Movie Recommendation System

CS 6550 Introduction to information retrieval project, Fall 2020

Sai Vamshi Dobbali
School of Computing
University of Utah
Salt Lake City, Utah, USA
u1266122@utah.edu

Abishek Krishnan
School of Computing
University of Utah
Salt Lake City, Utah, USA
u1261980@utah.edu

ABSTRACT

In this project, we have explored various information retrieval techniques to

a) Rank and find similarities among movies.
b) Predict the ratings of a particular user, based on the rating history of similar users.
c) Recommend Top-N new movies to a user, based on rating history of similar users.

Broadly, we have implemented,

i) Statistical filtering
ii) Content based filtering
iii) Collaborative filtering
iv) Top-N recommendation

and used RMSE as a metric to evaluate our models.

DATASET

We have used

a) The Movies Dataset from Kaggle and
b) MovieLens 1M dataset from GroupLens [1]

METHODOLOGY

1 Statistical filtering

The first step was to determine how to score a movie based on the ratings in the database, for which we have used IMDB's [2] weighted rating formula,

$$\mathrm{WR} = \frac{v}{v+m} * R + \frac{m}{v+m} * C$$

v - number of votes for a movie

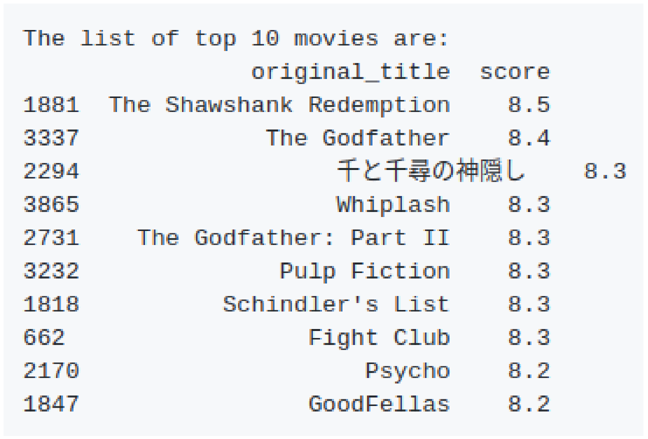
m - minimum number of votes for a movie
(We choose the 75th percentile)

R – rating of the movie

C - mean rating of all the movies
(For our dataset - ~5.6 on a scale of 10)

In addition to criterion of choosing m, we only considered movies which had a running time greater than 25 mins.

Based on these criteria, we sorted the movies in decreasing order to get the top 10 movies in our dataset.



```
The list of top 10 movies are:
                original_title  score
1881  The Shawshank Redemption    8.5
3337              The Godfather    8.4
2294                   千と千尋の神隠し    8.3
3865                   Whiplash    8.3
2731     The Godfather: Part II    8.3
3232               Pulp Fiction    8.3
1818           Schindler's List    8.3
662                  Fight Club    8.3
2170                     Psycho    8.2
1847                  GoodFellas    8.2
```

Figure 1: Top 10 movies based on statistical filtering

2 Content based filtering

The first step was to determine which content to choose, e.g., plot description, genre, director etc. We choose the plot description as our criterion to find similarities among movies.

Next, we cleaned our plot data by removing stop words and stemming. We proceeded to convert this description into feature vectors, by applying TF-IDF vectorization.

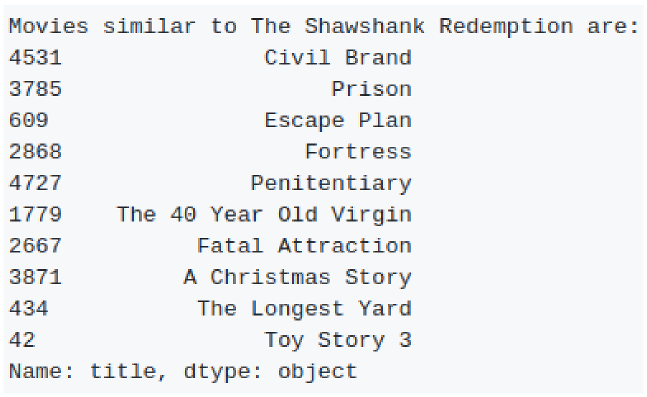
$$w_{ij} = tf_{ij} * \log\frac{N}{df_i}$$

w_{ij} - is the weight of word “i” in plot description “j”
df_i - is the number of plot descriptions that contain the term “i”

N - is the total number of movies in our dataset

We then calculated the pairwise cosine similarity among the movies.

For this implementation, we need to input a valid movie name, we then return the top 10 similar to it.



```
Movies similar to The Shawshank Redemption are:
4531               Civil Brand
3785                    Prison
609                Escape Plan
2868                  Fortress
4727              Penitentiary
1779    The 40 Year Old Virgin
2667           Fatal Attraction
3871          A Christmas Story
434             The Longest Yard
42                 Toy Story 3
Name: title, dtype: object
```

Figure 2: Top 10 movies similar to “The Shawshank Redemption”

3 Collaborative filtering

Both the above two techniques do not take user preference into consideration, they suggest same movies to all users, in order to suggest movies per user based on their preferences we performed user based collaborative filtering, it is a technique used to predict the movies that a user might like on the basis of ratings given to movies by the other users who have similar taste with that of the target user.

We predicted user rating for an unrated movie based on their cosine similarity.

Initially we hardcoded the rating of all the unrated movies to 3.0, for which we achieved a RMSE of 1.3.

To improve this, we predicted movie rating by taking average of all the user rating given to that movie, this improved the RMSE to 1.02.

Finally, we represented each user as vector of movies the user rated and computed pairwise cosine similarity matrix between the user’s feature vectors. We used these similarities as weights for each user rating and predicted rating for active user as weighted average of ratings given by other users.

Below is the formula we used for predicting the ratings:

$$Predicted\ rating(u,m) = \sum Similarity(u,u') * \frac{rating(u',m)}{Similarity(u,u')}$$

u – active user.

u’ – users who rated movie m.

This technique improved our RMSE to 1.01.

All the RMSE values are calculated using *k-fold* cross validation. We divided the dataset into 5 folds, trained the model on four folds and tested on the left-over fold.

4 Top-N recommendations

To implement this, we used SVD algorithm on the rated data to predict ratings for unrated movies.

Rated data is a matrix with rows as user-ids and columns as movie-ids. All unrated movies were given a rating of zero. After performing SVD on this matrix, we reconstructed back original matrix, which gave ratings for all the unrated movies.

We got a RMSE of 0.8721 using this technique on MovieLens 1 Million dataset.[1]

To recommend top 10 movies for a user, we filtered out all the predicted ratings for that user and recommended top 10 rated movies out of all predicted ratings for that user.

Below are the top 10 recommendations for user 10 along with the predicted ratings:

Movie	Predicted Rating
Running Free (2000)	3.194
Kansas City (1996)	2.561
Ladybird Ladybird(1994)	2.250
Big Blue, The (1998)	2.133
Dog of Flanders, A (1999)	2.091
Braindead (1992)	2.013
Cell, The (2000)	1.721
Class of Nuke 'Em High (1996)	1.643
Wonderland (1999)	1.553
Alien (1979)	1.373

Figure 3: Top 10 movies predicted for user-10 in our data

RESULTS

For statistical filtering, we were able to get the top 10 movies from the dataset based on IMDB's rating formula. For content-based filtering, given a movie title, we were able to retrieve top 10 similar movies based on the TF-IDF vectorization of the plot description. For collaborative filtering, for a hardcoded rating 3.0 to an unpredicted movie, gave a RMSE of 1.3, which we the improved to 1.02 by taking the average of all the user rating given to that movie. Using cosine similarity on feature vectors derived from a user improved our RMSE value to 1.01. Finally, for our Top-N recommendation system using SVD algorithm, we improved the RMSE to 0.8721. We successfully implement the above techniques to retrieve top movies to watch for a user, and our Top-N recommendation performed the best.

ACKNOWLEDGMENTS

We would like to thank the School of Computing, University of Utah, and Prof. Qingyao Ai for giving us this opportunity to explore information retrieval techniques as part of the class project for CS 6550 Introduction to information retrieval course, Fall 2020.

REFERENCES

[1] https://grouplens.org/datasets/movielens/1m/
[2] https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV#ratings