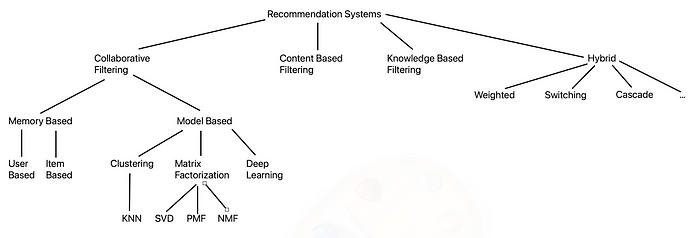
**Recommendation System**

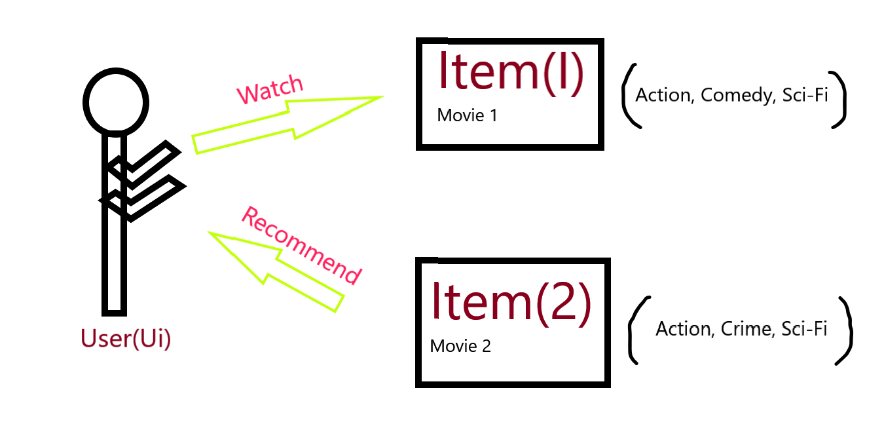
**INITIAL DRAFT**

A recommendation system is a data filtering tool that recommends the most relevant items to a particular user or a customer using machine learning algorithms.

There are three main types of recommendation systems or engines: Content-Based filtering, Collaborative filtering and Hybrid recommendation systems. We discuss each of them in details:

**Content-based filtering systems:**

Simple filtering systems that use the data of a single user and observations from the data. They uses item features to recommend other items similar to what the user likes based on user’s interactions, history, preferences and feedback. The recommendation is mainly based on a profile of the user’s choices and a description of an item. This makes it an Item-User based filtering. In order to create a user profile, the systems focus on two types of information:

1. A model of the user’s preference.
2. A history of the user’s interaction with the recommendation system.

Of course, the more information the user provides, the higher the accuracy of the system.

Content-based filtering is powerful as it does not need any data about other users since the recommendations are specific to one user. This makes it easier to scale to a large number of users.

Unfortunately, content-based filtering is not used on a large scale. This is because the model can only make recommendations based on the current user interests, which means it has limited ability to expand on the users’ interests.

There are multiple algorithms that are used in content-based filtering. The following is a brief explanation of some of those algorithms:

TF-IDF:

TF-IDF is a numerical statistic that is widely used to estimate the importance of a term within a document relative to a corpus. It has two main components, Term Frequency (TF) and Inverse Document Frequency (IDF). TF measures how frequently a term appears in a document, while IDF measures the rarity of a term across a collection of documents by penalizing common terms. TF-IDF makes recommendations based on the importance of terms within the contents of an item. If a user has shown interest in an item which contains some terms, TF-IDF recommends other items with high TF-IDF scores for those same terms. TF-IDF is simple and effective in capturing the content’s topical relevance. It does not need user ratings or explicit feedback. This is especially useful where there is a lack of user data. TF-IDF is capable of handling the varying length of documents by normalizing the term frequencies. However, TF-IDF has some limitations. Since it depends on term frequencies, it tends to overlook the semantic relationships between words. It does not correctly recognize synonyms or words with similar meaning. It also suffers from the cold-start problem. A problem where recommendations for new or unrated items are challenging due to lack of data about those items.

**Collaborative filtering systems:**

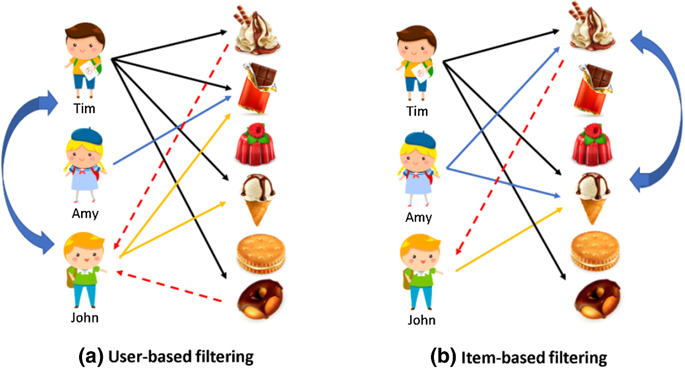
collaborative filtering systems mainly perform three things, they collect ratings or recommendations of objects, recognize patterns between users based on their ratings and generate new recommendations based on user-to-user comparisons. It is based on gathering and analyzing information about the user’s behavior, activities or preferences and predicting what they will like based on similarity with other users. Collaborative systems analyze user and/or item similarities. Similarity between users is determined by correlation. There are two main types of collaborative filtering, which are memory-based and model-based.

**Memory-Based:**

There are two types of memory-based filtering:

* **User-User Collaborative Filtering:**

Provides recommendations based on the user’s preferences that are similar to other users. In other words, it is based on searching for similar users and offering suggestions based on this similarity. This algorithm is very effective but takes a lot of time and resources.

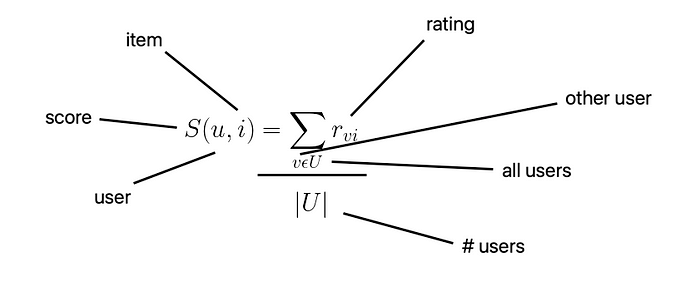


* **Item-Item Collaborative Filtering:**

It suggests items similar to other items the user liked. In other words, it is based on searching for items similar to other items users have previously liked or positively interacted with. It requires fewer resources and time compared to user-user collaboration filtering.

**Similarity:**

Typically, when someone wants to buy something, he asks his friends if they recommend it. He then takes the average of the ratings and goes ahead with the purchase if the average rating is positive. The user-item matrix (or the ratings matrix) is a matrix where each row represents a user while each column represents an item. Matrix cells contain the ratings that users have given to items. If this is to be expressed mathematically, the following formula is going to be used:



A black and white diagram

Description automatically generated with medium confidenceThe formula means that the rating for the item i for user u is equal to the sum of ratings for this item divided by the total number of users in the dataset. The formula does not take into consideration how similar one user is to another. In order to take similarity into account, we need to multiply each rating by some weight. This weight corresponds to how much another user v taste are similar to that of the user u. We then divide by the sum of the absolute value of the weights instead of the number of users in the dataset.

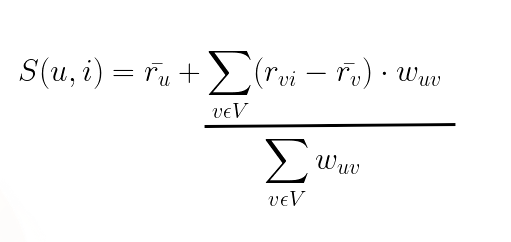
There are many other ways to measure the similarity between users’ preferences such as **cosine similarity.**

**Neighbourhoods:**

If m = |U| users and n = |I| items, the time complexity for computing pairwise correlations is O(m2n). Which makes memory-based collaborative filtering unscalable. This is because when the number of users and items grows large, calculating pairwise correlations is computationally expensive and henceforth slow. This is the reason why the neighbourhood concept is introduced. Instead of calculating for all users in the dataset, a neighbourhood is a subset of the users that is used in predicting the rating. We can choose which users should be included in the neighbourhood using many ways:

* Choose users that have a similarity score above a specific threshold.
* Choose randomly.
* Choose the top N users ranked by similarity score.
* Choose users within the same cluster.

Neighbourhood V is a collection of users similar the user we are trying o predict. The formula for calculating the rating is:



singular value decomposition, Singular Value Decomposition (SVD) & Its Application In Recommender System**Model based collaborative filtering:**

Single Value Decomposition (SVD):

SVD is used in collaborative filtering to decompose the user-item matrix into latent factors, which then can be used to make recommendations. SVD is originally dimensionality reduction technique from linear algebra. It is used in machine learning. It is a matrix factorization technique that reduces the number of features in a dataset from n dimensions to k dimensions where k < n. In the context of recommendation systems, SVD is useful in analyzing and representing the user-item matrix in a more compact form. SVD decomposes the user-item matrix into three other matrices. A is an m x n utility matrix. U is an m x r orthogonal left singular matrix. It represents the relationship between users and latent factors. S is an r x r diagonal matrix. It determines the strength of each latent factor. V is an r x n diagonal right singular matrix. It describes the similarity between items and latent factors. In this case, the latent factors are the characteristic of the items. SVD decreases the dimensions of the utility matrix A by extracting its latent factors. This is due to the mapping of each user and each item into an r-dimensional latent space, which facilitates a clear representation of relationships between users and items. Unfortunately, SVD cannot deal with missing values in the data. However, other techniques that are built on top of SVD can be used to solve this problem such as imputation, weighted SVD, SVD++, Funk SVD with biases.

A close-up of a chart

Description automatically generatedAlternating Least Squares (ALS):

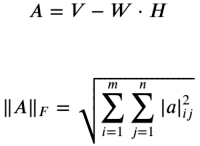
ALS is used to factorize the user-item matrix into two lower-dimensional matrices. The first one is the user matrix U while the second is the item matrix V. The two matrices capture latent features that represent the hidden patterns or characteristics of users and items. The dimensions of the latent feature space are determined by the rank of the factorization. The rank represents the number of latent features used to describe each user and item.

The main goal of ALS is to find these latent features such that the product of the user matrix U and the item matrix V approximates the original user-item matrix R. In other words, it aims to minimize the difference between the original user-item matrix R and the product of the user matrix U and the item matrix V. ALS is especially significant when the user-item matrix is sparse. ALS alternates between fixing either the user or item matrix and optimizing the other until convergence. The optimization involves a set of least-squares problems. Once reaching convergence to a satisfactory approximation, predictions for missing entries in the original matrix can be obtained by taking the dot product of the user and item vectors. Recommendation system can use those predictions to recommend items to users. For example, it may recommend items with the highest predicted rating that the user has not interacted with. ALS is highly scalable and has a good ability to handle sparce matrices.

Non-negative matrix factorization (NMF):

A black background with a black square

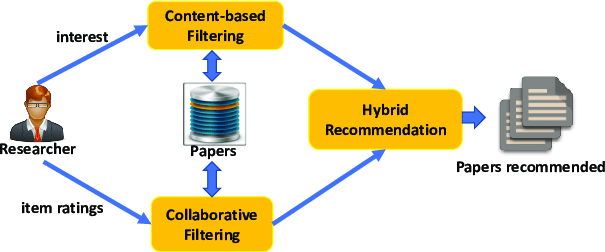
Description automatically generated with medium confidence

NMF is used to factorize the ratings matrix into two lower dimensions matrices. The first matrix is W and the second is H. The goal is to approximate the ratings matrix R by the dot product of the two matrices W and H. In other words, the goal is to minimize the difference between the original ratings matrix and the dot product of the two factor matrices W and H. All the values of R, W, and H should be greater than or equal to zero. The optimization process happens using iterative techniques such as gradient descent. When W and H converge to an acceptable approximation, their dot product can be used to make recommendations.

NMF is similar to ALS in multiple ways. The key differences are:

* NMF enforces non-negativity constraints on the factor matrices W and H. This is useful when dealing with data that should be naturally non-negative, such as user-item ratings. This makes NMF more interpretable as it allows for a clear separation of user and item characteristics.
* NMF uses iterative optimization approaches such as gradient descent to minimize the difference between the original ratings matrix and the dot product of the factor matrices W and H.
* Parallelization of NMF is more difficult than parallelizing ALS as ALS is naturally parallelizable. However, there are parallel and distributed algorithms that have been developed to address this issue.

**Hybrid Systems:**

A special type of system that combines multiple recommendation techniques such as collaborative, content-based filtering, and other approaches and uses them simultaneously. This provides more accurate and diverse recommendations to users as it utilizes the strength of different approaches. It can be used to overcome major problems such lack of data. Some hybridization techniques include:

* Weighted: It assigns different weights to recommendations from both techniques based on their respective performance.
* Switching: It chooses among recommendation techniques and applies the selected one.
* Mixed: It presents together recommendations from different recommenders to give the final recommendation.