Movie Recommendation using Machine Learning

Joshua Bolcar
Department of Engineering Student
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060
bjosh@vt.edu

Abstract

There are many media streaming services today that do not have their own form of a recommendation system. A movie streaming service may have hundreds of thousands of movies. A dataset of this size takes a significant amount of time to read each entry. This experiment utilizes machine learning techniques to optimize the problem of filtering large dataset. The experiment used a large dataset of movies with their respective information and tags and sorted these values into a matrix. Through further filtering techniques, the algorithm can find a list of movies with similarities to the one given and ranks each movie by assigned weights.

1 Problem statement

As stated in the abstract, we will use machine learning filtering techniques to reduce the size and obtain a dataset that is manageable. We want to maintain a high accuracy, accuracy in this case will be recommending ten movies that are like the ones given and have a low compilation time. We expect that reducing the datasets by filtering out unwanted data will drastically reduce compilation times. We plan to use cosine similarity with matrices to accomplish our tasks and achieve our goal stated above.

2 Analysis of results

2.1 Initial testing

We originally selected our data from

https://www.kaggle.com/danielgrijalvas/movies

Which contains a csv file of 220 of the most popular movies from each year (ranging from 1986 to 2016) for a total of 6820 movies. The utilized pandas library to read in all of our movie information from a csv file and formatted it into a data frame. We then filtered our all the movies that has a lower rating because they are less likely to be attractive to the user. We implemented our algorithm to find similar movies to a movie given using the genre of a movie. We were able to form a list of movies that had similar genre and ranked in descending order based on the average rating given.

2.2 Problems with initial dataset

We tried to further our algorithm by applying new techniques, but we found some flaws with our dataset. While this dataset was concise, it lacked in some describing features we planned to use as tags utilized in our improved algorithm. This dataset had only a singular genre for each movie, many movies have multiple genres and can not be properly represented by one. The dataset also only had one actor listed for each movie. Many movies can have multiple star actors and we planned to use this as another feature for our algorithm. It also lacked a movie description where we could find key words. The final flaw was it only had a score based on critic reviews. We planned to use a weighted rating for each movie based on both user and critic reviews.

2.3 Improving dataset

We found a new dataset from

https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset

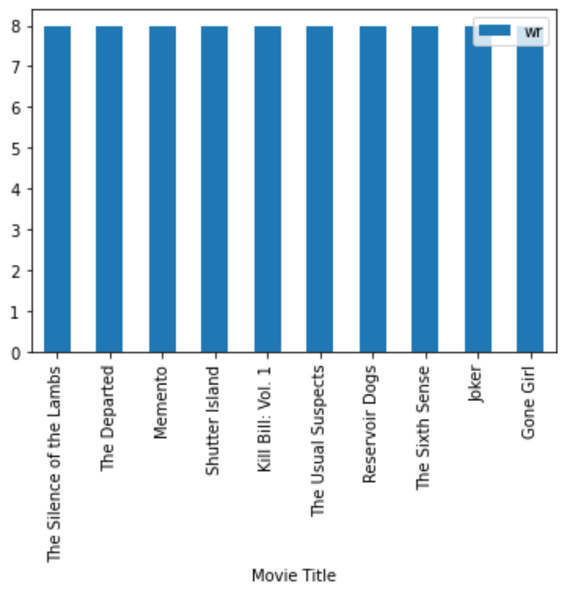
and this dataset contained 85855 movies. A larger sample size can improve our results and this dataset had the descriptors we were looking for. It has multiple genres, multiple actors, a movie description, directors, writers, and reviews from both users and critics listed. It also has top movies from different geographic locations so we can filter movies based on the user's preferences on location or language.

2.4 Filtering out unnecessary data

We took the average of the movie reviews and obtained an average rating of just over 5 out of 10. Initially we planned on filtering out movies below a 7 rating, we chose to set the threshold a bit lower to not filter out possible sequels or similar movies to the ones given. We filtered out all movies that received less than 5000 votes. The votes ranged from 99 to 2278845 votes, and we tossed the data with low vote counts to improve accuracy. Finally, we filtered out all movies that lacked in information necessary to apply our algorithm. We filtered the original 85855 movies down to a manageable 8655 movies.

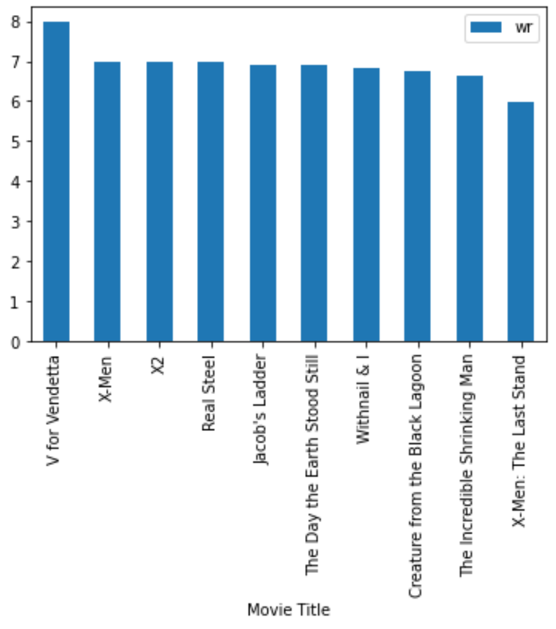
2.5 First algorithm

Initially we used k nearest neighbor classifier as a baseline experiment. We hypothesized it to be slower than our planned algorithm. It had long compilation times and our hypothesis was confirmed. The first step in creating our algorithm was to take a movie that the user enjoyed and find all movies similar with similar genres. This would filter the matrix to only contain movies with genres like the one given and rank them in descending order based on our weighted rate. The following is our results from running a genre recommendation for "Thriller" genre movies.



2.6 Improving the algorithm

We can now improve our algorithm using key movie descriptors as tags. We combine movie actors, directors, and genres together to form our tag descriptors. Then create a CountVectorizer and use our tag descriptors to create a matrix by transforming our vectorizer. Now this matrix can be used to perform cosine similarity to find a movie like one given. We tested the algorithm using the movie *Logan* which is an action/adventure movie with Hugh Jackman and Patrick Stewart as actors, and James Mangold as director. The top 10 similar movies were



There are multiple movies with Hugh Jackman and Patrick Stewart in the results, all the movies have the action, adventure, or thriller tag.

3 Conclusion

Our algorithm achieved our goal and found 10 movies like the given movie based on movie descriptors obtained using tags. We suggest that further research can be done by using co-occurrence with embedding key words from the movie description. This will be able to find words that commonly are found in the context of others. For example, an improved algorithm would be able to identify well known movie characters and identify similar movies based on spin-offs of the given movie or a movie that contains similar fictional characters.

4 Citations

Used libraries pandas, scikit, and matlab pyplot

https://scikit-learn.org/stable/

https://pandas.pydata.org/

https://matplotlib.org/tutorials/introductory/pyplot.html