Methods for Recommending and Analyzing Movies

Thomas Theisen

December 18, 2017

## Introduction

Recommender systems have proliferated in recent years with respect to the exponential amount of data generated by users like you and I. Companies like Netflix and Facebook use recommendation systems to predict the likelihood a user will watch a movie or click on an ad. Under the umbrella of recommendation systems sit two primary methods of prediction: Collaborative Filtering and Content-Based Filtering. In the following report, movie rating data, provided by Grouplens.org, will be used to explore the methods mentioned previously. In addition, movie information from IMDB.com will be utilized to determine if the words contained in a movie's plot can classify the genre in which the movie falls.

## Data Description

There were three primary datasets used in this analysis. The "ratings" dataset contains 100,000 observations, where each row is comprised of a user id, movie id, and the rating given by the user for the movie. All 671 unique users had rated at least 20 movies. A sample of the ratings dataset is shown below:

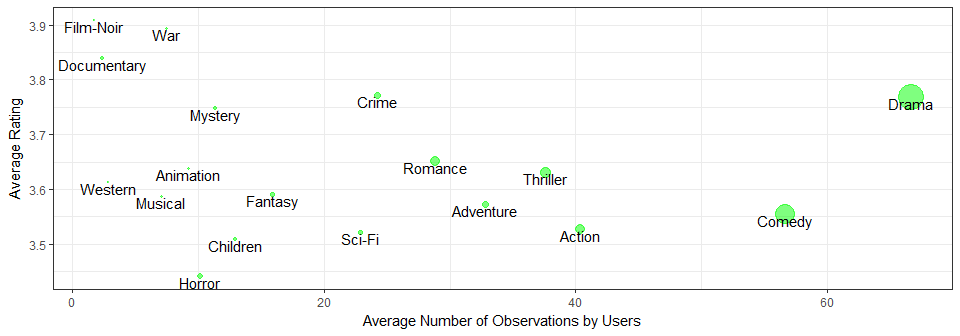
## userId movieId rating  
## 1 1 31 2.5

The "movies" dataset contains the information on 9,125 movies. This information includes a movie id, movie title, movie genre, and the movie release date.

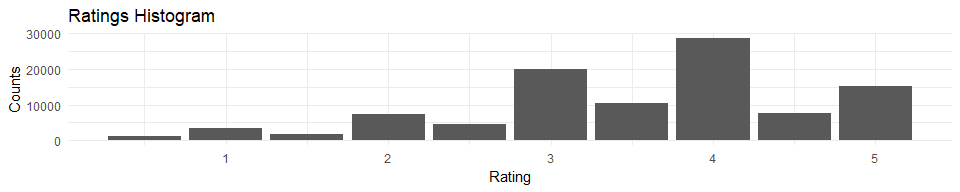
## movieId title genres   
## 1 "1" "Toy Story (1995)" "Adventure|Animation|Children|Comedy|Fantasy"

The "links" dataset was used to extract additional information from the IMDB API. Each movie had a corresponding imdb id, which allowed for the opportunity to gather additional information such as the movie's rating, runtime, director, writer, actors, plot, and imdbRating. This information was stored in a new data frame called "info".

The chart below shows that movies categorized as 'Dramas' are on average the most watched genre across all users. The size of each data point describes the number of total movies with that genre specification. Therefore, it makes sense that the most available genre of movies is also the most watched on average. Although documentary, war, and film-noir movies are watched less than 10 times by users on average, they maintain the highest average rating. Horror movies are by far the lowest rated genre. The primary difficulty faced while using this data was how a movies genre was specified. Very few movies are described using a single genre like "Action", but instead, a combination of genres such as "Action, Romance, War". In total, there are 902 unique genre combinations, with the large majority of combinations only occuring once. Therefore, the average ratings on the chart below are not strictly a single genre such as "Action", but the average rating of all movies that contained "Action" somewhere in the genre list. That is, each genres final average rating is influenced by external genres.



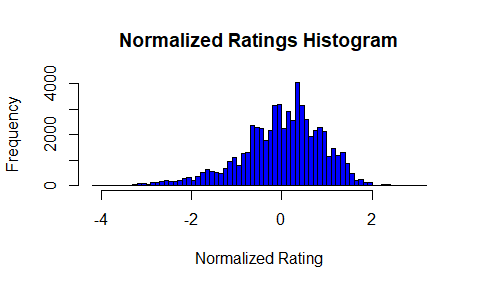
Below is a histogram describing the movie ratings given in the ratings dataset. It seems people are more likely to rate a movie on a whole number basis. The most observed ratings are 4.0, 3.0, and 5.0.



## Explicit Recommedation Systems

### User Rating Bias

Before beginning the recommendation process, a few pre-processing measures can significantly improve movie recommendations. Significant improvements in predictions can be made by normalizing user ratings to remove user bias. For example, person A might be a real movie critic who considers a rating of 3 to be quite good, reserving ratings above 3 for outstanding movies. Person B may be a casual observer who might give a rating of 4 if the movie is really bad, otherwise almost always a 5. This normalization process removes an individual users personal scale for what he or she considers good versus bad in terms of rating. The two popular methods of normalization are 'centering' where the average rating for all movies given by user u is subtracted from each individual rating and Z-score normalization. When computing recommendations in this analysis, the Z-score normalization method was used to account for variance in a users ratings. Below is a histogram describing the left skewed distribution of the normalized user ratings.



### Item Based versus User Based Collaborative Filtering

Item based collaborative filtering is an algorithm-based method of producing recommendations calculated by assessing the relationship between items (movies) given in the ratings dataset. The user based collaborative filtering method is identical in computation except that recommendations are generated based on the relationship between users. By transforming the ratings dataset from a long format to a wide format, a matrix is created where each user is a row, and the columns represent every possible movie the user could have rated. The ratings matrix itself was quite sparse because users on average watched only 149 movies of the possible 9,125. Due to memory issues within R, I had to remove any movies that had been watched less than 25 total times.

The three most popular similiarity computation methods are cosine similiarity, pearson correlation-based similiarity, and Jaccard similiarity. Cosine similiarity is calculated by taking two vectors and calculating the cosine of the angle between them. The correlation method is much the same, expect the correlation between the two vectors is computed. Jaccard similiarity is calculated as the cardinality of the intersection of two vectors divided by the union of those two vectors. Larger final values for any of these computations are indicitive of a higher similiarity between vectors.

In item based collaborative filtering, these two vectors are comprised of users who have watched both movies x and y. User based collaborative filtering uses the opposite approach, where the two vectors are based on each movie both user a and b have watched. Ultimately, the similarity between each pair of movies (pair of users) in the ratings dataset will be computed in order to begin recommending movies the users haven't rated before.

In both item and user based collaborative filtering the parameter k or n are used to dictate the number of most similar movies or users used in the recommendation computation, respectively. These parameters are tuned to optimize recommendations based upon the dataset. In order to demonstrate the process of collaborative filtering, the R package "recommenderlab" will be used.

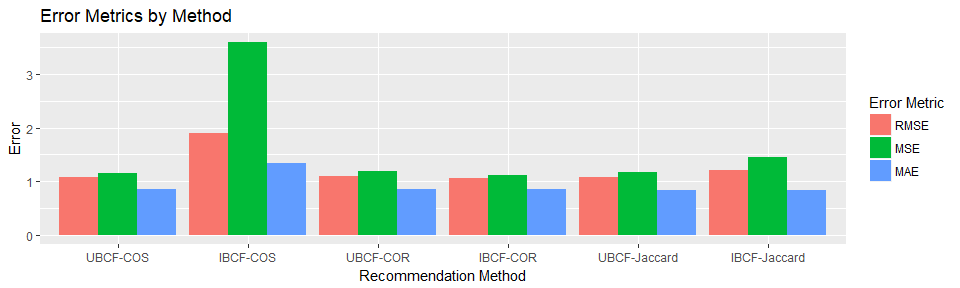
Below are the top 3 recommended movies for user 2. The results of item and user based collaborative filtering are given in order below.

## movieId title genres  
## 66 70 From Dusk Till Dawn (1996) Action|Comedy|Horror|Thriller  
## 7210 68157 Inglourious Basterds (2009) Action|Drama|War  
## 7662 81591 Black Swan (2010) Drama|Thriller

## movieId title genres  
## 643 778 Trainspotting (1996) Comedy|Crime|Drama  
## 889 1097 E.T. the Extra-Terrestrial (1982) Children|Drama|Sci-Fi  
## 6917 58559 Dark Knight, The (2008) Action|Crime|Drama|IMAX

### Accessing Error Between Collaborative Filtering Methods

The three most popular metrics of assessing recommendation system error are root mean squared error, mean square error, and mean absolute error. Using cross validation as a method of computing these metrics, the error statistics for user and item based collaborative filtering are given below including the similiarity method used to compute them. Overall, it appears the cosine similiarity, user based methodology is the most appropriate method for recommending movies for users in this dataset.



### Problems with Explicit Recommendation Systems and a Case for Content-Based Recommendation Models

Content-based systems use metadata or the properties of items to generate similarities. For movies these properties include details like release date, directors, writers, and actors. The dataset "info" contains this information, but the time needed to clean and transform this data into usable form is substantial giving the varying number of directors, writers, and actors in each movie. Thus, I am leaving this up to future work and research.

Content based recommendation models are generally considered superior to collaborative filtering methods for the following reasons: The results tend be highly relevant and transparent due to the similiarities between movie properties and not explicit movie ratings, and new movies can be recommended immediately because the properties of the new movie are known. Collaborative filtering recommendations give no direct indication to why a given recommendation may be relavent to a given user, as witnessed in the recommendations above. User 2 was given multiple recommendations that were action heavy, when in fact he or she rates western, war, crime, and animation movies more highly.

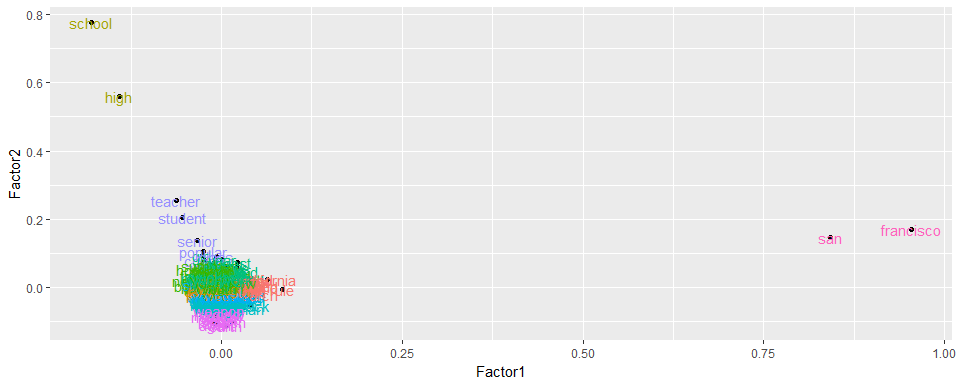
The similarity properties that drive true and understandable recommendations are those where plot descriptions are similiar, actors or directors overlap, or similiar genres are reported. As many researchers say, collaborative filtering gives 'dumb' recommendations. Content based modeling gives Netflix the opportunity to list relavent '1980's mystery romances' that are similiar to other '1980's mystery romances' a user has already watched. Out-of-the-box recommendations coming from companies like Netflix utilize an off-by-one approach where an individuals love for '1980's mystery romances' might suggest to Netflix this individual will enjoy '1980's mystery thrillers' as well.

## Classifying Movie Genres using Movie Plot Descriptions

To begin the process of movie genre classification, pre-modeling steps such as string cleaning and data structure alterations were necessary. The first step in the process was converting all the plot descriptions to lower case, removing all punctuation, and any 1 or 2 letter words. Next, "stopwords" were removed. Stopwords are high frequency words that create minimal additional meaning to most sentences. Next, each individual word in each plot description was checked to see if it could be reduced down to a root word through a process known as stemming. For example, "accounts" would become "account". Next, the data frame structure was coerced into a structure known as a document term matrix. This structure is formatted to allocate a single row to each movie, where 1's or 0's are denoted on the columns as a method to indicate the presence of a word in the movie's plot description. To further reduce the number of word features, any word that didn't occur more than 25 times throughout all movie plots was removed. Future research into the appropriate word frequency cutoff would be interesting. After this whole process, 1096 words remained. A glimpse of the document term matrix is given below.

## Terms  
## Docs abandon abil abus academi accept accid accident account accus across  
## 114 0 0 0 0 0 0 0 0 0 1

Given the finalized data structure, factor analysis (using a 'varimax' rotation) was used as a dimensionality reduction method to create factor loadings for each word. The cumulative variance explained by the first three loadings was only 0.006, which is evident in the plot below. Future research regarding methods of dimensionality reduction that could bolster the amount of cumulative variance explained could lead to significantly better genre predictions. Next, the factor loadings were supplied to the kmeans algorithm to create latent variables or "superwords". A scree plot indicates roughly 10 clusters are appropriate. That is, each of the 10 superwords is comprised of a collection of individuals words that were closest to one another in the feature space. It's interesting to see the word pairs high school, teacher student, senior popular, and san francisco were correctly grouped together, in reference to our own knowledge of pairs of words that are most closely associated in everyday life. Otherwise, every word had factor loadings, in two dimensional space, between -0.125 and 0.125 on the x-axis and -0.125 and 0.10 on the y-axis.



To create the final data structure, a new data frame was created to denote which superwords occured in each movie plots description.

The naive bayes algorithm was used as the method of classification. This classification techinque is based on Baye's Theorem with an assumption of independence among predictors. That is, each superword feature is regarded as unrelated to the presence of any other superword. Laplace estimation is utilized as a smoothing techinque to solve the problem of any zero frequency features between training and test sets. Individual naive bayes models are created for each genre. That is, the results from 18 separate naive bayes models are used to predict the final genre classification vector for each movie. Model probability cutoffs were decided upon for each model by choosing the cutoff value that returned the highest testing accuracy.

Final model classification accuracy was slightly underwhelming. The final model resolved to an over classification (rate) error metric of 1.96, and an under classification metric of 1.76. That means the model was predicting roughly the correct number of listed genres for each movie, but often the wrong genres. Only 20.7% of movies were predicted perfectly. In total, the model predicted at least 1 genre correctly 47.9% of the time. These percentages fluctuate when a seed isn't set before using k-means.

## At Least One Genre Predicted All Genres Predicted  
## 1 0.4788983 0.2070191

## Conclusion

Recommendation systems have become the driving force behind many companies revenue streams. Collaborative filtering methods are common, but content based recommendations allow for diverse and transparent recommendations between business and user. More time spent tuning model parameters and investigating alternative methods of text clustering could lead to significant improvements in overall genre prediction accuracy. Aggregating synonyms and removing words of neutral sentiment have been shown to create more robust and accurate classification models. Future research using this data will be spent creating a content based model.