**Personalization using Collaborative Filtering using similar users**

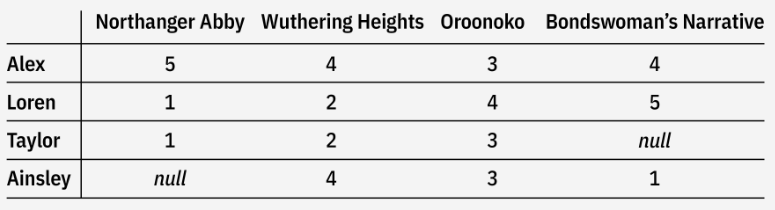
Imagine that we want to recommend a movie to our friend ***Stanley***. We could assume that similar people will have similar taste. Suppose that I and ***Stanley*** have seen the same movies, and we rated them all almost identically. But Stanley hasn’t seen *‘The Godfather: Part II’*and I did*.*If I love that movie, it sounds logical to think that he will too. With that, we have created an artificial rating based on our similarity.

**Working**

Collaborative filtering uses a matrix to map user behavior for each item in its system. The system then draws values from this matrix to plot as data points in a vector space. Various metrics then measure the distance between points as a means of calculating user-user and item-item similarity.

**User Item Matrix**

In a standard setting of collaborative filtering, we have a set of *n* users and a set of *x* items. Each user’s individual preference for each item is displayed in a user-item matrix (sometimes called a user rating matrix). Here, users are represented in rows and items in columns. In the *Rij*matrix, a given value represents the behavior of user *u*toward item *i*. These values may be continuous numbers provided by users (for example ratings) or binary values that signify whether a given user viewed or purchased the item. Here is an example user-time matrix for a bookshop website:

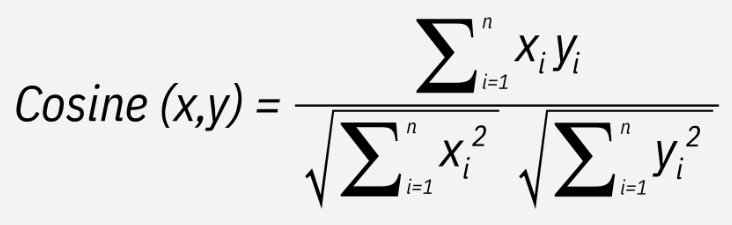


This matrix displays user ratings for different books available. A collaborative filtering algorithm compares user’s provided ratings for each book. By identifying similar users based on those ratings, it predicts ratings for books a target user has not seen—represented by *null* in the matrix—and recommend (or not recommend) those books to the target user according.  
  
The example matrix used here is full given it's restricted to four users and four items. However, in real world scenarios known users’ preferences for items are often limited, leaving the user-item matrix sparse.

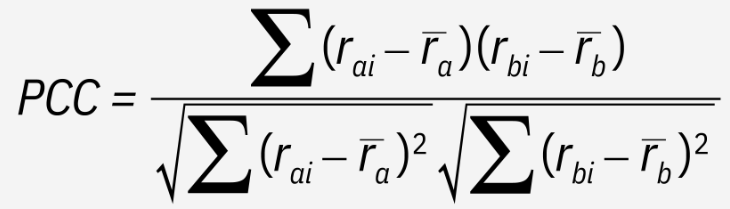
**Measuring Similarity:**

There are mainly two functions to measure similarity,

1. **Cosine Similarity:** Cosine similarity signifies the measurement of the angle between two vectors. Compared vectors comprise a subset of ratings for given user or item. The cosine similarity score can be any value between -1 and 1. The higher the cosine score, the more alike two items are considered. Some sources recommend this metric for high-dimensional feature spaces. Cosine similarity is especially useful for sparse data, as it captures relationships based on interaction patterns rather than absolute values.



1. **Pearson correlation coefficient (PCC):** PCC helps measure similarity between items or users by computing the correlation between two users’ or items’ respective ratings. PCC ranges between -1 and 1, which signify negative to identical correlation. It’s worth noting that this metric is typically used when user rating patterns are mean-adjusted because it removes any bias that might happen when different users have different rating baselines. Unlike cosine similarity, PCC uses all the ratings for a given user or item. For example, if calculating PCC between two users, we use this formula, in which *a* and *b* are different users, and *rai* and *rbi* are that user's rating for item *i*:



**Types of Collaborative Filtering:**

Collaborative filtering techniques can be broadly categorized into memory-based and model-based approaches.

* **Memory Based Approaches:** Memory-based systems can be divided into two sub-types:
  + ***User-based filtering*** recommends items to a target user based on the preferences of behaving users. The recommendation algorithm compares a target user’s past behavior to other users. Specifically, the system assigns each user a weight representing their perceived similarity with the target user—this is the target user’s neighbors. It then selects *n* users with the highest weights and computes a prediction of the target user’s behavior from a weighted average of the selected neighbors’ behavior. The system then recommends items to the target user based on this prediction. User-based similarity functions are computed between rows in the user-item matrix.
  + ***Item-based filtering*** recommends new items to a target user based on that user’s behavior toward similar items. Note, however, that in comparing items, the collaborative system does not compare item features (as in content-based filtering) but instead how users interact with those items.

For instance, in a movie recommendation system, the algorithm may identify similar movies based on correlations between all user ratings for each movie (correcting for each user’s average rating). The system will then recommend a new movie to a target user based on correlated ratings. That is, if the target user rated movie *a* and *b* highly but has not seen movie *c*, and other users who rated the former two highly also rated movie *c* highly, the system will recommend movie *c* to the target user. In this way, item-based filtering calculates item similarity through user behavior. Item-based similarity functions are computed between columns in the user-item matrix.

* **Model Based Approaches:** The model-based methods create a predictive machine learning model of the data. The model uses present values in the user-item matrix as the training dataset and produces predictions for missing values with the resultant model. Model-based methods thus use data science techniques and machine learning algorithms for suggesting items.
  + Matrix factorization is a widely discussed collaborative filtering method often classified as a type of latent factor model. As a latent factor model, matrix factorization assumes user-user or item-item similarity can be determined through a select number of features.

Some of the advantages are:

* Enables personalized recommendations without requiring item metadata.
* Identifies hidden patterns beyond direct item similarity.
* **Domain** **Independence**: Collaborative filtering doesn't depend on detailed item metadata, making it adaptable across diverse industries

Some of the challenges include:

* **Cold** **Start** **Problem**: Difficulty in recommending items to new users with limited data
* **Data** **Sparsity**: Large user-item matrices often contain many missing values
* Performance may degrade as the number of users and items increases.

**Hybrid recommender systems:** Hybrid recommendation systems combine collaborative filtering and content-based filtering to enhance accuracy and address the limitations of each approach individually. By merging user interaction patterns with item attributes, these systems provide more robust recommendations.