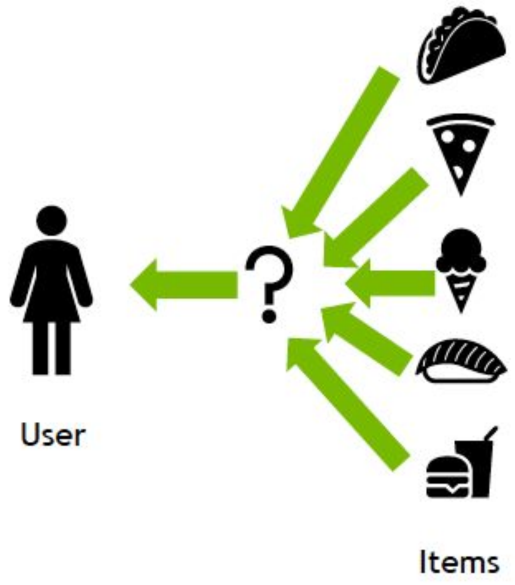
Machine learning

Lecture 7 - Recommendation system

28.04.2025

Toker Gilat



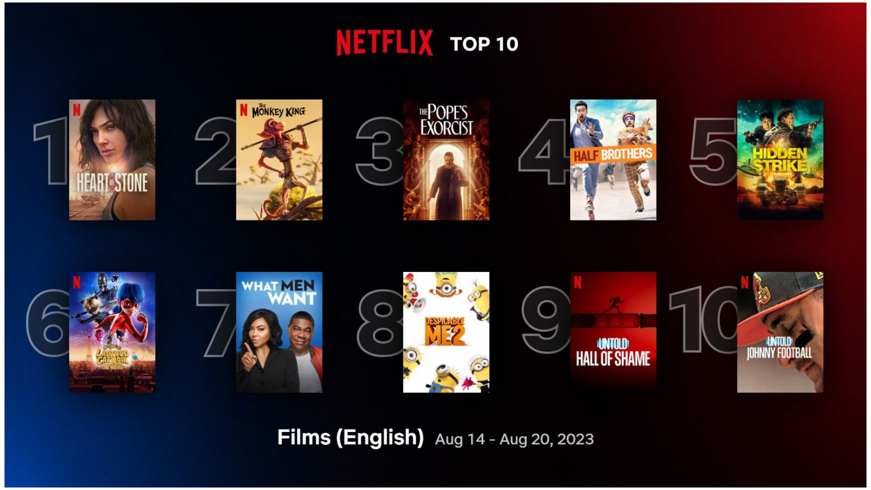
Recommendation system

Recommendation systems are algorithms aimed at suggesting relevant items to users based on their preferences or behavior. Recommender Systems aim to help a user to select items **from a crowded item** space. They address the problem of information overload.

Recommendation system types:

Popularity Based:

This is a basic system in which movies/shows which are rated high will be recommended to all the users in a certain demographic region. Eg: Netflix Top Trending will show top 10 movies trending in that particular country to ever user. No personalization.



Recommendation system types:

Content-Based Filtering:

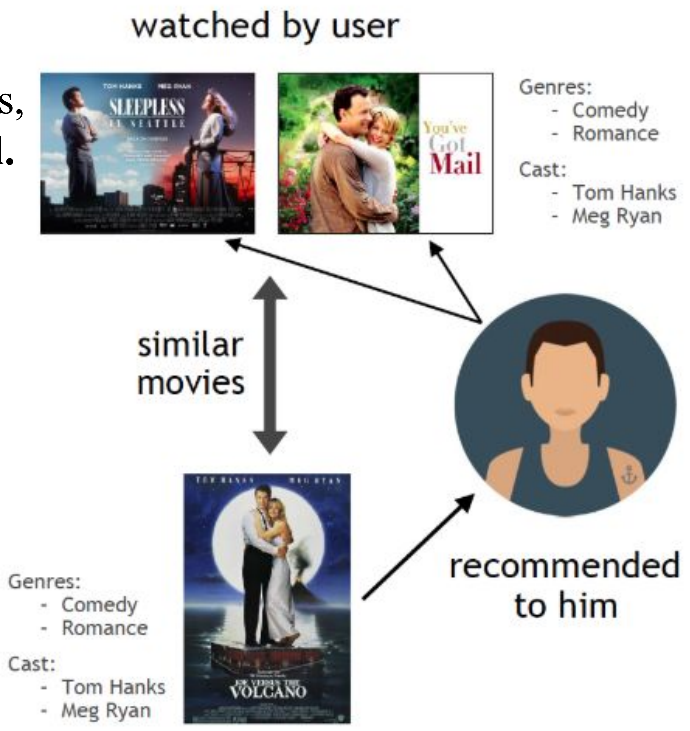
recommends items based on item features and user preferences, f**ocusing on the characteristics of the items a user has liked.**

<u>Mechanism:</u>

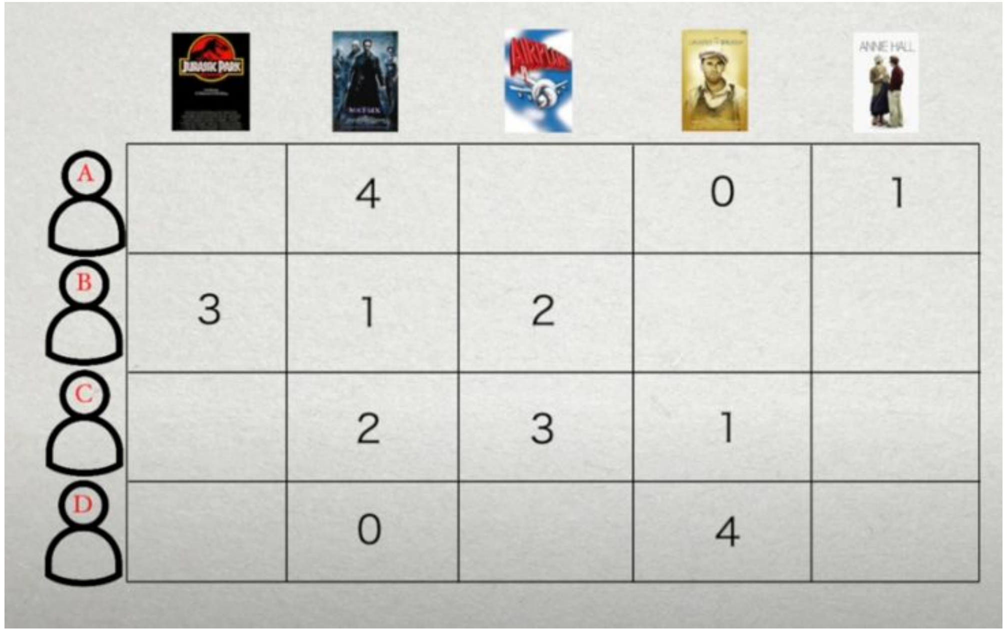
- Extracts features of items (e.g., movie genres, book descriptions).
- Builds a profile for each user based on features of the items they've interacted with.
- Recommends items that are similar to what the user has previously liked.

<u>Limitations:</u>

- **Cold-start problem for items**
- **Limited exploration**
- **Feature engineering**

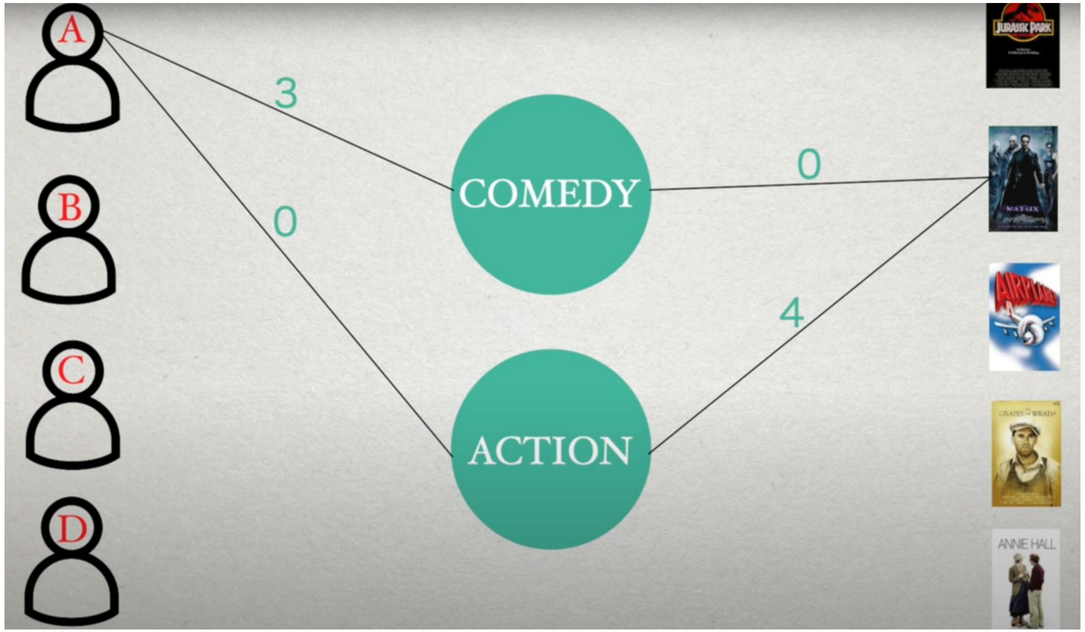


Recommendation system types:

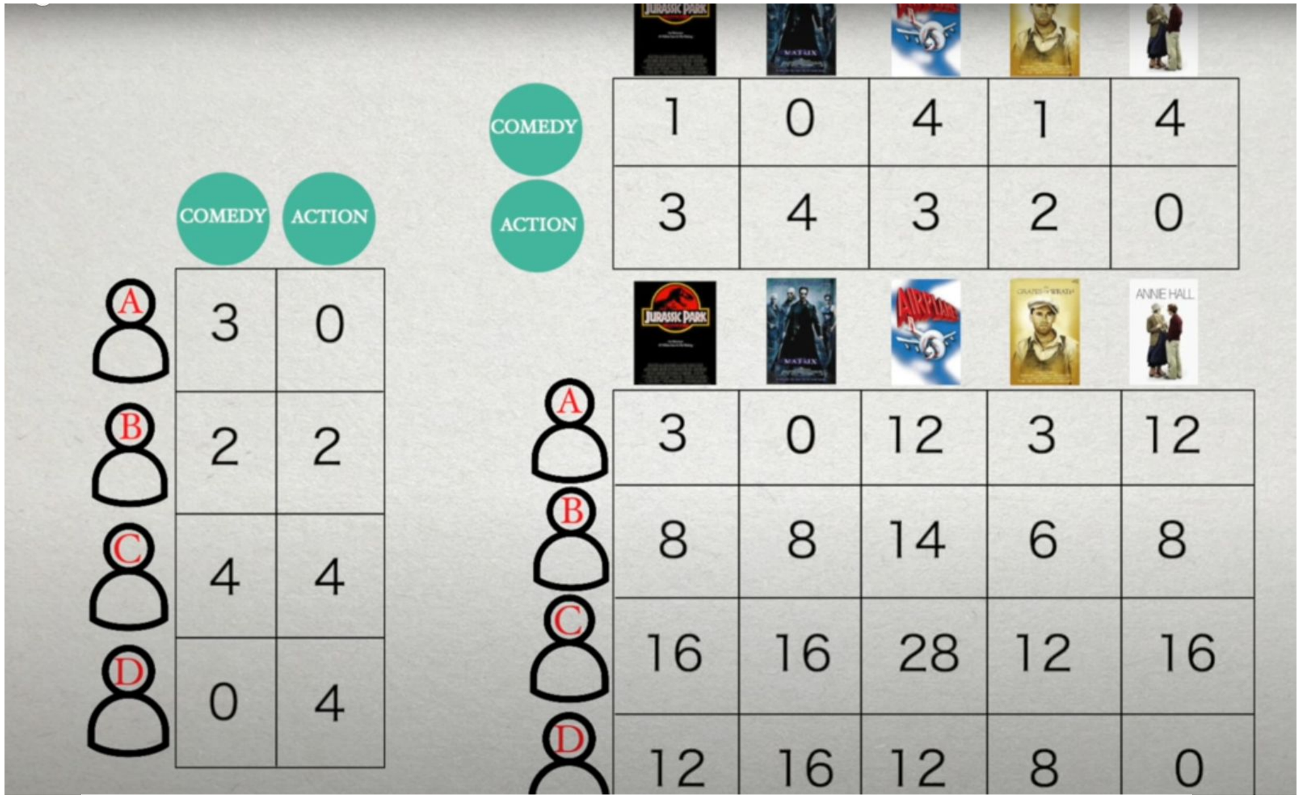


A		4		0	1
B	3	1	2		
C		2	3	1	
D		0		4	

Recommendation system types:



Recommendation system types:



	COMEDY	ACTION
A	3	0
B	2	2
C	4	4
D	0	4

COMEDY	1	0	4	1	4
ACTION	3	4	3	2	0

A	3	0	12	3	12
B	8	8	14	6	8
C	16	16	28	12	16
D	12	16	12	8	0

User-Item Matrix (Predicted) = User-Feature Matrix × Feature-Item Matrix

Recommendation system types:

Collaborative Filtering:

Collaborative Filtering (CF) leverages user interactions (e.g., ratings, clicks, purchases) to make recommendations, assuming users with similar past behaviors will share similar future preferences. Collaborative filtering algorithms recommend items (this is the filtering part) based on preference information from many users (this is the collaborative part).

<u>Types of Collaborative Filtering:</u>

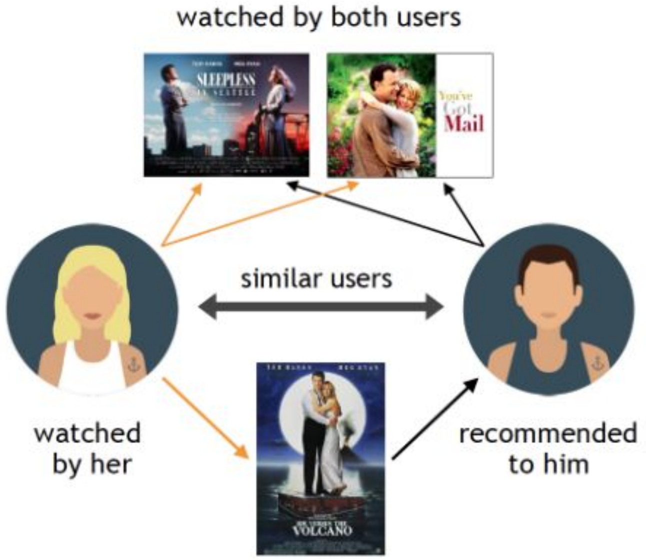
- User-User Collaborative Filtering
- Item-Item Collaborative Filtering

Advantages:

- No need for explicit feature extraction.
- Can uncover surprising relationships

Limitations:

- Cold-start problem
- Scalability
- Sparsity



Recommendation system

Collaborative Filtering

User-User Collaborative Filtering: In User-User Collaborative Filtering, the algorithm finds users with similar interaction patterns (e.g., ratings, clicks) and recommends items that these similar users have liked but the target user has not yet interacted with.

Algorithm Steps:

1. Prepare the User-Item Interaction Matrix.
2. Calculate Similarity Between Users (Cosine Similarity, Pearson Correlation).
3. Identify Top-N Similar Users.
4. Aggregate Preferences - Predict ratings for items the target user hasn’t interacted with.
5. Recommend Items - Rank items based on their predicted ratings.

Recommendation system

Collaborative Filtering

Algorithm Steps:

1. **Prepare the User-Item Interaction Matrix.**

User\Movie	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4	-	-	5	1	-	-
B	5	5	-	2	-	5	-
C	-	-	-	2	4	5	3

Recommendation system

Collaborative Filtering

<u>Algorithm Steps:</u>

2. Calculate Similarity Between Users (Cosine Similarity, Pearson Correlation).

Common Similarity Metric:

- <u>Cosine Similarity:</u> Measures the cosine of the angle between two user vectors in the feature space.

$$\text{sim}(A, B) = \frac{\sum_i r_{A,i} \cdot r_{B,i}}{\sqrt{\sum_i r_{A,i}^2} \cdot \sqrt{\sum_i r_{B,i}^2}}$$

  * Sensitive to missing data (unrated items).

- <u>Pearson Correlation Coefficient:</u> Measures the linear correlation between two user vectors after subtracting their mean ratings.

$$\text{sim}(A, B) = \frac{\sum_i (r_{A,i} - \bar{r}_A)(r_{B,i} - \bar{r}_B)}{\sqrt{\sum_i (r_{A,i} - \bar{r}_A)^2} \cdot \sqrt{\sum_i (r_{B,i} - \bar{r}_B)^2}}$$

  * Accounts for user biases (e.g., one user always rates high, another always rates low).
  * Measures the linear correlation between users' centered rating vectors

Recommendation system

Collaborative Filtering

Algorithm Steps:

2. Calculate Similarity Between Users (Cosine Similarity, Pearson Correlation).

- User A: $r_A = [4, 0, 0, 5, 1, 0, 0]$
- User B: $r_B = [5, 5, 0, 2, 0, 5, 0]$

1. **Dot Product:**

$$r_A \cdot r_B = (4 \cdot 5) + (0 \cdot 5) + (0 \cdot 0) + (5 \cdot 2) + (1 \cdot 0) + (0 \cdot 5) + (0 \cdot 0) = 20 + 0 + 0 + 10 + 0 + 0 + 0 = 30$$

2. **Magnitude of r_A:**

$$||r_A|| = \sqrt{(4^2) + (0^2) + (0^2) + (5^2) + (1^2) + (0^2) + (0^2)} = \sqrt{16 + 0 + 0 + 25 + 1 + 0 + 0} = \sqrt{42}$$

3. **Magnitude of r_B:**

$$||r_B|| = \sqrt{(5^2) + (5^2) + (0^2) + (2^2) + (0^2) + (5^2) + (0^2)} = \sqrt{25 + 25 + 0 + 4 + 0 + 25 + 0} = \sqrt{79}$$

4. **Cosine Similarity:**

$$\text{sim}(A, B) = \frac{r_A \cdot r_B}{||r_A|| \cdot ||r_B||} = \frac{30}{\sqrt{42} \cdot \sqrt{79}} \approx 0.55$$

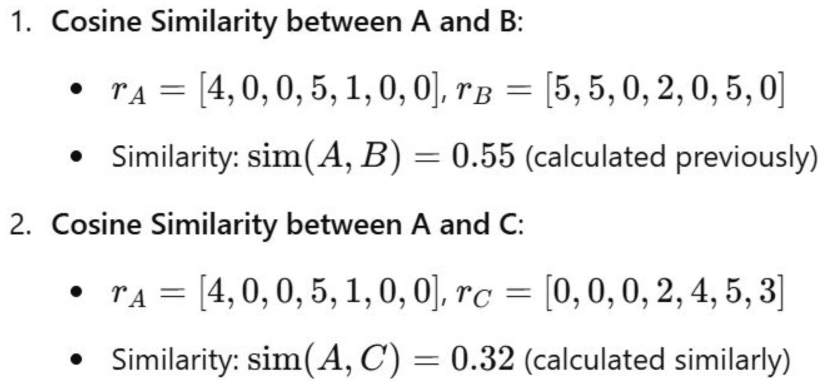
Recommendation system

Collaborative Filtering

Algorithm Steps:

3. Identify Top-N Similar Users.

- Single Nearest Neighbor: Use the ratings of the single most similar user (nearest neighbor) to make recommendations.
- Top-N Similar Users: Use the ratings of the top-N most similar users, where N>1.
- Weighted Aggregation: Calculate a weighted average of ratings from multiple users, where the weight is based on their similarity score.



1. **Cosine Similarity between A and B:**
   - $r_A = [4, 0, 0, 5, 1, 0, 0]$, $r_B = [5, 5, 0, 2, 0, 5, 0]$
   - Similarity: $\text{sim}(A, B) = 0.55$ (calculated previously)
2. **Cosine Similarity between A and C:**
   - $r_A = [4, 0, 0, 5, 1, 0, 0]$, $r_C = [0, 0, 0, 2, 4, 5, 3]$
   - Similarity: $\text{sim}(A, C) = 0.32$ (calculated similarly)

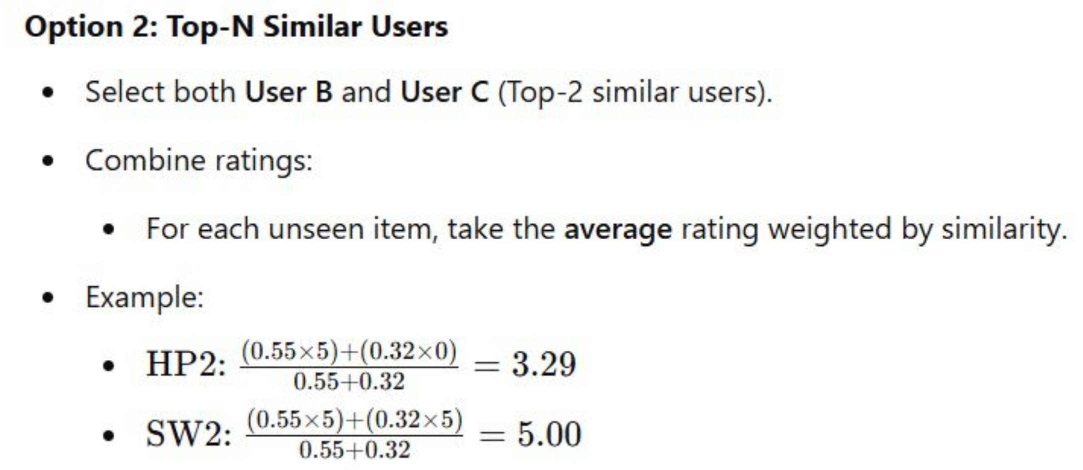
הטכניון
ביה״ס ללימודי המשך
של הטכניון ע״ש עזריאלי

Recommendation system

Collaborative Filtering

Algorithm Steps:

4. Aggregate Preferences - Predict ratings for items the target user hasn’t interacted with.



Option 2: Top-N Similar Users

- Select both **User B** and **User C** (Top-2 similar users).
- Combine ratings:
    - For each unseen item, take the **average** rating weighted by similarity.
- Example:
    - HP2: $\frac{(0.55\times5)+(0.32\times0)}{0.55+0.32} = 3.29$
    - SW2: $\frac{(0.55\times5)+(0.32\times5)}{0.55+0.32} = 5.00$

Recommendation system

Collaborative Filtering

<u>Algorithm Steps:</u>

5. Recommend Items

Recommend ‘SW2’ movie to user 1

Recommendation system

Collaborative Filtering

Item-Item Collaborative Filtering: In Item-Item Collaborative Filtering, the algorithm recommends items based on their similarity to items that the target user has already interacted with. It assumes that similar items will appeal to the same user.

<u>Algorithm Steps:</u>

1. Prepare the User-Item Interaction Matrix.
2. Calculate Similarity Between **Items** (Cosine Similarity, **Jaccard Similarity**).
3. Identify Top-N Similar items.
4. Predict Ratings
5. Recommend Items - Rank items by predicted ratings and recommend the top-N items.

Recommendation system

Collaborative Filtering

2. Calculate Similarity Between Items

- Jaccard Similarity: Measures the proportion of shared users who rated both items.

$$\mathrm{sim}(i,j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$$

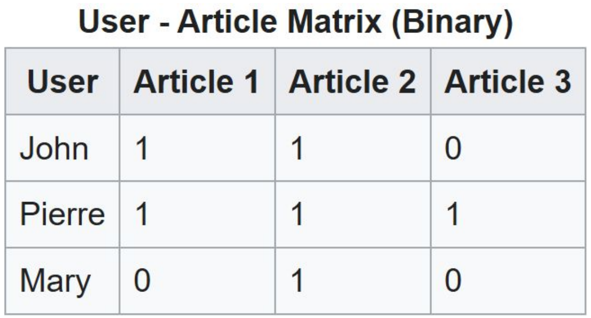
- U_i: Users who rated item i.
- U_j: Users who rated item j.
- Best for binary data or datasets where only interactions are logged (e.g., clicks, purchases).

Recommendation system

Collaborative Filtering

2. Calculate Similarity Between Items

Example:



User - Article Matrix (Binary)

User	Article 1	Article 2	Article 3
John	1	1	0
Pierre	1	1	1
Mary	0	1	0

$$\text{Jaccard}(1,2) = \frac{2}{3} \approx 0.6667$$

$$\text{Jaccard}(1,3) = \frac{1}{2} = 0.5$$

$$\text{Jaccard}(2,3) = \frac{1}{3} \approx 0.3333$$

Recommendation system

Collaborative Filtering

In Theory: Dual Approaches - Both User-User and Item-Item Collaborative Filtering are conceptually symmetric. They rely on the same idea: finding similarities, either between users or between items.

In Practice: Item-Item Often Outperforms:
Many use cases demonstrate that Item-Item Collaborative Filtering tends to deliver better performance. This is primarily because of the stable and interpretable nature of item relationships compared to user preferences, which can vary widely.

Why?

- Items Are Simpler
- Scalability
- Data Sparsity

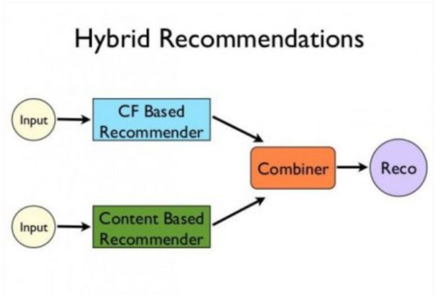
Recommendation system types:

Hybrid Systems:

Hybrid Systems combine Collaborative Filtering and Content-Based Filtering to overcome the limitations of both approaches.

<u>Mechanism:</u>

- Uses collaborative filtering to leverage interaction data.
- Incorporates content-based filtering to factor in item metadata.
- Combines predictions from both methods (weighted sum, stacking).



<u>Advantages:</u>

- Mitigates the cold-start problem by using item metadata.
- Provides more robust and accurate recommendations.

<u>Limitations:</u>

- Computationally expensive due to the combination of multiple models.
- Requires integration of different types of data (interaction data + metadata).

Recommendation system

Matrix Factorization

Overview: Matrix Factorization (MF) is a powerful technique commonly used in recommendation systems. It works by breaking down a large user-item interaction matrix into smaller matrices that capture latent features (hidden patterns) of users and items. These latent features help predict user preferences for items they haven't interacted with yet.

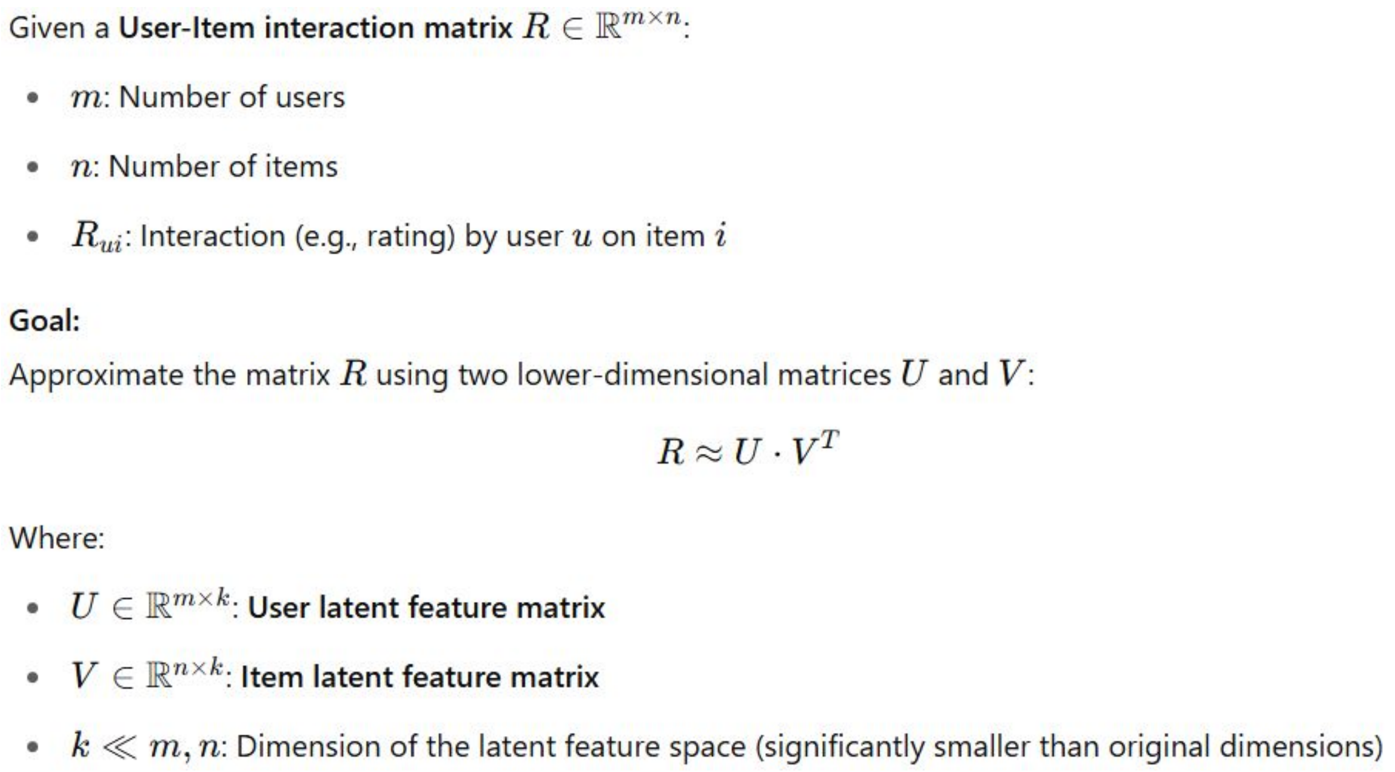
Key Idea: Instead of directly comparing users and items, MF maps them into a shared latent feature space, making it easier to find similarities and make predictions.

Why It’s Important?

- **Handles Sparse Data:** MF can work effectively even when most of the user-item matrix is empty (common in recommendation systems).
- **Uncover Hidden Relationships:** By learning latent features, MF identifies patterns that are not directly observable in the raw data.
- **Scalability:** Modern MF techniques are computationally efficient and scale well for large datasets.

Recommendation system

Matrix Factorization - Technical Overview



Given a **User-Item interaction matrix** $R \in \mathbb{R}^{m \times n}$:

- m: Number of users
- n: Number of items
- R_{ui}: Interaction (e.g., rating) by user u on item i

Goal:

Approximate the matrix R using two lower-dimensional matrices U and V:

$$R \approx U \cdot V^T$$

Where:

- $U \in \mathbb{R}^{m \times k}$: **User latent feature matrix**
- $V \in \mathbb{R}^{n \times k}$: **Item latent feature matrix**
- $k \ll m, n$: Dimension of the latent feature space (significantly smaller than original dimensions)

Recommendation system

Matrix Factorization - Technical Overview

MF matrices (U, V) are learned by minimizing the following objective function:

$$\min_{U,V} \sum_{(u,i)\in\mathcal{K}} (R_{ui} - U_u \cdot V_i^T)^2 + \lambda(||U_u||^2 + ||V_i||^2)$$

Where:

- $\mathcal{K}$: Set of known (observed) user-item interactions
- λ: Regularization parameter to prevent overfitting (L2 Regularization)

Typical optimization algorithms include:

- **Stochastic Gradient Descent (SGD)**: iterative updates of U and V
- **Alternating Least Squares (ALS)**: optimizes one matrix at a time holding the other fixed, iteratively

Kaggle competition

MovieLens 100K Dataset

Kaggle competition

MovieLens 100K Dataset

Goal: Build a recommendation system **to predict movie ratings** for users based on their past interactions.
Dataset: The MovieLens 100K dataset provides user-item interactions, including explicit ratings for movies from 1 to 5.
Challenge: Use the provided data to predict the ratings for movies not yet rated by users and recommend top movies they are likely to enjoy.

https://www.kaggle.com/datasets/prajitdatta/movielens-100k-dataset

Kaggle competition

MovieLens 100K Dataset

The dataset is divided into two main files:

U1.base: Training dataset containing user-movie interactions.

Each row includes:

- User ID
- Movie ID
- Rating (1-5)
- Timestamp
-

U.item: Metadata about movies.

- Each row includes:
- Movie ID
- Movie Title
- Release Year
- Genre Information

Kaggle competition

MovieLens 100K Dataset

Competition Details

- After training your recommendation model, your task is to predict ratings for movies using the provided test dataset (u1_test_untagged_1000.csv).
- The test file contains only the columns userId and movieId. You need to use your model to predict the missing ratings and add these predictions under a new column named predicted_rating.
- Your model's performance will be evaluated using Root Mean Squared Error (RMSE)

Until 20:30: Work on developing and finalizing your recommendation model for the competition.
20:30 - 20:45: Receive the test set and submit your results file containing the predicted ratings.
20:45 – 21:00: Break while the results are being calculated and evaluated.
21:00 – 21:30: Class discussion and student presentations of their solution approaches.

https://www.kaggle.com/datasets/prajitdatta/movielens-100k-dataset/data