Introduction to Search and Recommender Systems

Chirag Shah

Recommendation Systems

Reading List:

- Thorat, P. B., Goudar, R. M., & Barve, S. (2015). Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, *110*(4), 31-36.
- Introduction to recommender systems | by Baptiste Rocca
- Recommender Systems

Information Filtering

Information filtering is a systematic approach to filter out useless/irrelevant information extracting a subset of information that a user is likely to find interesting or useful from a large stream of dynamically generated information. IF system uses various types of filtering methods such as content-based filters, collaborative filters, and hybrid filters.

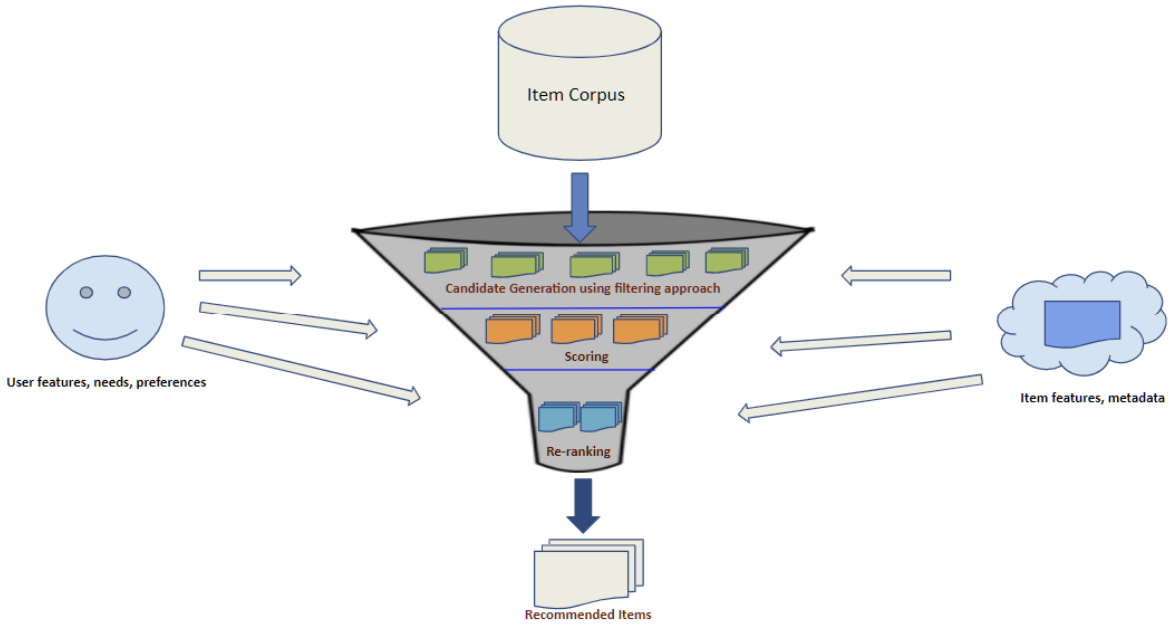
The main methodological difference between the information filtering and information retrieval is that the information retrieval process retrieves useful information from a fixed data or document set. On the other hand, information filtering selects relevant information or rejects irrelevant information from a stream of incoming data. However, both the approaches support users in finding useful information, reducing information overload and search times.

Recommender/Recommendation Systems

Recommender or recommendation system is an information filtering system which uses various filtering techniques to (1) find interesting content by filtering essential information out of a large stream of information according to user's preferences, interest, or observed behavior about the content; and (2) present the information in the form of suggestions.

Furthermore, recommendation systems utilize various prediction models to predict whether a particular user would prefer certain content or not based on the user's profile or long-term information needs, preferences and deliver recommendations. Not only does the recommender system provide users with the content they want, but it can also provide content that users have never thought of searching for themselves.

Basic Architecture of a Recommendation Systems



A recommendation system generally consists of all or a few of the following basic components, divided up in three phases: candidate generation, scoring, and re-ranking.

Phase 1: Candidate Generation (Data collection and learning phase)

The first step in creating a recommendation system is determining and generating/filtering a smaller subset of relevant candidate items for recommendations from a potentially huge corpus. The filtering technique usually generates the candidate subset(s) by quickly evaluating either the user profiles, their needs, item representations or user-item relationships. A given model may provide multiple candidate generators, each nominating a different subset of candidates. There are three most popular candidate generation approaches: collaborative filtering, content-based filtering, and a hybrid of them. These approaches use different sources to generate candidates, such as

User Profile/Features and User Query/Preference Representation (Also used in Personalization)

To generate relevant candidate items, many recommender systems rely on representing the users' interests and preferences. Recommendation systems usually collect user preference or build user profiles in two ways -- by collecting users' explicit feedback, such as user past activities, past and current ratings, reviews, and/or personal information such as gender, age and other information provided by the users, or collecting implicit feedback such as user interaction with items, clicks on an item or link, spending time on an item description, device, location, time, dates, etc. Based on various user-related characteristics, the system creates user profiles. Various vector-based embedding techniques have been used to represent users.

Item Representation

Information items also known as content or documents are the entities a system recommends (e.g., movie, song, books and so on) are the entities a system recommends. Many recommendation systems

rely on learning an appropriate embedding representation of the items and their descriptive metadata to generate candidate items to recommend. Also, various vector-based embedding techniques have been used to represent items and their metadata.

User-item Relationship Mapping: Similarity Measures for Candidate Generation

Both content-based and collaborative filtering approaches map each item and each user (user need or context) to an embedding vector in a common low-dimensional embedding space and captures some latent structure of the item or user set.

Then the recommendation system measures similarity *s* between a pair of user and item embeddings for candidate generation E x E → R as a scalar measuring their similarity. The embeddings can be used for candidate generation as follows: given a query embedding u ∈ E, the system looks for item embeddings I ∈ E that are close to *u*, that is, embeddings with high similarity *s(u ,i)*.

The following are some common techniques to measure the degree of user-item similarity for recommendation.

Cosine similarity

The cosine of the angle between user and item vectors *s(u,i)=cos(u,i)*

Dot Product

The dot product of two vectors *s(u,i) =||i|| ||u|| cos(u,i*).

Euclidean distance

The distance between user and item in Euclidean space $s(u,i) = \|u - i\| = [\sum_{x=1}^{d} (u_x - i_x)^2]^{\frac{1}{2}}$. A smaller distance means higher similarity.

Also, there are many machine learning and deep learning models to predict the similarity for a user and item such as various clustering algorithms, Matrix Factorization, deep neural networks, softmax models and so on.

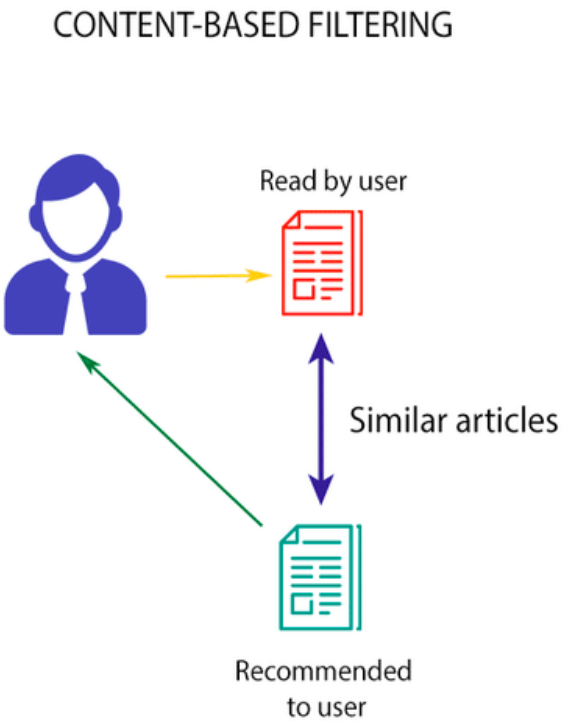
Based on these measures, the system also computes sources for candidate generation such as related items from a matrix factorization model, popular or trending items, items liked or recommended by friends.

Popular Candidate Generation Filtering Approaches

Content-based filtering (also called cognitive filtering)

This kind of filtering is based on additional information about users and/or items and recommends items that are similar to the items that the user likes. The system finds the similarity between items based on its description or other contextual features. The user's previous history is taken into account to find similar items the user may like. The user preferences can be identified using two methods - by gathering explicit

feedback or implicit feedback. For instance, if the user watches a lot of cooking videos, then the system can recommend other cooking videos to that user.

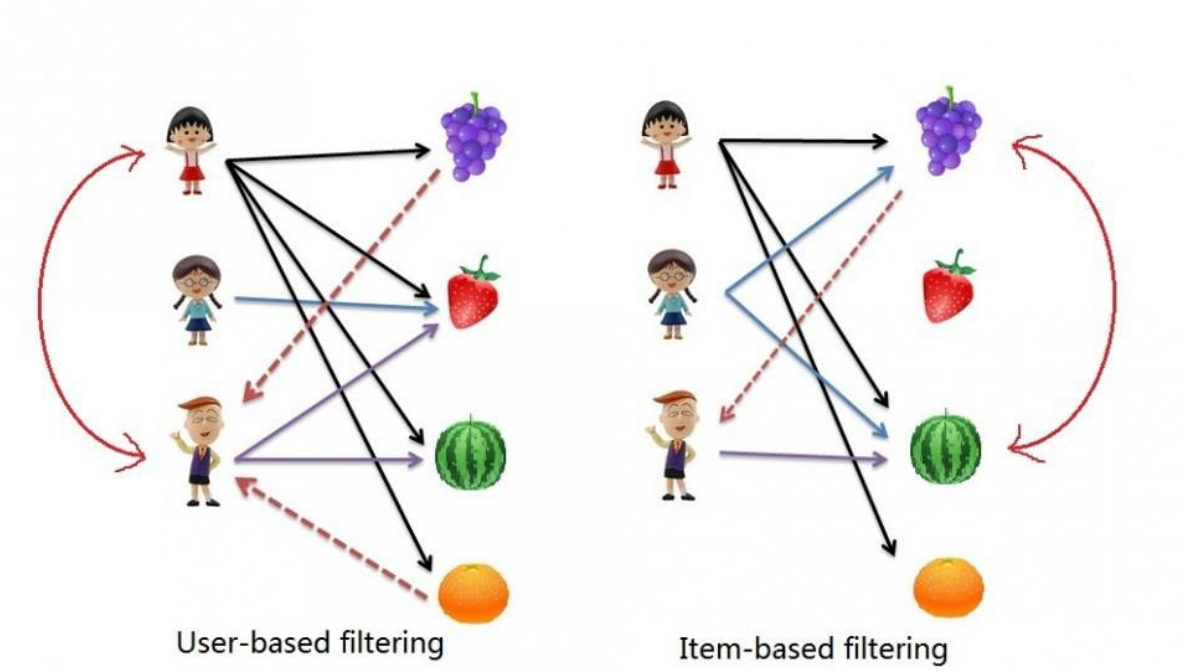


For this model, two types of information are used to generate candidates. First, the previous preferences of the user, their interest, user's interaction history, and/or user's personal information. This data is represented by the user vector. Second, information related to the item as an item vector. Then it generates a feature matrix where each column represents a feature, and each row represents an item. The item vector contains the features of all items based on which similarity between them can be calculated using Vector Space Model such as TFIDF or probabilistic models such as Naïve Bayes Classifier, Decision Trees or Neural Networks. During the recommendation process, the similarity score will be calculated based on the vectors to determine whether the item is relevant or not to the user.

This filtering method is based on a single user's history of interactions and preferences and does not require other users' data since it is treated as a user-specific classification problem. It can also capture a user's interest and recommend relevant items even if it is rarely interested by other users. However, with an approach like this, the more information that the user provides, the higher the accuracy and the disadvantage is that the item's feature representation requires rich descriptive item metadata.

Collaborative filtering (also called social filtering)

This approach makes recommendations based on other users with similar tastes or situations. This kind of filtering assumes that people who like similar items in the past will also like similar items in the future. Therefore, this method recommends items based on the similarity of both the users and items and the past interactions recorded between users and items. These interactions are stored in the "user-item interactions matrix".



For example, if user A is similar to user B as recognized in the system, and user A likes cooking videos, then the system can recommend cooking videos to user B. Then it prepares a matrix where each column represents an item, and each row represents a user to calculate similarity measures. However, since this method requires users' past history, it suffers from the cold-start problem, which means that it cannot recommend things to a new user due to lack of sufficient personal and interaction data.
There are two types of collaborative filtering:

Memory-based or neighborhood-based methods

This is a model to recollect ratings of other users to predict a user's ratings.

Two types of memory-based collaborative filtering techniques are used -- user-user collaborative filtering, and item-item collaborative filtering.

User-User collaborative filtering ("Customers who bought this item, also bought these items"

The model forms a pool of similar users and averages their ratings of the target item. In this, the user vector includes all the items purchased by the user and rating given for each particular item. The similarity is calculated between users using an n*n matrix in which n is the number of users present. Now, the recommended matrix is calculated. In this, the rating is multiplied by the similarity between the users who have bought this item and the user to which item has to be recommended. This value is calculated for all items that are new for that user and are sorted in descending order. Then the top items are recommended to that user.

Item-Item collaborative filtering

This model forms a pool of similar items and averages the target user's ratings of those items. In this, rather than considering similar users, similar items are considered. If the user 'A' loves 'Inception' they may like 'The Martian' as the lead actor is similar. Here, the recommendation matrix is m*m matrix where m is the number of items present.

Model-based Collaborative Filtering

Model-based methods use machine learning and data mining techniques to generate candidates such as association rules-based models, Singular Value Decomposition (SVD), Matrix Completion technique, clustering, decision trees, latent-factor models, and neural network models.

Knowledge-based Recommender Systems

Knowledge-based recommender systems are similar to content-based recommender systems in that they construct a personalized model for each user and use it to find relevant items for them. However, instead of using the rating history of the user to construct that model, they require the user to explicitly state their preferences, perhaps through keywords or a series of requirements. They are useful in a cold start scenario, or in a complex item domain with many attributes, where a user might not immediately know what they want.

Hybrid

Such recommender systems combine content-based filtering, collaborative filtering, and other approaches. It can be implemented through many ways: run each filtering method separately and combine them, add one filtering technique to another filtering, unify the approaches to one model.

Such methods are introduced to solve some common problems such as cold-start, sparsity, and over-specialization problems, and also improves the accuracy and efficiency of original approaches.

The hybrid recommender systems can be classified into the following categories based on their combination methods: Weighted, Switching, Mixed, Feature combination, Cascade, Feature augmentation, and Meta-level.

Candidate Ranking and Retrieval

As described above, given the user preference embedding u, the system search for item embeddings *i* that are close to *u* in the embedding space (nearest neighbors), and return a subset of potentially relevant candidate items according to the similarity score *s(u ,i)*.

To compute the nearest neighbors in the embedding space based on the similarity scores, the system exhaustively scores every potential candidate. Scoring every candidate item is very expensive in terms of time and computing resources, therefore, often recommender systems use approximate nearest neighbors or perform the scoring process offline and store a list of top potential candidates for each user.

Phase 2: Scoring (Prediction and recommendation phase)

After generating a list of potential candidate items based on similarity measures, a model scores and ranks the generated candidates in order to select a set of items (e.g., top 10 items) to display to the user. Since this time, the ranking and scoring happen on a relatively small subset of items, the system can use a more precise model relying on additional users.

Scoring Function

The choice of scoring function can dramatically affect the ranking of items, and ultimately the quality of the recommendations. Some scoring functions can bring positional bias (items displaying on top of the screen are more likely to be clicked vs. items lower on the screen) in scoring. Some scoring functions generate position independent ranking to avoid the problem.

Phase 3: Re-ranking

In the final stage of a recommendation process, the system re-ranks the candidates or filters some candidates considering additional criteria or constraints. For example, in this stage, the system can remove items that a user explicitly disliked. Re-ranking can also help ensure diversity, and fairness in recommendation so that a user can get a wide range of items.

The common challenges a recommender system faces

Cold start

This refers to a situation where a recommender does not have adequate information about a user or an item in order to make relevant predictions.

Sparsity of data

This is the problem that occurs as a result of lack of enough information, that is, when only a few of the total number of items available in a database are rated by users. This always leads to a sparse user-item matrix. So, finding ways to use denser parts of the data set and those with information is critical.

Latent association and Synonymy

Synonymy is the tendency of very similar items to have different names or entries or incorrect labelling. Same items with different labelling can be ignored or incorrectly consumed, meaning that the information does not get incorporated correctly.

Scalability

The computational approach has become overwhelmed by the multiplicity of items and users. This becomes a challenge as data sets widen and can lead to performance reduction.

Evaluation

Conventional measurement techniques include measures of accuracy or coverage measures. The type of metrics used depends on the type of filtering technique.

Accuracy

Accuracy can be described as the fraction of correct recommendations out of the total possible recommendations.

Statistical accuracy metrics evaluate accuracy of a filtering technique by comparing the predicted ratings directly with the actual user rating. Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Correlation are commonly used as statistical accuracy metrics.

Decision support accuracy metrics that are popularly used are Reversal rate, Weighted errors, Receiver Operating Characteristics (ROC) and Precision Recall Curve (PRC), Precision, Recall and F-measure. These metrics help users in selecting items that are of very high quality out of the available set of items. The metrics view prediction procedure as a binary operation which distinguishes good items from those items that are not good.

Coverage

Coverage measures the number of items or users that the system is actually able to provide recommendation for.