## **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.
* Content-based recommender systems try to match users to items that are similar to what they have liked in the past. This similarity is not necessarily based on rating correlations across users but on the basis of the *attributes*  of the objects liked by the user
* At the most basic level, content-based systems are dependent on two sources of data:
  + The ﬁrst source of data is a description of various items in terms of content-centric attributes. An example of such a representation could be the text description of an item by the manufacturer.
  + The second source of data is a user proﬁle, which is generated from user feedback about various items. The user feedback might be explicit or implicit. Explicit feedback may correspond to ratings, whereas implicit feedback may correspond to user actions.
* The cold-start problem, which describes the difficulty of making recommendations when the users or the items are new.
* **How can we overcome the cold start problem?**
  + A basic idea would be to recommend the most popular / frequently bought items using a frequency-based approach. But this is a very vague approach.
* Consider the case of a new user.
  + Even though we do not have any information regarding this user's interactions with different items, we do have other additional information about this new user.
  + **Location**
    - This can be used to get an idea of the items used/purchased by other users in that area.
    - A swiggy recommender engine can make an assumption that Idli-Dosa is more probable to be liked by a user residing in Southern India.
    - We can get the location of users from IP Addresses.
    - Most platforms do ask for your location before letting you sign up.
  + **Gender**
    - Useful in recommending clothes and accessories.
  + **Age**
  + **Type of Credit Card**
    - This too can help get a lot of information about the data, like their spending habits, their credit limit, the brand of credit card, etc.
  + **The device being used to access the platform**
    - We can assume that a user using Apple Macbook would have more spending power than a user using a cheap Chinese smartphone.
* We form a new d'-dimensional vector that holds all this data, and then use **user-user similarity** on it, and recommend accordingly.
* This is known as **User-user similarity-based Content Filtering** Recommender systems.
* Consider that there is a new product on Amazon, and though there is no data about user ratings, we still have additional information like
  + **Product Description**
    - This would potentially be stored as a BoW(Bag of Words).
  + **Price**
  + **Category of product**
    - Like electronics, clothing, sports, etc…and so on.
* We form a new d-dimensional vector that holds all this data, and then use **item-item similarity** on it, and recommend accordingly.
* Hence this is called **item-item similarity-based content filtering** Recommender systems.
* We can recommend this new item to those users who bought similar items from the same categories until we have sufficient information.
  + This additional information is known as **metadata**.
* This process of finding user-user or item-item similarities, using metadata in order to recommend items to users is called a **Content-based Recommendation system**.

**Why is it called a "Content-based" Recommendation**

* Because we are not using the purchase data for finding similarities and then recommending.
* Instead, we are using **features extracted from**/provided in the **content** of the user/item to form a d-dimensional vector representing the user/item metadata.
* The point of content-based filtering is that we have to know the content of both the users and the items.

**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 is 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 recommends anything from other categories.