**Movie Recommendation System**

**Thomas Rauzi**

**Data Wrangling**

To build the movie recommendation system, we utilized three data sets: movie metadata, keywords and ratings. The data is from Rounak Banik’s Kaggle page. Movie metadata originally contained 45,466 movies with 24 variables. After dropping 18 variables as well as turning movie genre into dummy variables, there are 33 variables. Additionally, after dropping some duplicated movies and movies missing data, I ended up with 45,432 movies. Keywords contained movie id and keywords for 46,419 movies; however, after dropping 987 duplicate movies, there were keywords for 45,432 movies. Ratings data set contains 26,024,289 ratings.

**Exploratory Data Analysis**

**Movie Metadata**

For movie metadata, there are 3154 different movie genres. The interquartile range indicates the middle 50% of genres only contain 1 through 3 movies, which suggests the genres are too specific. Additionally, there are 2652 movies that lack genre data, which accounts for about 5.8% of the movies. The movies with unique value for genre will not yield reliable predictive results since there is a very limited number of them. For example, there is only one movie that is a drama, comedy, romance, family, and foreign, so that movie cannot be directly related to other films.

Since all movie genres are a combination of 18 genres, I created a column for each genre. Then for each of the 3152 genres, I put a one in the column if the genre contained that term. For example, if the genre is action comedy, then both the action and comedy column have a one while the remaining genre columns have zeros.

Average vote ranges from 0 to 10 where zero means the movie has not been rated. The mean vote average is 5.6 while the median is 6. The data is slightly left skewed likely due to the 2,993 missing values, which are encoded as 0. The IQR is 1.8, which suggests the data is mostly centered, and the majority of data ranges between a rating of 5 and 6.8. There are 3,597 potential outliers that are less than 2.3 or larger than 9.5. Vote count data is heavily rightly skewed where 75% of movies have only 34 or less votes, and the max movie has 14075. Approximately 6% of movies have 0 votes.

**Ratings**

There are 26,024,289 ratings that range between .5 and 5. The mean and median vote is 3.5, which suggests the data is symmetrical. There are 270,896 users with the median rating of 4 and mean vote of 3.7, which suggests the data is slightly left skewed, which is confirmed by the boxplot. There is one user who rated 18,276 movies while the mean and median number of movies rated is 92 and 30 movies, respectively.

The boxplot of median ratings for each user shows 2,268 potential outliers on the left side of the distribution. If a user only rates a few movies, their limited ratings could cause the low median rating. However, after plotting vote count and median rating while coloring by potential outlier status, shows there vote count dose not explains the outliers (figure 1).

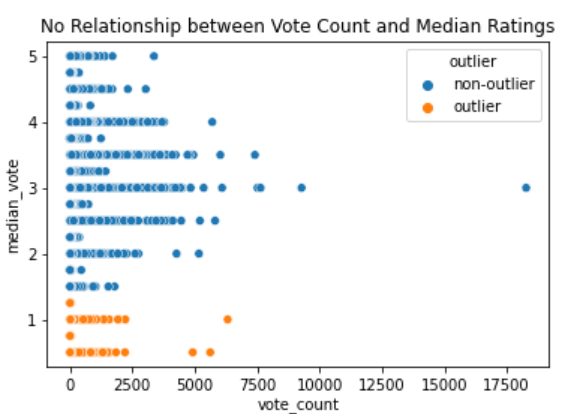


Figure 1

**Models**

When building the recommendation system, I focused on content and collaborative recommendation systems. Both systems have their strengths and weaknesses. Content based systems compare movies based only on their features. Since a new movie’s features are compared to other movies, content-based systems are easily adaptable. Conversely, item based collaborative filtering compares movies by how they are rated by users. Thus, a movie that is rated similarly by users are considered similar.

Although collaborative filtering doesn’t adapt quickly to new movies, it provides more robust results. When people enjoy more than one type of movie, their ratings will be similar among the different types of movies. Therefore, a collaborative type of filtering will introduce users to a wide range of movies. Before, we can apply the modes, the data must be formatted. The data has to be formatted differently based on which method is utilized.

**Content Recommendation**

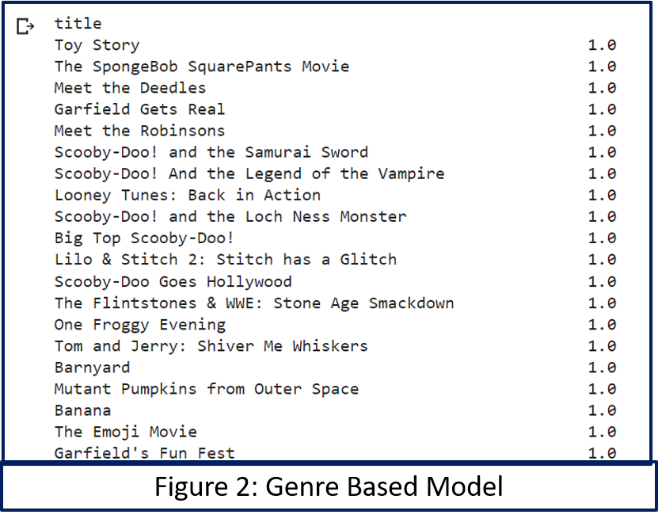
For the content-based recommendations, we utilized three different sets of variables. First set of variables only contains genre. Second set has genre and popularity while the last set has genre and keywords.

Due to computing power limitations, we are unable to run a model with all 45,418 movies, so we will select half of the movies. Initially, we were going to randomly sample the movies; however, this resulted in a lot of obscure movies. Since we are not familiar with a lot of the movies that were left, we were unable to judge how well the recommendation system performed. Thus, we decided to select the 22,709 most popular movies. Movie popularity is based on the number of votes to avoid issues with people's opinions. The more votes a movie has the more people who have seen the movie, so even if the movie did not perform well, with enough votes the movie should be well enough known.

Selecting the movies based on number of votes does mean that obscure movies will be left out; thus, the recommendation system will not perform well for people who enjoy those types of movies. However, we feel the majority of people like more well-known movies, which is why the movies are well known. To cater to the largest number of people, we decided to go with the more well know movies.

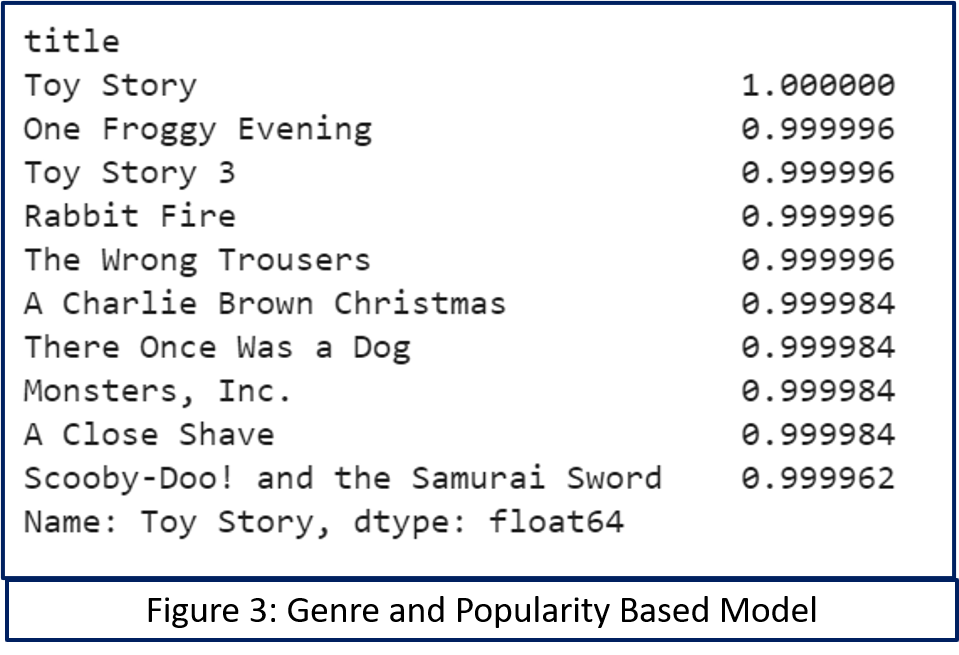
The first model will use only genre to make a recommendation; therefore, the assumption is that a person chooses to watch a movie because they enjoy a particular genre. Although this assumption is likely true to an extent, its recommendations will be limited because people enjoy movies based on more than just genre, and they like more than one genre. Although the recommendation will be limited, it will provide a good base model. Since the genre data is binary, we use Jaccard similarity for this first model.

Although the recommendations based solely on genre is very basic, it seems to have some relevant suggestions. As shown in figure 2, it recommends SpongeBob and Scooby-Doo for Toy story, which are animated kids movies. Since there are so many movies ranked as 1, you have to go through a lot of movies before it recommends one of the other Toy Story movies. Thus, there are a lot of really close movies that are missed.



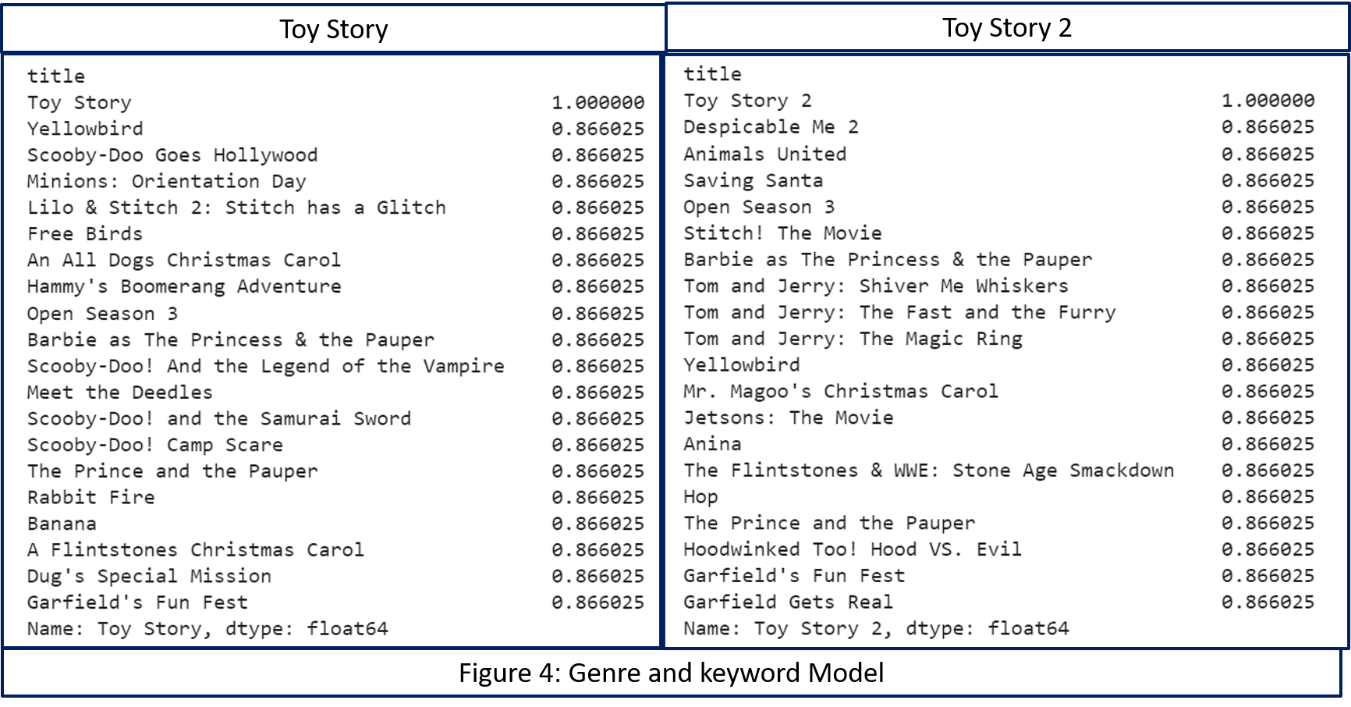
For the second model, average vote and genre will be included. Since genre only based model produced so many movies ranked as one, we included the average vote, so if a person watches a popular movie, it will recommend other popular movies in the same genre.

Since the vote count data ranges between 1 and 10, the data is normalized. normalizing the data will ensure the distance calculations do not put more weight on vote average than the dummy variables. Figure 3 shows the Genre and average vote model performed better because it recommended Toy Story 3 in the top 10 similar movies to Toy Story, and it recommended two other Toy Story movies in the top 50. However, there are still a lot of movies with a similarity score close to 1.



Although the recommendation system with both genre and average vote count was able to recommend a sequel of Toy Story in the top 10 results, it did miss Toy Story 2, which maybe a result of the lower popularity of the second movie. Also, since it only uses genre and popularity, there are a lot of results to go through. To help the system better recommend sequels and more similar movies, we added keywords as a feature. We utilized Term Frequency inverse Document Frequency Vectorizer (TfidfVectorizer) package to deal with keywords.

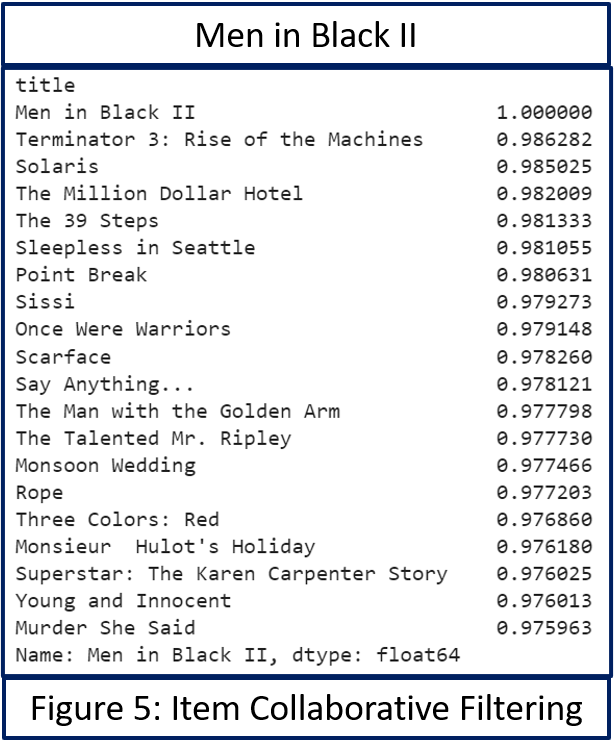
The recommendation system does return Toy Story of Terror for both Toy Story and Toy Story 2. Toy Story has a score of .85, .75, and .87 with Toy Story 2, Toy Story3, and Toy Story of Terror, respectively. Thus, the system does give out reasonable scores; however, it seems like the system is still scoring more based on genre; which is why there are many movies with scores close to 1.



**Item Based Collaborative Recommendation**

For the item collaborative system, we are using only the users, the movies they watched and the ratting they gave the move. To perform collaborative filtering, we must get the data into an item matrix. Item matrix has a row for each movie and column for each user, and each cell has the user’s rating. Before we can create the model, we merged the title and movie id columns of movie metadata with the ratings data frames, which left us with 9,939,845 ratings given by 264,541 users over 3,993 movies.

When trying to reshape the new dataset into an item matrix, our system kept crashing. To reduce the dataset, we decided to drop users with only few reviews; however, after trial error, we had to drop any users who had fewer than 94 ratings, which were 90 percent of users. Additionally, we removed any movie with fewer than 32 reviews, which is the 30th percentile. After reducing the dataset, 26,666 users rating 3,967 movies for a total of 5,019,231 ratings. Since the item matrix was 95% sparse, we filled the missing values with zeros Using zeros to fill the missing values made sense because a rating of zero implies the person has not seen the movie.

 By reducing the dataset, we lost the Toy Story movies, which means we cannot directly compare the previous results with collaborative filtering results; however, we can still judge the effectiveness of the recommendations based on the movie it recommends. Item based collaborative filtering recommended Terminator 3: Rise of the Machines as the top pick for people who watched Men in Black II (figure 5). Since Terminator and Men in black are sci-fi action movies it seems the recommendation system performed well. However, the system also put Sleepless in Seattle in the top 5 picks for Men in Black II.

Reducing the data or relying only on users’ ratings may have caused the mixed results for the collaborative filtering. When we reduced the number of users based on how many movies they reviewed, we may have removed other movies more similar to Terminator. Alternatively, since Terminator and Sleepless in Seattle were rated similarly by people who watched both, it is possible that the two movies share something in common.

Although recommending Sleepless in Seattle based on Terminator seems strange, the diversity of the recommendations is one of the benefits to collaborative filtering. Collaborative filtering introduces people to movies they would not likely have watched, but still enjoy. In other words, this seemly strange recommendation may not actually be that strange.

**Future Work**

Both the content based and collaborative filtering were good first steps in creating a movie recommendation system, we would like to use more features in the content-based approach and try a hybrid system. Since we only utilized keywords and genre, our results were quite vague; thus, I would like to include either a description or plot for each movie. We then can use natural language process (NLP) to find similar movies.

Although exposing users to a wide array of movies is a benefit of collaborative filtering, ideally the model would recommend mostly similar movies with a few diverse one. Most of the recommendation in our collaborative item-based model were diverse movies. To improve this aspect, we would like to create a hybrid system that uses ratings as well as movie features to make suggestions.