Recommender Systems

Recommender systems are systems that are designed to recommend things to the user based on many different factors. These systems predict the most likely product that the users are most likely to purchase and are of interest to.

Companies like Netflix, Amazon, Instagram, etc. use recommender systems to help their users to identify the correct product or movies for them.

Why recommender systems?

- Increase in revenue based on personalization.
- Better user experience.
- More time spent on the platform
- Help websites improve user engagement.

Recommender systems applications

- Netflix to recommend movies
- E-commerce websites to recommend the products.
- Social media platforms to recommend feeds/ blogs/news/songs…etc. Ex: Instagram, Facebook.
- Food recommendations by Zomato, Swiggy.
- Songs recommendations by Spotify and Wynk music.
- Dating apps recommending people.

Types of recommender systems

1) Apriori Algorithm
2) Content-based filtering system.
3) Collaborative-based filtering system.
4) Similarity-based filtering system.

5) Matrix factorization
6) Popularity-based recommender system.

1. Market-Basket Analysis

- Market basket analysis is used to analyze the combination of products which have been bought together.
- This is a technique used for purchases done by a customer. This identifies the pattern of frequent purchases of items by customers.
- This analysis can help to promote deals, offers, sale by the companies, and data mining techniques helps to achieve this analysis task.

Apriori Algorithm

- Apriori algorithm refers to the algorithm which is used to calculate the association rules between objects. It means how two or more objects are related to one another. we can say that the apriori algorithm is an association rule learning that analyzes that people who bought product A also bought product B.
- It is a Frequency-based algorithm. Generally, the apriori algorithm operates on a database containing a huge number of transactions.
  Ex: People who bought iPhones also bought Airpods.

Examples where the apriori algorithm can be used:

- Telecommunications
- Banking / Insurance
- Medical

- E-Commerce
- Retail

Association rules

- Association Rules are widely used to analyze retail basket or transaction data and are intended to identify strong rules discovered in transaction data using measures of patterns, based on the concept of strong rules.

- There are various metrics in place to help us understand the strength of the association between an antecedent and consequent:

- The **IF component** of an association rule is known as the antecedent. The **THEN component** is known as the consequent.

1. Support:

It is calculated to check how popular a given item is. It is measured by the proportion of transactions in which an item set appears. It is also used to measure abundance or frequency.

$$Support(x) = \frac{Number\ of\ transactions\ with\ x}{Total\ number\ of\ transactions}$$

Drawback: If ‘x’ is very popular then many items will have high support w.r.t to x.

2. Confidence:

It is calculated to check how likely item X is purchased when item Y is purchased. This is measured by the proportion of transactions with item X, in which item Y also appears.

$$Confidence(X{-}{>}\ Y) = \frac{Number\ of\ transactions\ with\ X\ and\ Y}{Number\ of\ transactions\ with\ X}$$

Drawback: If ‘y’ is very popular then it will have high confidence w.r.t to many items.

3. Lift:

It is calculated to measure how likely item Y is purchased when item X is purchased while controlling for how popular item Y is.

$$Lift(X -> Y) = \frac{Confidence\ (X -> Y)}{Support\ (Y)}$$
$$= \frac{Support\ (X \cap Y)}{Support\ (X) * Support\ (Y)}$$

- lift(X→Y) = 1 if X and Y are independent
- lift(X→Y) < 1, unlikely to be bought together.
- lift(X → Y) > 1, likely to be bought together
- Gives Bi-directional recommendation.

4. Leverage or Piatetsky-Shapiro:

Leverage computes the difference between the observed frequency of X and Y appearing together and the frequency that would be expected if X and Y were independent. A leverage value of 0 indicates independence.

$$Leverage(X -> Y)$$
$$= Support\ (X \cap Y) - Support\ (X) * Support\ (Y)$$

Though it is similar to lift, leverage is easier to interpret.
The leverage value lies in the range of -1 to +1, whereas the lift value ranges from 0 to infinity.

5. Conviction:

Conviction is another way of measuring association, although it is a bit harder to get your head around. It compares the probability that X appears without Y if they were independent of the actual frequency of the appearance of X without Y.

It can be calculated as the ratio of the expected frequency that X occurs without Y if X and Y were independent divided by the observed frequency of incorrect predictions.

A high conviction value means that the consequent is highly dependent on the antecedent.

$$Conviction(X -> Y) = \frac{\mathbf{1} - Support\ (Y)}{\mathbf{1} - Confidence\ (X -> Y)}$$

Steps to implement the apriori algorithm:

1) Create a frequency table of all the items that occur in all transactions.
2) Create a pivot matrix representing 1 if the item is present and 0 if the item is not present.
3) Encode the matrix, return 0 for all the items that have the value less than or equal to 0 and return 1 for all the values that are greater than or equal to 1.
4) Use the library mlxtend.frequent_patterns and import apriori.
5) Calculate frequent itemsets using the metric min_support.
$$min\_support = \frac{Number\ of\ times\ subset\ has\ occurred}{Number\ of\ transactions}$$
6) Use the library mlxtend.frequent_patterns and import association_rules
7) With the help of association_rules select the appropriate metric (ex: support, confidence, lift … default = confidence) to recommend the items.

Advantages:

1) It is the most simple and easy to understand and implement.

2) It doesn’t require labeled data as it is completely unsupervised and hence this can be used for many different situations as we can find unlabeled data quite often.
3) It is used to calculate large item sets.

Disadvantages:

1) Apriori algorithm is an expensive method to find support since the calculation has to pass through the whole database.
2) It is computationally expensive.
3) Complexity grows exponentially.
4) Cold start problem.

2. Content-Based recommender System

- Content-based filtering makes recommendations based on the similarity of items. It uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback.
- Content-based filtering does not require other users data during recommendation to one user.

Advantages:

1) 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.
2) No cold start problem.
3) No sparsity problem
4) The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.

Disadvantages:

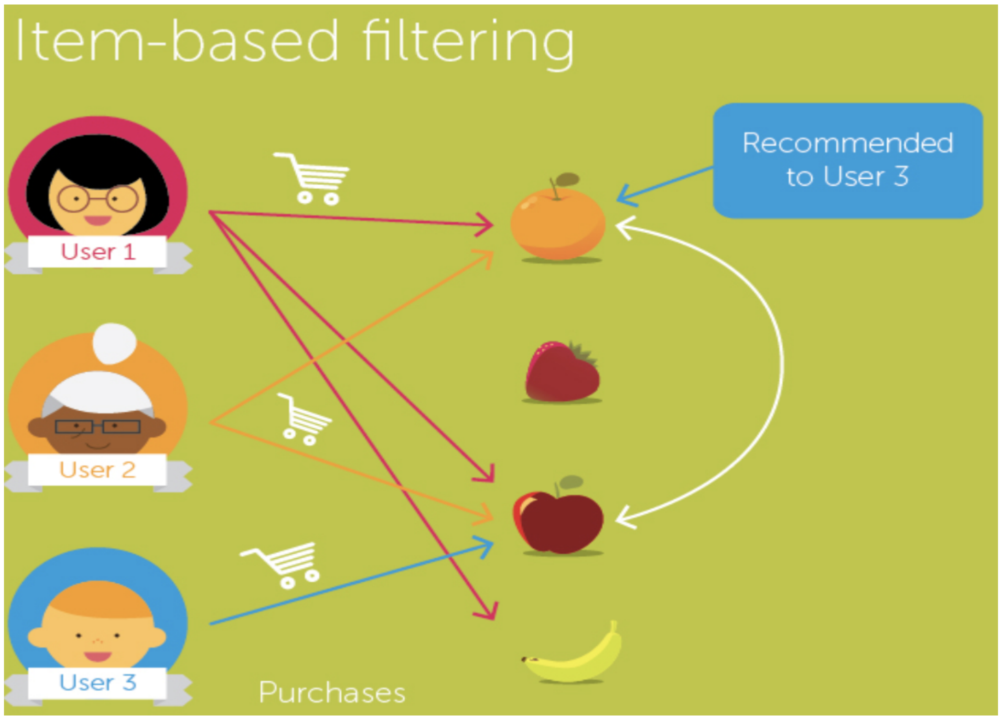
1) Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge.
2) It always recommends items related to the same categories, and never recommend anything from other categories.

3. Collaborative Filtering System

- To address some of the limitations of content-based filtering, collaborative filtering uses similarities between users and items simultaneously to provide recommendations.
- This system can filter out items that a user might like on the basis of reactions by similar users and makes recommendations based on user interactions.
- This filtering works on the assumption that people who like similar things have similar taste. There are different approaches we can adopt to implement CF recommender systems

3(a). Item - Item Based recommender System

- Item-item collaborative filtering, or item-based, or item-to-item, is a form of collaborative filtering for recommender systems based on the similarity between the items is calculated.
- “Users who liked this item also liked…”



Ex :

- User 1 bought Orange, Strawberry and Apple.
- User 2 bought Orange and Apple.
- User 3 purchases Apple so we calculate similarity between the fruits and recommend Orange.

To calculate similarity between two items, we use hamming distance.

- **Hamming Distance**: Hamming distance is a metric for comparing two binary data strings. While comparing two binary strings of equal length, Hamming distance is the number of bit positions in which the two bits are different. The Hamming distance between two strings, a and b is denoted as d(a,b).

  Ex: We have two strings in binary data

  Str1 = [0,0,1,0,0,1]

Str2 = [0,1,1,0,1,1] .

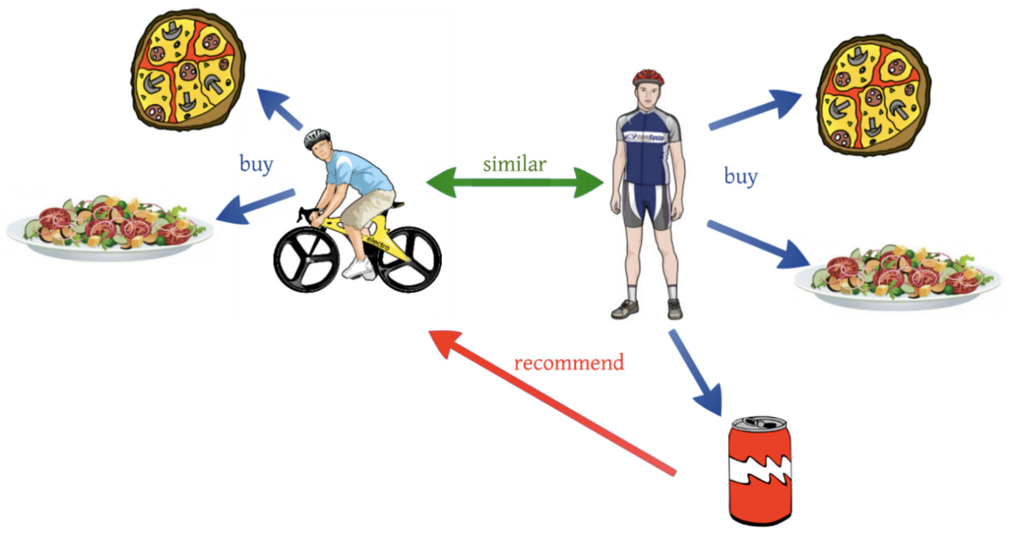
So, there are two positions in which bits are different hence the distance will be 2.

How Item-based works?

- Firstly, we compute similarities between items using hamming distance.
- Secondly, based on the computed similarities, items similar to already consumed/rated are looked at and recommended accordingly.

3(b). User - User Based recommender System

- User-user collaborative filtering, or user-based, or user-to-user, is a form of collaborative filtering for recommender systems based on the similarity between the users is calculated.
- “ Users who are similar to you also liked…”



Ex:

- There is user 1 who bought pizza, salad and soft drink and there is user 2 who bought pizza and salad. So, we calculate similarity between two users and recommend soft drink to user 2.

  To calculate similarity between two items, we use euclidean distance or cosine-similarity.

- **Euclidean Distance**: The Euclidean distance is already familiar, the Euclidean distance is calculated between two 2-dimensional vectors x = (x1, x2) and y = (y1, y2) is given by:

$$Euclidean\ dist\ = \sqrt{(x_2 -\ x_1)^2 + (y_2 - y_1\ )^2}$$

- **Cosine Similarity**: Cosine similarity is the cosine of the angle between two vectors and it is used as a distance evaluation metric between two points in the plane.
  Cosine similarity ranges from -1 to 1.
  1 indicates the items are the same whereas -1 represents the compared items are dissimilar.

$$Cosine\ similarity\ = \frac{a\ .\ b}{||a||\ .\ ||b||}$$

How User-based works?

- Firstly, we compute similarities between users using euclidean distance or cosine similarity.
- Secondly, based on the computed similarities between the users, the items of similar users are recommended.

Drawbacks:

- Data Sparsity: In case of a large number of items, the number of items a user has rated reduces to a tiny percentage making the correlation coefficient less reliable.
- User profiles change quickly and the entire system model has to be recomputed which is both time and computationally expensive.
- To overcome these drawbacks we use an item - item based recommender system.

3(c). User - Item Based recommender System

- User-Item collaborative filtering is a form of collaborative filtering for recommender systems based on the similarity between the users and items is calculated. Data contains a set of users and items and ratings/reactions in the form of a user-item interaction matrix.
- If the user-item interaction matrix is sparse it is not easy to find the users that have rated the same items.
- To deal with this sparsity problem we use matrix factorization.

4. Matrix Factorization

- Matrix factorization is a way to generate latent features when multiplying two different kinds of entities. Collaborative filtering is the application of matrix factorization to identify the relationship between items and user entities.
- With the input of users' ratings on the items, we would like to predict how the users would rate the items so the users can get the recommender based on the prediction.
- Since not every user gives ratings to all the items, there are many missing values in the matrix and it results in a sparse matrix.

- Hence, the null values not given by the users would be filled with 0 such that the filled values are provided for the multiplication.

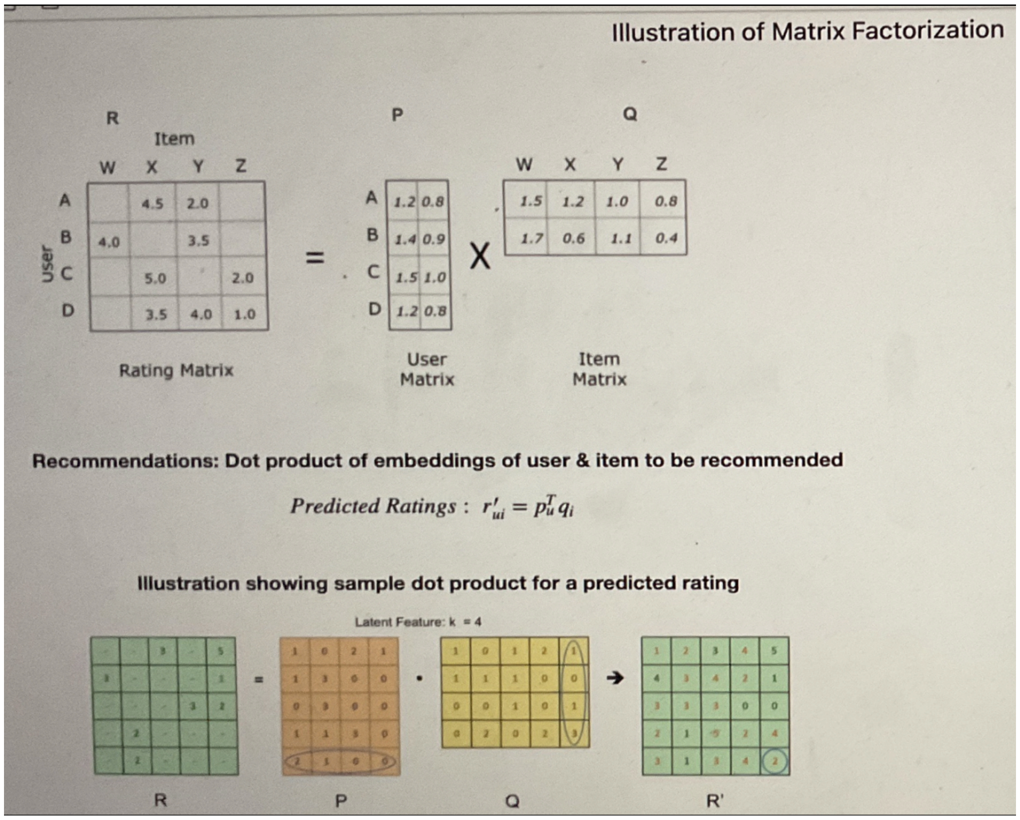


Illustration of Matrix Factorization
R
Item
W X Y Z
A B C D
User
4.5 2.0
4.0 3.5
5.0 2.0
3.5 4.0 1.0
Rating Matrix
=
P
1.2 0.8
1.4 0.9
1.5 1.0
1.2 0.8
User Matrix
X
Q
1.5 1.2 1.0 0.8
1.7 0.6 1.1 0.4
Item Matrix
Recommendations: Dot product of embeddings of user & item to be recommended
Predicted Ratings : $r'_{ui} = p_u^T q_i$
Illustration showing sample dot product for a predicted rating
Latent Feature: k = 4
R
P
Q
R'

- Matrix P represents the association between a user and the features while matrix Q represents the association between an item and the features.
- We can get the prediction of a rating of an item by the calculation of the dot product of the two vectors.
- To get two entities of both P and Q, we need to initialize the two matrices and calculate the difference and we minimize the difference through the iterations. The method is called gradient descent, aiming at finding a local minimum of the difference.

$$\min_{U,I} \sum_{i} \sum_{j} \left(R_{ij} - U_i \cdot I_j \right)^2$$

- The user-item interaction matrix is very sparse since every user does not rate all items. The missing entries in the matrix would be replaced by the dot product of the factor matrices. Therefore, we know what to recommend to the users with the unseen movies based on the prediction.

Advantages of CF:

1) No domain knowledge necessary
   - We don't need domain knowledge because the embeddings are automatically learned.
2) Serendipity
   - The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item.
3) The system doesn't need contextual features.

Disadvantages of CF:

1) Fails in case of cold start: the model requires enough of other users already in the system to find a good match.
2) Collaborative filtering models are computationally expensive.
3) It's a bit difficult to recommend items to users with unique tastes.

5. How to evaluate the performance of recommender systems?

5(a). Qualitative evaluation through metrics

- There is a common practice to keep a hold out set and evaluate the quality using human evaluators
- Serependity, Diversity, and Novelty are other attributes that are also important for a recommender system
- Serependity: Ability of the model to help users discover new interests.

  If the ML system treats the user in isolation, it may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item when it looks at aggregated preferences.

- Recommender systems that should suggest novel, relevant and unexpected items also help in diversification and popularity bias.

5(b). Quantitative evaluation through metrics

- **Precision at k:** Proportion of recommended items in the top-k set that are relevant

  Precision@k = (# of recommended items @k that are relevant) / (# of recommended items @k)

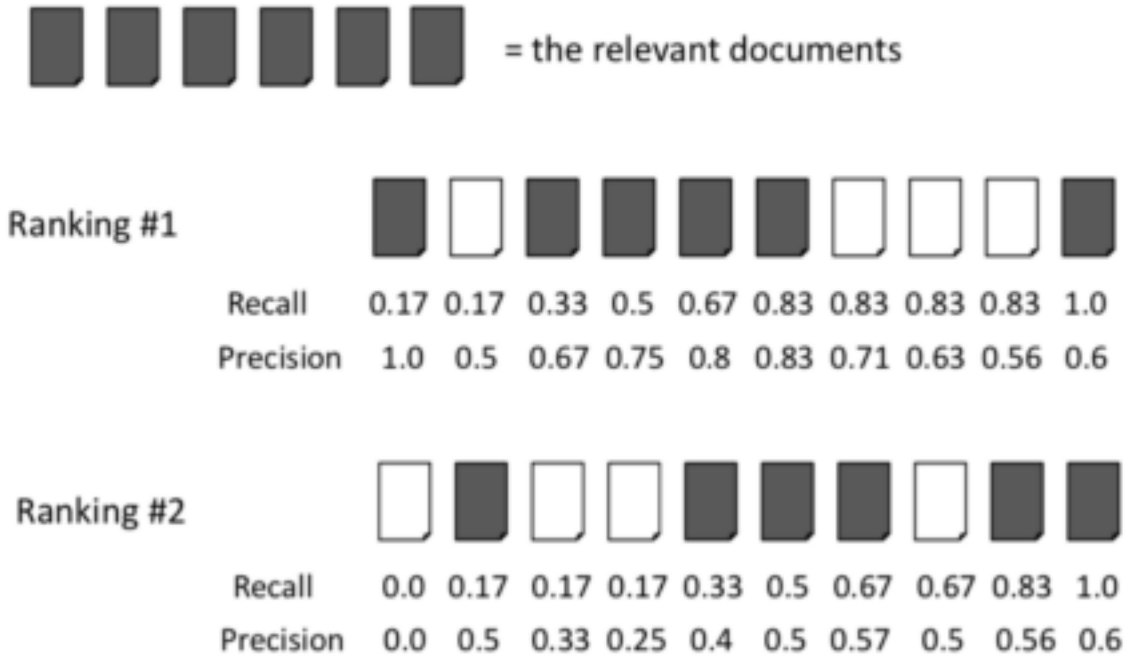
  Ex: If precision at 10 in a top-10 recommender problem is 80% - 80% of the recommender I make are relevant to the user.

- **Recall at k:** Proportion of relevant items found in the top-k recommenders

  Recall@k = (# of recommended items @k that are relevant) / (total # of relevant items)

  Ex: If recall at 10 is 40% in our top-10 recommender system. This means that 40% of the total number of relevant items appear in the top-k results.

- Illustration of precision and recall at 10 for two ranking models



Ranking #1

Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2

Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

- **Precision vs. Recall**

  Precision as a metric in recommenders optimizes the ability of how good we are in retrieving relevant items (already well rated) but too much precision lowers the ability to give unique and new recommenders (false positives).

  Optimizing for recall makes sense when the number of relevant items is less than recommended items. Here let us look at 100 recommend items & let us look at both precision and recall
- **Metrics for ranking:**

  Coverage, Hit Rate, and F1 are some other accuracy-related metrics for recommender systems MAP, DCG, NDCG, and MRR are some other metrics that also considers order (rank of items) while evaluating recommender systems: Please refer to this article:

  https://towardsdatascience.com/ranking-evaluation-metrics-for-recommender-systems 263d0a66ef54