

## Table of Contents

• Introduction: .....	1
• Details .....	1
Content based: .....	2
Advantages .....	2
Disadvantages .....	2
Collaborative based:.....	2
• User-User filtering.....	2
• Item-Item based filtering: .....	3

### • Introduction:

In the modern world we are surrounded by electronic gadgets like phones, smart devices, IoT devices which are generating the data at a pace we have never seen before. Everyday new products/services are coming which are helping users to do their job in a better and efficient way. Companies are doing a close loop where they provide the services/products and captures the signal(data) by logging (implicit) or taking explicit feedback. Take an example of Amazon, where every day millions of sellers trying to reach out to millions of users with their products, so next question how does Amazon know which product to show to which user at what time that will increase the likelihood for the user to buy it Or How YouTube recommends the next video and it was spot on or the one you were looking for!! Or how Netflix recommends a movie to users. Its either based on user's watch history or find similar users to the user and recommend the movies they watch to the user.

In both of these examples the common thing is the recommendation to user either based on their buying history or people who bought the product the user bought and recommends the other products which these similar users bought to the user. In case of YouTube, it again based on your watch history or people how are similar to the user and watched videos are shown to the user.

According to research by McKinsey [Link2], 35 percent of what consumers purchase on Amazon and 75 percent of what they watch on Netflix come from product recommendations based on such algorithms.

As more and more users are getting on the digital footprint and producing more and more data of their interaction, it gets tougher for the users to find the right content/product on time. So either user search and go through each data point, which seems unrealistic given the kind of data we are producing or system proactively finds the best content/product for the user based on their preference and suggest, which seems to be best solution that saves the user time and effort to get the best content. We are going to see more and more personalization content/product in the near future that helps company to reach to users with right content/product.

### • Details

Now, we understand what recommendation is and how it plays a role in personalizing the content/product, let's understand under the hood. For any of the recommendation to work, we need data of user, product or user product interaction.

Traditionally there are two ways the recommendation is done:

### 1. Content based:

In this we do the feature engineering say if we have to recommend the movie to a user we will take the movie and create the different features like Who is the Actor, Director, Genre, Actress etc, Now if we know which movie a user watched, we can find the similar movie by using various similarity metrics ( Euclidian distance, Dot Product or Cosine ) and shows the top k records as a recommendation to the user. The main drawback with this is that the recommendation is as good as the features we can create. However, the advantage with this is that we can personalize the user content very accurately.

#### Advantages

The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.

The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.

#### Disadvantages

Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features. The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

### 2. Collaborative based:

Here we are not relying on the feature of the product, instead we rely on the user's response to the product. It can be collected by two means: explicitly and implicitly.

- Explicit user feedback is anything that requires user effort, like leaving a review/rating or initiating a complaint or product return (often from customer relationship management, CRM, data).
- By contrast, implicit user feedback is information that can be gathered about a user's preferences without them actually specifying those preferences. For example, past purchase history, time spent looking at certain offers, products, or content, data from social networks, etc.
- Explicit feedback can be very clear: a user has literally stated their preferences, likes, or dislikes. But by the same token, it's inherently biased; a user doesn't know what he doesn't know (in other words, he might like something but has never tried it and therefore wouldn't list it as a preference or interact with that type of item or content normally).
- By contrast, implicit feedback is the opposite — it can reveal preferences that a user didn't — or wouldn't — otherwise, admit to in a profile (or perhaps their profile information is stale). On the other hand, implicit feedback can be more complicated to interpret; just because a user spent time on a given item doesn't mean that (s)he likes it, so it's best to rely on a combination of implicit signals to determine preference.


























There are two ways to do the collaborative recommendation:

- A. **User-User filtering:** recommends items by finding similar users to the *active user* (to whom we are trying to recommend a movie). A specific application of this is the user-based Nearest Neighbor algorithm. This algorithm needs two tasks:

- Find the K-nearest neighbors (KNN) to the user **a**, using a similarity function **w** to measure the distance between each pair of users:

$$\text{Similarity}(a, i) = w(a, i), i \in K$$

- Predict the rating that user **a** will give to all items the **k** neighbors have consumed but **a** has not. We Look for the item **j** with the best predicted rating.
- In other words, we are creating a User-Item Matrix, predicting the ratings on items the active user has not see by filling the matrix with the predicted rating, based on the other similar users. This technique is **memory-based**.

					
A					
B					
C					
D					
E					

- There are different ways we can fill this matrix:
  - We can do matrix factorization methods like SVD to approx. the user-item matrix with the missing value populated. A Great video to understand the [SVD](#)

#### Advantages:

- Easy to implement.
- Context independent.
- Compared to other techniques, such as content-based, it is more accurate.

#### Disadvantages:

- Sparsity:** The percentage of people who rate items is really low.
- Scalability:** The more **K** neighbors we consider (under a certain threshold), the better my classification should be. Nevertheless, the more users there are in the system, the greater the cost of finding the nearest K neighbors will be.
- Cold-start:** New users will have no to little information about them to be compared with other users.

**B. Item-Item based filtering:** In this we focus on what items from all the options are more similar to what we know a user enjoys. This new focus is known as Item-Based Collaborative Filtering (IB-CF).

We could divide IB-CF in two sub tasks:

- Calculate similarity among the items:
  - Cosine-Based Similarity:** In this case, two items are thought of as two vectors in the m dimensional user-space. The similarity between them is measured by computing the cosine of the angle between these two vectors. Formally, in the m x n ratings matrix where m are users and n are items, similarity between items i and j, denoted by sim(i, j) is given by

$$\text{sim}(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

where “.” denotes the dot-product of the two vectors.

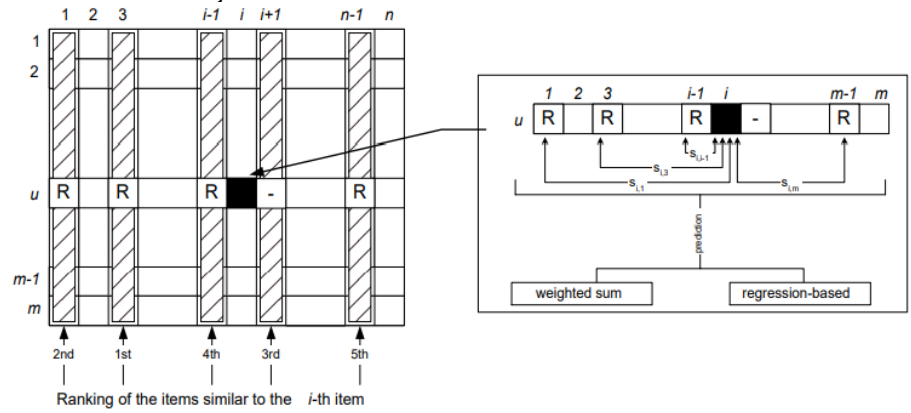
- Correlation-Based Similarity:** In this case, similarity between two items i and j is measured by computing the Pearson-r correlation corr i,j . To make the correlation computation accurate we must first isolate the co-rated cases (i.e.,

cases where the users rated both  $i$  and  $j$ ) in the  $m \times n$  ratings matrix where  $m$  are users and  $n$  are items. Let the set of users who both rated  $i$  and  $j$  are denoted by  $U$  then the correlation similarity is given by

$$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_i)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_j)^2}}.$$

Here  $R_{u,i}$  denotes the rating of user  $u$  on item  $i$ ,  $\bar{R}_i$  is the average rating of the  $i$ -th item

3. **Adjusted Cosine Similarity:** One fundamental difference between the similarity computation in user-based CF and item-based CF is that in case of user-based CF the similarity is computed along the rows of the matrix but in case of the item-based CF the similarity is computed along the columns i.e., each pair in the co-rated set corresponds to a different user (Figure 2). Computing similarity using basic cosine measure in item-based case has one important drawback—the difference in rating scale between different users are not taken into account. The adjusted cosine similarity offsets this drawback by subtracting the corresponding user average from each co-rated pair. Formally, the similarity between items  $i$  and  $j$  is



$$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_u)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_u)^2}}.$$

Here  $\bar{R}_u$  is the average

- Calculation of Prediction:
  1. **Weighted Sum:** As the name implies, this method computes the prediction on an item  $i$  for a user  $u$  by computing the sum of the ratings given by the user on the items similar to  $i$ . Each ratings is weighted by the corresponding similarity  $s_{i,j}$  between items  $i$  and  $j$ . Formally, using the notion shown in Figure shown above we can denote the prediction  $P_{u,i}$  as

$$P_{u,i} = \frac{\sum_{\text{all similar items, } N} (s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, } N} (|s_{i,N}|)}$$

Basically, this approach tries to capture how the active user rates the similar items. The weighted sum is scaled by the sum of the similarity terms to make sure the prediction is

2. **Regression:** This approach is similar to the weighted sum method but instead of directly using the ratings of similar items it uses an approximation of the ratings based on regression model. In practice, the similarities computed using cosine or correlation measures may be misleading in the sense that two rating

vectors may be distant (in Euclidean sense) yet may have very high similarity. In that case using the raw ratings of the “so called” similar item may result in poor prediction. The basic idea is to use the same formula as the weighted sum technique, but instead of using the similar item N’s “raw” ratings values  $R_{u,N}$ ’s, this model uses their approximated values  $\bar{R}_{u,N}$  based on a linear regression model. If we denote the respective vectors of the target item  $i$  and the similar item  $N$  by  $R_i$  and  $R_N$  the linear regression model can be expressed as:

$$\bar{R}_N = \alpha \bar{R}_i + \beta + \epsilon$$

The regression model parameters  $\alpha$  and  $\beta$  are determined by going over both of the rating vectors.  $\epsilon$  is the error of the regression model.

The difference between User-User- and this method is that, in this case, we directly pre-calculate the similarity between the co-rated items, skipping K-neighborhood search.

- **Conclusion:**

Recommender systems are a powerful new technology for extracting additional value for a business from its user databases. These systems help users find items they want to buy from a business. Recommender systems benefit users by enabling them to find items they like. Conversely, they help the business by generating more sales. Recommender systems are rapidly becoming a crucial tool in E-commerce on the Web. Recommender systems are being stressed by the huge volume of user data in existing corporate databases and will be stressed even more by the increasing volume of user data available on the Web. New technologies are emerging in this space which uses complex architecture of neural networks and also leveraging embedding layer to reduce the very sparse data to more dense representation.

- **Reference**

1. <https://sourcesofinsight.com/information-overload-is-not-the-problem-its-filter-failure/>
2. <https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers#>
3. <https://developers.google.com/machine-learning/recommendation/collaborative/matrix>
4. <https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f>
5. [http://glaros.dtc.umn.edu/gkhome/fetch/papers/www10\\_sarwar.pdf](http://glaros.dtc.umn.edu/gkhome/fetch/papers/www10_sarwar.pdf)