# Movie recommendations based on rating history, movie metadata

Ryan McGuinness

## Introduction

There are two major strategies when offering personalized movie recommendations. These are content-based filtering, where movies are recommended based on how similar they are in relation to the movies the user likes, and collaborative filtering, where movies are recommended by finding other users with similar viewing preferences and recommending movies that those similar users rated highly. In practice, personalized movie recommendation systems combine these strategies.

Data for this project is the MovieLens dataset provided by GroupLens. The dataset contains 25 million ratings between 162 thousand viewers across 62 thousand movies. In this project, collaborative filtering and content-based filtering will be used to produce movie recommendations. When recommending a movie to a viewer, we will use a movie that user liked to find similar movies to what the viewer likes. This project does not address recommending movies to a viewer without a rating history.

Python is my tool of choice for such projects, and the pandas implementation of data frames is used throughout the entirety of this project. All algorithms related to principal component analysis, including preprocessing algorithms, are implemented using scikit-learn. While the full data set has 25 million ratings in it, and while it is in that data set we can do most of the data exploration, this model is built off a version with only 1 million ratings due to computational needs.

## Initial Data Exploration

Data is provided across six .csv files: `movies.csv`, `ratings.csv,` `tags.csv`, `links.csv`, `genome-scores.csv`, and `genome-tags.csv`. All movies are given a unique numeric ID consistent across `movies.csv`, `ratings.csv`, `tags.csv`, and `links.csv`. All individual users are given a unique numeric ID consistent across `ratings.csv` and `tags.csv`.

`movies.csv` contains the ID, title, and genre for every movie, for a total of 62,423 movies. The Title column of this file is entirely non-null, and consists of the movie title with the year it was released in parentheses. Titles are encoded in UTF-8. Where the movie's original name is not the same as its English name, the original name is listed in parentheses after the English name and before the year released. For movies with multiple genres listed, the genres are pipe-separated. There are no null objects in this column, but some movies do not list any genres at all. For these movies, the genres column reads "(no genres listed)". The list of genres used in this file is relatively small, consisting of only 18 different genres. These are Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, and Western.

`ratings.csv` contains one record for every rating of one movie by one user, totaling 25,000,095 records. Each record consists of the user ID, the movie ID, the rating given, and the time the rating was given. Ratings are expressed on a scale of 0.5 – 5.0 and the only possible ratings are integer and half-integer ratings. The time the rating is given is expressed in seconds after midnight of January 1, 1970, UTC.

`tags.csv` contains one record for every user-defined tag for every movie, totaling 1,093,360 records. Each record consists of the user ID, the movie ID, the tag given, and the time the tag was given. Some records have null tags. The time the tag is given is expressed in seconds after midnight of January 1, 1970, UTC.

`links.csv` contains IDs to link each movie in this dataset to imdb.org and themoviedb.org. This resource is not used in this project.

The Tag Genome is included in the full dataset only and is split into two files: `genome-scores.csv` and `genome-tags.csv`. The Tag Genome is a system to encode how relevant a set of tags are to each movie on a continuous scale between 0 and 1. A relevance of 0 indicates that the tag does not apply at all to a movie, and a tag of 1 indicates that the tag applies very strongly to the movie. Every movie has a relevance score for each of 1128 tags.[1]

`genome-scores.csv` has one record for every movie in the dataset and tag in the genome, for a total of 15,584,448 records, and each record consists of the movie ID, the tag ID, and the relevance of the tag. The relevance is expressed as a float between 0 and 1.

`genome-tags.csv` simply relates tag IDs to a short description of the tag. These short descriptions are not needed for the purposes of creating a movie recommender system, so `genome-tags.csv` is not used in this project.

## Data Transformation

### MOVIE INFORMATION

The `movies.csv` file originally contains information on the ID of the movie, title of the movie, the year the movie is produced, and the genres each movie belongs to as a data frame of shape (62422, 3). This information is transformed into 2 data frames, one which only contains information on the ID and title of the movie, and one which contains the ID, year produced, and genres. The movie titles data frame will not be used as a part of the recommender system, except to translate between the movie IDs and readable titles. The genres in the movie information data frame are encoded into 18 separate columns such that a 1 indicates the film belongs to the genre and a 0 indicates the film does not belong to the genre.

### TAG GENOME

The genome scores are originally expressed in a data frame with shape (15584448, 3), where the 3 columns represent the movie ID, the tag ID, and the relevance scores. This is transformed into a pivot table of shape (13816, 1128), such that each row of the data frame corresponds to one movie and each column of the data frame corresponds to one tag. Each row is indexed by the corresponding movie ID, and similarly each column is indexed by the corresponding tag ID. However, not every movie in the dataset is expressed in this data frame: there are 62,422 unique movie IDs in the `movies.csv` file, but only 13,816 unique movie IDs in this pivot table.

As certain tags can be (and often are) correlated with other specific tags, there is much redundancy in the tag genome data as is. For this reason, principal component analysis is performed on the genome scores pivot table. As shown in Figure 1, 238 principal components are needed to explain 80% of the variance within the pivot table. Using these PCs, we can reduce the dimensionality of the tag genome by nearly a factor of 5.

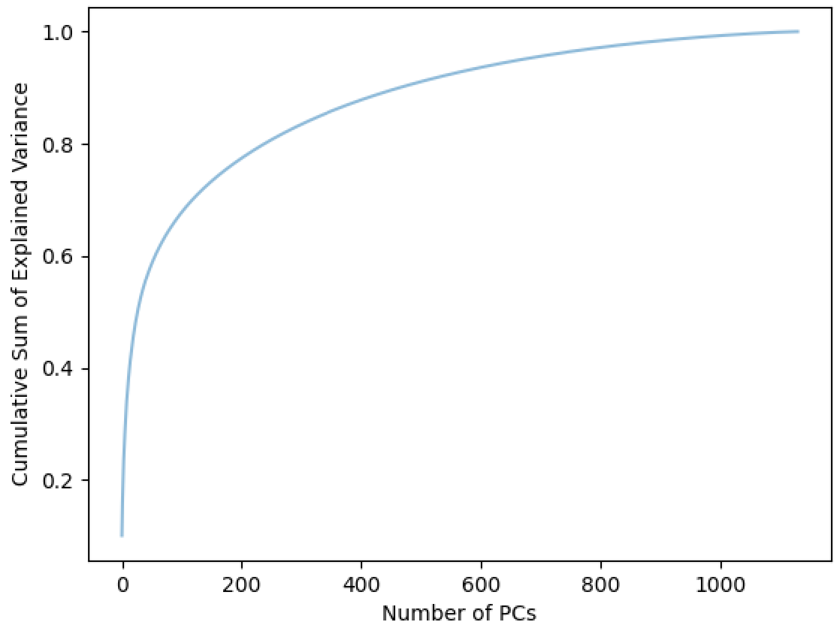

**Figure 1**: Explained variance of Tag Genome principal components. 238 PCs explain 80% of variance.

RATINGS

Ratings are originally expressed in a data frame with shape (25000095, 4), where the 4 columns correspond to the user ID, the movie ID, the rating given on a scale of 5, and the timestamp the rating was submitted. The first transformation is to discard the timestamp information. This data may be useful for finding trending movies, but our goal is to utilize a user's rating history to find personalized recommendations, not to recommend what's trending. We then remove ratings from users who have not rated many movies, and remove movies which have not received many ratings in order to reduce noise in the data set. The resulting data fame is then transformed into a pivot table where the rows are indexed by user ID and each column is headed by a movie ID. Of course, there are many null values as each user can only feasibly rate a small subset of the movies in this dataset. These values are set to 0 since we don't want the recommendation system to recommend movies that similar users have not interacted with.

## Theory and Methods

There are some technical challenges arise when using the full data set due to its size. When loading a .csv file as a data frame using Pandas, the values are always read in with data type object before they can be casted to the data type int. However, the data type object has a higher memory usage than the type int. As it happens, the computer I have access to can handle the pivot table expressing ratings given to each movie by each user, but it cannot read such a pivot table directly from a .csv file. Therefore, transforming the ratings data into a pivot table and finding similar movies using Pearson correlation must happen in the same python session. However, this method runs into the problem of creating too many large data frames to handle at once. For this reason, we are limited to the smaller dataset with 1 million ratings for model building.

### CONTENT-BASED FILTERING

Content-based filtering directly uses features of the movies the user has rated well to recommended movies with similar features.[2] The features used in this project are genres and tags (while tags are user-defined and generally a collaborative feature, the Tag Genome allows us to consider the relevance to these tags as features ofs the movies). Where the data is available, one could implement a content-based filtering algorithm using actors, directors, producers, movie title and descriptions, and anything that is directly a feature of the movie, but I am limiting the scope of this project to genres and tags as that is the information the dataset comes with. Furthermore, the Tag Genome is only available for the full dataset of 25 million ratings, which my computer simply cannot handle, so we will stick to using the genres here.

A similarity matrix is created with one row and one column for every movie ID, and the values are populated with the cosine similarity between those movies. Cosine similarity will compute the similarity score as the cosine of the angle between vectors representing them so that a score of 1 indicates perfect correlation and a score of 0 indications no correlation at all (no scores are negative here as components of the vectors are either 0 or positive). The components of the vectors are computed using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. This vectorization is used so that the more movies there are that belong to a particular genre, the less important that genre becomes for determining similarity. Figures 2 and 3 show this similarity matrix.

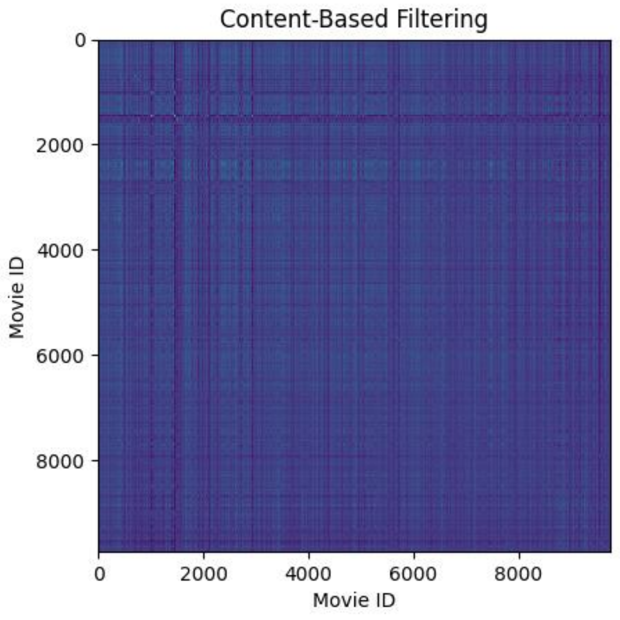

**Figure 2:** Content-based filtering similarity matrix for movie genres. Note that some movies have high similarity to many movies, while some movies show almost no similarity to other movies.

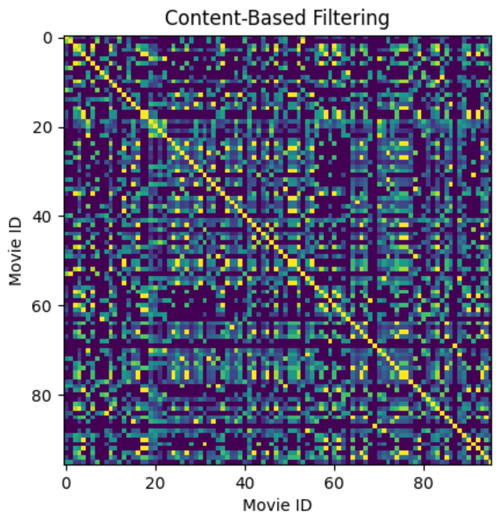

**Figure 3:** Content-based filtering similarity matrix zoomed in to see some details.

## COLLABORATIVE FILTERING

Collaborative filtering can be understood by taking rating history from the user seeking recommendations, finding viewers with a similar rating history, and selecting a movie that those viewers recommend. In practice, a similarity matrix is constructed for the movies long before the user seeking recommendations does anything.[2] I construct this similarity matrix by finding the Pearson correlation coefficient *r* between every pair of movies, and that matrix is used to find similar movies to recommend. Using this similarity matrix, we can implement similar techniques as in content-based filtering. When implementing this system, there is a question of how often to update this matrix, as it should change slightly every time a user submits a new rating on a movie, but that question is not addressed here.

On the whole, any pair of movies generally has a quite low *r* value in this data set. This is because the ratings pivot table is filled with a zero anywhere a user has not rated a movie. However, it is no problem that *r* values are low across the board, because finding best recommendations entails finding the *highest* values for *r*. The highest *r* values in the column for the movie the user likes (not corresponding to that movie itself) can then be recommended by the model.

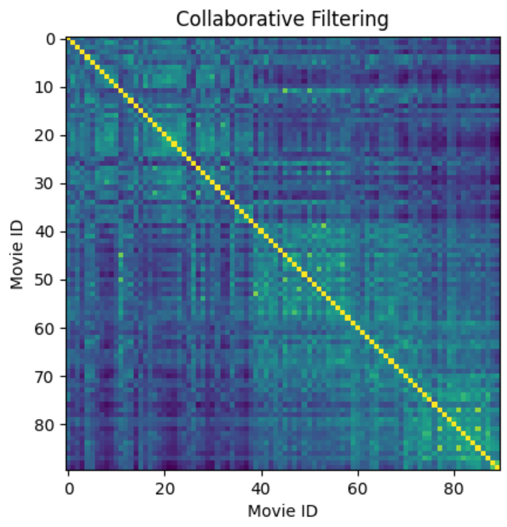

**Figure 4**: Collaborative filtering similarity matrix.

## Conclusions

Filtering on genres alone does not make for a great personalized recommendation system based on a single movie. This is because there are relatively few genres, and movies in the same genres are effectively a distance of 0 away from one another, so a very similar list of movies will be recommended for many movies given to the recommendation system. Despite that, the algorithm does indeed return a list of similar movies. For example, a user who likes *Finding Nemo* will be recommended a set of animated adventures.

Filtering on ratings results in the system offering a broader range of genres for any particular movie. Therefore, a viewer who consults this kind of recommendation system in hopes of actually being recommended something new is more likely to receive just that. However, this method has some serious problems with scalability. Even in the smaller data set I used to build these models, movies with lower rating counts need to be cut out of the system. Only 90 movies make it into this data set to keep it from crashing or from taking an unreasonably long time to train, but with a more powerful machine this can be overcome.

Unfortunately, I was unable to use the tag genome for anything beyond data exploration due to the genome only being available for the complete dataset, which is far too large for my computer to handle. However, having implemented content-based filtering once this project, and having done some data exploration and transformation with the tag genome data frame, I am content in leaving the implementation for a more powerful computer to handle.

## References

[1]Vig, J., Sen, S., and Riedl, J. 2012. *The tag genome: Encoding community knowledge to support novel interaction.* ACM Trans. Interact. Intell. Syst. 2, 3, Article 13 (September 2012), 44 pages. DOI = 10.1145/2362394.2362395 http://doi.acm.org/10.1145/2362394.2362395

[2]Roy, Abhijit. *Introduction To Recommender Systems- 1: Content-Based Filtering And Collaborative Filtering.* Medium. https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421