## **“RECOMMENDER SYSTEM” – FOR MOVIELENS MOVIES DATABASE**

# INTRODUCTION

Recommender systems or recommendation systems are a subclass of information filtering system that seek to predict the "rating" or "preference" that a user would give to an item.

Recommender systems have become extremely common in recent years, and are utilized in a variety of areas: some popular applications include movies, music, news, books, research articles, search queries, social tags, and products in general. However, there are also recommender systems for experts, collaborators, jokes, restaurants, financial services, life insurance, romantic partners (online dating), and Twitter pages.

One popular application of Recommender systems is in movie prediction based on a user’s profile.

MovieLens is one such a recommender system and virtual community website that recommends movies for its users to watch, based on their film preferences using advanced recommendation algorithms. MovieLens was created by GroupLens Research, a research lab in the Department of Computer Science and Engineering at the University of Minnesota, to gather research data on personalized recommendations.

Recommender systems typically produce a list of recommendations in one of two ways – through collaborative and content-based filtering or the personality-based approach.

**Collaborative filtering approaches** building a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in. e.g.

* Last.fm creates a "station" of recommended songs by observing what bands and individual tracks the user has listened to on a regular basis and comparing those against the listening behavior of other users. Last.fm will play tracks that do not appear in the user's library, but are often played by other users with similar interests. As this approach leverages the behavior of users, it is an example of a collaborative filtering technique.

**strengths and weaknesses**

Last.fm requires a large amount of information on a user in order to make accurate recommendations. This is an example of the cold start problem, and is common in collaborative filtering systems.[

**Content-based filtering approaches** utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties.

* Pandora uses the properties of a song or artist (a subset of the 400 attributes provided by the Music Genome Project) in order to seed a "station" that plays music with similar properties. User feedback is used to refine the station's results, deemphasizing certain attributes when a user "dislikes" a particular song and emphasizing other attributes when a user "likes" a song. This is an example of a content-based approach.

**Strengths and weaknesses**

While Pandora needs very little information to get started, it is far more limited in scope (for example, it can only make recommendations that are similar to the original seed).

These approaches are often combined (see Hybrid Recommender Systems).

The personality-based approach derives product and service preferences from a user's personality.

\* Cold start problem

Cold start is a potential problem in computer-based information systems which involve a degree of automated data modelling. Specifically, it concerns the issue that the system cannot draw any inferences for users or items about which it has not yet gathered sufficient information.

The cold start problem is most prevalent in recommender systems. Recommender systems form a specific type of information filtering (IF) technique that attempts to present information items (movies, music, books, news, images, web pages) that are likely of interest to the user. Typically, a recommender system compares the user's profile to some reference characteristics. These characteristics may be from the information item (the content-based approach) or the user's social environment (the collaborative filtering approach).

In the content-based approach, the system must be capable of matching the characteristics of an item against relevant features in the user's profile. In order to do this, it must first construct a sufficiently-detailed model of the user's tastes and preferences through preference elicitation. This may be done either explicitly (by querying the user) or implicitly (by observing the user's behaviour). In both cases, the cold start problem would imply that the user has to dedicate an amount of effort using the system in its 'dumb' state – contributing to the construction of their user profile – before the system can start providing any intelligent recommendations.

In the collaborative filtering approach, the recommender system would identify users who share the same preferences (e.g. rating patterns) with the active user, and propose items which the like-minded users favoured (and the active user has not yet seen). Due to the cold start problem, this approach would fail to consider items which no-one in the community has rated previously.

The cold start problem is also exhibited by interface agents. Since such an agent typically learn the user's preferences implicitly by observing patterns in the user's behaviour – "watching over the shoulder" – it would take time before the agent may perform any adaptations personalised to the user. Even then, its assistance would be limited to activities which it has formerly observed the user engaging in.

# Colloborative Filtering

One approach to the design of recommender systems that has wide use is collaborative filtering. Collaborative filtering methods are based on collecting and analyzing a large amount of information on users’ behaviors, activities or preferences and predicting what users will like based on their similarity to other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself. Many algorithms have been used in measuring user similarity or item similarity in recommender systems. For example, the k-nearest neighbor (k-NN) approach and the Pearson Correlation as first implemented by Allen.

Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past.

When building a model from a user's behavior, a distinction is often made between explicit and implicit forms of data collection.

Examples of **explicit data collection** include the following:

* Asking a user to rate an item on a sliding scale.
* Asking a user to search.
* Asking a user to rank a collection of items from favorite to least favorite.
* Presenting two items to a user and asking him/her to choose the better one of them.
* Asking a user to create a list of items that he/she likes.

Examples of **implicit data collection** include the following:

* Observing the items that a user views in an online store.
* Analyzing item/user viewing times[24]
* Keeping a record of the items that a user purchases online.
* Obtaining a list of items that a user has listened to or watched on his/her computer.
* Analyzing the user's social network and discovering similar likes and dislikes.

The recommender system compares the collected data to similar and dissimilar data collected from others and calculates a list of recommended items for the user. Several commercial and non-commercial examples are listed in the article on collaborative filtering systems.

One of the most famous examples of collaborative filtering is item-to-item collaborative filtering (people who buy x also buy y), an algorithm popularized by Amazon.com's recommender system. Other examples include:

As previously detailed, Last.fm recommends music based on a comparison of the listening habits of similar users.

Facebook, MySpace, LinkedIn, and other social networks use collaborative filtering to recommend new friends, groups, and other social connections (by examining the network of connections between a user and their friends). Twitter uses many signals and in-memory computations for recommending who to follow to its users.

Collaborative filtering approaches often suffer from three problems: cold start, scalability, and sparsity.

* Cold start: These systems often require a large amount of existing data on a user in order to make accurate recommendations.
* Scalability: In many of the environments in which these systems make recommendations, there are millions of users and products. Thus, a large amount of computation power is often necessary to calculate recommendations.
* Sparsity: The number of items sold on major e-commerce sites is extremely large. The most active users will only have rated a small subset of the overall database. Thus, even the most popular items have very few ratings.
* Synonyms: Synonyms refers to the tendency of a number of the same or very similar items to have different names or entries. Most recommender systems are unable to discover this latent association and thus treat these products differently. For example, the seemingly different items "children movie" and "children film" are actually referring to the same item. Indeed, the degree of variability in descriptive term usage is greater than commonly suspected.[citation needed] The prevalence of synonyms decreases the recommendation performance of CF systems. Topic Modeling (like the Latent Dirichlet Allocation technique) could solve this by grouping different words belonging to the same topic.
* Gray sheep: Gray sheep refers to the users whose opinions do not consistently agree or disagree with any group of people and thus do not benefit from collaborative filtering. Black sheep are the opposite group whose idiosyncratic tastes make recommendations nearly impossible. Although this is a failure of the recommender system, non-electronic recommenders also have great problems in these cases, so black sheep is an acceptable failure.
* Shilling attacks: In a recommendation system where everyone can give the ratings, people may give lots of positive ratings for their own items and negative ratings for their competitors. It is often necessary for the collaborative filtering systems to introduce precautions to discourage such kind of manipulations.
* Diversity and the long tail: Collaborative filters are expected to increase diversity because they help us discover new products. Some algorithms, however, may unintentionally do the opposite. Because collaborative filters recommend products based on past sales or ratings, they cannot usually recommend products with limited historical data. This can create a rich-get-richer effect for popular products, akin to positive feedback. This bias toward popularity can prevent what are otherwise better consumer-product matches. A Wharton study details this phenomenon along with several ideas that may promote diversity and the "long tail."Several collaborative filtering algorithms have been developed to promote diversity and the "long tail" by recommending novel, unexpected, and serendipitous items.

A particular type of collaborative filtering algorithm uses matrix factorization, a low-rank matrix approximation technique.

Collaborative filtering methods are classified as memory-based and model based collaborative filtering. A well-known example of memory-based approaches is user-based algorithm and that of model-based approaches is Kernel-Mapping Recommender.

## 1.1 Methodolgy

Collaborative filtering systems have many forms, but many common systems can be reduced to two steps:

**User-based collaborative filtering**

1. Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user.

A specific application of this is the user-based **Nearest Neighbor algorithm**.

**Item-based collaborative filtering**

Alternatively, item-based collaborative filtering (users who bought x also bought y), proceeds in an item-centric manner:

1. Build an item-item matrix determining relationships between pairs of items.
2. Infer the tastes of the current user by examining the matrix and matching that user's data.

For example, the **Slope One item-based collaborative filtering family**.

Another form of collaborative filtering can be based on implicit observations of normal user behavior (as opposed to the artificial behavior imposed by a rating task).

These systems observe what a user has done together with what all users have done (what music they have listened to, what items they have bought) and use that data to predict the user's behavior in the future, or to predict how a user might like to behave given the chance.

These predictions then have to be filtered through business logic to determine how they might affect the actions of a business system. For example, it is not useful to offer to sell somebody a particular album of music if they already have demonstrated that they own that music.

Relying on a scoring or rating system which is averaged across all users ignores specific demands of a user, and is particularly poor in tasks where there is large variation in interest (as in the recommendation of music). However, there are other methods to combat information explosion, such as web search and data clustering.

**Content-based filtering**

Another common approach when designing recommender systems is content-based filtering. Content-based filtering methods are based on a description of the item and a profile of the user’s preference. In a content-based recommender system, **keywords are used to describe the items and a user profile is built to indicate the type of item this user likes**. In other words, these algorithms try to recommend items that are similar to those that a user liked in the past (or is examining in the present). In particular, various candidate items are compared with items previously rated by the user and the best-matching items are recommended. This approach has its roots in information retrieval and information filtering research.

To abstract the features of the items in the system, an item presentation algorithm is applied. A widely used algorithm is the tf–idf representation (also called vector space representation).

To create a user profile, the system mostly focuses on two types of information: 1. A model of the user's preference. 2. A history of the user's interaction with the recommender system.

Basically, these methods use an item profile (i.e. a set of discrete attributes and features) characterizing the item within the system. The system creates a content-based profile of users based on a weighted vector of item features. The weights denote the importance of each feature to the user and can be computed from individually rated content vectors using a variety of techniques. Simple approaches use the average values of the rated item vector while other sophisticated methods use machine learning techniques such as Bayesian Classifiers, cluster analysis, decision trees, and artificial neural networks in order to estimate the probability that the user is going to like the item.

Direct feedback from a user, usually in the form of a like or dislike button, can be used to assign higher or lower weights on the importance of certain attributes (using Rocchio classification or other similar techniques).

A key issue with content-based filtering is whether the system is able to learn user preferences from users' actions regarding one content source and use them across other content types. When the system is limited to recommending content of the same type as the user is already using, the value from the recommendation system is significantly less than when other content types from other services can be recommended. For example, recommending news articles based on browsing of news is useful, but would be much more useful when music, videos, products, discussions etc. from different services can be recommended based on news browsing.

As previously detailed, Pandora Radio is a popular example of a content-based recommender system that plays music with similar characteristics to that of a song provided by the user as an initial seed. There are also a large number of content-based recommender systems aimed at providing movie recommendations, a few such examples include Rotten Tomatoes, Internet Movie Database, Jinni, Rovi Corporation, Jaman and See This Next. Document related recommender systems aim at providing document recommendations to knowledge workers, for example Noggle and Google Springboard.

**Hybrid recommender systems**

Recent research has demonstrated that a hybrid approach, combining collaborative filtering and content-based filtering could be more effective in some cases. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model (see[16] for a complete review of recommender systems). Several studies empirically compare the performance of the hybrid with the pure collaborative and content-based methods and demonstrate that the hybrid methods can provide more accurate recommendations than pure approaches. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem.

**Netflix is a good example of the use of hybrid recommender systems. They make recommendations by comparing the watching and searching habits of similar users (i.e. collaborative filtering) as well as by offering movies that share characteristics with films that a user has rated highly (content-based filtering)**.

A variety of techniques have been proposed as the basis for recommender systems: **collaborative, content-based, knowledge-based, and demographic techniques**. Each of these techniques has known shortcomings, such as the well known cold-start problem for collaborative and content-based systems (what to do with new users with few ratings) and the knowledge engineering bottleneck in knowledge-based approaches. A hybrid recommender system is one that combines multiple techniques together to achieve some synergy between them.

* Collaborative: The system generates recommendations using only information about rating profiles for different users. Collaborative systems locate peer users with a rating history similar to the current user and generate recommendations using this neighborhood.
* Content-based: The system generates recommendations from two sources: the features associated with products and the ratings that a user has given them. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on product features.
* Demographic: A demographic recommender provides recommendations based on a demographic profile of the user. Recommended products can be produced for different demographic niches, by combining the ratings of users in those niches.
* Knowledge-based: A knowledge-based recommender suggests products based on inferences about a user’s needs and preferences. This knowledge will sometimes contain explicit functional knowledge about how certain product features meet user needs.
* The term hybrid recommender system is used here to describe any recommender system that combines multiple recommendation techniques together to produce its output. There is no reason why several different techniques of the same type could not be hybridized, for example, two different content-based recommenders could work together, and a number of projects have investigated this type of hybrid: NewsDude, which uses both naive Bayes and kNN classifiers in its news recommendations is just one example.

**Seven hybridization techniques**:

* Weighted: The score of different recommendation components are combined numerically.
* Switching: The system chooses among recommendation components and applies the selected one.
* Mixed: Recommendations from different recommenders are presented together.
* Feature Combination: Features derived from different knowledge sources are combined together and given to a single recommendation algorithm.
* Feature Augmentation: One recommendation technique is used to compute a feature or set of features, which is then part of the input to the next technique.
* Cascade: Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.
* Meta-level: One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique.

**Personality-based recommender systems**

A new approach stems from Buettner who proposed a personality-based product recommender (PBPR) framework analyzing social media data in order to predict a user's personality and to subsequently derive its personality-based product preferences.

**Success criterion for recommender systems**

Typically, research on recommender systems is concerned about finding the most accurate recommendation algorithms. However, there is a number of factors that are also important.

* Diversity – Users tend to be more satisfied with recommendations when there is a higher intra-list diversity, e.g. items from different artists.
* Recommender persistence – In some situations it is more effective to re-show recommendations, or let users re-rate items, than showing new items. There are several reasons for this. Users may ignore items when they are shown for the first time, for instance, because they had no time to inspect the recommendations carefully.
* Privacy – Recommender systems usually have to deal with privacy concerns because users have to reveal sensitive information. Building user profiles using collaborative filtering can be problematic from a privacy point of view. Many European countries have a strong culture of data privacy, and every attempt to introduce any level of user profiling can result in a negative customer response. A number of privacy issues arose around the dataset offered by Netflix for the Netflix Prize competition. Although the data sets were anonymized in order to preserve customer privacy, in 2007 two researchers from the University of Texas were able to identify individual users by matching the data sets with film ratings on the Internet Movie Database. As a result, in December 2009, an anonymous Netflix user sued Netflix in Doe v. Netflix, alleging that Netflix had violated United States fair trade laws and the Video Privacy Protection Act by releasing the datasets. This led in part to the cancellation of a second Netflix Prize competition in 2010. Much research has been conducted on ongoing privacy issues in this space. Ramakrishnan et al. have conducted an extensive overview of the trade-offs between personalization and privacy and found that the combination of weak ties (an unexpected connection that provides serendipitous recommendations) and other data sources can be used to uncover identities of users in an anonymized dataset.
* User demographics – Beel et al. found that user demographics may influence how satisfied users are with recommendations. In their paper they show that elderly users tend to be more interested in recommendations than younger users.
* Robustness – When users can participate in the recommender system, the issue of fraud must be addressed.
* Serendipity – Serendipity is a measure of "how surprising the recommendations are". For instance, a recommender system that recommends milk to a customer in a grocery store might be perfectly accurate, but it is not a good recommendation because it is an obvious item for the customer to buy.
* Trust – A recommender system is of little value for a user if the user does not trust the system. Trust can be built by a recommender system by explaining how it generates recommendations, and why it recommends an item.
* Labelling – User satisfaction with recommendations may be influenced by the labeling of the recommendations. For instance, in the cited study click-through rate (CTR) for recommendations labeled as "Sponsored" were lower (CTR=5.93%) than CTR for identical recommendations labeled as "Organic" (CTR=8.86%). Interestingly, recommendations with no label performed best (CTR=9.87%) in that study.

Catch phrase – Spatial and temporal correlation

**Performance measures**

Evaluation is important in assessing the effectiveness of recommendation algorithms. The commonly used metrics are the **mean squared error** and **root mean squared error**, the latter having been used in the Netflix Prize. The information retrieval metrics such as **precision and recall or DCG are useful to assess the quality of a recommendation method**. Recently, diversity, novelty, and coverage are also considered as important aspects in evaluation. However, many of the classic evaluation measures are highly criticized. Often, results of so-called offline evaluations do not correlate with actually assessed user-satisfaction. The authors conclude "we would suggest treating results of offline evaluations [i.e. classic performance measures] with skepticism".

**Multi-criteria recommender systems**

Multi-criteria recommender systems (MCRS) can be defined as recommender systems that incorporate preference information upon multiple criteria. Instead of developing recommendation techniques based on a single criterion values, the overall preference of user u for the item i, these systems try to predict a rating for unexplored items of u by exploiting preference information on multiple criteria that affect this overall preference value. Several researchers approach MCRS as a multi-criteria decision making (MCDM) problem, and apply MCDM methods and techniques to implement MCRS systems. See this chapter for an extended introduction.

# DATA SETS

The dataset used was from MovieLens, and is publicly available at -<http://grouplens.org/datasets/movielens/latest>.

There are two sets of data having different number of observations –

1. Small dataset - It contains 105339 ratings and 6138 tag applications across 10329 movies. These data were created by 668 users between April 03, 1996 and January 09, 2016.
2. Large dataset- It contains 22884377 ratings and 586994 tag applications across 34208 movies. These data were created by 247753 users between January 09, 1995 and January 29, 2016.

For initial model building and testing, the smaller dataset is used.

**Formatting and Encoding** -

The dataset files are written as files with a single header row. Columns that contain commas (`,`) are escaped using double-quotes (`"`). These files are encoded as UTF-8. If accented characters in movie titles or tag values (e.g. Misérables, Les (1995)) display incorrectly, make sure that any program reading the data, such as a text editor, terminal, or script, is configured for UTF-8.

**User Ids**

MovieLens users were selected at random for inclusion. Their ids have been anonymized. User ids are consistent between `ratings.csv` and `tags.csv` (i.e., the same id refers to the same user across the two files).

**Movie Ids**

Only movies with at least one rating or tag are included in the dataset. These movie ids are consistent with those used on the MovieLens web site (e.g., id `1` corresponds to the URL <https://movielens.org/movies/1>). Movie ids are consistent between `ratings.csv`, `tags.csv`, `movies.csv`, and `links.csv` (i.e., the same id refers to the same movie across these four data files)

The files present in this data set are described below.

|  |  |
| --- | --- |
| **File Name** | **File Description** |
| ratings.csv | All ratings are contained in the file `ratings.csv`. Each line of this file after the header row represents one rating of one movie by one user, and has the following format:  userId,movieId,rating,timestamp  The lines within this file are ordered first by userId, then, within user, by movieId.  Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).  Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970. |
| tags.csv | All tags are contained in the file `tags.csv`. Each line of this file after the header row represents one tag applied to one movie by one user, and has the following format:  userId,movieId,tag,timestamp  The lines within this file are ordered first by userId, then, within user, by movieId.  Tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. The meaning, value, and purpose of a particular tag is determined by each user.  Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970. |
| movies.csv | Movie information is contained in the file `movies.csv`. Each line of this file after the header row represents one movie, and has the following format:  movieId,title,genres  Movie titles are entered manually or imported from <https://www.themoviedb.org/>, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.  Genres are a pipe-separated list, and are selected from the following:  \* Action  \* Adventure  \* Animation  \* Children's  \* Comedy  \* Crime  \* Documentary  \* Drama  \* Fantasy  \* Film-Noir  \* Horror  \* Musical  \* Mystery  \* Romance  \* Sci-Fi  \* Thriller  \* War  \* Western  \* (no genres listed) |
| links.csv | Identifiers that can be used to link to other sources of movie data are contained in the file `links.csv`. Each line of this file after the header row represents one movie, and has the following format:  movieId,imdbId,tmdbId  movieId is an identifier for movies used by <https://movielens.org>. E.g., the movie Toy Story has the link <https://movielens.org/movies/1>.  imdbId is an identifier for movies used by <http://www.imdb.com>. E.g., the movie Toy Story has the link <http://www.imdb.com/title/tt0114709/>.  tmdbId is an identifier for movies used by <https://www.themoviedb.org>. E.g., the movie Toy Story has the link <https://www.themoviedb.org/movie/862>.  Use of the resources listed above is subject to the terms of each provider. |

For our recommender models we will be using the ratings.csv and movies.csv.