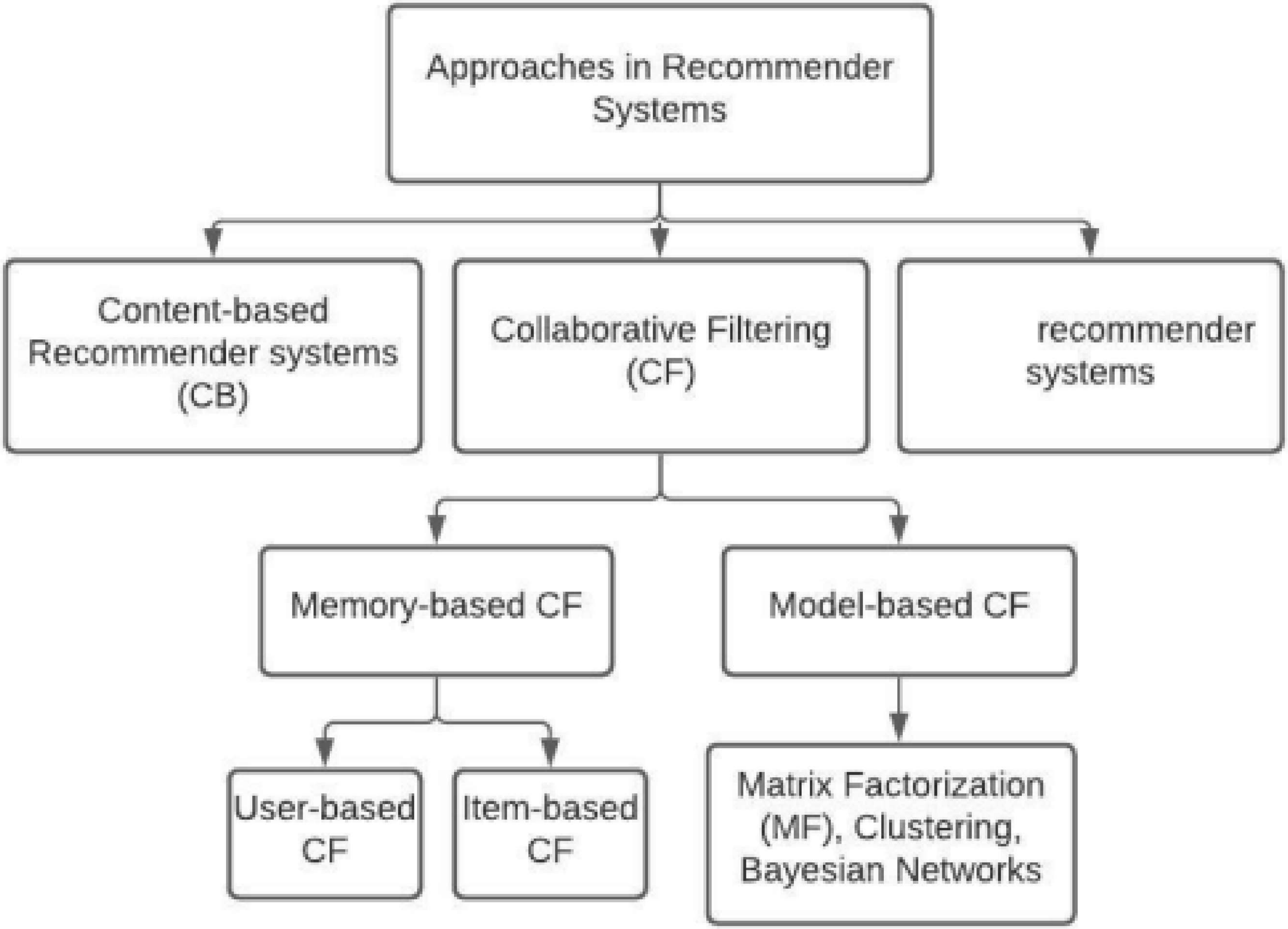
Class 6 - 8

Approaches in Recommender Systems



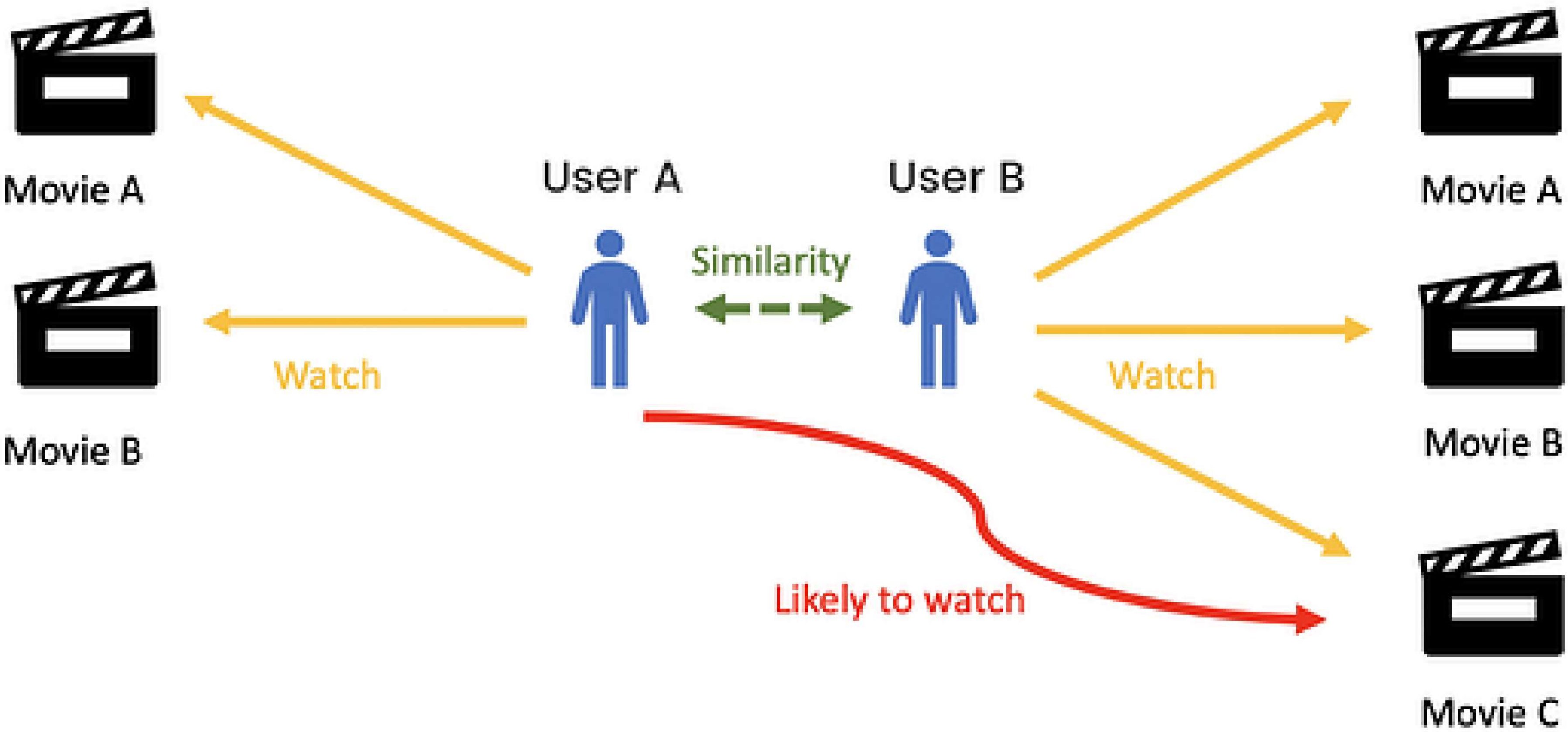
Collaborative Filtering - From Word-of-Mouth to Digital Recommendations

- Roots of Collaborative Filtering
  - *Human Nature*
- Evolution of Collaborative Filtering
  - *From Small Circles to the Internet*
- How It Works
  - *Collect Data .*
  - *Analyze Patterns*
  - *Make Recommendations*
- Why It's Powerful
  - *Collective Wisdom*
  - *Personalization*
  - *Real-Time Adaptability*
- Real-World Applications
  - *E-commerce*
  - *Entertainment*
  - *Social Media*
  - *Travel*

What is Collaborative Filtering?

- Predicts user preferences by analyzing patterns in user-item interactions.
- Example: If User A and User B both liked movies X and Y, recommend movie Z (liked by User B) to User A.

Example of CF



Key Components

- User-Item Matrix
- Similarity Measures
- Prediction
- Recommendation

Notation and Definitions

- U: Set of users in the system.
- I: Set of items in the system.
- r_{ui}: Rating given by user u for item i.
- U_i: Subset of users who rated item i.
- I_u: Subset of items rated by user u.
- I_{uv}: Items rated by both users u and v.
- U_{ij}: Users who rated both items i and j.

Memory-Based Collaborative Filtering

Also called as Neighborhood based CF

- Relies on the entire user-item matrix to make predictions.
- Two main approaches:
  - **User-based Collaborative Filtering**:
    - Find similar users based on their ratings.
    - Predict ratings for a user based on the ratings of similar users.
    - Example: If User A and User B have similar movie preferences, recommend movies liked by User B to User A.
  - **Item-based Collaborative Filtering**:
    - Find similar items based on user ratings.
    - Recommend items similar to those a user has liked.
    - Example: If Movie X and Movie Y have been liked by many of the same users, and User A liked Movie X, then recommend Movie Y to User A.

Memory-Based Collaborative Filtering

Also called as Neighborhood based CF

- Relies on the entire user-item matrix to make predictions.
- Two main approaches:
  - **User-based Collaborative Filtering**:
    - Find similar users based on their ratings.
    - Predict ratings for a user based on the ratings of similar users.
    - Example: If User A and User B have similar movie preferences, recommend movies liked by User B to User A.
  - **Item-based Collaborative Filtering**:
    - Find similar items based on user ratings.
    - Recommend items similar to those a user has liked.
    - Example: If Movie X and Movie Y have been liked by many of the same users, and User A liked Movie X, then recommend Movie Y to User A.

People who have similar tastes will like similar things.

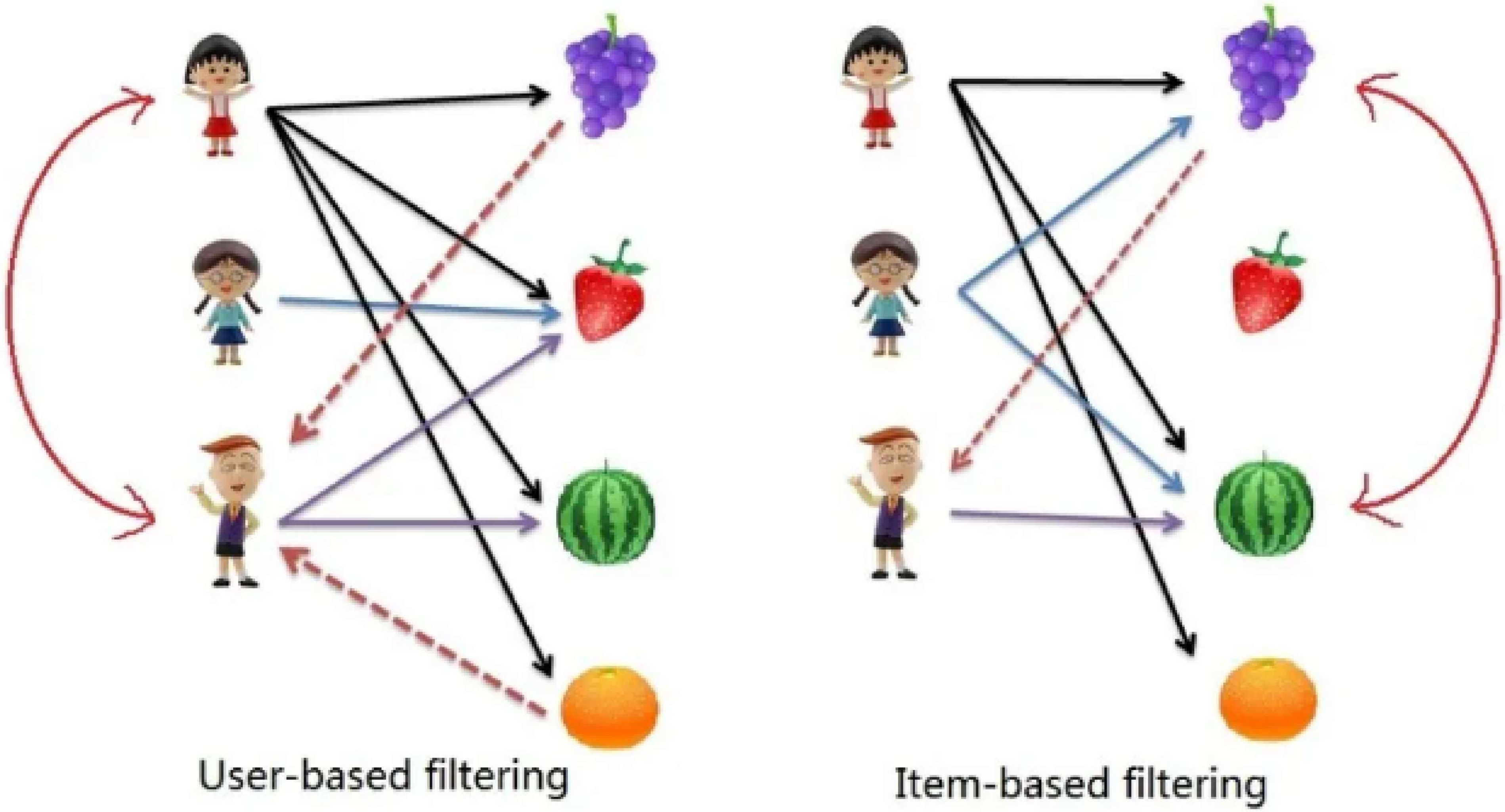
Memory-Based Collaborative Filtering

Also called as Neighborhood based CF

- Relies on the entire user-item matrix to make predictions.
- Two main approaches:
  - **User-based Collaborative Filtering**:
    - Find similar users based on their ratings.
    - Predict ratings for a user based on the ratings of similar users.
    - Example: If User A and User B have similar movie preferences, recommend movies liked by User B to User A.
  - **Item-based Collaborative Filtering**:
    - Find similar items based on user ratings.
    - Recommend items similar to those a user has liked.
    - Example: If Movie X and Movie Y have been liked by many of the same users, and User A liked Movie X, then recommend Movie Y to User A.

People who have similar tastes will like similar things.

If two items are similar, a user who likes one will likely like the other.



https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f

Example Workflow

- Step 1: Collect user-item rating data.
- Step 2: Compute similarity between users/items.
- Step 3: Select nearest neighbors.
- Step 4: Predict unknown ratings.
- Step 5: Recommend top-N items.

User-Item Matrix

The user-item rating matrix is a central data

User	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
U1	5	3	4	4	-	1
U2	3	1	2	3	3	2
U3	4	3	4	3	5	-
U4	3	3	1	5	4	-
U5	1	5	5	2	1	4

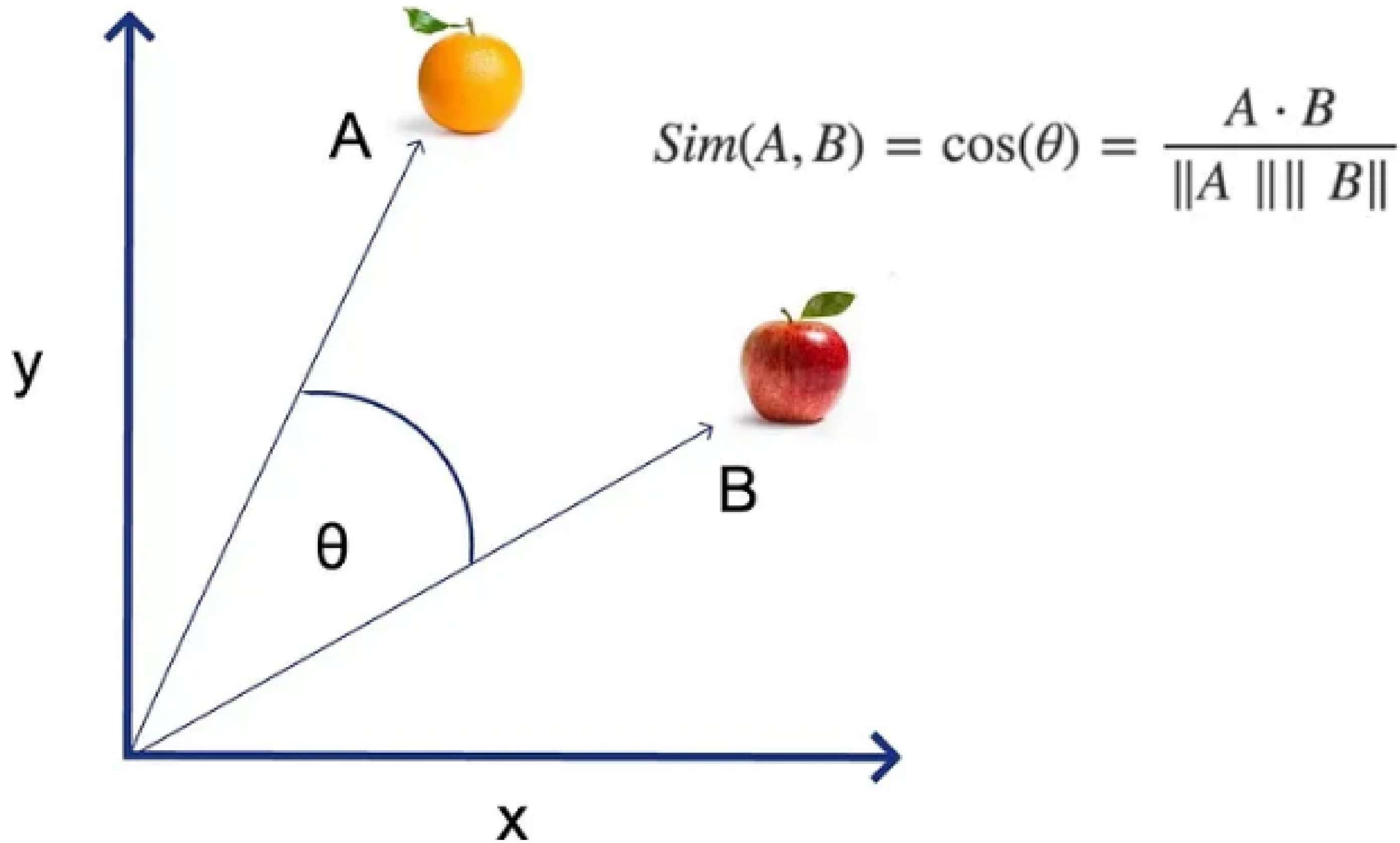
- Rows: Users (U1 to U5).
- Columns: Items (Item 1 to Item 6).
- Ratings: On a scale of 1 to 5. '-' indicates missing ratings.

User-User Similarity

We calculate similarity between User 1 (U1) and User 2 (U2) using:

- Cosine Similarity
- Pearson Correlation
- Jaccard Similarity

Cosine Similarity



Cosine Similarity in Collaborative Filtering

- Measures the angle between vectors. **Formula:**

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

- I_{uv}: Set of co-rated items.
- r_{ui}, r_{vi}: Ratings of users u and v for item i.

Example: Cosine Similarity

User Ratings:

User	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
U1	5	3	4	4	-	1
U2	3	1	2	3	3	2

- Rows: Users (U1 and U2).
- Columns: Items (Item 1 to Item 6).
- Ratings: On a scale of 1 to 5. '-' indicates missing ratings.

Co-rated Items: $\{1, 2, 3, 4, 6\}$

Computing Cosine Similarity

Ratings for Co-rated Items:

- $U1 = [5, 3, 4, 4, 1]$
- $U2 = [3, 1, 2, 3, 2]$

$$\begin{aligned}
\text{sim}(U_1, U_2) &= \frac{5 \cdot 3 + 3 \cdot 1 + 4 \cdot 2 + 4 \cdot 3 + 1 \cdot 2}{\sqrt{5^2 + 3^2 + 4^2 + 4^2 + 1^2} \cdot \sqrt{3^2 + 1^2 + 2^2 + 3^2 + 2^2}} \\
&= \frac{15 + 3 + 8 + 12 + 2}{\sqrt{25 + 9 + 16 + 16 + 1} \cdot \sqrt{9 + 1 + 4 + 9 + 4}} \\
&= \frac{40}{\sqrt{67} \cdot \sqrt{27}} \\
&\approx 0.939
\end{aligned}$$

- Does not account for rating scale differences (one user may rate higher on average than another).

Pearson Correlation

- PCC improves over cosine similarity by normalizing ratings.
- Measures the linear correlation between two users' ratings.
- Centers the data by subtracting the mean rating.
- Handles user rating scale differences better.

$$\text{sim}(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}$$

$$\text{sim}(U_1, U_2) =$$

(5-3.4)(3-2.2) + (3-3.4)(1-2.2) + (4-3.4)(2-2.2) + (4-3.4)(3-2.2) + (1-3.4)(2-2.2) $\frac{}{\sqrt{(5-3.4)^2+(3-3.4)^2+(4-3.4)^2+(4-3.4)^2+(1-3.4)^2} \cdot \sqrt{(3-2.2)^2+(1-2.2)^2+(2-2.2)^2+(3-2.2)^2+(2-2.2)^2}}$
≈ 0.85

Jaccard Similarity

- Jaccard similarity is ideal when the dataset consists of implicit feedback rather than explicit ratings.
- Implicit feedback typically includes binary data indicating whether a user interacted with an item (e.g., clicked, purchased, or viewed).

$$\text{Jacc}(U1, U2) = \frac{|I_{U1} \cap I_{U2}|}{|I_{U1} \cup I_{U2}|}$$

Sets:

$$I_{U1} = \{Item1, Item2, Item3, Item4, Item6\}$$

$$I_{U2} = \{Item1, Item2, Item3, Item4, Item5, Item6\}$$

Calculation:

$$|I_{U1} \cap I_{U2}| = 5, \quad |I_{U1} \cup I_{U2}| = 6$$

$$\text{Jacc}(U1, U2) = \frac{5}{6} \approx 0.833$$

Selecting Neighborhood

- Select Users with a Similarity Score Above a Certain Threshold
- Select at Random
- Select the Top-k Users Ranked by Similarity Score
- Select Users Within the Same Cluster

Another example for User-User Similarity Computation

User-Id	1	2	3	4	5	6	Mean	Cosine(i, 3)	Pearson(i, 3)
1	7	6	7	4	5	4	5.5	0.956	0.894
2	6	7	?	4	3	4	4.8	0.981	0.939
3	?	3	3	1	1	?	2	1.0	1.0
4	1	2	2	3	3	4	2.5	0.789	-1.0
5	1	?	1	2	3	3	2	0.645	-0.817

Table: User-user similarity computation between user 3 and other users.

Bias in Recommender Systems

Bias refers to unfair preference or systematic errors in recommendation due to skewed data or algorithms.

- Discuss different types of bias
    - Popularity Bias (highly-rated items get more recommendations).
    - Selection Bias (only active users influence recommendations).
    - Exposure Bias (some items never get recommended).

Normalization in Recommender Systems

- Normalization helps adjust ratings for
  - Different user preferences.
  - Different rating scales.
- Different types of normalization
  - Mean-centering (used in PCC) - Subtracts the user's average rating from each rating to remove individual biases.
  - Z-score normalization - Standardizes ratings so that each user's ratings have a mean of 0 and a standard deviation of 1.

Z-Score Normalization

- In statistics, z-score is the signed number of deviations from mean indicating that a datum is above the mean if positive and below the mean otherwise.
- It is suitable for situations where minimum and maximum of item ratings are unknown.
- Used when rating scales vary across users.
- In user-based collaborative filtering (CF), the original rating r_{uj} is normalized to z_{uj}.

Z-Score Normalization Formula

$$z_{uj} = \frac{r_{uj} - \mu_u}{\sigma_u}$$

$$\sigma_u = \sqrt{\frac{\sum_j (r_{uj} - \mu_u)^2}{|I_u| - 1}}$$

- μ_u is the mean rating of user u.
- σ_u is the standard deviation of user u's ratings.
- Adjusts for user rating biases.

Example: Z-Score Normalization

Given User Ratings:

User	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
1	7	6	7	4	5	4
2	6	7	?	4	3	4
3	?	3	3	1	1	?
4	1	2	2	3	3	4
5	1	?	1	2	3	3

Step 1: Compute Mean Rating and Standard Deviation for User 1:

$$\mu_1 = \frac{7+6+7+4+5+4}{6} = 5.5$$

$$\sigma_1 = \sqrt{\frac{(7-5.5)^2+(6-5.5)^2+(7-5.5)^2+(4-5.5)^2+(5-5.5)^2+(4-5.5)^2}{6-1}} \approx 1.38$$

Step 3: Compute Z-scores

$$z_{1j} = \frac{r_{1j} - \mu_1}{\sigma_1}$$

User	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
1	$\frac{7-5.5}{\sigma_1}$	$\frac{6-5.5}{\sigma_1}$	$\frac{7-5.5}{\sigma_1}$	$\frac{4-5.5}{\sigma_1}$	$\frac{5-5.5}{\sigma_1}$	$\frac{4-5.5}{\sigma_1}$

$$z_1(7) = \frac{7 - 5.5}{1.38} \approx 1.09$$

$$z_1(6) = \frac{6 - 5.5}{1.38} \approx 0.36$$

$$z_1(4) = \frac{4 - 5.5}{1.38} \approx -1.09$$

$$z_1(5) = \frac{5 - 5.5}{1.38} \approx -0.36$$

Z-score values

User	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Mean (μ)	Std Dev (σ)
User 1	1.09	0.36	1.09	-1.09	-0.36	-1.09	5.5	1.38
User 2	1.41	0.00	0.71	-0.71	-1.41	0.00	4.0	1.41
User 3	0.27	0.80	1.34	-1.34	-0.80	-0.27	4.5	1.87
User 4								
User 5								

Table: Z-score Normalization for User Ratings

Prediction

- Prediction in recommender systems involves estimating unknown ratings based on similar users/items..
- Helps generate personalized recommendations.
- Different approaches:
  - Standard Weighted Average Prediction.
  - Mean-Centered Neighborhood-Based Prediction.
  - z-score based prediction.
  - Significance Weighting based prediction

Standard Weighted Average Prediction

$$\hat{r}_{uj} = \frac{\sum_{v \in P_u(j)} r_{vj} \cdot \text{Sim}(u, v)}{\sum_{v \in P_u(j)} |\text{Sim}(u, v)|}$$

$P_u(j)$ (peers) is the set of k closest users to target user u who have specified ratings for item j.

- Uses raw ratings without mean adjustment.
- Can be biased due to different user rating styles.

Example Calculation:

$$\hat{r}_{31} = \frac{7 \times 0.894 + 6 \times 0.939}{0.894 + 0.939} \approx 6.49$$

$$\hat{r}_{36} = \frac{4 \times 0.894 + 4 \times 0.939}{0.894 + 0.939} = 4$$

Mean-Centered Neighborhood-Based Prediction

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \text{Sim}(u, v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\text{Sim}(u, v)|}$$

- Adjusts for user bias.
- Uses mean rating adjustments.
- More robust for users with different rating scales.

Example calculation:

$$\hat{r}_{31} = 2 + \frac{1.5 \times 0.894 + 1.2 \times 0.939}{0.894 + 0.939} \approx 3.35$$

$$\hat{r}_{36} = 2 + \frac{-1.5 \times 0.894 - 0.8 \times 0.939}{0.894 + 0.939} \approx 0.86$$

Z-score based prediction

$$\hat{r}_{uj} = \mu_u + \sigma_u \cdot \frac{\sum_{v \in P_u(j)} \mathrm{Sim}(u,v) \cdot z_{vj}}{\sum_{v \in P_u(j)} |\mathrm{Sim}(u,v)|}$$

Example Calculation of Prediction

$$\hat{r}_{31} = 2 + (4.5)\frac{(1.09 \times 0.894) + (1.41 \times 0.939)}{0.894 + 0.939}$$

- Z-score normalization helps mitigate user rating biases.
- It standardizes ratings making similarity calculations more reliable.
- Useful when the rating scale boundaries are unknown.

Significance Weighting for Reliable Similarity

- Similarity scores may be unreliable when users have few common ratings.
- Apply a **discount factor** to similarity scores based on the number of common ratings.
- **Significance Weighting Formula**:

$$\text{DiscountedSim}(u, v) = \text{Sim}(u, v) \cdot \frac{\min\{|I_u \cap I_v|, \beta\}}{\beta}$$

  - $|I_u \cap I_v|$: Number of items rated by both users u and v.
  - β: Threshold for minimum common ratings (e.g., $\beta = 5$).
  - If $|I_u \cap I_v| < \beta$, the similarity score is discounted.

Why Include Significance Weighting?

- *Improves Reliability:* Ensures that similarity scores are based on a sufficient number of common ratings.
- *Reduces Noise:* Minimizes the impact of user pairs with few common ratings, which may introduce noise into the predictions.
- *Enhances Prediction Accuracy:* Leads to more accurate and meaningful recommendations.
- **Example**:
  - Suppose $\beta = 5$ and two users have 3 common ratings.
  - Discount factor $= \frac{3}{5} = 0.6$.
  - If raw similarity score $= 0.8$, discounted score $= 0.8 \cdot 0.6 = 0.48$.
- **Application**:
  - Use discounted similarity in the prediction formula:

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \text{DiscountedSim}(u, v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\text{DiscountedSim}(u, v)|}$$

  - $P_u(j)$: Set of users similar to u who have rated item j.

Similarity Function Variants

Raw Cosine Similarity:

$$\text{RawCosine}(u,v) = \frac{\sum\limits_{k \in I_u \cap I_v} r_{uk} \cdot r_{vk}}{\sqrt{\sum\limits_{k \in I_u \cap I_v} r_{uk}^2} \cdot \sqrt{\sum\limits_{k \in I_u \cap I_v} r_{vk}^2}}$$

Alternative:

$$\text{RawCosine}(u,v) = \frac{\sum\limits_{k \in I_u \cap I_v} r_{uk} \cdot r_{vk}}{\sqrt{\sum\limits_{k \in I_u} r_{uk}^2} \cdot \sqrt{\sum\limits_{k \in I_v} r_{vk}^2}}$$

PCC:

$$\text{pcc}(u,v) = \frac{\sum\limits_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum\limits_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}$$

Significance Weighting:

$$\text{DiscountedSim}(u,v) = \text{Sim}(u,v) \cdot \frac{\min(|I_u \cap I_v|, \beta)}{\beta}$$

Prediction Function Variants

Mean-Centered Prediction:

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \text{Sim}(u, v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\text{Sim}(u, v)|}$$

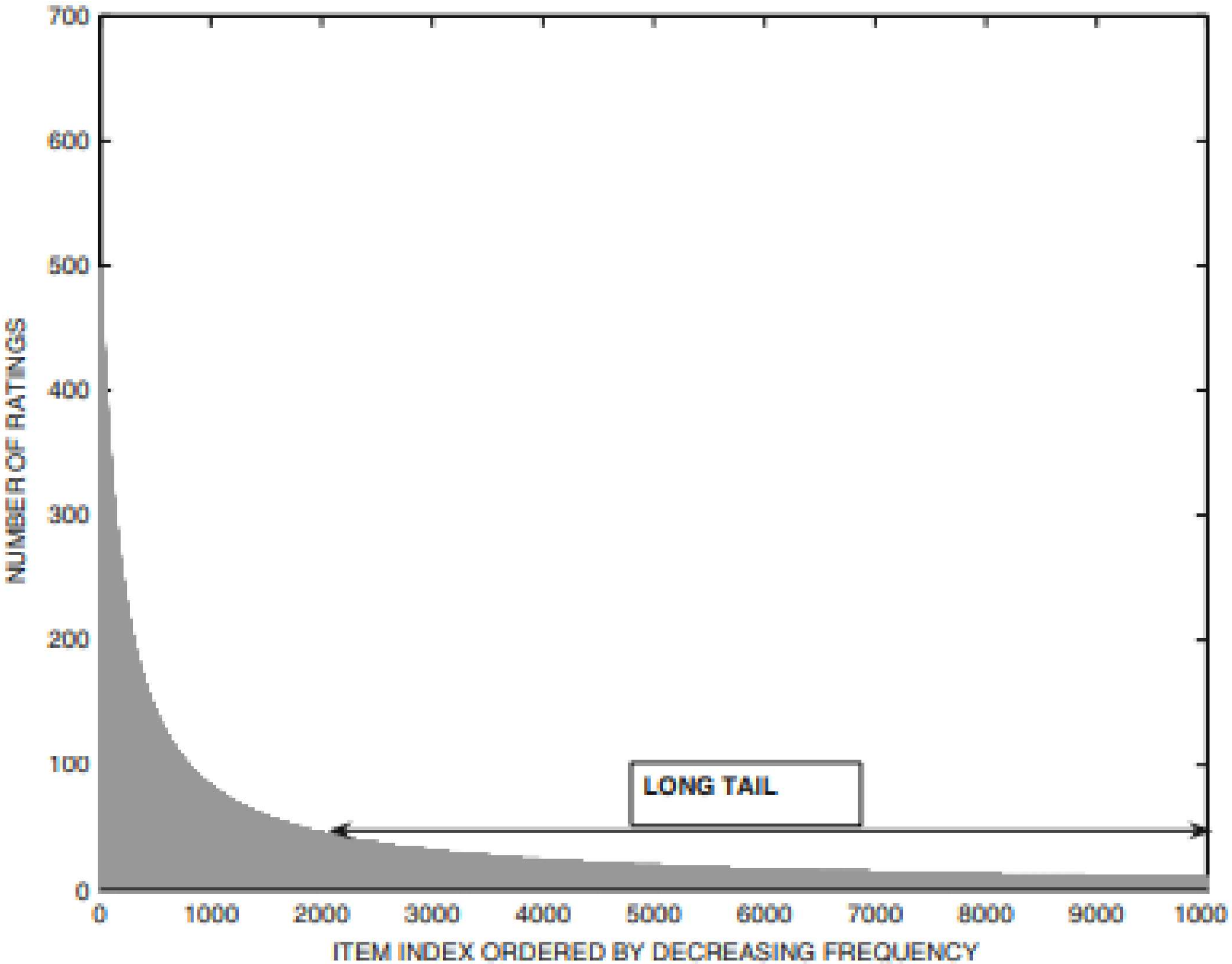
Z-Score based prediction:

$$\hat{r}_{uj} = \mu_u + \sigma_u \cdot \frac{\sum_{v \in P_u(j)} \text{Sim}(u, v) \cdot z_{vj}}{\sum_{v \in P_u(j)} |\text{Sim}(u, v)|}$$

Which is better? Raw rating, z-score or something else

- Normalization improves prediction
- If a function $g(\cdot)$ is applied during rating normalization, then its inverse needs to be applied during prediction.
- Mean-Centering - Simple and effective in reducing user biases, improving similarity calculations.
- Z-Score Normalization - Standardizes ratings for comparability but may lead to predictions outside the valid range.

Long Tail



700
600
500
400
300
200
100
0
NUMBER OF RATINGS
LONG TAIL
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
1000
ITEM INDEX ORDERED BY DECREASING FREQUENCY

The Long Tail in Recommender Systems

Definition: The **long tail** refers to the phenomenon where a small number of popular items receive most interactions, while a large number of niche items receive very few interactions.

Impact of the Long Tail:

- Popular items dominate recommendations.
- Niche items are often overlooked.
- Users may not discover relevant but less-known items.

Long Tail Business Models

What is the Long Tail?

- Selling small amounts of many different items instead of a few bestsellers.
- Works well online because digital stores have unlimited space.

How Does It Work?

- Online shops like Amazon can sell rare books that physical stores don't stock.
- Streaming services can offer many niche movies, not just blockbusters.
- Amazon (Books & Video Streaming)
  - Sells many different books, even those with very few buyers.
  - Offers a mix of popular and niche movies on Prime Video.

Why Is It Useful?

- More choices for customers.
- Businesses make money from many small sales instead of a few big ones.

https://www.youtube.com/watch?v=AnEGkRePp0Q&t=140s

Addressing Long Tail problem

- Popular items may dominate similarity computations.
- **Solution**: Use Inverse User Frequency (IUF) weighting
- IUF weight for item j is given by,

$$w_j = \log\left(\frac{m}{m_j}\right)$$

where:

- m = total number of users.
- m_j = number of users who rated item j.

Inverse User Frequency (IUF) Weighting

Inverse User Frequency (IUF) adjusts the influence of popular items in collaborative filtering:

- Frequently rated items contribute less to user similarity. IUF down-weights popular items, reducing their dominance.
- Less frequently rated items get higher weights, making them more influential in recommendations.
- Encourages diversity in recommendations.

The IUF weight for item j is given by:

$$w_j = \log \frac{m}{m_j}$$

Rare items have a stronger impact in similarity computations, reducing bias toward popular items.

Weighted Pearson Correlation

The similarity between users u and v is given by:

$$Pearson(u, v) = \frac{\sum_{k \in I_u \cap I_v} w_k (r_{uk} - \mu_u)(r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} w_k (r_{uk} - \mu_u)^2} \times \sqrt{\sum_{k \in I_u \cap I_v} w_k (r_{vk} - \mu_v)^2}}$$

where:

- I_u and I_v are the sets of items rated by users u and v.
- $I_u \cap I_v$ represents items rated by both users.
- w_k is the weight of item k (e.g., IUF weight).
- r_{uk} and r_{vk} are ratings of user u and user v for item k.
- μ_u and μ_v are the average ratings of users u and v.

Weighted Pearson Correlation Incorporates IUF

- Traditional Pearson correlation treats all items equally.
- Instead of treating all items equally, less common items have a stronger impact on similarity scores.
- This prevents popularity bias, where frequently rated items dominate recommendations.
- Improves similarity accuracy by reducing the effect of frequently rated items.

Item-Based CF

- **Key Idea**:
  - Peer groups are constructed based on items (not users).
  - Similarities are computed between items (columns in the ratings matrix).
- **Process**:
  - Mean-center the ratings matrix by subtracting the average rating of each item.
  - Compute adjusted cosine similarity between items.

Adjusted Cosine Similarity

- Measure similarity between items.

$$\text{AdjustedCosine}(i,j) = \frac{\sum_{u \in U_i \cap U_j} s_{ui} s_{uj}}{\sqrt{\sum_{u \in U_i \cap U_j} s_{ui}^2} \times \sqrt{\sum_{u \in U_i \cap U_j} s_{uj}^2}}$$

Mean-Centered Ratings:

$$s_{ui} = r_{ui} - \mu_u$$

- U_i: Set of users who rated item i.
- Adjusted cosine is preferred over Pearson correlation for item-based models.

Adjusted Cosine Calculation between Items 1 and 3

User-Id	1	2	3	4	5	6
1	1.5	0.5	1.5	-1.5	-0.5	-1.5
2	1.2	2.2	?	-0.8	-1.8	-0.8
3	?	1	1	-1	-1	?
4	-1.5	-0.5	-0.5	0.5	0.5	1.5
5	-1	?	-1	0	1	1
Cosine(1, j)	1	0.735	0.912	-0.848	-0.813	-0.990
Cosine(6, j)	-0.990	-0.622	-0.912	0.829	0.730	1

Table: Ratings matrix with mean-centering for adjusted cosine similarity computation.

$$\text{AdjustedCosine}(1,3) = \frac{(1.5 \times 1.5) + (-1.5 \times -0.5) + (-1 \times -1)}{\sqrt{(1.5^2 + (-1.5)^2 + (-1)^2)} \times \sqrt{(1.5^2 + (-0.5)^2 + (-1)^2)}}$$

$$= 0.912$$

Predicting Ratings

- **Steps**:
  1. Identify top-k most similar items to the target item t.
  2. Use weighted average of user u's ratings on these similar items.
- **Formula**:

$$\hat{r}_{ut} = \frac{\sum_{j \in Q_t(u)} \text{AdjustedCosine}(j, t) \cdot r_{uj}}{\sum_{j \in Q_t(u)} |\text{AdjustedCosine}(j, t)|}$$

Rating Prediction

- **Prediction for Item 1**:

$$\hat{r}_{31} = \frac{3 \cdot 0.735 + 3 \cdot 0.912}{0.735 + 0.912} = 3$$

- **Prediction for Item 6**:

$$\hat{r}_{36} = \frac{1 \cdot 0.829 + 1 \cdot 0.730}{0.829 + 0.730} = 1$$

- **Conclusion**:
  - Item 1 is more likely to be preferred by user 3.
  - Predictions are consistent with user 3's rating history.

Adjusted Cosine vs PCC

User	Item A	Item B
$U1$	4	5
$U2$	2	3
$U3$	5	6

- PCC normalizes by item averages, capturing how users deviate from average item ratings.
- Adjusted Cosine normalizes by user averages, better reflecting relative preferences and user biases.
- Adjusted Cosine is generally more accurate for item-based collaborative filtering.

Summary

- *User-Based CF:* Recommends items by finding users with similar preferences. This approach relies heavily on user-user similarity but struggles with high computational complexity and sparsity due to diverse user behaviors.
- *Item-Based CF:* Focuses on item similarity. It recommends items similar to those a user has already liked. This method is more scalable, as items are generally fewer and more consistently rated.
- User-Based: Finds similar users.
- Item-Based: Finds similar items.

Indirect Similarities

- Indirect similarities reveal hidden connections.
- If A and C have low direct similarity but both are highly similar to B, they likely share similar preferences.
- This can be due to data sparsity, where few common interactions between A and C mask their true similarity.

Transferring Similarity in Collaborative Filtering

- Transferring similarity captures both direct and indirect similarities between users.
- Direct similarity only considers direct relationships.

$$t_{ij} = \epsilon \sum_{v} s_{iv} t_{vj} + s_{ij},$$

  where:
  - s_{ij}: Direct similarity between users i and j.
  - t_{ij}: Transferring similarity between users i and j.
  - ϵ: Decay factor for indirect similarities.
- In matrix form:

$$\mathbf{T} = (\mathbf{I} - \epsilon \mathbf{S})^{-1} \mathbf{S}.$$

Direct Similarity Matrix $\mathbf{S}$

Using cosine similarity, the direct similarity matrix $\mathbf{S}$ is:

$$\mathbf{S} = \begin{bmatrix} 1 & 0.981 & 1.0 & 0.789 & 0.645 \\ 0.981 & 1 & 1.0 & 0.789 & 0.645 \\ 1.0 & 1.0 & 1 & 0.789 & 0.645 \\ 0.789 & 0.789 & 0.789 & 1 & 0.645 \\ 0.645 & 0.645 & 0.645 & 0.645 & 1 \end{bmatrix}$$

- s_{ij}: Cosine similarity between users i and j.

Transferred Similarity Calculation

$$T = (I - \epsilon S)^{-1} S$$

where, ϵ is the decay factor - controls the influence of indirect similarities, and I is the identity matrix. Let $\epsilon = 0.01$

$$T = \begin{bmatrix} 1.0418 & 1.0228 & 1.0420 & 0.8261 & 0.6771 \\ 1.0228 & 1.0418 & 1.0420 & 0.8261 & 0.6771 \\ 1.0420 & 1.0420 & 1.0421 & 0.8263 & 0.6773 \\ 0.8261 & 0.8261 & 0.8263 & 1.0342 & 0.6744 \\ 0.6771 & 0.6771 & 0.6773 & 0.6744 & 1.0277 \end{bmatrix}$$

Using T for Recommendations

- Step 1: Predict Missing Ratings

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}(u)} t_{uv} \cdot r_{vi}}{\sum_{v \in \mathcal{N}(u)} |t_{uv}|}$$

- Step 2: Generate Recommendations
  - Predict ratings for all unrated items.
  - Recommend the top-k items with the highest predicted ratings.

Using $\mathbf{T}$ for Neighborhood Selection

- Step 1: Select Top-k Similar Users
  - For each user u, select the top-k users with the highest transferring similarity t_{uv}.
- Step 2: Use Neighborhood for Predictions
  - Use these neighbors to predict ratings or generate recommendations.
- Advantages:
  - Captures indirect relationships for better recommendations.

Using $\mathbf{T}$ for Cold Start Problems

- Step 1: Compute Transferring Similarities
    - For a new user u, compute t_{uv} to all existing users v.
- Step 2: Predict Ratings
    - Use these similarities to predict ratings for the new user.
- Advantages:
    - Mitigates cold start problems by leveraging indirect similarities.

What if We Use **S** Instead?

- Step 1: Predict Missing Ratings

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}(u)} s_{uv} \cdot r_{vi}}{\sum_{v \in \mathcal{N}(u)} |s_{uv}|}$$

- Step 2: Generate Recommendations
  - Predict ratings for all unrated items.
  - Recommend the top-k items with the highest predicted ratings.

Comparison of **T** and **S**

- Advantages of **T**:
  - Captures indirect relationships for better recommendations.
  - Mitigates cold start problems.
  - Improves recommendation diversity.
- Advantages of **S**:
  - Simpler and faster to compute.
  - Easier to interpret.
- When to Use **T**:
  - For large datasets with significant indirect relationships.
  - When higher recommendation accuracy is required.
- When to Use **S**:
  - For small datasets or when computational resources are limited.

Summary

- Transferring similarity **T** captures both direct and indirect relationships.
- It is useful for:
  - Generating recommendations.
  - Selecting better neighborhoods.
  - Visualizing user relationships.
  - Mitigating cold start problems.
- Direct similarity **S** is simpler but less powerful.
- Choose **T** or **S** based on the dataset size and requirements.