UNIT III: User-Based collaborative filtering, Similarity Function Variants, Variants of the Prediction Function, Item-Based Collaborative filtering, Comparing User-Based and Item-Based Methods, Strengths and Weaknesses of Neighborhood-Based Methods

Neighborhood-Based Collaborative Filtering

Introduction

- **Neighborhood-based collaborative filtering** (also called **memory-based filtering**) relies on user and item similarity.
- Two main types:
  - **User-based collaborative filtering**: Predicts ratings based on similar users' ratings.
  - **Item-based collaborative filtering**: Predicts ratings based on a user's ratings of similar items.

Key Differences

- **User-based filtering**: Uses peer users' ratings (rows of **rating matrix**).
- **Item-based filtering**: Uses the same user's ratings on similar items (columns of rating matrix).
- They are complementary but produce different recommendation types.

Problem Formulation

- **Predicting missing ratings**: Estimate the unknown rating for a user-item pair.
- **Finding top-k items or users**:
  - More practical in real-world applications (e.g., recommending top-k items to users).
  - Top-k users can help merchants with targeted marketing.

Key Properties of Ratings Matrices

1. Definition and Structure of Ratings Matrices

- The ratings matrix **R** is an **m** × **n** matrix where **m** represents users and **n** represents items.
- Ratings are typically **sparse**, with only a small subset of the entries specified.
- **Specified entries = Training data**; **Unspecified entries = Test data**.
- Recommendation is a **generalization** of classification and regression problems.

Example of a Sparse Ratings Matrix:

User \ Item	Item 1	Item 2	Item 3	Item 4	Item 5
User A	4	5	?	2	?
User B	?	?	3	?	5
User C	2	?	?	4	?

Here, **"?" represents missing ratings**, meaning users have not rated those items.

2. Types of Ratings

Continuous Ratings

- Ratings can take any value within a range (e.g., Jester joke system: -10 to 10).
- **Drawback**: Users find it difficult to choose from an infinite set of values.

Interval-Based Ratings

- Ratings are selected from a fixed scale (e.g., 1-5, -2 to 2, 1-7).
- Assumes equal distance between rating levels.

Ordinal Ratings

- Categorical but ordered values (e.g., "Strongly Disagree" to "Strongly Agree").
- No assumption that differences between categories are equal

- **Forced choice method**: Omits neutral options to ensure decisive responses.

Binary Ratings

- Only two options (e.g., Like/Dislike, Thumbs up/Thumbs down).
- Found in systems like **Pandora Radio**.
- **Forced choice** is imposed, as users cannot express neutrality.

Unary Ratings

- Users express only **positive** preferences (e.g., Facebook "Like").
- Often derived from implicit feedback (e.g., purchasing an item implies a positive rating).
- No explicit **negative** feedback option.

3. Implicit Feedback & Unary Ratings

- **Implicit feedback**: User actions (e.g., purchases, clicks) are interpreted as preferences.
- **More common** than explicit ratings, as users interact more frequently than they rate.
- Can be seen as a **positive-unlabeled (PU) learning** problem in classification.

4. The Long-Tail Property in Ratings Distribution

- **Observation**: A **small fraction** of items are rated frequently (popular items), while the majority have **few ratings** (long-tail items).
- **Graph representation**:
  - X-axis: Items ranked by frequency of ratings.
  - Y-axis: Number of ratings per item.
  - Results in a **skewed distribution**.

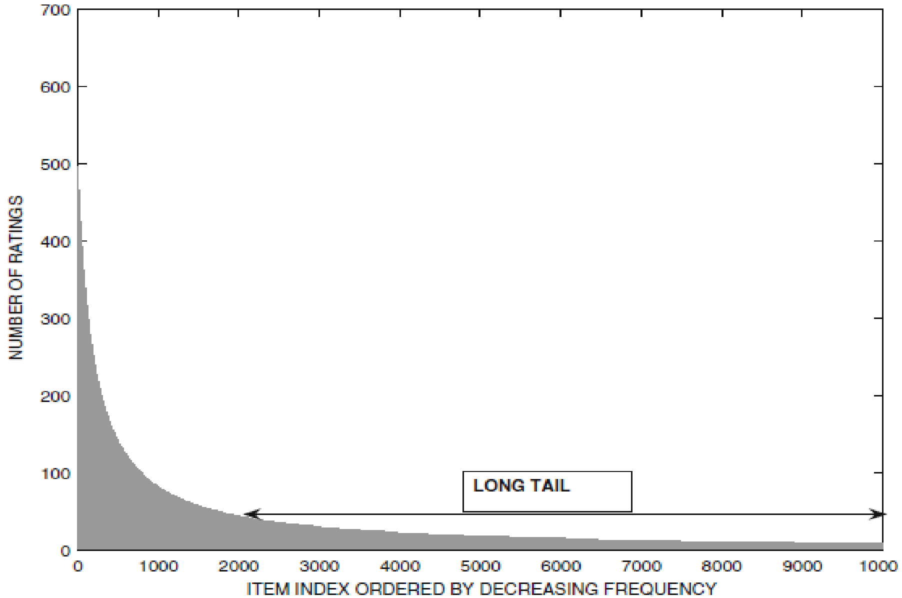

Figure 2.1: The long tail of rating frequencies

5. Implications of the Long-Tail Property

- **Merchant Profitability**
  - Popular items are competitive but **low-profit**.
  - **Less popular items (long-tail)** often have **higher profit margins** (e.g., Amazon's strategy).

- **Difficulty in Long-Tail Predictions**
  - Sparse ratings in the **long tail** make predictions **less accurate**.
  - Many recommendation algorithms **favor popular items**, reducing diversity.
- **Impact on Neighborhood-Based Filtering**
  - **High-frequency items** define neighborhoods, leading to **biased predictions**.
  - Frequent items do not always represent rare items, causing **misleading recommendations**.
  - **Evaluation metrics** may also become **misleading** due to this bias.

Predicting Ratings with Neighborhood-Based Methods

1. Concept of Neighborhood-Based Methods

- Uses **user-user similarity** or **item-item similarity** to make recommendations.
- Relies on the principle that **similar users** or **similar items** have similar ratings.

2. Two Basic Principles

- **User-Based Models**
  - Users with similar rating patterns tend to rate items similarly.
  - Example: If **Alice and Bob** have rated movies similarly in the past, Alice's rating for **"Terminator"** can predict Bob's rating for the same movie.

- **Item-Based Models**
  - Similar items receive similar ratings from the same user.
  - Example: Bob's ratings for "Alien" and "Predator" can predict his rating for "Terminator."

3. Connection to Nearest Neighbor Classification

- Collaborative filtering is a **generalization of classification/regression modeling**.
- Neighborhood-based models are similar to **nearest neighbor classifiers** in machine learning.
- Unlike classification, collaborative filtering determines nearest neighbors using **both rows (users) and columns (items)**.

4. User-User Similarity Computation (Example from Table 2.1)

- **User similarity measures**:
  - **Cosine similarity**
  - **Pearson correlation**
- Users with higher similarity scores are considered **closer neighbors**.

Table 2.1: User-user similarity computation between user 3 and other users

Item-Id ⇒ / User-Id ⇓	1	2	3	4	5	6	Mean Rating	Cosine(i, 3) (user-user)	Pearson(i, 3) (user-user)
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

5. Item-Item Similarity Computation (Example from Table 2.2)

- **Adjusted cosine similarity** is used for item similarity calculations.
- Items are compared after **mean-centering** ratings to eliminate user bias.
- **Cosine similarity scores** between items indicate their similarity levels.

Item-Id ⇒ / 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$(1, j)$ (item-item)	1	0.735	0.912	-0.848	-0.813	-0.990
Cosine$(6, j)$ (item-item)	-0.990	-0.622	-0.912	0.829	0.730	1

Scenario: Movie Recommendation

Consider a movie rating system where users rate movies on a **1 to 5 scale**. The goal is to predict the missing rating for a user using **neighborhood-based collaborative filtering**.

Ratings Matrix (Users × Movies)

User	Movie A	Movie B	Movie C	Movie D	Movie E
Alice	5	3	?	4	2
Bob	4	5	4	3	?
Charlie	2	1	3	?	5
David	3	3	2	5	?

Step 1: Choose a Method (User-Based or Item-Based)

Let's predict **Alice's missing rating for Movie C** (denoted as "?").

1. **User-Based Approach:**
    - Identify users **most similar to Alice** (e.g., Bob, Charlie, and David).
    - Compute **similarity** (e.g., using Pearson correlation or Cosine similarity).
    - Use ratings from similar users to predict **Alice's rating for Movie C**.
2. **Item-Based Approach:**
    - Identify **movies similar to Movie C** (based on ratings from all users).
    - Use Alice's ratings for those **similar movies** to predict the missing rating.

Step 2: Compute Similarities

For a **user-based approach**, similarity can be calculated using **cosine similarity** or **Pearson correlation.**

Example Cosine Similarity Between Alice & Bob:

$$\text{Similarity}(Alice, Bob) = \frac{(5 \times 4) + (3 \times 5) + (4 \times 3) + (2 \times ?)}{\sqrt{(5^2 + 3^2 + 4^2 + 2^2)} \times \sqrt{(4^2 + 5^2 + 4^2 + ?^2)}}$$

Similarly, we compute the **similarities with Charlie and David**.

Step 3: Predict Alice's Rating for Movie C

Using **weighted average of ratings from similar users**, the missing rating is predicted as:

$$\hat{r}_{Alice,C} = \frac{\sum_{u \in Neighbors} \text{Similarity(Alice, u)} \times \text{Rating of Movie C by user u}}{\sum_{u \in Neighbors} \text{Similarity(Alice, u)}}$$

If **Bob is the most similar user**, and he rated **Movie C as 4**, then Alice's predicted rating might be **around 4**.

Alternative: Item-Based Approach

Instead of finding similar **users**, we find similar **movies** to Movie C (e.g., Movie A and Movie D) and use **Alice's ratings on those movies** to predict her rating for Movie C.

Final Prediction

- If **user-based filtering** is used → Alice's rating for **Movie C ≈ 4** (based on Bob's rating).
- If **item-based filtering** is used → Alice's rating for **Movie C ≈ 3.5-4** (based on similarity with Movies A & D).

User-Based Neighborhood Models

1. Concept of User-Based Neighborhoods

- Defines **user neighborhoods** by identifying **similar users** to the **target user**.
- Uses these similar users' ratings to **predict missing ratings** for the target user.
- A similarity function is required, but it must account for **different rating scales** among users.

2. Key Challenges in User-Based Similarity Computation

- **Different rating scales:** Some users consistently give higher or lower ratings than others.
- **Sparse ratings:** Many users rate only a **small subset** of items, making similarity computation challenging.
- **Mutual rating sets:** Similarity is computed only for the **overlapping rated items** between two users.

3. Steps to Compute User Similarity

- **Define Rated Items for Each User**
  - I_u = Set of items rated by user u.
  - $I_u \cap I_v$ = Items rated by **both** users u and v.
- **Compute Mean Rating (μ_u) for Each User**
  - The mean rating of a user is computed as:
  - $\mu_u = \frac{\sum_{k \in I_u} r_{uk}}{|I_u|}$
  - This ensures normalization across different rating scales.
- **Calculate Pearson Correlation Similarity**
  - Pearson similarity between two users u and v is computed as:

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

  - Measures how strongly correlated two users' rating patterns are

- **Find Top-k Similar Users for Each Item Prediction**
  - The k most similar users who have rated the target item are selected.
  - Users with negative or very low similarity may be excluded for better predictions.

4. Predicting Missing Ratings Using Neighborhood-Based Approach

- Ratings need to be mean-centered to avoid bias from different rating scales:

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

The final predicted rating ($\hat{r}_{uj}$) for user u on item j is computed as:

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

where $P_u(j)$ is the set of **top-k similar users** who have rated item j.

5. Variations & Enhancements

- Some implementations compute **mean ratings dynamically** based on overlapping items.
- **Heuristic filtering** removes users with **low or negative similarity** to improve accuracy.
- The method allows for **different similarity measures** and **weighting strategies** to fine-tune recommendations.

Summary: Example of User-Based Collaborative Filtering Algorithm

1. Problem Statement

- The goal is to **predict missing ratings** for **User 3** in **Table 2.1**.
- Specifically, we need to compute:
  - $\hat{r}_{31}$ → User 3's predicted rating for **Item 1**.
  - $\hat{r}_{36}$ → User 3's predicted rating for **Item 6**.
- **Approach:**
  - Use **User-Based Collaborative Filtering** by computing similarity scores and applying **weighted average prediction**.

2. Step 1: Compute Similarity Between Users

- Similarity is calculated between **User 3** and all other users using:

  1. **Cosine Similarity**
  2. **Pearson Correlation Coefficient**

Example Calculations for User 1 and User 3:

1. **Cosine Similarity Calculation**

$$\text{Cosine}(1,3) = \frac{(6 \times 3) + (7 \times 3) + (4 \times 1) + (5 \times 1)}{\sqrt{6^2 + 7^2 + 4^2 + 5^2} \times \sqrt{3^2 + 3^2 + 1^2 + 1^2}} = 0.956$$

2. **Pearson Correlation Calculation**

$$\text{Pearson}(1,3) = \frac{(6 - 5.5)(3 - 2) + (7 - 5.5)(3 - 2) + (4 - 5.5)(1 - 2) + (5 - 5.5)(1 - 2)}{\sqrt{(1.5)^2 + (1.5)^2 + (-1.5)^2 + (-0.5)^2} \times \sqrt{(1)^2 + (1)^2 + (-1)^2 + (-1)^2}} = 0.894$$

- **Similarities for all users** are stored in the last two columns of **Table 2.1**.
- The **top-2 most similar users** to **User 3** are:
  - **User 1** (Pearson = 0.894)
  - **User 2** (Pearson = 0.939)

3. Step 2: Predict Missing Ratings Using Weighted Average

- The missing ratings are computed using a **weighted sum of ratings from similar users.**

Raw Prediction (Without Mean-Centering)

1. **Predict $\hat{r}_{31}$ (User 3's rating for Item 1):**

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

2. **Predict $\hat{r}_{36}$ (User 3's rating for Item 6):**

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

- **Interpretation:**
  - **Item 1 (6.49)** > **Item 6 (4)**, so **Item 1 is recommended over Item 6.**
  - However, this **does not account for rating biases.**

4. Step 3: Mean-Centering for Improved Prediction

- **Mean-centering** adjusts ratings to **account for individual rating biases.**
- The adjusted rating is computed as:

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

- The new **weighted mean-centered predictions** are:

1. **Predict $\hat{r}_{31}$ using mean-centered ratings:**

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

2. **Predict $\hat{r}_{36}$ using mean-centered ratings:**

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

Example from Table 2.2 (Mean-Centered Ratings for Users 1 & 2):

User	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Mean (μ_u)
User 1	1.5	0.5	1.5	-1.5	-0.5	-1.5	5.5
User 2	1.2	2.2	?	-0.8	-1.8	-0.8	4.8

4. Predicting $\hat{r}_{31}$ (User 3's Rating for Item 1)

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

Breaking it down:

- $\mu_3 = 2$ (User 3's mean rating)
- Users **1 and 2** have rated **Item 1** and are the top-2 most similar users.
- **Mean-centered ratings of Item 1:**
  - User 1: 1.5
  - User 2: 1.2
- **Weighted sum of mean-centered ratings:**
  - $(1.5 \times 0.894) = 1.341$
  - $(1.2 \times 0.939) = 1.1268$
- **Denominator (sum of similarities):**
  - $0.894 + 0.939 = 1.833$
- **Final prediction:**

$$\hat{r}_{31} = 2 + \frac{1.341 + 1.1268}{1.833} = 2 + 1.35 = 3.35$$

Thus, **User 3's predicted rating for Item 1 is $\mathbf{3.35}$.**

5. Predicting $\hat{r}_{36}$ (User 3's Rating for Item 6)

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

Breaking it down:

- **Mean-centered ratings of Item 6:**
  - User 1: -1.5
  - User 2: -0.8
- **Weighted sum of mean-centered ratings:**
  - $(-1.5 \times 0.894) = -1.341$
  - $(-0.8 \times 0.939) = -0.7512$
- **Denominator (sum of similarities):**
  - $0.894 + 0.939 = 1.833$
- **Final prediction:**

$$\hat{r}_{36} = 2 + \frac{-1.341 - 0.7512}{1.833} = 2 - 1.14 = 0.86$$

Thus, **User 3's predicted rating for Item 6 is $\mathbf{0.86}$.**

5. Key Observations & Insights

1. **Item 1 is still ranked higher than Item 6** → So Item 1 is recommended.
2. **Mean-centering reduces bias:**
   - The original **unadjusted** prediction for **Item 6** was **4**, but after mean-centering, it dropped to **0.86**, which is **more realistic.**
   - This is because **User 3's closest peers (Users 1 & 2) also rated Item 6 lower than their average ratings.**
3. **Mean-centering prevents incorrect recommendations:**
   - The **raw approach falsely suggested Item 6 was highly rated (4).**
   - Mean-centering **correctly reflects that Item 6 is not a good recommendation** for User 3.
4. **Predicted value outside rating range:**
   - $\hat{r}_{36} = 0.86$, but the valid rating scale is **1 to 7**.
   - In real-world systems, this prediction can be **corrected to the nearest valid rating.**

Similarity Function Variants

1. Raw Cosine Similarity

- **Computes similarity using raw ratings** instead of mean-centered ratings.
- **Formula (Mutually Rated Items Only):**

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

- Alternative Formula (All Rated Items Used for Normalization):

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

2. Preference for Pearson Correlation

- **Pearson correlation is better than raw cosine** because it **adjusts for user bias** using mean-centering.
- Accounts for **differences in users' rating tendencies** (e.g., generous vs. strict raters).

3. Significance Weighting for Similarity Adjustment

- **Issue:** Similarity scores are unreliable if users have very **few common ratings**.
- **Solution:** Apply a **discount factor** when the number of common ratings $(|I_u \cap I_v|)$ is low.
- **Formula:**

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

- β = predefined threshold.
- Ensures similarity is reduced when common ratings are low.
- **Range:** Always between **0 and 1**.

4. Usage of Discounted Similarity

- Used in:
  - **Selecting peer groups** for recommendations.
  - **Computing weighted predictions** for missing ratings.

Variants of the Prediction Function

1. Z-score Normalization for Ratings

- **Z-score** is used as an alternative to mean-centering:
  - **Standard deviation (σu)** of a user's ratings is computed as:

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

  - **Standardized rating** (Z-score) is calculated as:

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

  - This normalizes the ratings further by scaling the deviations from the mean.

2. Z-score Prediction Formula

- **Prediction using Z-score** normalization:

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

  - Here, $P_u(j)$ is the set of **top-k similar users** who have rated item **j**.
  - **Z-score ratings** z_{vj} are weighted by similarity scores.
  - The **standard deviation** σ_u is applied to scale the prediction back to the original rating scale.

3. Comparison of Mean-Centering vs. Z-score

- **Z-score** may sometimes provide higher-quality results, but there are **conflicting conclusions** in studies.
- **Problems with Z-score:**
  - Predicted ratings might fall **outside the permissible rating range** (e.g., 1-7).
  - Despite this, predictions **can still be useful for ranking items** based on desirability.

4. Exponentiating Similarity Weights (α)

- **Similarity weighting** can be **amplified** using an exponent:

$$Sim(u, v) = \text{Pearson}(u, v)^{\alpha}$$

  - By setting $\alpha > 1$, the importance of similarity is **amplified** during prediction, giving more weight to highly similar users.

5. Neighborhood-based Collaborative Filtering as Regression

- This approach is closer to **nearest neighbor regression** because the predicted values are treated as **continuous variables**.
- **Classification alternative:**
  - **Treat ratings as categorical values** and ignore the ordering among ratings.
  - **Vote-based prediction:** The most frequent rating among the peer group is chosen as the prediction.
  - **Advantage**: More effective with **small distinct ratings** (e.g., "Agree," "Neutral," "Disagree").
  - **Limitations**: Loses **ordering information** with **high-granularity ratings** (e.g., on a 1-7 scale).

Item-Based Neighborhood Models

1. **Concept of Item-Based Neighborhood Models**
   - Instead of **finding similar users**, this model **finds similar items**.
   - Similarity is computed **between items (columns in the ratings matrix)** rather than users.
   - **Each row is mean-centered** before computing similarities.
2. **Mean-Centering Process**
   - Similar to **user-based filtering**, but performed **column-wise** (on items).
   - The **average rating of each item** is subtracted from individual ratings:

$$s_{uj} = r_{uj} - \mu_j$$

- μ_j = Mean rating of item j.
- s_{uj} = Mean-centered rating of user u for item j.

3. Adjusted Cosine Similarity for Items

- **Adjusts cosine similarity** by mean-centering ratings before computing similarity.
- Formula:

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

  - U_i = Users who have rated **item i**.
  - $U_i \cap U_j$ = Users who have rated **both items i and j**.
  - **Pearson correlation can also be used**, but adjusted cosine generally performs better.

4. Predicting Missing Ratings

- To predict user **u**'s rating for item **t**:
  - Find the **top-k most similar items** to item t.
  - Select only the **items that user u has rated**.
  - Compute a **weighted average** of user u's ratings on these similar items.

- Formula:

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

  - $Q_t(u)$ = **Top-k most similar items** that user **u** has rated.
  - **Weighting is based on adjusted cosine similarity.**

5. Example: Movie Recommendation

- If a user has **rated several sci-fi movies**, the model can predict their rating for another **similar sci-fi movie**.
- **Item similarity** ensures recommendations align with the user's **interests and rating patterns**.

6. Similarities to User-Based Models

- **Same core structure**, but items replace users in the similarity computation.
- **Variants of similarity and prediction functions** (like Z-score and weighting adjustments) can also be applied to item-based filtering.

Example: Item-Based Collaborative Filtering Algorithm

- **Item-Based Collaborative Filtering** predicts **missing ratings** for **User 3** using **Table 2.1** and its **mean-centered form (Table 2.2)**.

1. Problem Setup

- User 3 **has missing ratings for Item 1 and Item 6**.
- We need to **predict these missing ratings** using **item-based collaborative filtering**.

2. Compute Adjusted Cosine Similarity Between Items

- Similarity between **items** is computed after **mean-centering**.
- The **mean-centered ratings matrix** is given in **Table 2.2**.
- **Adjusted Cosine Similarity Formula**:

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

Example: Compute Adjusted Cosine Similarity Between Item 1 and Item 3

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

$$= \frac{(2.25) + (0.75) + (1)}{\sqrt{(2.25 + 2.25 + 1)} \times \sqrt{(2.25 + 0.25 + 1)}}$$

$$= \frac{4}{\sqrt{5.5} \times \sqrt{3.5}} = 0.912$$

- Other **item-item similarities** are computed similarly and **shown in Table 2.2**.
- Items **2 and 3 are most similar to Item 1**.
- Items **4 and 5 are most similar to Item 6**.

3. Predicting User 3's Missing Ratings

- Predictions are made by taking a **weighted average** of User 3's ratings on the **most similar items**.

Predict $\hat{r}_{31}$ (User 3's Rating for Item 1)

$$\hat{r}_{31} = \frac{(3 \times 0.735) + (3 \times 0.912)}{0.735 + 0.912}$$

$$= \frac{(2.205) + (2.736)}{1.647} = \frac{4.941}{1.647} \approx 3$$

Predict $\hat{r}_{36}$ (User 3's Rating for Item 6)

$$\hat{r}_{36} = \frac{(1 \times 0.829) + (1 \times 0.730)}{0.829 + 0.730}$$

$$= \frac{(0.829) + (0.730)}{1.559} = \frac{1.559}{1.559} = 1$$

4. Key Observations

- **Item-Based Filtering Predicts:**
  - Item 1 $\longrightarrow$ Rating 3
  - Item 6 $\longrightarrow$ Rating 1

- **Comparison with User-Based Filtering:**
  - **User-Based Prediction for Item 6** was **0.86**, which was **out of the valid range**.
  - **Item-Based Prediction for Item 6** is **1**, which is **within the allowed range**.
  - **Item-Based Filtering uses User 3's own ratings**, so predictions **align better with her past ratings**.

- **Item-Based Filtering Improves Stability:**
  - Item similarities remain **more stable over time** than user similarities.
  - This leads to **better prediction accuracy** in many cases.
  - Even though the **top-k recommended items are similar**, the **predicted ratings can differ**.

Comparing User-Based and Item-Based Methods

1. Accuracy Comparison

- **Item-Based Methods** often provide **more accurate recommendations** because they use a **user's own past ratings** to predict new ratings.
- **User-Based Methods** rely on **other users' ratings**, which might introduce **bias due to different interests**.
- Item-based filtering works well when **similar items** can be clearly identified (e.g., recommending historical movies based on past historical movies).

2. Robustness to Shilling Attacks

- **Item-Based Methods** are **more resistant to shilling attacks** (fake user profiles attempting to manipulate recommendations).
- **User-Based Methods** are **more vulnerable** to such attacks.

3. Diversity in Recommendations

- **User-Based Methods** tend to provide more **diverse recommendations** than item-based methods.
- **Diversity ensures**:
  - Users **do not receive overly similar recommendations**.
  - They discover **new and unexpected items** (serendipity).
- **Item-Based Methods** sometimes recommend **obvious choices** or items **too similar** to what the user has already consumed.

4. Explanation of Recommendations

- **Item-Based Filtering** allows for **clear explanations**, e.g.,
  - *"Because you watched X, we recommend Y."* (like Netflix does).

- **User-Based Filtering** explanations are harder:
  - Example: A histogram of **neighboring users' ratings** can be shown to explain why a movie is recommended.
  - However, these **anonymous neighbors are not personally known** to the user, reducing trust in the explanation.

5. Stability of Recommendations

- **Item-Based Recommendations are More Stable** because:
  - **Fewer items exist than users**, making item similarity calculations **more reliable**.
  - **User-Based Methods are sensitive** to new ratings, as a **few new ratings** can change similarity scores significantly.
  - **User-Based Methods require frequent updates** due to the continuous addition of new users.
  - **Item-Based Models need less frequent updates** because items are added at a much slower rate than users.