IMDB Rating Sentiment Prediction

Ayo Ayiloge, Odeya Russo, Joseph Crosby, Caitlin Ree, Suraj Rajan, Wyatt Stone
UCLA Department of Statistics and Data Science

Fall 2023

Introduction

If you're looking for public opinion before selecting a film to watch for movie night, you might want to take a look through IMDb. The website is an online database of information related to films, television series, podcasts, home videos, video games, and streaming content online. In fact, IMDb is the world's most popular and authoritative source for movie, TV and celebrity content, at least according to... IMDb. Key features of the website include:

1. Movie/TV Show Listings
2. Actor & Crew Profiles
3. User Ratings & Reviews
4. Release Dates
5. Awards information, etc.

For our project, we take a look at 50,000 movie reviews pulled from IMDb. The dataset contains one column for the reviews and another with the sentiment, either "positive" or "negative". Our goal is to analyze the keywords in the reviews and use them to predict the sentiment of each review.

Pre-Processing and EDA

Lexicon and ETL

To conduct the binary sentiment classification task on the IMDB movie reviews, we make use of Kaggle lexicons made available to us[1]. We start by loading the IMDB review CSV file as well as the negative and positive lexicons. To better prepare the IMDB review dataset for transformation, we removed some remaining HTML formatting (line breaks: `<br /><br />`) from the reviews. We also added a review_num column to simplify the process of working with the dataset.

The negative and positive lexicons each only contained a list of words so we added a column to both lexicons that contained the corresponding sentiments ("positive" or "negative"), then we merged the two lexicons into one larger opinion_lexicon.

After preparing our lexicon, we break down all our reviews into its individual words, grouped by review number. Each row in this new dataset represents one word from one review. Using this new data frame, we also quickly calculated the total word count for each review. This would be used later for Term Frequency(TF) calculations. Then came a crucial step: taking this data frame of review words and performing an inner join to our opinion lexicon words. This filters our data frame of reviews broken down by word for only the words that are in our opinion lexicon. This is huge because it saves us from having to worry about removing any personal pronouns, determiners, coordinating conjunctions, and prepositions from the reviews. After performing this step, we take a deeper look at our new data frame using a frequency chart and word cloud:

[1]Lexicon was provided by professor and can be found **here**

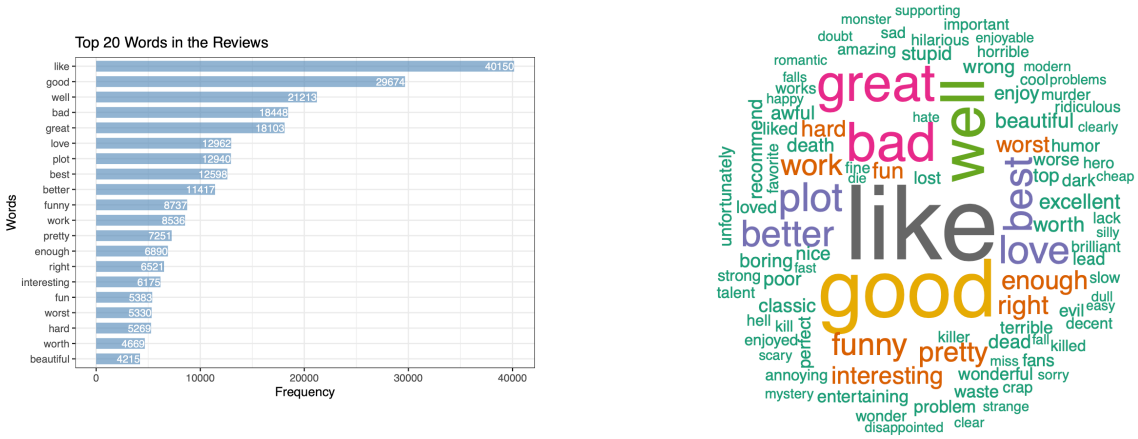

Figure 1: (a) Top 20 Lexicon words found in reviews; (b) Wordcloud with top words larger words more common

We can see that the most common word is 'like', showing up over 33% more times than the next most common word. Other frequent words include "good", "well", "bad", and "great". We don't see any stop words among our most frequent words, indicating a successful dataframe join.

TF-IDF and Numericalization

Although our reviews are now filtered to only include words from the lexicon, we still need to remove some more words; those that don't give us much useful information about the sentiment of the reviews. We have over 5800 unique words from the lexicon in our dataset and utilizing them all as features will be far too computationally expensive.

This is where Term Frequency-Inverse Document Frequency(TF-IDF) comes into play. We use this to perform dimensionality reduction on our dataset and remove words outside of a set bound. For the Term Frequency calculation, rather than finding the # of times a word appears in a review / # of words in the review, we found the # of times each word appeared in the entire dataset / the # of words in the entire dataset. Doing this essentially gives us an average TF for each word and limits us to only one TF-IDF for each unique word rather than 50,000.

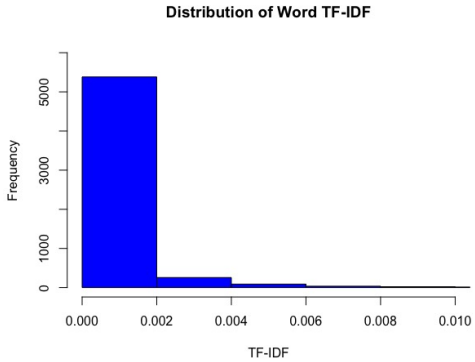

Figure 2: Distribution of TF-IDF for lexicon words in our reviews

The TF-IDF values ranged from 1.1e-5 to about 0.0315. The vast majority of the 5817 lexicon words, almost 5400 words, had a TF-IDF below 0.002. Since only a small portion of our words had a TF-IDF above this, we decided to make this a minimum cutoff. We did not need a maximum cutoff for the TF-IDF since the stop words had already been taken care of. When filtering for words with a TF-IDF >0.002, we are left with 435 words or "important features".

With these remaining words, we get a count of how many times each word appears in each review, then we pivot this data frame so that each column is one of 435 words, each row is a review number, and each cell value is the frequency of that word in the review. This new numericalized data frame is combined with the columns from our original dataset (excluding the review column) to create our final dataset. 71 of the 50,000 reviews did not contain any of the 435 words so the NA values in those rows were replaced with zeros.

The last step in our preprocessing work was to split our dataset of 50k rows and 437 columns (1 row_num + 1 sentiment + 435 words) into testing and training sets of 25k rows each. We now have balanced, numericalized datasets ready for modeling.

Analysis

With our data now properly filtered and numericalized to the relevant 435 lexicon words, we were able to begin modeling and predicting the sentiment of each review in the testing dataset based on the training dataset. We elected to evaluate four different models: logistic regression, linear discriminant analysis, quadratic discriminant analysis, and a random forest. This gave us an even split between discriminative models and generative models for the purpose of study.

Logistic Regression

Logistic regression is a discriminative model often employed for classification problems, much like this one where we are attempting to classify whether a movie review had a "positive" or "negative" sentiment based on the contents of the review.

The first step of employing logistic regression was a slight alteration of the data. We had to convert the "sentiment" variable from a character denoting "positive" and "negative" to an integer – 1 for "positive" and 0 for "negative." This allowed us to employ the function to actually do the logistic regression.

After conducting the logistic regression on the training data, we were able to apply it to the testing data to predict the sentiment of each review in the testing data. These were the results of the data:

		Predicted	
		Negative	Positive
Actual	Negative	10198	2276
	Positive	1763	10763

Figure 3: Logistic Regression confusion matrix

As we can see, the model was better at accurately predicting the positive reviews than the negative. It had an accuracy of 85.9% for the positive reviews, against just an 81.8% accuracy for the negative reviews. Overall, the logistic regression yielded a total accuracy of 83.8%.

Cross Validation and Regularization

After conducting the initial logistic regression on the training and testing datasets, we wanted to make sure that our model did not overfit to the training data and can generalize well to other, new data. To do this, we looked to L2 (Ridge) Regularization. To determine the parameter, lambda, which controls the strength of the penalty term, we used 5-fold cross validation. Interestingly, the cv kept selecting the smallest λ in our range as the best choice, even as we continued to feed the cv model a lower range. It became clear the the best results/accuracy came from a model with no penalty. In fact, when using 5-fold cv directly on our training

dataset with no penalty term, we get an accuracy of 83.9%, slightly higher than the initial model. This tells us that our model is stable and generalizes well across different subsets of data.

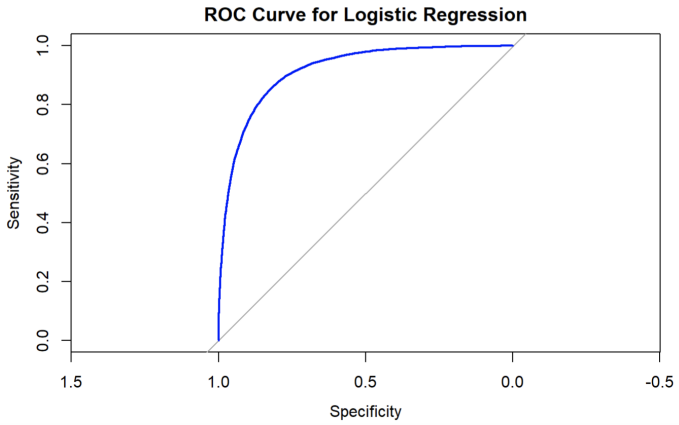

Figure 4: Logistic Regression ROC curve

The ROC curve for logistic regression shows a relatively large area under the curve, indicating that the model fits well.

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a generative model, unlike the discriminative logistic model, which studies the joint probability distribution as opposed to the purely conditional probability in generative models.

First, we must make an assumption that the conditional probability of the review sentiment based on the words found in the review follows a multivariate normal distribution with a fixed mean vector and covariance matrix. This step of the process will also be repeated when we conduct Quadratic Discriminant Analysis (QDA) later. However, the main difference between the LDA and the QDA is that here, we will assume that the classes have a common covariance matrix between them.

The methods of conducting LDA are largely the same as the logistic regression, as we use the same data set which alters "positive" and "negative" to 1's and 0's, and the process of generating the table is virtually the same by using the training data to predict the testing data.

		Predicted	
		Negative	Positive
Actual	Negative	10100	2374
	Positive	1750	10776

Figure 5: LDA confusion matrix

The accuracy of predicting a positive sentiment on the testing data was 81.9%, and the accuracy of predicting a negative sentiment on the testing data was 85.2%, so compared to the logistic regression, both the true positive rate and true negative rate increased, and the true negative rate was higher than the true positive rate. The overall accuracy of this model is about 83.5%, which is an improvement from the logistic regression model.

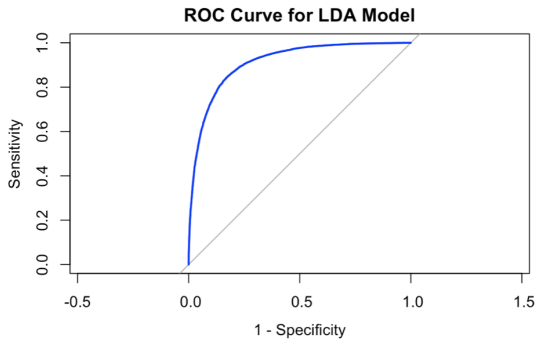

Figure 6: LDA ROC curve

This is the Receiver Operating Characteristic (ROC) curve for the LDA model, and what this helps visualize are the true positive and false positive rates. The more bowed out the curve is, the better the model performed in terms of accuracy, and the one of the LDA model is very bowed out, with a large area underneath the curve, so we can assume this model was strong in predicting the sentiment of the review based on the words in the review.

Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA), another type of generative model, is closely related to LDA. QDA retains one of the core LDA assumptions that the observations within each class are drawn from a multivariate Gaussian distribution. However, as stated previously, an important distinction that differentiates QDA from LDA is that each class has its own covariance matrix. Typically, QDA is utilized when the training dataset is very large so that the variance of the classifier is not a major concern. QDA serves as a middle ground between non-parametric methods like KNN and linear approaches like LDA and logistic regression. While it assumes a quadratic decision boundary, it still provides more flexibility than the linear methods, giving it the potential to capture more complex relationships[2].

After training the QDA model on the training dataset of 25,000 samples, we evaluated its performance on the testing data, which consists of the same amount of samples. The response variable Sentiment was converted into a factor before fitting the model and making predictions, with levels "Positive" and "Negative," to ensure that it is treated as a binary classification variable.

		Predicted	
		Negative	Positive
Actual	Negative	9624	2067
	Positive	2887	10422

Figure 7: QDA confusion matrix

As we can see from the confusion matrix, the accuracy of predicting a positive sentiment is around 78.3% (True Positives/Total Positives), while the accuracy of predicting a negative sentiment is around 82.3% (True Negatives/Total Negatives). These accuracies are both lower than QDA's linear counterpart LDA's TP and TN accuracies. The overall accuracy of the QDA model is 80.2%, slightly less than both Logistic Regression and LDA.

[2]Source: ISLR Textbook pg. 151

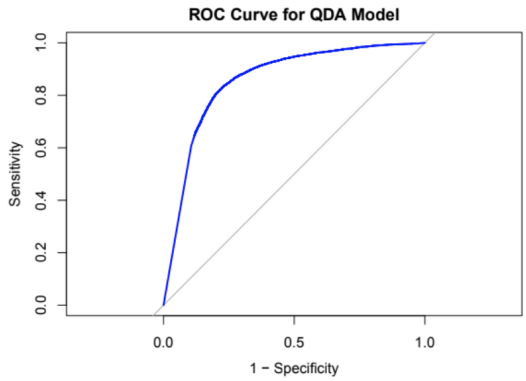

Figure 8: QDA ROC curve

A ROC curve plots the True Positive Rate (TPR) vs False Positive Rate (FPR) at different classification thresholds. "Sensitivity" refers to the TPR, whereas 1 - Specificity refers to the FPR. A side-by-side comparison of the QDA Model ROC to those of Logistic Regression and LDA shows that both models outperform QDA; their curves are more bowed out towards the top-left corner, indicating a higher sensitivity (TPR)[3].

Why does Logistic Regression and LDA outperform QDA?

As we saw from the results, LDA and Logistic Regression slightly outperform QDA. While the difference may be marginal, there are possible explanations that might help us understand the differing accuracies. The assumption of varying covariance matrices for different classes may introduce higher variance, leading to possible overfitting and reduced generalization ability. The comparatively simpler LDA and logistic regression models might find it easier to generalize, underscoring a potential bias-variance tradeoff that favors these models.

Random Forest

Decision trees are constructed by repeated splits of subsets of the data (the root node) into two descending subsets (child nodes). A terminal node, or leaf node, ideally occurs when that node is purely made up of one class. By combining hundreds or even thousands of decision trees, we can create a random forest that contains trees with many different subsets of the data. Random forest models are considered one of the more accurate predictive models, although the computing time is quite long and inefficient compared to other models (i.e. this algorithm is computationally expensive).

After splitting the dataset into 50% for training and 50% for testing, we first did Random Search Cross Validation in order to determine some ideal hyperparameters for our model. The results suggested the maximum number of features for each tree be 4.

Our random forest model with the updated hyperparameter had an accuracy of 83.4%, which was roughly 2% better than the model with default hyperparameters. Shown below are the results of the model and its ROC curve.

[3]Source: **Classification: ROC curve and AUC**

		Predicted	
		Negative	Positive
Actual	Negative	10221	2279
	Positive	1866	10634

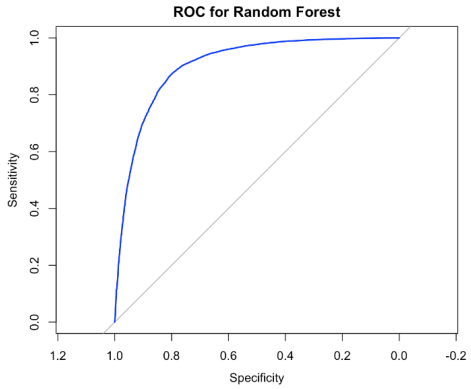

Figure 9: (a) Random Forest confusion matric; (b) Random Forest ROC curve

The true positive rate (sensitivity) was 85.1%, and the false positive rate (specificity) was 81.8%.

Conclusion

	Accuracy
LogR	83.8%
LDA	83.5%
QDA	80.2%
RF	83.4%

Table 1: Comparison of model accuracies

All four chosen models performed generally well, with each exceeding the threshold of 80% accuracy. Logistic Regression performed the best. LDA came in second, narrowly returning a higher accuracy than the random forest. However, the 0.1% difference between the two is not large enough to conclusively say that one is better than the other. QDA came in last, with an accuracy just narrowly crossing the 80% threshold. The lower accuracy of QDA tells us that varying covariance matrices between classes/sentiments was not a beneficial assumption to make. The QDA model likely overfit to the training data, resulting in under-performance on the testing dataset. The accuracy of LDA, on the other hand suggests that it was relatively safe to assume that observations within each class come from a multivariate Gaussian distribution.

Based on the results and accuracy, we would definitely recommend a Logistic Regression model. The next best models, LDA and Random Forest, still had lower accuracies and are more computationally expensive in this case. This could make a significant difference if these models were being put into production and/or used on a much larger dataset.

Future Analysis

While we did end up evaluating four different modeling techniques, future iterations of this study could include a K nearest neighbors analysis, or perform K-fold cross validation on the models other than Logistic Regression

and Random Forest. However, cross validation would have made the models even more computationally expensive. In addition, KNN likely would not have performed as well as some of the models we implemented. KNN is a very flexible model and it likely would have overfit to the training data like QDA.

Once again, our models performed fairly well, especially considering that we only used an opinion lexicon and term frequencies to numericalize our reviews. If we were looking to improve our accuracy, possibly north of 90%, we could have also accounted for the context surrounding words. This would help us determine if a word really reflected positive or negative sentiment. In addition, we could have taken punctuation, such exclamation marks, and capitalized words into account. Although relatively uncommon and difficult to detect, the presence of sarcasm could also be sought out. This would also make us reconsider the true sentiment a word or sentence displays.

Lastly, keeping more words from our lexicon would have helped us in improving the accuracy of our models. Reducing our unique words to 435 did ensure that our models ran more efficiently but the TF-IDF cutoff used to make that selection was arbitrarily selected. This reduction also led to some reviews no longer having any words that were "important features". Keeping more words, at least enough that every review has a word, would be a good decision if we were to perform this analysis again.