Recommender Systems Summative

Thomas Pettit   
Computer Science  
Durham UniversityDurham  
thomas.w.pettit@durham.ac.uk

# Introduction

## Domain of the Application

The domain of this application is movie/film recommendation. In particular, providing personalised recommendations to an active user of the system.

## Purpose and Aim of the Application

Movie and film streaming is a market that is growing very quickly, with an estimated market share of 60 billion dollars worldwide, and is predicted to reach 220 billion dollars by 2028 [2]. Therefore, it is in streaming companies’ interest to improve their service in order to maximise their portion of this market share. One way in which companies can get ahead of their competitors is to provide better and more personalized recommendations to their users, as this means their users will be more inclined to stay on their platform to have access to these improved suggestions, instead of heading to a competitor platform. With the huge amount of data that companies such as Netflix and Amazon Prime Video now have on user ratings, and genres and tags for each film on their platform, their ability to provide such recommendations has increased vastly, and so recommender systems that harness this data are improving quickly.

The aim of this application is therefore to demonstrate how, using large amounts of data, recommendations can be provided to the active user, using content-based and collaborative-filtering approaches, that maximise both accuracy and diversity.

# Methods

## Data Description

The dataset used is the MovieLens 25 Million Dataset [1]. This dataset includes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service.

The dataset includes 25000095 ratings, and 1093360 tags across 62423 movies, from 162541 different users. In order to ensure that there is enough ratings data for training the recommender systems for each user, each user in the dataset has rated at least 20 different movies. In order to hide the identity of the users, only an ID can identify each user, which has no reference to their actual identity elsewhere.

The dataset itself contains various files, namely: genome-scores, genome-tags, links, movies, ratings and tags. In this paper, the genome-scores, genome-tags, movies, ratings and tags files were used.

The dataset includes a Tag Genome. This is a data structure that contains the relevance of each tag to each movie, with each movie having a value for every tag in the genome. The tags indicate a range of features for a film, including: atmospheric, thought-provoking, realistic and more. The genome itself was calculated using a machine learning algorithm, trained upon user-contributed content such as tags, ratings and textual reviews, as described in [3]. This Tag Genome is represented by the files: genome-scores.csv and genome-tags.csv. The genome-scores.csv file contains the calculated relevance of each tag to each movie, and the genome-tags.csv file provides the description of each tag from each tag ID.

The movies file contains the title and genres associated with each movie ID.

The ratings file contains all the ratings given by users on the movies, as well as a timestamp indicating when the rating was left on the given film, represented in seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

The tags file contains all the tags given by users on films, as well as the timestamp that the user gave the tag for the corresponding film, also formatted in seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

## Data Preparation and Feature Selection

The datasets themselves contained no missing values in their columns. No pre-processing had to be done on any of the data files.

There were 1128 tags for each movie, each with their own relevance score for each movie. All of these tags were used to train both the content-based and collaborative-filtering recommender systems. This is because the more tags, the more accurate the recommender systems can be, as they can learn from a greater volume of data to learn user preferences from.

## Recommendation Algorithms

**Content-Based Filtering Approach**

In this approach, firstly the user profile is created. All the ratings for the active user are then collected. All the IDs of the movies that this user rated are then gathered into an array. The Tag Genome relevancy scores for each of the tags for each of these movies is then collected, and a dataframe is created to represent this information. The average rating for the active user is then calculated. In order to determine whether a rating is positive or negative, (whether the active user liked or disliked the film given their rating of it), the average rating of the user is then subtracted from the rating given by the user for a particular film, as shown below:

Where rui represents the rating given by user u on item i, and represents the average rating of user u. However, this is only applied if the user has more than 5 ratings, as if the user only has 1 rating, the average will be the one rating they have given, and so this would cause their rating to be 0, not providing any information to the system, and with less than 5 ratings, there isn’t enough data available to accurately determine an average rating for the given user.

Then, the user profile is created by calculating the dot product between every tag’s relevance on every rated movie and the rating the user has given on every movie. The user profile therefore then represents the active users’ preferences for every one of the tags in the Tag Genome.

Then, in order to make recommendation predictions, the dot product of the user profile and the movie Tag Genome scores for each of the possible movies is calculated, and saved in a new dataframe. The sum of all the columns for every movie is then calculated, and this represents the prediction score for each movie. These values are then scaled using Min-Max Feature Scaling, using the Sklearn MinMaxScaler [5], to ensure all the prediction values are between 0 and 1. Then, the films that the active user has seen previously are removed, so that they are not recommended movies they have already seen. The dataframe is then sorted in descending order, off of the predicted score column, and the first 10 movies are returned, as these movies have the highest predicted scores for the active user. The prediction score is then multiplied by 100, to make the score now a percentage, and then limited to only 1 decimal place. This makes for a more readable output for the active user.

In order to make the predictions more explainable, the 10 tags with the highest relevancy values in the user profile are displayed, as well as the 5 tags with the highest relevancy scores for each of the top 10 predictions for the active user. An example of this can be shown below in Figure 1 on the next page.

**Collaborative-Filtering Approach**

In this approach, a model-based collaborative-filtering recommender system is created, based upon a Matrix-Factorisation algorithm, specifically SVD, popularized by Simon Funk in the Netflix Prize [4]. This approach works in the following way:

The predicted rating for a movie i by user u, is given by the following formula:

where 𝜇 represents the average rating of all the movies in the dataset, bu represents the average rating given by user u, bi represents the average rating of item i, qi  represents the latent item vector and pu represents the latent user vector. Both the latent user and item vectors were set to 100 latent factors.

During training, the SVD minimises the following regularized squared error:

This is performed via stochastic gradient descent:

Where eui represents the difference between the actual rating given by user u on item i, and the predicted rating by the SVD. The model in this paper was trained using the learning rate, 𝛾, of 0.005, and the regularisation term, 𝜆, of 0.02.

As with the content-based approach, the top 5 tags of the recommended films are displayed in the table shown to the user, as shown below in Figure 2 on the next page.

## Evaluation Methods

**Accuracy: RMSE**

In order to evaluate the accuracy of the models, Root-Mean Squared Error was used, using the sklearn metrics function mean\_squared error [6]. This follows the following formula:

**Accuracy: Content-Based Filtering Approach RMSE**

Initially, the entire ratings.csv file was split into training and testing data, with a 75:25 ratio. The issue with this approach is that building a user profile for every potential user in the training set is very time consuming. Thus, a different approach was taken. In this approach, 5 users are selected at random from the entire ratings dataset. All the ratings given by each of these 5 users are then collected. 75% of these ratings are then used to create the user profile, and 25% are used to test the algorithm. The user profile is created using the 75% split, and then the user profiles are then used to create the predictions for each of the films in the 25% split, in the same method as described in section II.C. However, unlike producing a match score measured in percentages, as shown in Figure 1, the predictions that were Min-Max Feature Scaled, were now multiplied by 5, to produce a rating between 0 and 5. The values were then rounded to the nearest 0.5, to be in the correct format to compare with the actual user ratings. Then, in order to evaluate the accuracy of this recommender system, Root-Mean Squared Error was calculated for each of these 5 users, using the predicted ratings for the films in the test set, and the actual ratings given by the users on the films in the test set. The 5 calculated RMSE values for each of the random 5 users is then averaged, and the overall RMSE value for the content-based recommender system is created.

**Accuracy: Collaborative-Filtering Approach RMSE**

In order to measure the RMSE of this approach, the sklearn surprise accuracy library [7] was used. The method for measuring RMSE for this approach was slightly different to that of content-based. In order to speed up the time taken to evaluate the model, the ratings.csv dataset was split into training and testing data, with 2.5% of the data being used to test the model and 7.5% for training. The model that was trained on the entire dataset was used to make predictions on this test set. The model’s predictions on this test set were then used, alongside the actual user ratings, using the RMSE formula, to calculate the model’s RMSE score.

**Diversity**

Diversity was the other metric used to evaluate the two approaches. Diversity determines how wide the spectrum of recommendations is. This was used in this paper as a recommender system that always recommends similar movies (for example movies that have the same genre), would lead to the user always watching the same type of films, and thus could quickly become bored of this film genre, and so the recommender system would become obsolete. There are different methods of measuring diversity of recommendations. In this paper, it was measured via the proportion of common users between any 2 of the recommended films. If 2 items in the recommended set have a high proportion of common users, the recommended items are most likely very similar, and so the diversity of the recommender system is low, as described in [9].

This can be calculated via Cosine Similarity. The cosine similarity between every pair of predictions from the recommender system can be calculated, and the average value created. The Cosine Similarity was calculated using the following formula, where i and j are two different films:

This shows the average similarity between any 2 predictions in the top 10 recommendations provided by the recommender system.

Then, diversity can be calculated via the following formula:

where Sim10 is the average cosine similarity score between any 2 predictions in the top 10 provided by the recommender system.

To use this metric on the 2 recommender systems in this paper, a user was first selected at random from the ratings dataset. Then, the top 10 recommendations were created for this user, and then the diversity metric was calculated following the method outlined above.

# implementation

## Input Interface

When the user first logs into the system, they are presented with the following menu:

The user can either login as an existing user in the dataset, or create a new user. However, if the user decides to create a new user, the collaborative filter approach for recommendations cannot be used, as it would require the whole SVD model to be retrained, which would be too time consuming. The user is warned of this issue, as shown in the screenshot above.

If the user decides to login as existing user, they enter their user ID, and if they decide to create a new user, they are told their new user ID, and are logged in. Upon logging in, the user is presented with the following menu:

Text

Description automatically generatedText

Description automatically generatedHere, the user can view any ratings they have already made, add a new rating, edit an existing rating, or view their recommendations. Viewing ratings prints a dataframe to the command line showing the movie title and their rating for every film they have rated. Add a new rating allows the user to enter a movie ID, and then the rating for that film, as shown in the screenshot below:

However, this new rating is not taken into account for the collaborative-filtering recommendations, due to the same reason mentioned previously, whereby the model would need to be retrained, which is too time consuming.

Text

Description automatically generatedThe user can also choose to edit a rating they have previously made, which can be shown below in the screenshot:

The user can also select option 4 to view their recommended films. Upon pressing 4, they are then presented with a menu that allows the user to select either content-based or collaborative filter to use to create recommendations.

The user can also choose to logout of the system via option 5 of the menu.

## Output Interface

Both the recommender system approaches present the table of top 10 recommendations to the active user in the same way, with the movie ID, movie title, match score, and the top tags associated with the films. This can be seen below:

Figure 1 shows the output of the Content-Based recommendations. This approach states the top tags for the active user, which helps the user understand why they have been recommended the films they have been.

Figure 2 shows the output of the Collaborative-Filtering recommendations.

The tables are easy to read, and also have the advantage of displaying the top tags of each film, to help the active user determine which film out of these recommendations they may start watching.

# Evaluation

## Evaluation Metrics’ Results

**Accuracy: RMSE**

|  |  |
| --- | --- |
| **Content-Based Approach** | **Collaborative-Filter Approach** |
| 1.977 | 0.698 |

**Diversity**

|  |  |
| --- | --- |
| **Content-Based Approach** | **Collaborative-Filter Approach** |
| 0.703 | 0.955 |

## RS Techniques’ Comparison

The collaborative-filter approach has both lower Root Mean Squared Error, and a higher diversity score. This means that, when tested upon the testing data, the model predicts a rating by users on a particular film more closely to the actual user rating than the content-based approach. The collaborative-filter approach also has a higher diversity score. This means that it provides a wider spectrum of recommended films. This is particularly beneficial in this case, as it introduces the active user to a wider range of potential films to watch, outside of their usual watching habits. In industry, this would encourage the user to stick to the platform with this recommender system, as it recommends new and different films, thus making this an important metric.

Therefore, due to the fact that Collaborative-Filter approach has both a lower RMSE score, and a higher diversity score, it is the better technique in this case, despite the time taken to train the model. However, the Collaborative-Filtering approach is not as explainable as the Content-Based approach. Thus, if the movie recommender system had to be more transparent, sacrificing accuracy and a small loss in diversity score may be worth it. The Collaborative-Filtering approach also suffers from the fact that new users or ratings or edited ratings would require the model to be retrained, which is a very time consuming process. Thus, if these weaknesses needed to be solved, then the small drawbacks in RMSE and diversity associated with the Content-Based approach may be bearable.

# conclusion

In conclusion, both content-based and collaborative-filtering approaches can be used to produce recommendations for the active user of a recommendation system. The Collaborative-Filtering approach performs best on both the evaluation metrics, however it’s practical use for when new users or ratings are created or users’ ratings are edited, is limited, as the SVD model would need to be retrained on the updated dataset, which is a very time-consuming process. Thus, if small sacrifices in RMSE and Diversity are allowable, then the Content-Based filter would potentially be best.

##### References

1. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. <https://doi.org/10.1145/2827872>
2. Research ltd and Markets. Global video streaming market size, share amp; trends analysis.
3. Jesse Vig, Shilad Sen, and John Riedl. 2012. The Tag Genome: Encoding Community Knowledge to Support Novel Interaction. ACM Trans. Interact. Intell. Syst. 2, 3: 13:1–13:44. <https://doi.org/10.1145/2362394.2362395>
4. <https://surprise.readthedocs.io/en/stable/matrix_factorization.html#unbiased-note>
5. <https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>
6. https://scikitlearn.org/stable/modules/generated/sklearn.metrics.mean\_squared\_error.html
7. <https://surprise.readthedocs.io/en/stable/accuracy.html>
8. <https://medium.com/fnplus/deep-learning-22d971e3ad43>
9. <https://eugeneyan.com/writing/serendipity-and-accuracy-in-recommender-systems/#:~:text=Serendipity%20is%20measured%20as%20unexpectedness,the%20user%20interacted%20with%20i%20>.