Capstone Report

By: Michael Rong

Introduction

This report is a condensed summary of our selected Capstone project that explores the use of Natural Language Processing (NLP) to improve movie rating predictions. The project consists of exploring and processing data to build several machine learning models which are able to accurately predict the ratings of movies based with the help of its reviews. The report will begin with background information revolving the topic and selection of the project problem statement, followed by a summary of where the dataset was obtained. The data cleaning and processing techniques will be covered in depth. The report will also further explain the models that were used as well as the results and insights obtained from the machine learning models, followed by a conclusion section that summarizes the findings and discusses the struggles encountered during the project.

Background and Project Selection

NLP is not a new field of technology, but the applications in industry have continued to advance. Large organizations have found significant value in using NLP to gain insights from customer reviews and social media trends. As a result, the use of NLP has become increasingly popular in many fields, including the media industry. Because of this, I believe that understanding the language used in movie reviews can provide valuable insights for filmmakers and scriptwriters, allowing them to cater their work to focus on specific genres or themes that are likely to receive positive reactions. Given my interest in media companies, I decided to explore this idea further by using NLP to analyze movie reviews and improve movie rating predictions.

The Datasets

The data in which we used was obtained from MovieLens, MovieLens is a company that does their own research on movies and collects and organizes them to give generated recommendations to users based on the movies they liked. The data was downloaded from a zip file consisting of data being used to analyze tags of movies which were given in the MovieLens website. We took only two of the various files which were provided, the 'raw' JSON file which consisted of 84661 entries and 7 columns originally, and a second JSON file 'reviews' which contained over 2.6 million individual reviews.

The raw dataset was an overview of the movie's credits, containing the title, starring actors, director, and average rating, along with other miscellaneous columns for identification. The review dataset only had the item_ID which was necessary to join the two datasets on later, and the review texts themselves.

Cleaning and Processing

The data cleaning and processing were first performed on each individual dataset, and then the finalization of the data was done after joining. The movie dataset required the most cleaning and processing, including feature engineering.

Initially, we only noticed missing data in the 'directedBy', 'starring', and 'dateAdded' columns, but we realized that some entries contained only white space. We replaced those entries with null values and dropped the 'dateAdded' column, as it was not relevant to the movie's quality.

Next, we dropped null values and checked for duplicate entries, but found none. We extracted the movie's year from the title, which was contained within parentheses, and created a new 'Year' column. Rows with null year values were also dropped.

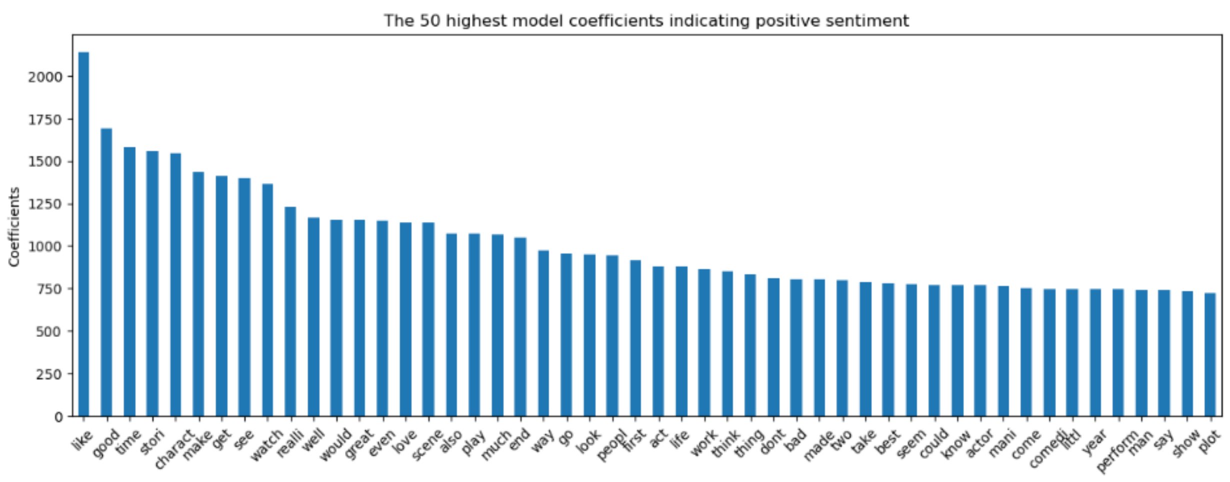
The 'avgRating' column had a scale of 0.5-5, which we converted to a scale of 1-10 for better interpretability. We removed the 0 values from the column and created a 'Sentiment' column, where a value of 1 represents a positive sentiment and 0 represents a negative sentiment.

We checked for outliers and reduced the number of starring actors to a maximum of 5, as most movies had 5 or fewer credited actors. However, there were some extreme outliers, such as having 30 directors. We created new features, including the number of movies credited to a director and a 'Celeb_score' that averaged the actors' appearances in each movie.

For the movie data, we produced a correlation heatmap check if the columns were appropriate for a linear regression model. Most elements had low correlation except for the 'Sentiment' and 'Avg rating' columns, which we decided to keep since one was the target variable, and they were related.

Next, we processed the review data by sampling it to reduce the size for computational efficiency and removing null values after joining. After the join, we performed our train, validation, and test splits. We used a Term Frequency-inverse document frequency vectorizer to determine the importance of words in the review texts and gained useful insights from it.

Below is a graph of the highest model words that indicated positive sentiment:



When performing the same analysis for negative sentiment, we encountered something strange, most of the search indicated the negative sentiment was due to numeric values such as years. This will be addressed in the areas to improve section later in the report.

By the end of the data processing, the vectorized words were joined in their respective train, validation, or test set. And these sets of data were saved to CSVs to be ready for modeling.

Modeling and Insights

The approach for our modeling process followed these steps:

1. We started by creating a linear regression model using the pre-nlp dataset. This model served as our baseline, which we aimed to beat in terms of performance metrics with consecutive models

2. Included the vectorized words into the data, and ran a ridge regression model.

3. We also ran a random forest regressor model.

4. The last model we would implement is a Support Vector Machine (SVM)

To evaluate the performance of each model, we used the prediction score or R-squared score (depending on the model), as well as the Mean Squared Error and Root Mean Squared Error.

To better understand this analysis, a brief summary of each performance metric will be given.

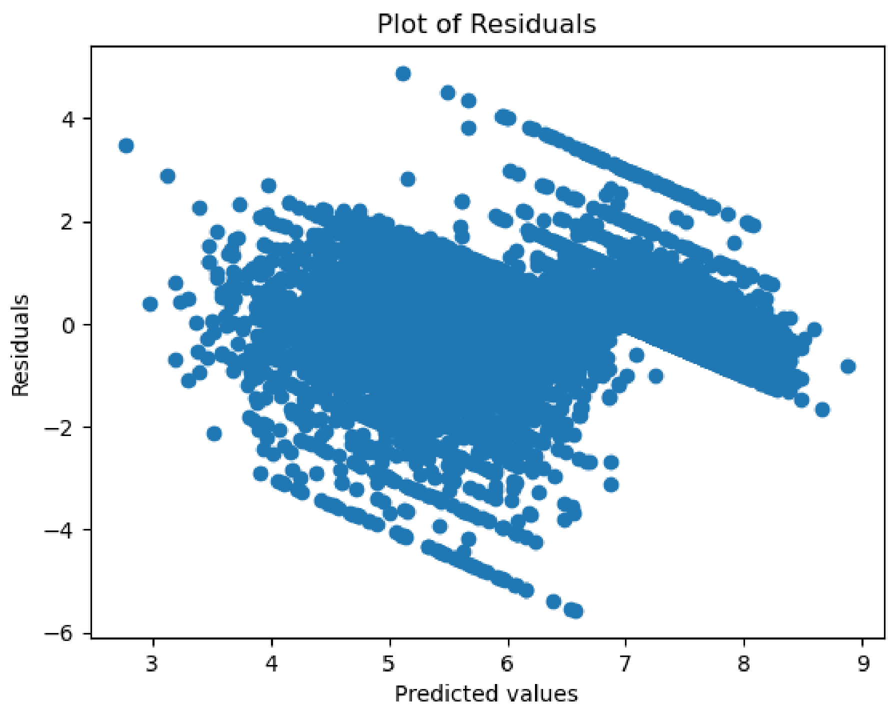
R squared measures how much of the variance in our dependent variable is explained by the model.

Mean Absolute Error (MAE) and and Root Mean Squared Error (RMSE) both measure the difference between predicted and actual values, with RMSE placing more weight on larger errors.

Our baseline model without NLP had an R squared value of 0.433 for both training and validation scores.

We then tried a more advanced model using ridge regression with hyperparameter tuning. This model had an R squared value of 0.62 on the training set and 0.52 on the test set, with an MAE of 0.65 and an RMSE of 0.92. This represented a substantial improvement over our baseline model. In addition, the MAE and RMSE seemed quite reasonable as this was across a range of 1-10

We also plotted the residuals, also known as the errors of the predicted values from the actual values and checked the shape of the scatterplot. The rule of Homoscedasticity suggests that this may not be the best data for a linear regression model, as a good linear model would show us data of residuals that look more randomized.



With this information now known, we used a more decision-based model, the random forest, to attempt to see if this would give us further improvements. Due to time constraints the number of estimators for this chosen was relatively low (10). The random forest model performed on our model gave us a train r squared of 0.91, a validation r squared of 0.6, MAE of 0.6 and a RMSE of 0.9

Although this is very overfitted towards the training data because of the low number of estimators, the performance did see an increase in the validation set, as well as a lower MAE as well as RMSE, overall, the random forest had performed better.

Lastly, we created a SVM model, however this piece of code was executing for a very long period of time, the model was unfortunately unable to run due to the hyperparameter optimization to be done in the pipeline that we had created.

Shortcomings/Improvements

During the project, several issues were encountered that could be improved in future iterations. Firstly, regular expressions could be used to remove numerical characters from the word search prior to vectorization, as their presence appeared unusual in the negative coefficients of word frequency.

Another issue that arose was the time taken for some models to complete. In some cases, these models took an excessive amount of time to complete, which impacted the efficiency of the project.

Aside from optimizing hyperparameters in my existing models, to further improve this project, I would look into the use of web scraping techniques to obtain additional information from the IMDB website. As well as looking into more advanced models such as XG boost or using methods such as neural networks to look for clusters in data.

Conclusion

This report is an executive summary of my Capstone Project, using NLP to improve movie rating predictions. This report explored all the processes done throughout my project as well as summarized the cleaning, feature engineering, modeling, and insights of the project. Overall, the findings of this report show the potential of using NLP to improve movie rating predictions, I believe the goal of this project was met.