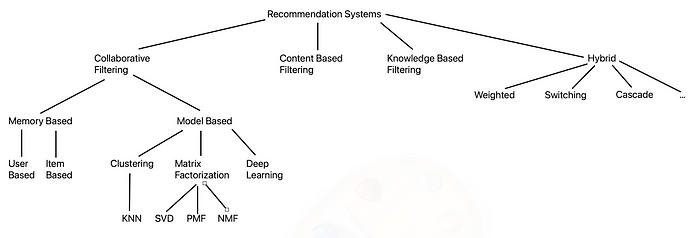
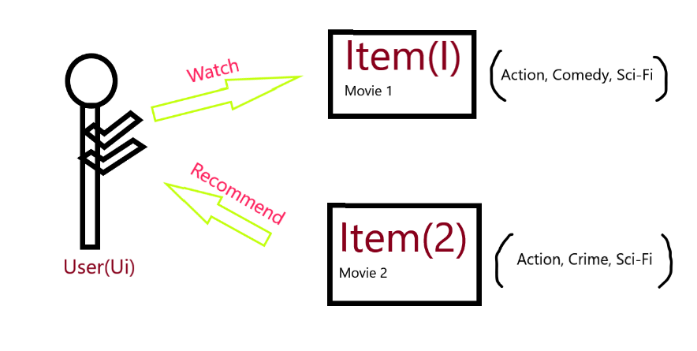
**Recommendation System**

**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 such as TF-IDF, Cosine similarity, vector space model, NLP, RNN. The following is a brief explanation of one 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:**

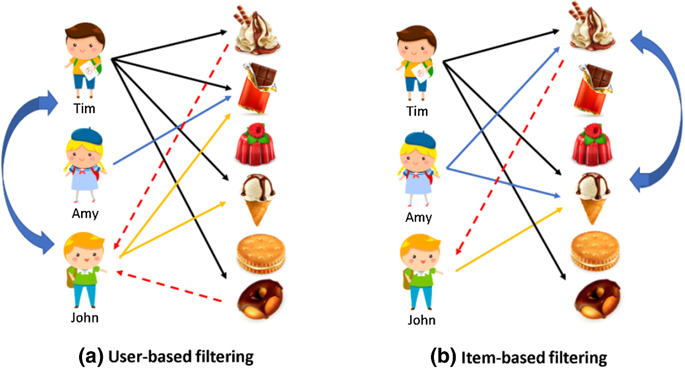
collaborative filtering systems mainly perform three things, they collect ratings or recommendations of objects, recognize patterns between users based on their ratings and generate new recommendations based on user-to-user comparisons. Collaborative systems analyze user and/or item similarities. Similarity 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:**

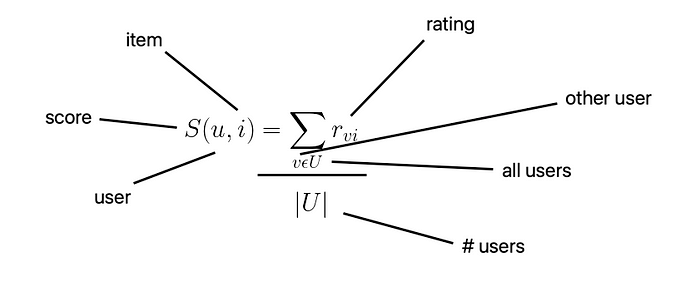
Provides recommendations based on the user’s preferences that are similar to other users. In other words, it is based on searching for similar users and offering suggestions based on this similarity. This algorithm is very effective but takes a lot of time and resources.



* **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:**

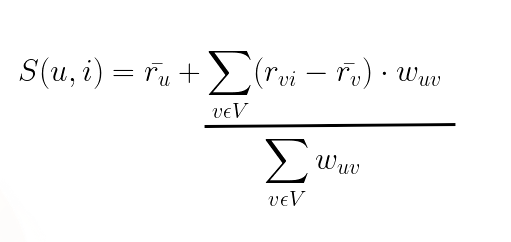
Typically, when someone wants to buy something, he asks his friends if they recommend it. He then takes the average of the ratings and goes ahead with the purchase if the average rating is positive. The user-item matrix (or the ratings matrix) is a matrix where each row represents a user while each column 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:

The 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 A black and white diagram

Description automatically generated with medium confidencesome 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 to the user we are trying to 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 a 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.

Alternating Least Squares (ALS):

A close-up of a chart

Description automatically generatedALS 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.

**Knowledge-Based Recommendation Systems (KB RS):**

A knowledge-based recommendation system is a system that make recommendations based on specific queries made by the user, not on a user’s rating history. It prompts the user to give some rules or guidelines on what the results should look like. The user may also give an example of an item. The system searches through its item database and tries to find a similar match. There are two main types of knowledge-based recommendation systems:

**3.1. Constraint-Based:**

It uses existing knowledge bases that contain explicit rules about how to relate user requirements with item features. It is very similar to an expert system in knowledge base systems. It is considered a constraint satisfaction problem.

**3.2. Case-Based:**

It depends on letting the user choose a target or an anchor item the algorithm finds a similar item to recommend. Results are usually treated as new target cases with some interactive modifications. This is similar to Content-Based systems where items similar to ones the user previously liked are suggested. The main difference is that most knowledge-based systems depend on the description of the items in the form of relational attributes in knowledge bases rather than as text keywords like in Content-Based systems.

**3.2.1. Similarity Metrics:**

Similarity metrics are needed to retrieve examples similar to the specified item. For continuous variables, similarity can be the difference between two numbers. For example, two different colleges are similar if the difference in fees is close to zero. It can also be more complex and require the use of statistical measures such as the standard deviation. When it comes to categorial variables, similarity is much more challenging. Domain hierarchies are often used in this case. Domain hierarchies are tree graphs where each category can have a parent category and subcategories. The similarity of two items can then be measured by the length of the path between the items.

**3.2.2. Critiquing:**

Once some result is found using similarity metrics and recommended to the user, the user is able to provide feedback and customizes the results to match what he is looking for. The user specifies a change request on some attributes of the item they like. The change request can be a directional critique or a replacement critique. A directional critique provides feedback on the direction in which the recommendation should be adjusted. For example, if the system recommended a college that is 100Km away, the user may request a shorter distance. On the other hand, a replacement critique suggests alternative items to replace the initially recommended item. For example, if the system recommended a medical college, the user may suggest recommending a computer science college instead.

**KB RS Strengths:**

* Complex Item Domain: KB RS is very proficient when the items have many complex aspects to consider. Those aspects often need expert knowledge in the domain. KB RS effectively captures those complex aspects by utilizing knowledge bases crafted by domain experts. This strength is especially significant when the item is of great importance and requires a lot of thought, such as a university or a college.
* Avoids Cold Start Problem: User data is not needed because what the user wants is explicitly defined. The recommendation process is accurate and can start and work well without requiring existing rating data.

**KB RS Weaknesses:**

* Knowledge Acquisition Bottleneck: A big problem lies in the creation of the knowledge base. It requires the conversion of the knowledge possessed by domain experts into consumable representations.

**Hybrid Recommendation 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 strengths and overcomes the weaknesses of different approaches. It can be used to overcome major problems such lack of data. Some hybridization techniques include:

Weighted:

A diagram of a diagram

Description automatically generatedIn the weighted hybrid recommendation system, multiple models are chosen such that they are able to interpret the dataset well. The system takes outputs from each selected model. It then combines the result by statistically assigning different weights to recommendations from chosen models based on how well each model represents the dataset.

Switching:

A diagram of a system

Description automatically generatedThe switching hybrid recommendation system chooses one among recommendation models and applies the selected one. The selection of the recommendation model is based on multiple criteria such as the user profile, the dataset and other features. Switching introduces an additional layer upon the recommendation system. This layer is responsible for selecting which model to use. The performance of the switching system is entirely dependent on the strengths and weakness of the selected recommendation model.

Mixed:

A diagram of a process

Description automatically generatedThe mixed hybrid recommendation system begins by taking both the user profile and features in order to generate different sets of candidate datasets. It then inputs each generated candidate into a selected recommendation model. Finally, it presents together recommendations from different recommenders to give the final recommendation. The mixed recommendation model has the ability to make a large number of recommendations simultaneously. It can fit each candidate dataset to a suitable model which results in a better performance.

Feature Combination:

A diagram of a system

Description automatically generatedThe feature combination hybrid recommendation system adds a virtual contributing recommendation model to a main recommendation model. The virtual model works as feature engineering towards the original user profile dataset. An example would be to inject features of a collaborative recommendation model into a content-based recommendation model. The combination is able to take into account the collaborative data along with the main content-based data.

Feature Augmentation:

A hexagon with black text

Description automatically generatedIn feature augmentation hybrid recommendation system, a contributing recommendation model is employed to generate an augmented profile. An augmented profile is basically a rating or a classification of the user-item profile. The augmented profile is taken as input by the main recommendation system to produce the final predicted result. Feature augmentation system is capable of improving the performance of the core system without needing to change the main recommendation model.

Meta-Level:

The meta-level hybrid recommendation system is similar to the feature augmentation system in that both have a contributing model that provides an augmented dataset to the main recommendation model. However, the meta-level system is different as it replaces the original dataset with an entire learned model from the contributing model as input to the main recommendation model. Whereas in the feature augmented system, the contributing model just generates some features as input for the main recommendation model.

Cascade:

A diagram of a diagram

Description automatically generatedThe cascade hybrid system is a strict hierarchical structure recommendation system. The main recommendation system produces a primary result. The secondary recommendation system takes the primary result as input to resolve minor issues that may exist such as breaking ties in the scoring. The cascade system is effective against sparse datasets since it resolves the issues of missing data and ties in scoring.

**Conclusion**

After comparing and contrasting different variations of recommendation systems, we found that the most suitable recommendation system type to utilize is the **hybrid recommendation system**. This is mainly because as illustrated earlier, it is very efficient when dealing with lack of data. Another reason is that it combines the strengths of multiple recommendation systems while covering their weaknesses. Among the types of hybrid recommendation systems, we found that mixed and cascade are the most suitable for our case.

The mixed hybrid recommendation system represents a good choice because it can fit each candidate dataset to a suitable model which results in a better performance. On the other hand, the cascade hybrid recommendation system serves as a convenient choice because it is effective against sparse datasets. It resolves the issues of missing data and ties in scoring.

Initially, we will depend heavily on a knowledge-based system. This is because of the cold start problem. We need users to specifically state which features they need in a college. After having provided enough recommendations to users and gathered enough feedback to construct a rating matrix, we will deploy other techniques such as content-based and collaborative filtering.

**\*\*todo: provide details about which recommendation system types are going to be used in mixed and cascade systems according to the chart of each\*\***

**We can use knowledge-based systems at first to solve the cold start problem then switch to other types when enough ratings are available.**

**Can content-based systems be used at the start to solve the cold start problem?**

**Make an appendix to put irrelevant parts. Only expose relevant parts in the main document.**