Recommender Systems

Memory-based Collaborative Filtering

Pr. El Habib Nfaoui

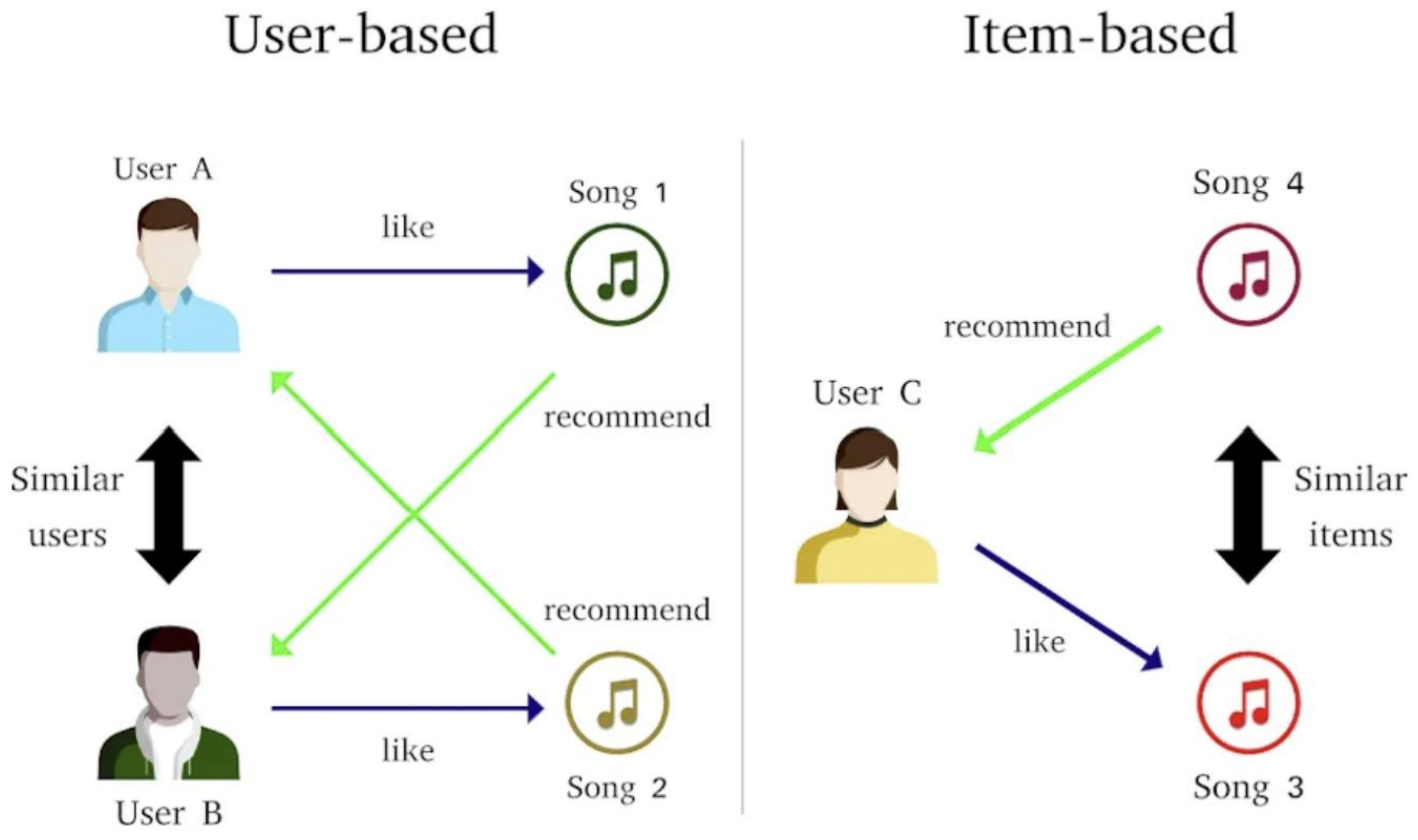
1. Memory-based Collaborative Filtering (or Neighborhood-based Collaborative Filtering)

- Memory-based approach (which is also called the K-Nearest Neighbor approach) utilizes the **entire user-item database** (the utility matrix) to **generate predictions directly**, i.e. there is **no model building**. This approach includes both **user-based** and **item-based** algorithms. In essence, such an algorithm explicitly **finds similar users or items** (which are called **neighbors**) and uses them **to predict the preference of the target user**. Specially, it uses k-nearest neighbor classifiers to predict user ratings or purchase propensity by measuring the correlation between the profile of the target user (which may be a set of item ratings or a set of items visited or purchased) and the profiles of other users in order to find users in the database with similar characteristics or preferences. Once the k nearest "neighbors" are found, predictions are made based on some **kind of aggregation of the values from these neighbors**.

- Both techniques have their advantages and use cases.

- Memory-based methods are widely used in industry (many commercial systems). They are popular because of their intuitiveness, ease of implementation, and good accuracy.

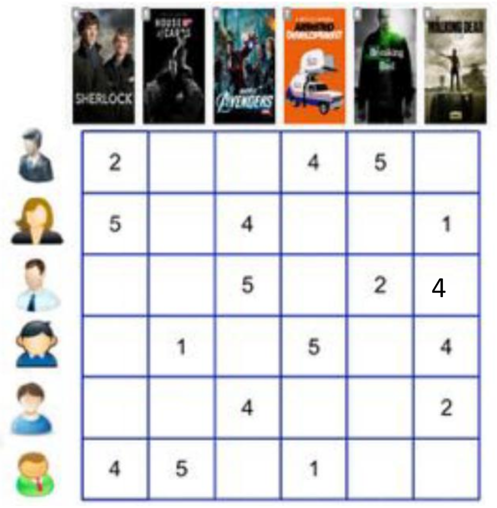
Working mechanism of memory-based approach



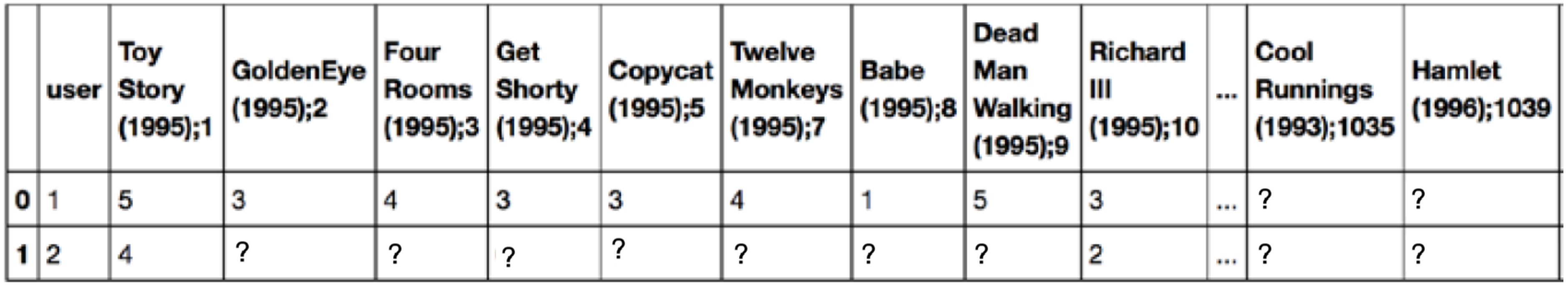
Example of a Utility Matrix

Neighborhood-based systems use the stored ratings directly in the prediction.

Utility matrix



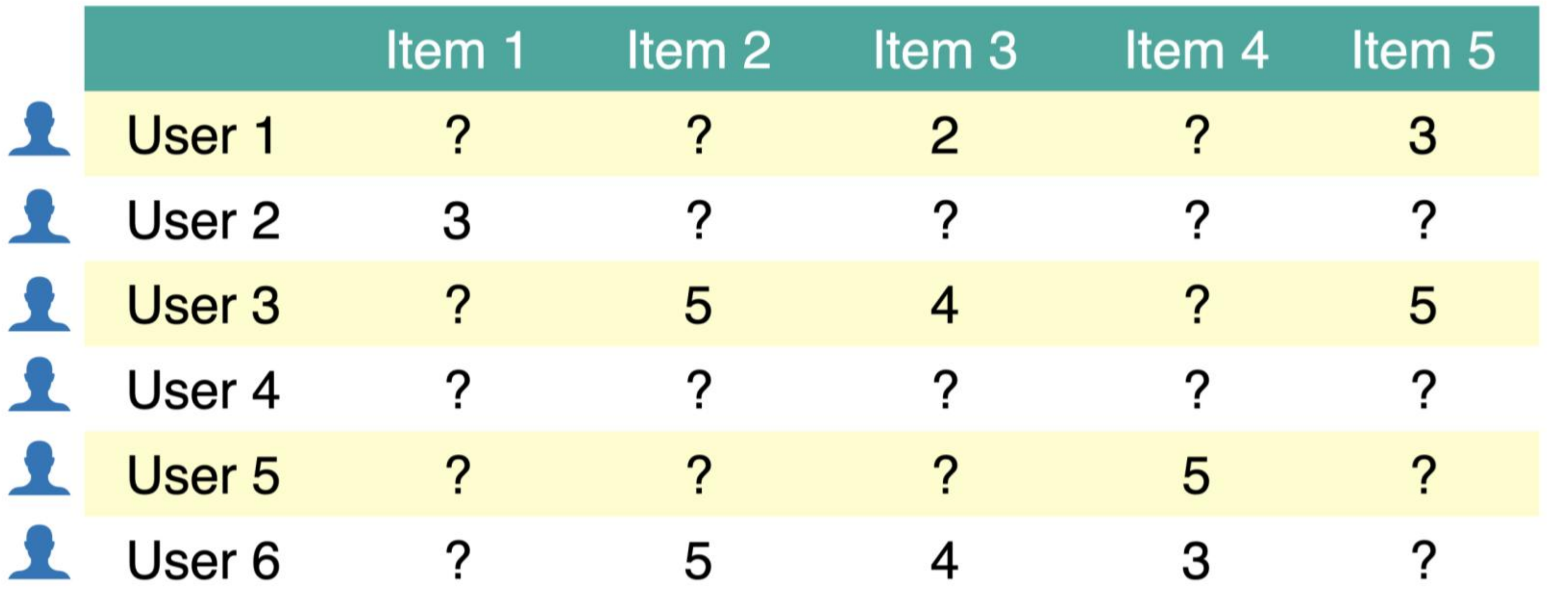
2			4	5	
5		4			1
		5		2	4
	1		5		4
		4			2
4	5		1		



	user	Toy Story (1995);1	GoldenEye (1995);2	Four Rooms (1995);3	Get Shorty (1995);4	Copycat (1995);5	Twelve Monkeys (1995);7	Babe (1995);8	Dead Man Walking (1995);9	Richard III (1995);10	...	Cool Runnings (1993);1035	Hamlet (1996);1039
0	1	5	3	4	3	3	4	1	5	3	...	?	?
1	2	4	?	?	?	?	?	?	?	2	...	?	?

What do we predict?

- Given a utility matrix of users and items, complete the utility matrix. In other words, **predict missing values in the matrix**.
- Once we have predicted ratings, we can recommend items to users they are likely to rate higher.

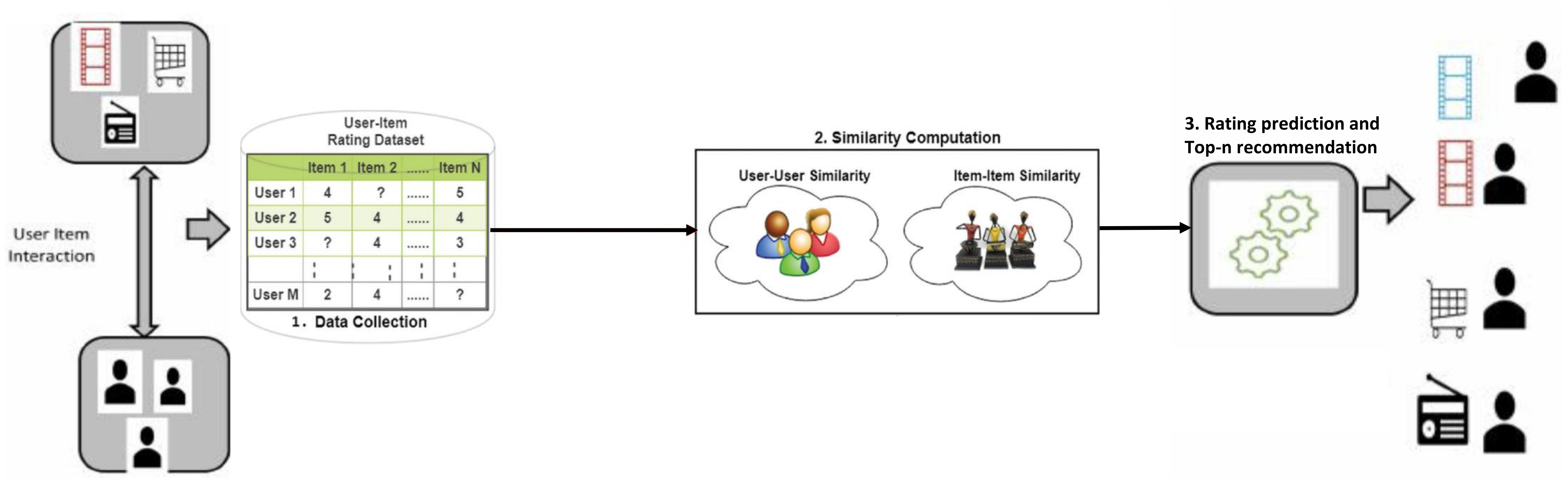


	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	?	?	2	?	3
User 2	3	?	?	?	?
User 3	?	5	4	?	5
User 4	?	?	?	?	?
User 5	?	?	?	5	?
User 6	?	5	4	3	?

2. Conceptual framework of neighborhood-based CF

- Figure below shows the conceptual framework of neighborhood-based CF. Neighborhood CF defines the closest neighbors using the following two methods:
  - User-Based CF: User similarity metric is used to find the nearest neighbors. The rating value of these neighbors and their similarity values are utilized in the prediction of unrated items of users for the formation of the Top-n list in the recommendation.

  - Item-Based CF: In the item-based CF algorithm, the nearest neighbors are determined using the similarity values of items, and these similarity values and rating values of these neighbors are used in the formation of the recommendation list to the user.



	Item 1	Item 2	......	Item N
User 1	4	?	......	5
User 2	5	4	......	4
User 3	?	4	......	3
	⋮	⋮	⋮	⋮
User M	2	4	......	?

A conceptual framework for neighborhood-based collaborative filtering

3. User-based Collaborative Filtering (user-user KNN)

- User-based collaborative filtering, one of the key methods in recommender systems, works by finding users who are similar to the target user (often based on past behavior like ratings, purchases, or views) and then recommends items that these similar users have liked or interacted with in the past. **The fundamental assumption here is that if two users agreed in the past, they are likely to agree again in the future**.
- The process of user-based collaborative filtering typically involves the following steps:
  1. Calculate the similarity between **all pairs of users**. This is often done using measures such as **cosine similarity or pearson correlation coefficient (popular measures are correlation and cosine))**.
  2. For a **given user (target user)**, find the most similar users, often **called neighbors**. The number of neighbors chosen can vary based on the specific application. The options are:
     - Take account of the top-k users (k-nearest neighbors). Usually a value of k between 20 and 50 is sufficient in most applications.
     - Take account of the users who similarity is above a defined threshold. It is common **to filter out neighbors with a similarity above than a specific threshold to prevent predictions being based on very distant or negative correlations**.
  3. Predict the target user's rating for an item (that they have not yet interacted with, i.e. items not yet rated by the target user) based on the ratings given to that item by the neighbors. Predictions are made based on some kind of aggregation of the values from these neighbors. The approaches are:
     - Average rating
     - Weighted average rating, using the similarities as weights

  4. Recommend the items with the highest predicted ratings (top-rated items).

3. User-based Collaborative Filtering

- A typical **user-based kNN collaborative filtering method** consists of two primary phases: the **neighborhood formation phase** and **the recommendation phase**.

a. The neighborhood formation

In the **neighborhood formation** phase, it compares the activity record of a target user (also called a visitor) with the historical records T of other users in order to find the top k users who have similar tastes or interests. The mapping of a visitor record to its neighborhood **could be based on similarity in ratings of items, access to similar content or pages, or purchase of similar items**. **In most typical collaborative filtering applications, the user records or profiles are a set of ratings for a subset of items**. Let the record (or profile) of the target user be **u** (represented as a vector) and the record of another user be **v** (v ∈ T). The top k most similar records to **u** are considered to be the neighborhood of **u**. For simplicity, we will also use **u** and **v** to denote the two users. The similarity between the target user, **u**, and a neighbor, **v**, can be calculated using **Pearson's correlation coefficient below:**

P.S. Notice that the user profile concept involves a lot of information: user attributes, ratings of items, pages or content access, purchase of items, films watched, etc.
A subpart of the information used in a recommender system (Content-based or CF) depends on its type.

Pearson's correlation coefficient: refresher

$$sim(\mathbf{u},\mathbf{v}) = \frac{\sum_{i\in C}(r_{\mathbf{u},i}-\bar{r}_{\mathbf{u}})(r_{\mathbf{v},i}-\bar{r}_{\mathbf{v}})}{\sqrt{\sum_{i\in C}(r_{\mathbf{u},i}-\bar{r}_{\mathbf{u}})^2}\sqrt{\sum_{i\in C}(r_{\mathbf{v},i}-\bar{r}_{\mathbf{v}})^2}}$$

where C is the set of items that are co-rated by users **u** and **v** (i.e., items that have been rated by both of them), $r_{\mathbf{u},i}$ and $r_{\mathbf{v},i}$ are the ratings (or weights) given to item i by the target user **u** and a possible neighbor **v** respectively, and $\bar{r}_{\mathbf{u}}$ and $\bar{r}_{\mathbf{v}}$ are the average ratings (or weights) of **u** and **v** respectively.

Note: we take only the ratings for the set of items, rated by both users (also to compute the average ratings)

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

where

- n is sample size
- x_i, y_i are the individual sample points indexed with *i*
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the sample mean); and analogously for $\bar{y}$.

- The Pearson correlation coefficient measures the linear relationship between two datasets. Like other correlation coefficients, this one varies between **-1 and +1** with **0 implying no correlation**. Correlations of **-1 or +1** imply an **exact linear relationship**. **Positive correlations** imply that **as x increases**, so **does y. Negative correlations** imply that as **x increases**, **y decreases**.

- Once similarities are calculated, the most similar users are selected. The number of neighbors ***K*** is an input parameter (**hyperparameter**), usually a value **between 20 and 50** is sufficient in most applications.
- It is also common **to filter out neighbors with a similarity of less than a specific threshold to prevent predictions being based on very distant or negative correlations**.

b. The recommendation phase

Once the most similar user transactions (neighbors) are identified, **the recommendation phase** can use the following formula to compute the rating prediction of item *i* for target user **u** (there are other possible formulas in the literature for this purpose):

$$p(\mathbf{u},i) = \bar{r}_{\mathbf{u}} + \frac{\sum_{\mathbf{v}\in V} sim(\mathbf{u},\mathbf{v}) \times (r_{\mathbf{v},i} - \bar{r}_{\mathbf{v}})}{\sum_{\mathbf{v}\in V} |sim(\mathbf{u},\mathbf{v})|}$$

The k-nearest-neighbors (k-NN) of u are the k users v with the highest similarity sim(u,v) to u. However, **only the users who have rated item i can be used** in the prediction of p(u,i), and we instead consider the **k users most similar to u that have rated i**. We can write this set of neighbors as $N_i(u)$ (i.e. V in this equation)

Where *V* is the set of *k* similar users (i.e. **k** most similar users **that have rated item i)**

$r_{\mathbf{v},i}$ is the rating of user **v** given to item *i*,

$\bar{r}_{\mathbf{u}}$ and $\bar{r}_{\mathbf{v}}$ are the average ratings of user **u** and **v** respectively to compensate for subjective judgment (some users are generous and some are picky). i.e., $\bar{r}_{\mathbf{u}}$ is the mean of all ratings given by user u. $\bar{r}_{\mathbf{v}}$ is the mean of all ratings given by user v.

sim(**u**, **v**) is the Pearson correlation described above. The formula basically computes the preference of all the neighbors weighted by their similarity and then adds this to the target user's average rating. The idea is that different users may have different "baselines" around which their ratings are distributed. Once the ratings are predicted, we simply choose those highly rated items to recommend to the user.

In the denominator, |*sim(u,v)*| is used instead of *sim(u,v)* because negative weights can produce ratings outside the allowed range.

b. The recommendation phase (continued)

- The formula basically computes the preference of all the neighbors weighted by their similarity and then adds this to the target user's average rating. The idea is that different users may have different "baselines" around which their ratings are distributed.

- Once the ratings are predicted, we simply choose **those highly rated items to recommend to the user**.

- Note that in case the utility matrix is **so sparse that no neighbors are found,** the mean rating of the user is predicted. It may happen that the predicted rating is beyond 5 or below 1, so in such situations the predicted rating is set to 5 or 1 respectively.

Algorithm

As we explained before, the approach uses a k-NN method to find the users whose past ratings are similar to the ratings of the chosen user so that their ratings can be combined in a weighted average to return the current user's missing ratings.

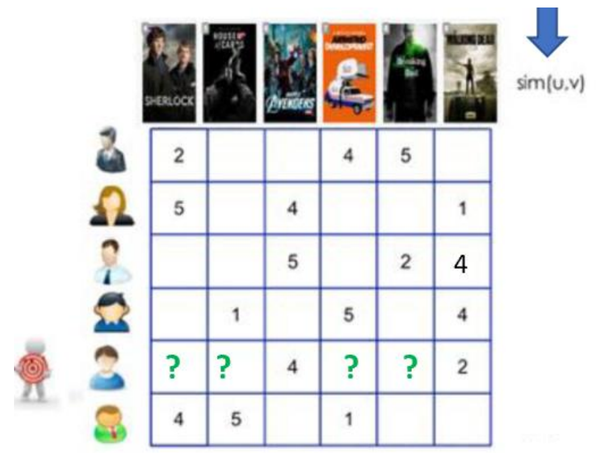
The basic logic behind user-based CF is summarized in the following algorithm:

For any given *user u (target user)* and *item i* not yet rated by this target user:

1. Find the ***K*** most similar users (nearest neighbors) **that have rated *i*** using a similarity metric ***s***.
2. Calculate the predicted rating for item *i* **not yet rated by *u*** as a weighted average over the ratings of the users *K*.
3. Recommend the top-rated k items.

The k-nearest-neighbors (k-NN) of u are the k users v with the highest similarity sim(u,v) to u. However, only the users who have rated item i **can be used** in the prediction of p(u,i), and we instead consider the k users most similar to u that have rated i. We can write this set of neighbors as $N_i(u)$ (i.e. V in this equation)

Utility matrix



2			4	5	
5		4			1
		5		2	4
	1		5		4
?	?	4	?	?	2
4	5		1		

Target user u (also called a **visitor**)

Example: Measuring User Similarity and Making Predictions

	Item1	Item2	Item3	Item4	Item5
User0	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0.85
sim = 0.70
sim = 0.00
sim = -0.79

Pearson's correlation coefficient between User0 and all other users

- To make a prediction for Item5, we first decide which of the neighbors' ratings we consider.
- In our this example, we take User1 and User2 as peer users to predict User0's rating.
- Hence the prediction for User0's rating for Item5 based on the ratings of nearest neighbors User1 and User2 will be:

$$p(\text{User0, Item5}) = 4 + \left(\left(0.85 * (3-2.4) + 0.70 * (5-3.8) \right) / \left(0.85 + 0.70 \right) \right) = \mathbf{4.87}$$

	Item1	Item2	Item3	Item4	Item5
User0	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

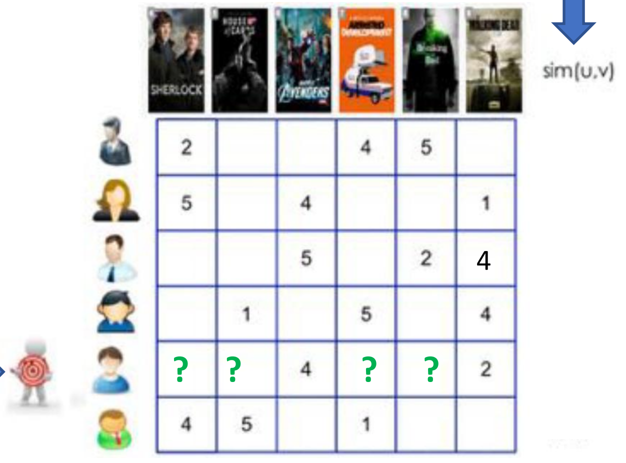
sim = 0.85
sim = 0.70
sim = 0.00
sim = -0.79

User-based CF: Exercise

Similarity between the target user u and user v

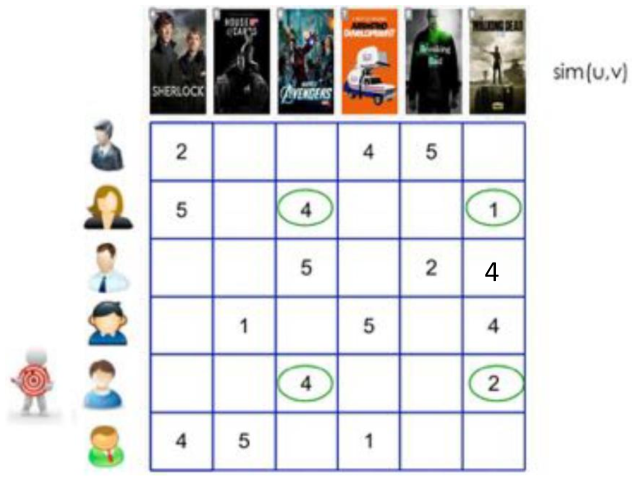
Utility matrix

Target user u (also called a **visitor**)

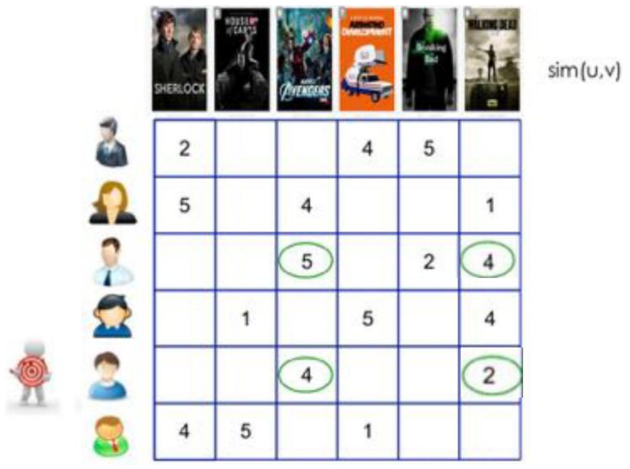


						sim(u,v)
2			4	5		
5		4			1	
		5		2	4	
	1		5		4	
?	?	4	?	?	2	
4	5		1			

Neighborhood formation phase:

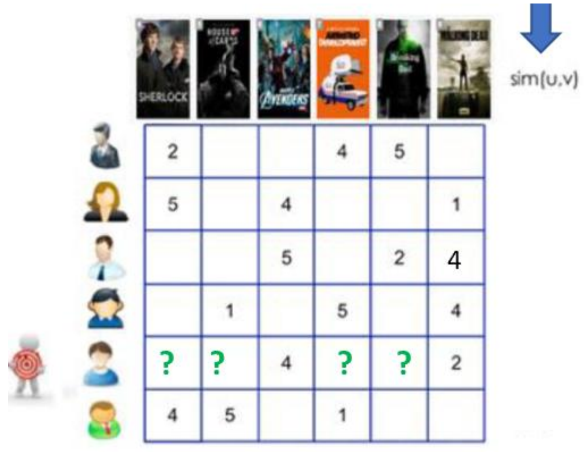


						sim(u,v)
2			4	5		
5		4			1	
		5		2	4	
	1		5		4	
		4			2	
4	5		1			



						sim(u,v)
2			4	5		
5		4			1	
		5		2	4	
	1		5		4	
		4			2	
4	5		1			

Recommendation phase:



						sim(u,v)
2			4	5		
5		4			1	
		5		2	4	
	1		5		4	
?	?	4	?	?	2	
4	5		1			

4. Evaluation of the recommendation systems

- The evaluation of a recommendation system can be executed:
  - **Offline** (using only the data in the utility matrix)
  - **or Online (using the utility matrix data and the new data provided in real time by each user using the website)**.

- Two **offline** tests often used to evaluate recommendation systems:
  - Root Mean Square Error on ratings (RMSE)
  - and ranking accuracy.

- For all the evaluations in which k-fold cross-validation is applicable, a k-fold cross-validation can be performed to obtain more objective results. The utility matrix will then be divided into k folds.

4.1 Evaluation with Numerical Ratings

- Popular Evaluation Measures
  - for **numerical ratings** – e.g., on a Likert scale between 1 and 5
    - - Mean Absolute Error (MAE)
    - - Root Mean Squared Error (RMSE)

- **Mean Absolute Error (*MAE*) computes the deviation between predicted ratings and actual ratings**

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | p_i - r_i |$$

- **Root Mean Square Error (*RMSE*) is similar to *MAE*, but places more emphasis on larger deviation**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (p_i - r_i)^2}$$

Example: MAE versus RMSE

Nr.	UserID	MovieID	Rating (r_i)	Prediction (p_i)	$\lvert p_i-r_i \rvert$	$(p_i-r_i)^2$
1	1	134	5	4.5	0.5	0.25
2	1	238	4	5	1	1
3	1	312	5	5	0	0
4	2	134	3	5	2	4
5	2	767	5	4.5	0.5	0.25
6	3	68	4	4.1	0.1	0.01
7	3	212	4	3.9	0.1	0.01
8	3	238	3	3	0	0
9	4	68	4	4.2	0.2	0.04
10	4	112	5	4.8	0.2	0.04

MAE = 0.46
RMSE = 0.75

emphasis on larger deviation

Root mean square error (RMSE) evaluation

- Given each rating r_{ij} in in the validation set (u_vals), the predicted rating $\hat{r}_{ij}$ is calculated using a recommendation method, the root mean square error is obtained:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i,j \in u\_vals} \left(r_{ij} - \hat{r}_{ij}\right)^2}{N_{val}}}$$

Here, N_{val} is the number of ratings in the `u_vals` vectors. The presence of the square factor in this formula highly penalizes the large errors, so the methods with low RMSE (best values) are characterized by small errors spread over all the predicted ratings instead of large errors on few ratings, like the mean absolute error MAE= $\frac{\sum_{i,j \in u\_vals} \left| r_{ij} - \hat{r}_{ij} \right|}{N_{val}}$ would prefer.

5. Memory-based Collaborative Filtering : Pros & Cons

- **Pros:**
  - ✓ Well-understood, works well in some domains
  - ✓ Works for any kind of item
    - No feature selection needed
  - ✓ Requires no explicit item descriptions or demographic user profiles

- **Cons:**
  - Requires user community to give enough ratings (many real-world systems thus employ implicit ratings)
  - No exploitation of other sources of recommendation knowledge (demographic data, item descriptions)
  - **Cold Start Problem**

The cold start problem occurs with new users or items that lack sufficient data for profiling.

  - how to recommend new items?
  - what to recommend to new users?
  - Need enough users in the system to find a match
- **First rater:**
  - Cannot recommend an item that has not been previously rated
- **Sparsity:**
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items
- **Popularity bias:**
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items
- **Approaches for dealing with the Cold Start Problem**
  - ask/force users to rate a set of items
  - use another method or combination of methods (e.g., content-based, demographic or simply non-personalized, hybrid recommendation) until enough ratings are collected.