Netflix Movie Recommendation

Donghyuk Kim
Computer Science
Worcester Polytechnic Institute
Worcester, Massachusetts
dkim8@wpi.edu

Shijing Yang
Data Science
Worcester Polytechnic Institute
Worcester, Massachusetts
syang6@wpi.edu

I. Introduction

Many successful online companies nowadays rely on their recommendation systems to provide personalized experience that users expect. Netflix, for instance, must cater to the wide variety of users who have different movie tastes. Given the importance of the topic, building a smart recommender system has been studied extensively in both industry and academia. In this report, we will use the Netflix ratings dataset to revisit some of the classical recommendation algorithms like collaborative filtering, and explore ways to improve it by utilizing an external IMDb dataset.

II. Dataset

We will be utilizing two datasets to build a recommender system: The Netflix dataset, which contain movie ratings from users, and the IMDB dataset, which contain features that describe movies.

A. Netflix Dataset

Netflix released this dataset to hold competition for the best algorithm to predict user ratings for movies. The dataset contains 17770 movies, 480189 users, and total over 100 million ratings scored between 1 and 5.

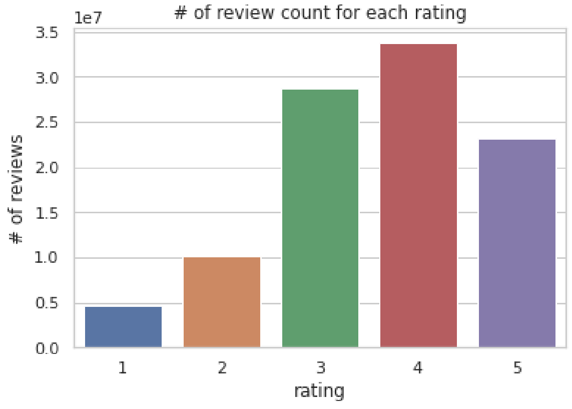

Fig. 1. Rating count for each ratings

As we can see in Figure 1, the most common ratings in the Netflix dataset is 3 and 4, which means users tend to rate movies with an average or above average rating. According to Figure 2 and Figure 3, the most of common number of given rating for a user is between 100 and 1000, while the most common number of received rating for a movie is also between 100 and 10,000.

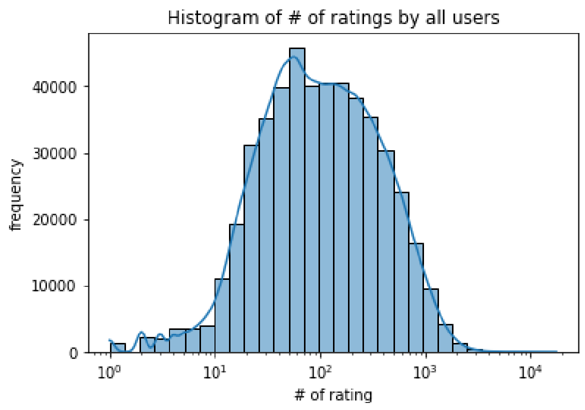

Fig. 2. Frequency of number of rating per user

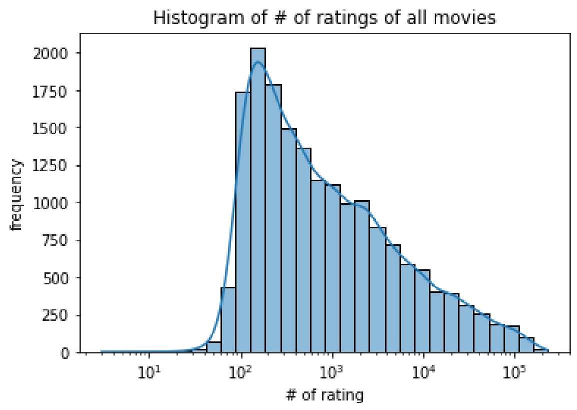

Fig. 3. Frequency of number of rating per user

B. IMDB Dataset

IMDb is the most popular movie website and it combines movie plot description, Metascore ratings, critic and user ratings and reviews, release dates, and many more aspects.IMDb dataset includes diverse metadata for 85,855 movies such as movie description, genre, director, actors, duration, date, etc.

III. Processing Data

A. Filtering Netflix Data

In order to perform classic memory-based collaborative filtering, we must build a user-item utility matrix where each row represents a user and each column represent a movie. With the entirety of the Netflix dataset, the utility matrix will have the size of around 17k x 480k. Due to the limitation of computational resources, we decided to filter our dataset to reduce the number of users and movies. We decided to keep the top 10% of movies that garnered most reviews, and top 10% of users that gave most reviews. After applying this filter, the number of ratings reduced approximately from 100 million to 30 million.

B. Filtering IMDb Data

We selected following metadata to be used as features that could potentially improve the recommender system: genre, director, actors, duration, year, and country.

C. Merging Two Datasets

Finally, we merged the two datasets based on title. Not all movies in the Netflix dataset were present in the IMDb dataset. After merging, the final dataset we used contained 1478 movies, 48033 users, and total of around 26 million reviews.

IV. Methodology

A. Collaborative Filtering Recommender

To predict what a particular user would rate a particular movie, collaborative filtering technique uses the known information of other similar users (or movies). For instance, if user A has similar liking as user B on a particular movie, A is likely to enjoy other movies that B has enjoyed. This would be an example of user-based collaborative filtering.

Using our filtered data (section II), we performed both user-based and item-based collaborative filtering. The first step for both CF methods is to build a utility matrix. Then, calculate the similarity between user (or item) vectors. Some similarity measures we used include cosine similarity, and Pearson correlation.

1) Item-based Collaborative Filtering: To predict a score a particular user for a particular movie using item-based CF, we find n most similar movies that user has rated. In our case, we set our neighbor size as 10 (n=10). Then, we use those 10 ratings to calculate a weighted average (1) for final prediction.

$$r_{xi} = \frac{\sum_{y \in N} S_{xy} r_{yi}}{\sum_{y \in N} S_{xy}} \tag{1}$$

The equation (1) is the formula to calculate the weighted average of item-based collaborative filtering. Let r_x be the vector of user x's ratings, then let N be the set of k movies most similar to movie i that have also rated by user i, and we are predicting the rating of movie i and user x.

2) User-based Collaborative Filtering: For a user-based collaborating filtering, instead of finding movies that are similar to the target movie, it finds users that have watched/rated the target movie and are similar to the target user. As the same as item-based collaborative filtering, we choose to use 10 as our neighbor size (n=10). After having the top 10 most similar users, we calculated the weighted average (2) of the ratings of selected users and got the predicted result of a target movie for a target user.

$$r_{xi} = \frac{\sum_{y \in N} S_{xy} r_{yi}}{\sum_{y \in N} S_{xy}} \tag{2}$$

The equation (2) is the formula to calculate the weighted average of user-based collaborative filtering. Let r_x be the vector of user x's ratings, then let N be the set of k users most similar to user x who have also rated movie i, and we are predicting the rating of movie i and user x.

3) Centered Cosine Similarity: While calculating the similarities of each vectors, we used *cosine similarity* to do so. However, since our user-movie matrix has a sparsity of 66%, therefore most of the values are empty. In order to calculate *cosine similarity* (3), we need to fill every empty cell with a value, which in our case, we filled them with 0s then calculate their cosine similarity accordingly.

$$S_c(A, B) := cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} \tag{3}$$

The drawback of using the method mentioned above is that it treats empty ratings as a negative rating, which makes it hard to distinguish users/movies that are extremely different.

There are another method, which is called *centered cosine similarity*. In this method, we calculate the average rating for each user using their history ratings, then subtract the mean for each rating for each user. The advantages of doing this are a) missing values are treated as 0s; b) It handles ”tough raters” and ”soft raters”.

4) Disadvantage of Collaborative Filtering: The biggest disadvantage of using collaborative filtering method as a recommender is that it will only recommend movies that at least have one review, which means it would have issue handling a completely new movie, since collaborative filtering strongly relies on ratings rated by others users for the predicted movie.

B. Content Based Recommender

Another direction is to build a content based recommender. The proposed formula for a content based recommender is shown as (4), where $r(i, j) = 1$ if user j has rated movie i, and 0 otherwise. $y^{(i,j)}$ represents the actual rating rated by user j on movie i. $\theta^{(j)}$ and $x^{(i)}$ are parameter vector for user j and feature vector for movie i. Therefore, for each user j and movie i, the predicted rating is defined as $\theta j^{(i)})^T x^{(i)}$. The final goal is to optimize $\theta^{(j)}$

$$\min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} ((\theta j^{(i)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (\theta_k^{(j)})^2 \tag{4}$$

After getting the parameter matrix for each user using (4), we are able to predict the rating every single movie given by any user.

1) Disadvantage of Content-based Recommender: Similar to collaborative Filtering Recommender, contend-based recommender also has its own issue. Even though content-based recommender is able to handle the cold-start movie issue that we mentioned earlier, however, it will only recommend similar movies that a user has watched before but never will recommend something new (e.g. a new genre, a new director etc.) to a user. Since content-based recommender is very user-dependent, hence we are proposing a method that can capture each users taste while is able to have other users input to recommend potential new items to a specific user.

C. Content-Boosted Collaborative Filtering Recommender

While CF is a simple and powerful algorithm, it has several limitations. One of those limitations is that it cannot address the cold-start problem, which means that new movies without any ratings will not be recommended. To address this issue, we implemented the content-boosted method where we use the IMDb dataset to augment our CF algorithm. In vanilla CF, vectors used in correlation calculations are very sparse. In content-boosted CF, we build a content-based predictor to fill in the sparse utility matrix prior to calculating the correlation.

To build the content-based predictor, we used the following features from the IMDb dataset: genre, director, actors, duration, year, and country. ”year” and ”duration” was transformed into categories based on the interval. Years were categorized into ”oldest”, ”old”, and ”new” based on intervals ”pre-1950”, ”1950-2000”, and ”2000-2000”. Duration was categorized into ”short”, ”medium”, and ”long” based on intervals ”0-89 min”, ”90-119min”, and ”120+ min”. After this transformation and prepossessing of all features, they were combined and vectorized into bag-of-words feature. Using these features, naive Bayes classifier was built to predict the score (1-5) of reviews and fill in the empty cells of the utility matrix. If actual rating already exists, it was left alone. Once the utility matrix became full, we perform correlation calculation as usual and run CF. The resulting MSE was 0.74, seeing a slight improvement of overall quality in recommendation.

1) Cold Start Problem: The main point of the content-boosted algorithm was, however, to address the cold-start problem. To evaluate this, we removed rating information from around 150 movies. In vanilla CF, these movies would never be recommended. Although we do not have ratings for these movies, we have feature information about them and thus we are able to predict scores for them using content-based predictor. Through the content-boosted algorithm, we were able to achieve MSE of .86 on withdrawn ratings.

V. Result

In this project, we implemented 5 different methods in total (user-based CF, item-based CF, content-based (linear regression), content-boosted (linear regression) as well as content-boosted (bag of words)). Due to the volume of data, we did not perform cross-validation of each of the method, instead of we performed 80-20 training-test data split. We used MSE as our result metrics to measure each method's performance.

Method	MSE
user-based CF	0.77
item-based CF	1.36
content-based (linear regression)	0.87
content-boosted (linear regression)	0.85
content-boosted (bag of words)	0.74

As we can see the table above, we can conclude that so far content-boosted using bag-of-words method would have the best performance.

VI. Challenges and Future Work

One of the limitations of our experiment was that we filtered our dataset and kept only the top movies and top users. As a result, our dataset was biased towards them. For instance, top movies that garner many reviews are likely to be movies released by popular production companies or blockbuster movies with high budget. Since the distribution of movie characteristics are skewed and less diverse than entire dataset, it may not have benefited as much it could have from the content-boosted algorithm. Only considering top users can also bring bias as top movie enthusiasts may have different behavior than regular users. In the future, we may consider model-based approaches to collaborative filtering where it is unnecessary to store giant utility matrix so that we can keep the entire dataset.

References

[1] https://www.kaggle.com/netflix-inc/netflix-prize-data
[2] https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset?select=IMDb+names.csv
[3] https://www.cs.utexas.edu/~ml/papers/cbcf-aaai-02.pdf