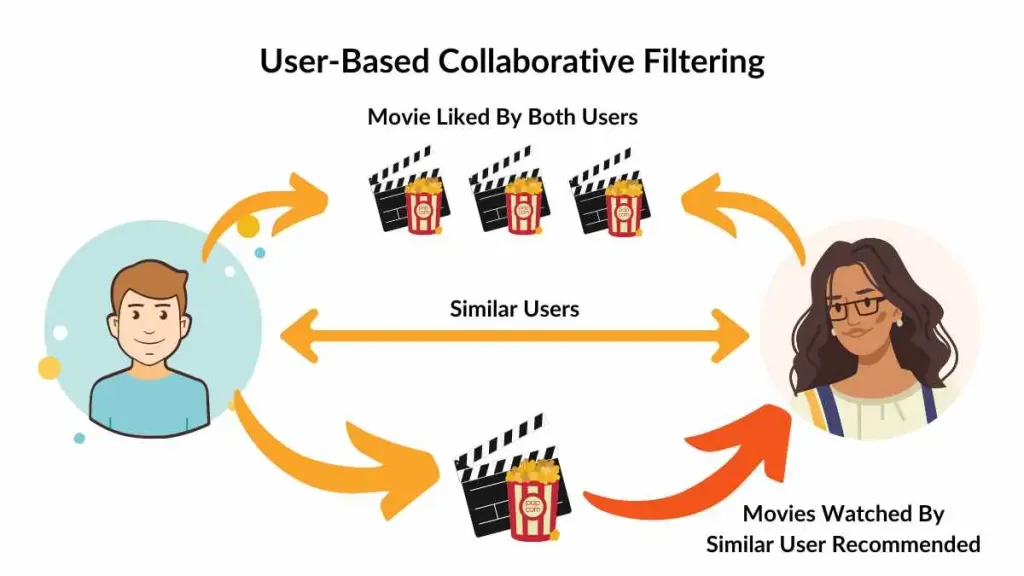
**ALGORITHM AVENGERS**

**DIY-Recommendation-Collaborative Matrix**

**User-Based Collaborative Filtering:**



User based collaborative filter is a very important filtering technique majorly used in recommendation systems.

User based Collaborative Filtering refers to used to predict the items that a user might like on the basis of ratings given to that item by other users who have similar taste with that of the target user.

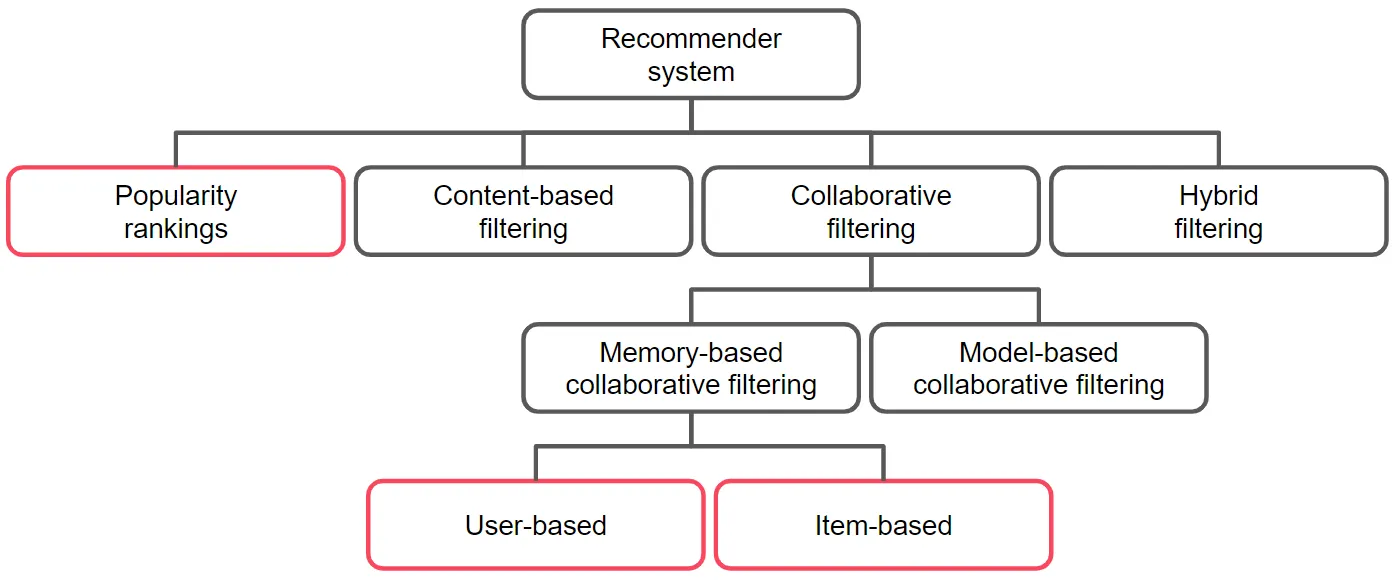
User-based collaborative filtering is a technique used to predict the items that a user might like on the basis of ratings given to that item by other users who have similar taste with that of the target user.

User based Collaborative Filtering building process :-

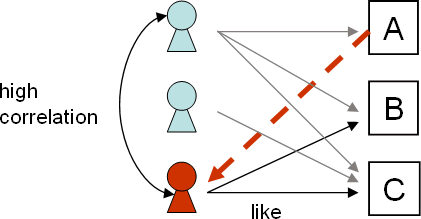
**Item Similarity**:The algorithm computes the similarity between items (e.g., movies or products) based on user ratings or interactions. It uses measures like **cosine similarity**, **Pearson correlation**, or **Jaccard index** to find items that are often rated similarly by the same set of users.

**Recommendations**:Once item similarity is established, if a user likes an item (e.g., a specific movie), the system will recommend other similar items based on those similarities. For example, if you like a particular movie, it will suggest other movies that are often liked by users who liked that movie.

Overview of the main topic:



Basic Example of using User-Based Collaborative Filtering:



* User 1 has watched the Comedies A, B, C, D and rated it according to her interests.
* User 2 has most probably also watched inter alia the same comedies as well and provided his preferences in the rating system.
* User 3 is browsing for recommendations and based on the history of their choices, the system matches this user’s ratings against any other users’ and finds the people with most “similar” tastes i.e. in this scenario User 1 and User 2.
* Accordingly, the system recommends Comedy D to User 3 as they are most likely to watch and like it.

**Working :-**

* Finding the similarity of users to the target user U. Similarity for any two users ‘a’ and ‘b’

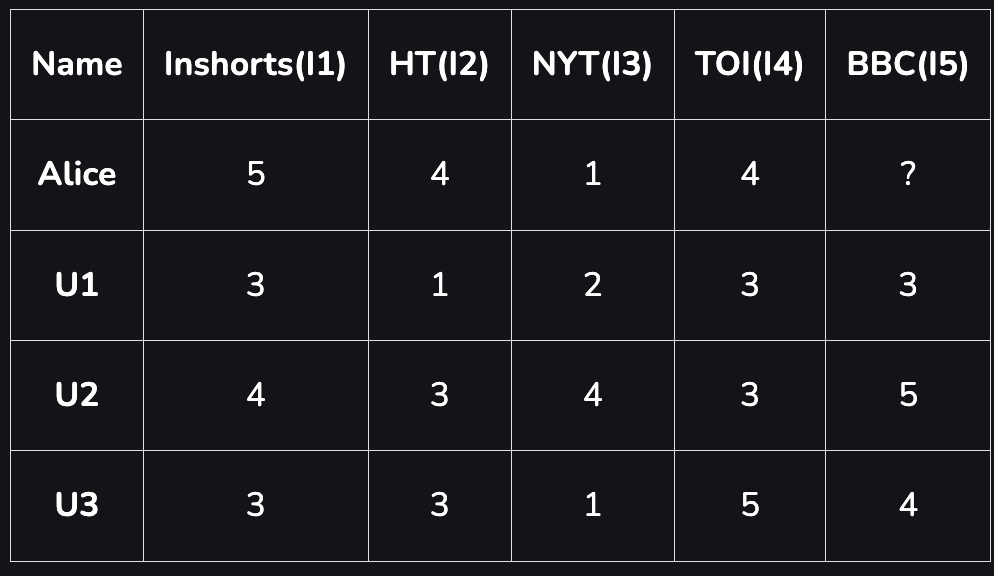


* Prediction of missing rating of an item Now, the target user might be very similar to some users and may not be much similar to others.

Hence, the ratings given to a particular item by the more similar users should be given more weightage than those given by less similar users and so on. This problem can be solved by using a weighted average approach. In this approach, you multiply the rating of each user with a similarity factor calculated using the above mention formula. The missing rating can be calculated as



Example: Consider a matrix that shows four users Alice, U1, U2 and U3 rating on different news apps. The rating range is from 1 to 5 on the basis of users



* Step 1: Calculating the similarity between Alice and all the other users At first we calculate the averages of the ratings of all the user excluding I5 as it is not rated by Alice.

The average : 

Averages for all 



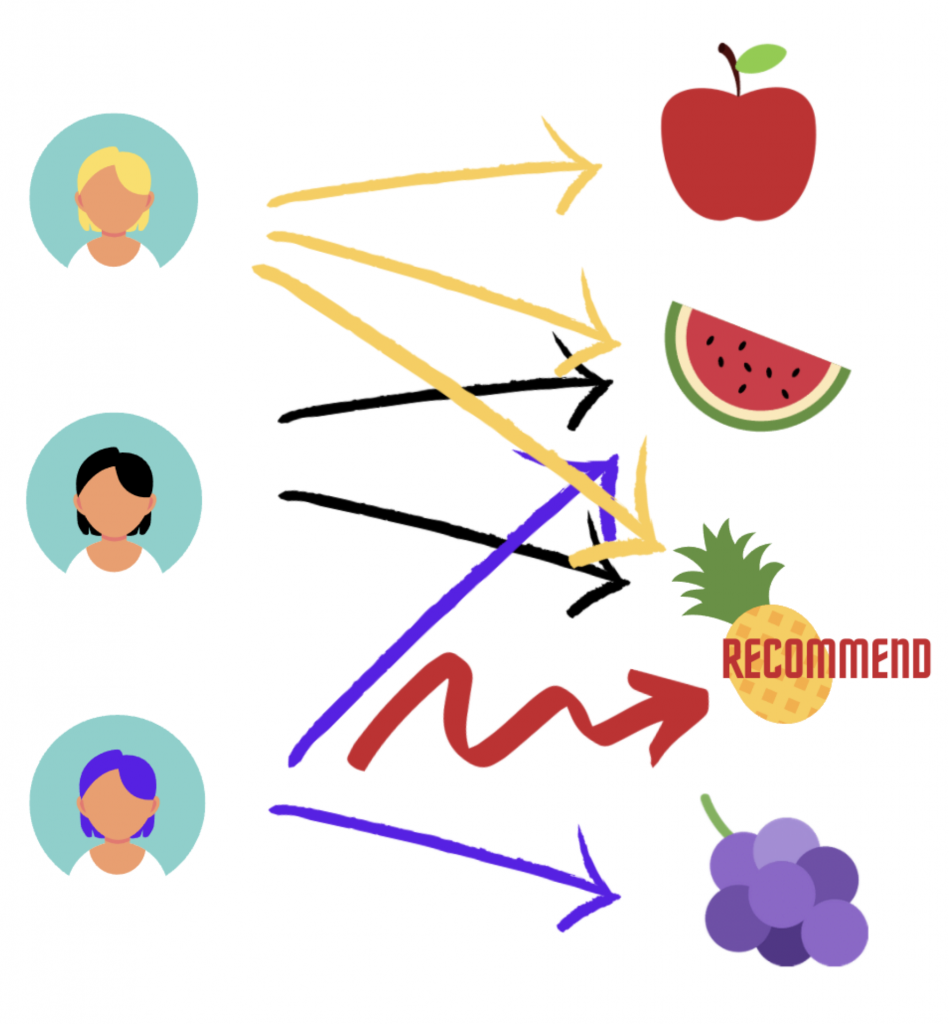
* Calculating similarities of alice with other users



* Predicting the rating of the app not rated by Alice Now, we predict Alice’s rating for BBC

News App,

**Item-Based Collaborative Filtering:**

****

Item-based collaborative filtering is a recommender system technique that suggests new items to a user based on how they interacted with similar items

It's a model-based algorithm that calculates the similarity between items in a dataset, and then uses those similarities to predict how a user will rate items they haven't interacted with yet.

1. Calculate item similarities The algorithm calculates how similar items are to each other based on how users have rated them. For example, in a movie recommendation system, the algorithm might calculate how similar two movies are based on how users have rated them.

2. Predict user ratings The algorithm uses the similarities between items to predict how a user will rate items they haven't interacted with yet. For example, if a user has rated movies A and B highly, and other users who rated those movies highly also rated movie C highly, the system might recommend movie C to the user. Item-based collaborative filtering is different from user-based collaborative filtering, which measures the similarity between users.

3.Item-based collaborative filtering is used by companies like Amazon and various movie recommendation sites.

Item VS User based Collaborative Filtering

| Item based Collaborative Filtering | User based Collaborative Filtering |
| --- | --- |
| Item similarity | User similarity |
| Between items based on user interactions | Between users based on item interactions |
| More scalable (fewer items than users) | Less scalable (especially with many users) |
| For new items | | For new users | | --- | |
| Large systems with many users and items | Systems with fewer users or frequent user interaction |

### **Neural Collaborative Filtering (NCF)**

**Neural Collaborative Filtering (NCF)** is a deep learning-based approach for building recommendation systems. Unlike traditional methods like matrix factorization, which rely on linear models, NCF leverages neural networks to capture the complex, non-linear interactions between users and items.

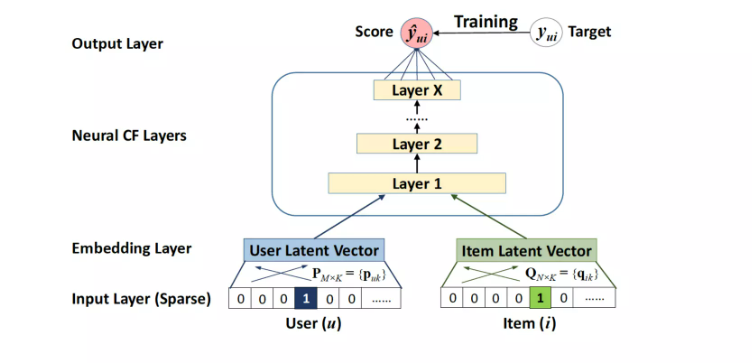
NCF enhances collaborative filtering by using deep neural networks to learn user and item embeddings, which are then combined through multiple layers to predict user preferences. This method is more flexible and expressive, as it can model complex relationships that traditional linear models may miss.

### **Key Components of NCF:**

1. **User and Item Embeddings**:
   * Similar to matrix factorization, NCF starts by learning latent factor embeddings for both users and items. Instead of using linear dot products (as in matrix factorization), these embeddings are learned through a neural network.
2. **Multi-Layer Perceptron (MLP)**:
   * NCF uses a multi-layer perceptron to model the non-linear interactions between the user and item embeddings. This neural network consists of several layers of neurons that apply activation functions to capture more intricate patterns in the data.
3. **Generalised Matrix Factorization (GMF)**:
   * NCF can be extended to include traditional matrix factorization (MF) as a special case. The **Generalised Matrix Factorization (GMF)** layer is often combined with the MLP layers to leverage both linear and non-linear interactions.
4. **Concatenation Layer**:
   * The outputs of the GMF and MLP layers are often concatenated to merge the strengths of both methods. This combined representation is then passed through a final output layer to predict the rating or preference.
5. **Loss Function**:
   * NCF typically uses **binary cross-entropy loss** for implicit feedback data (e.g., clicks or purchases), where the task is to predict whether a user interacted with an item or not. For explicit feedback data (e.g., ratings), the loss function could be **mean squared error (MSE)**.

### **NCF Architecture:**

The general architecture of an NCF model consists of the following steps:

1. **Input Layer**:
   * User and item IDs are passed as input.
2. **Embedding Layer**:
   * Separate embedding layers are used to learn dense representations of both user and item IDs. These embeddings are typically learned during the training process.
3. **GMF and MLP Layers**:
   * The **GMF** layer performs a linear dot product of the user and item embeddings.
   * The **MLP** layer consists of several dense layers, with non-linear activations like ReLU, applied to the concatenated user and item embeddings.
4. **Fusion Layer**:
   * The outputs from both the GMF and MLP branches are combined, typically through concatenation or addition.
5. **Output Layer**:
   * A final layer with a sigmoid activation is used to predict a binary label (interaction or no interaction) or a real-valued score (rating).

### **Applications of NCF:**

1. **Recommendation Systems**:
   * NCF is widely used in recommendation systems, particularly in collaborative filtering scenarios, where it can predict user-item interactions more effectively than traditional matrix factorization.
2. **Implicit Feedback Data**:
   * NCF is well-suited for handling implicit feedback data, such as clicks, views, and purchases, where binary labels (1 for interaction, 0 for no interaction) are used.
3. **Cold-Start Problem**:
   * NCF can be extended to address the cold-start problem by incorporating side information such as user demographics or item metadata, improving its ability to make recommendations for new users or items.