# **Introduction**

## **Recommender System**

On the Internet, where the quantity of decisions is overpowering, there is a need to have a filter which can prioritize the information and can efficiently deliver that information to overcome the problem of overloaded information, which creates problems to the Internet users. Recommender system tackles this problem by progressively looking through substantial volume of data to provide its users with more personalized content and services. Recommender System has the ability to predict whether a particular user would prefer the item or not based on user’s profile. It is beneficial for both the users and the organization providing services. It reduces the costs of finding and selecting items in an online shopping environment, and also have proved to improve the decision making process and the quality.

## **Approaches for designing Recommendation System**

There are some popular and mostly used approaches by which we can design Recommendation Systems.

1. **Collaborative Filtering**

It is a method of making predictions to a particular user by collecting information about other users, and then finding the similar user to that user. It does not rely on themselves but how the other users has reacted to that same product. Its basic assumption is that if a user X has the same opinion as a user Y on a certain issue, then X is more likely to have the same opinion as that of Y’s on some other different issue also. Its basic principle is that if two users possesses similar interest in the past, then they would possess similar interest in the future also.

For e.g. Let’s say that we have a portal for watching movies online and giving reviews on them, and in that portal we have 4 users (Jason, Andi, Sarah, Sam), who have watched 4 movies (Inside Out, Minions, Avengers and Ant-Man) and also had given some reviews on them.

Now there is a new user named Scaz, who has watched only Inside Out movie, and had given a higher rating to it. Now he wants to see other movies also but is confused on seeing which movie. So his confusion can be solved by the recommendation system by recommending him some movie which he can watch and will like it. Here “Yes” means that a user has given higher rating to that movie, and “No” means that user has given lower rating to that movie.



Now recommendation will be done by finding the user similar to the Scaz, who possesses interest to Scaz, now as Jason and Sarah both has given higher ratings to the Inside Out movie, and Andi and Sam has not given higher rating to it. So, we find that Jason and Sarah possesses same interest as that of Scaz,



https://lh3.googleusercontent.com/resEZiaEPO2O__EW6xqsVS6KPBcSkVdsAAKMewQ3qr2tYHpfrHBO26fv4b8qZeHcNbJ_0DRx28qrK_F2S3SXIhNG7DFQDYjl-T71r1XkK3qZQGe83j2ajV8UWwsY9uWcVWPYtbc9

Now we will find the movie which have been given higher ratings by both Jason and Sarah, and that movie is Avengers, So Avengers would be recommended to Scaz.

**TYPES OF COLLABORATIVE FILTERING**

1. **Memory Based** - It is categorized into two parts, User based, and Item based. With the help of the given U - I rating matrix, a user based CF approach predicts a user’s rating on a target item by aggregating the ratings that the similar users have given previously. In contrast to the user - based CF, item based CF recommends item based on the information about the other items that a user has previously rated.
2. **Model Based -** This type uses some machine learning algorithms to train the data and make models based on them.
3. **Hybrid -** This model uses both the characteristics of the Memory Based and Model Based Collaborative Filtering.

1. **CONTENT BASED FILTERING**

This filtering method is based on the description of the items based on the user’s profile,  i.e in this approach similarity between the items are determined using their attributes only. For e.g. if a let’s say if a user X has watched a movie ABC , then the movies which are similar to it in their attributes would be recommended to that user.

1. **HYBRID**

This uses both the Collaborative and Content Based Filtering for recommendation.

## **CROSS DOMAIN**

Till now what all we have talked about is recommending items belonging to one domain. Now these techniques can be extended for recommending items of different domains also.

**NOTION OF DOMAIN**

Here domain can be either can be of two different things having some common attributes like Movies and Books, or it can be of same items, information of which are collected at two different sites.  

**LEVEL OF DOMAIN**

1. **Attribute level (Action - Fantasy)**

Same type of items, different values of certain attribute

**2.  Type level (Music - Books)**

Similar types, sharing some attributes

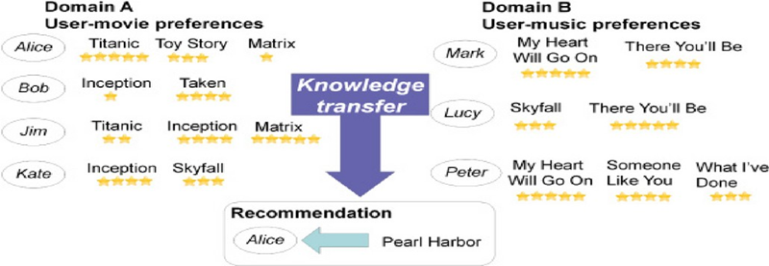
**3.   Item level (Books - Hotels)**

Distinct types, differing in most, if not all attributes

**4.  System level (IMDB - Rotten Tomatoes)**

Almost the same items, collected in different ways and/or from different operators

Here in this project we had used System level domains.

****

Now let’s take a look at an example of Cross Domain  Collaborative Filtering

Here in this example Domain A is of Movie and Domain B is of Music. Let’s say in Domain A, Alice has watched “Titanic" and has given it a 5 stars,”Toy Story” and given it a  3 stars, and “Matrix” and has given it only one star, now we would like to recommend some more movies to Alice.

So to do this first we will find the user similar to Alice from Domain A only, as we can see that there is no similar user to Alice in domain A, so will go to the other domain, Domain B and will try to find the similar user from that domain also. In that domain we find that Mark and Peter are similar to the Alice as both has given 5 stars “My Heart Will Go On” which is the title song of the movie  “Titanic”, so one of the recommendation that can be given to Alice is “Pearl Harbour” because Mark has also given 4 stars to the “There You’ll Be” which is the title song of the movie Pearl Harbour.

## **CHALLENGES OF CROSS DOMAIN**

1. **What to Share -** To decide what information is to be shared between our target  domain and source domains.
2. **How To Share -** To decide how efficiently we can transfer the information between  domains.

## **DATASET USED**

1. Movie Lens dataset has over 10000 movies listed and ratings from Rotten Tomatoes, where users have given ratings on the scale of 1-5. It also contains a User Dataset which has over 10000+ users where each user has rated at least 20 movies.
2. The Kaggle Dataset has 5000 movies which has ratings from IMDB, where users have given rating on the scale of 1-10.

## **OBJECTIVE**

Till recently the research on cross domain has only been focused on finding solution to the cold-start problem, which is to make recommendation to the new user who doesn't have any previous data related to the domain which can be used to make recommendation to him, or to tackle the cold-item problem which refers to finding the best suited user for a new item to the existing set of users who might be interested in buying that item. But our approach to cross domain also includes the scalability issue which in itself has a great scope for improvements.

Scalability is a big concern when it comes to having a lot of data considering we are using multiple domain to get the relevant data to make the best possible recommendation to the user, but with so much of data the processing speed of the server is affected a lot. This is just the current scenario, but with the ever increasing population and with more and more of people going online for even the most basic of their necessities the need for cross domain collaboration is increased and with that come the efficient management of that information, so that it can used at the right time and at the right place and put to the best use of it. That is what we are trying to do by dividing the entire data from all the domain into smaller chunks of more relevant data which would be a better way to use the processing speed of the server and even tackle the problem of increase in data in the future.

We aim to reduce the size of the data that is being processed by the server at a particular point in time because that make the process must more faster and hence the overall efficiency would also increase. But we also need to keep in mind that the quality of the recommendation should not be hampered while we are at it. So to reduce the size we are dividing the data into smaller clusters of similar nature which will be fed to the querying algorithm to make the relevant recommendation to the user.  Hence the overall efficiency of the implementation would be better and faster, and would be ready to face the increase in the data input of the database in near future.