Project 2: Movie Recommender System

Lara Clasen  
Spring Term 2020  
LLClasen.github.io

**Abstract**

The enclosed data analysis project aims to build a recommendation system for movie choices using R programming language. Recommendation systems are algorithms that work to offer relevant suggestions to the user based on a particular industry or product area. There are a few different types of recommendation systems that can be used for this sort of data. A simple recommender is pretty basic and offer suggestions based on what the general public “likes”. However, to get a bit deeper we can build either a content-based or collaborative filtering recommendation system for this movie data. Both of these use the specific user’s behavior to make suggestions. While content-based recommenders base their suggestions directly on user preferences for product features, collaborative filtering (as the name suggests) makes assumptions based on the user’s preferences as they relate to other user preferences. Because we have quite a bit of “content” to work with in this data set, we should be able to build either type of algorithm. The resulting recommendation for this project system falls into the category of item-to-item, or item-based, collaborative filtering. These data are important to analyze because it combines two huge pieces of the entertainment experience in today’s world: recommendations and movie information. It will allow users to obtain accurate recommendations for movie choices without the requirement of doing anything but looking at their phone or using the app which utilizes such an algorithm. The data source is a research group out of the University of Minnesota called GroupLens. This group specializes in recommender systems and other online computing systems and researches their use and abilities.

**Data Pre-Processing and Exploration**

The data set I have used contains rating and tagging activities from MovieLens and consists of 105,339 ratings and 6,138 tagging applications across 10,329 movies. Users are represented simply by an ID without any other demographic detail. All users had rated at least 20 movies to be selected. To begin, we retrieve our data from two distinct CSV files and translate them into dataframes. The two files we will be working from are named *movie\_data.csv* and *rating\_data.csv*, containing movie-specific data and rating-specific data, respectively.

One of the first ways we can become familiar with our data is by running the head and summary functions. The summary function gives us an overview of our dataframes while the head function provides the first 6 rows of each data set.

A screenshot of a cell phone

Description automatically generated

Figure 1: head function for movie\_data and rating\_data

By looking at a summary of our *movie\_data* dataframe (Figure 1) we can see that we need to adjust the genre feature so that it is in a more workable format. To do this, we use one-hot encoding to create a matrix consisting of genres for each film. From the rating\_data summary we see that the userId and movieId columns are represented as integers.

Next for pre-processing our data, we want to create a search matrix to allow us to easily search the films by specifying the genre feature. We do this using the cbind function in R which involves merging two data frames, with an equivalent number of rows, together into a single data frame. A majority of the movies connect with more than one genre. For example, Jumanji associates with the genres Adventure, Children, and Fantasy. In order to make sense of the movie ratings, we must convert our matrix into a sparse matrix. This new matrix is of the class 'realRatingMatrix'.

At this point we can review some important parameters that give us options for building our recommender system. With the recommenderlab package in R, we can easily manage our methods using the registry mechanism. We take a look at the recommenderRegistry for this project and we obtain the output shown in Figure 2. We see IBCF, or Item Based Collaborative Filtering, as an option here. This is the single model that we will implement for this project.

A picture containing table, room, holding, board

Description automatically generated

Figure 2: recommenderlab registry mechanism

Collaborative Filtering is an approach commonly used for recommender systems. This approach is a method of making predictions about interests of one user based upon the interests of another, separate user that appears to be similar to the first. The *collaborative* aspect makes sense because the underlying function combines multiple viewpoints to obtain an accurate assumption. There are two main types of collaborative filtering: user-to-item filtering and item-to-item filtering. Item-to-item filtering is what is being used here. This type involves making suggestions to users based on the preferences of many other users. For example, if User A likes action films and so does User B, the movies that User A watches will be recommended to User B and vice-versa. Ultimately the recommendation comes down to the similarity between two users. With the recommenderlab package we can compute these similarities using the Cosine operator (you could also use Pearson or Jaccard). Using the Cosine operator we calculate the similarities shared between users and movies. From there, we can extract the most unique ratings by utilizing movie rating counts. This is interesting to us because we are able to see which rating value was given to movies the least amount of times. We see that a rating of 0.5 was given the least often, followed closely by 1.5. Our users are much more likely to rate a movie around 4-stars than any of the other values between 0 and 5 (Figure 3).



Figure 3: Ratings based on uniqueness

To perform some additional data exploration, we can take a look at the most viewed movies in our data set. We count the number of views for a film and then organize them into a table in descending order. We can then visualize the views for our top movies in a bar plot (Figure 4). From our bar-plot we can see that Pulp Fiction is the most-watched film followed by Forrest Gump. We can also visualize a heatmap of the movie ratings and have the heat map visualization contain the first 25 rows and 25 columns.

A close up of a device

Description automatically generated

Figure 4: Top viewed movies

**Data Preparation**

There is some data preparation that we will need to perform in order to have the best data set to work with. Data preparation steps will include selecting useful data, normalizing data, and binarizing the data. To limit our data to more valuable points we will set a threshold such that a movie must have been rated by a min of 50 users and have a minimum of 50 views. This way, we have filtered a list of watched films from the least watched ones. After implementing this threshold, we now have 420 users and 447 films compared to the previous 668 users and 10,325 films. We can now delineate our matrix of relevant users. We produce a heat map of our top users and top movies. We also create a chart displaying the distribution of the average ratings per user (Figure 5). This visualization reflects our understanding that a user is more likely to rate a movie between 3.5 and 4 than any other rating available.

A picture containing screenshot

Description automatically generated

Figure 5: Average rating by user

In the next step of data preparation, we want to normalize our data. For some of our users they may have consistently rated all movies either very high or very low. This can create a strong bias within our data set, and so it is something that we want to work through a bit before proceeding. To remove this bias, we standardize the numerical values for a feature to a common scale value. The reduces the distortion in the range of our values. By normalizing we achieve an average value across all ratings of 0. We can then create a heat map displaying our now normalized rating feature.

Finally, we can binarize our data to simplify the process and allow our recommendation system to work more efficiently. Binarization is used when you want to convert a numerical feature vector into a Boolean vector, or binary representation. Binarizing means that we will end up with two discrete values: 0 and 1. We will define a matrix that will consist of 1 if the rating is above 3 and 0 if the rating is below 3.

**Recommendation System**

We can now begin to develop our recommender system using item based collaborative filtering. This method finds similarity between items based on user ratings of those items. The algorithm builds a table of similarity which is then fed into the system. To build this algorithm we must first split the data set into a test and train set split by 80/20. In the first step the algorithm identifies the *k* most similar items and store their number. Here, *k* is equal to 30. We use the Cosine method, which is the default method, but we could have also used the Pearson method.

Now that we have defined our model, using the getModel() function we can retrieve the model in the next step. Then we will find the class and dimensions of our similarity matrix. We can generate a heat map from this step containing the top 20 items and visualize the similarities between them. Next, we will create a new variable to consist of our top recommendations. We will then use the predict() function to identify items that are similar and rank them appropriately.

**Summary**

Recommendation systems are an incredibly common machine learning method, improving upon traditional classification algorithms with the ability to make accurate predictions based on user behavior. We have been able to successfully build a recommendation system to predict and recommend movie choices to users. This model provides suggestions to users through a filtering system based on user preferences and historical behavior, finding similarities between different products or, in our case, movies. The input is collaborative user information and the output is the resulting recommendation.

**Appendix**

**Main R packages used:**

* recommenderlab: Provides infrastructure to test and develop recommender algorithms
* ggplot2: Data visualization package for R; Widely used for creating custom plots
* data.table: Allows us to interface with the Javascript library *Datatables*
* reshape2: Allows us to easily transform data between wide and long formats

**References**

References

1. Rocca, B. (2019, June 12). Introduction to Recommender Systems. Retrieved from <https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>
   1. Offers a detailed overview of the different recommendation systems that are available and in what situations you might use them.
2. Poudyal, R. (2018, June 27). Latent Factor based method in collaborative filtering. Retrieved from <https://medium.com/@rabinpoudyal1995/latent-factor-based-method-in-collaborative-filtering-77756a02f675>
   1. This article reviews just one type of recommendation system and uses generalized examples to explain how it works.
3. Das, S. (2015, September 24). Beginners Guide to learn about Content Based Recommender Engine. Retrieved from <https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/>
   1. This article reviews content-based recommender engines popular services such as Facebook to demonstrate it usage.
4. Kirzhner, E. (2018, May 11). Machine Learning. Explanation of Collaborative Filtering vs Content Based Filtering. Retrieved from <https://codeburst.io/explanation-of-recommender-systems-in-information-retrieval-13077e1d916c>
   1. This article is very helpful for my case as it takes the time to compare two types of more complex recommendation algorithms.
5. Ansari, A., & Kohli, R. (2000, August). Internet recommendation systems. Retrieved from [https://www0.gsb.columbia.edu/mygsb/faculty/research/pubfiles/385/Internet Recommendation Systems.pdf](https://www0.gsb.columbia.edu/mygsb/faculty/research/pubfiles/385/Internet%20Recommendation%20Systems.pdf)
   1. An older article, this write-up offers some more numerical insight into the recommendation system.
6. Chauhan, N. S. (2019, July 9). Implement K-Nearest Neighbors classification Algorithm. Retrieved from <https://towardsdatascience.com/implement-k-nearest-neighbors-classification-algorithm-c99be8f14052>
   1. This article offers a valuable example of using the K-Nearest Neighbors classification algorithm.
7. Recommender Systems. (n.d.). Retrieved from <http://recommender-systems.org/>
   1. A very basic yet useful resource for understanding different recommendation system types.
8. Liao, K. (2018, November 19). Prototyping a Recommender System Step by Step Part 1: KNN Item-Based Collaborative Filtering. Retrieved from <https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea>
   1. Another helpful resource for the utilization of the K-Nearest Neighbors algorithm, specifically for use with collaborative filtering.
9. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (n.d.). Item-Based Collaborative Filtering Recommendation Algorithms. Retrieved from <http://files.grouplens.org/papers/www10_sarwar.pdf>
   1. This paper discusses item to item collaborative filtering specifically.
10. Ricci, F. (n.d.). Item-to-Item Collaborative Filtering and Matrix Factorization. Retrieved from <https://www.ics.uci.edu/~welling/teaching/CS77Bwinter12/presentations/course_Ricci/13-Item-to-Item-Matrix-CF.pdf>
    1. An interesting presentation presenting item to item collaborative filtering in comparison with other similar methods.