Project 3 Report

Paul Holaway (paulch2), Albert Li (xiangl9), & Matthew Schroeder (mas5)

November 27th, 2022

Statement of Contribution

All group members have contributed to this project. The coding and implementation for this project was done by Paul and Matt. The results summarized in the report were done by all group members. The procedure for selecting `myvocab` was done by Paul. The final report was written by Albert and Matt with the proofreading being done by Paul.

Overview

This project looks at IMDB movie reviews which are classified as either a positive review or a negative review. Our task was to build a binary classification model to predict whether a given movie review will be positive or negative. The data set consists of 50,000 rows and four columns. Each row represents a distinct review and the four columns are `id` which represents the review identification number, `sentiment` which a binary variable where 0 represents a negative review and 1 represents a positive review, `score` where 1-4 are negative reviews and 7-10 are positive reviews, and `review` which is the text of the review. For full points, the targets for this project are two-fold:

1. Use a vocabulary size of less than or equal to 1,000 words.
2. Produce an AUC of greater than 0.96 for all five of the test splits.

An AUC of > 0.96 means that at least 24,000 of the 25,000 test observations for each split are correctly classified. The same vocabulary needs to be used for all of the training and testing splits. We were able to achieve both measures, as for all of the five splits, our AUC was > 0.96 and our vocabulary remained consistent at $< 1,000$ words.

Technical Details

First, let's examine the data pre-processing. The data pre-processing for this project consists of splitting the data into five training and testing splits based on the `project3_splits.csv` file. For each of the five splits, the data is put in a folder (`split_1` and so on) and the files in each folder are `test_y.tsv`, `test.tsv`, and `train.tsv`. The `score` column is not included in the training data, which is intentional, as it should not be included as a feature in the model. Furthermore, we removed special characters (`<.*?>`) from the data set as they would not be helpful in our analysis. We created a DTM (document term matrix) using a custom function called `customDTM`. The argument that this function takes in is the data set of reviews. The first thing it does is identify stop words, which are common in the text, but do not provide much useful information. Words like I, their, were, is, the, and it were included, along with others.
The next step in this function utilized the `itoken` function. This creates an iterator over the `review` column

of the data and is an input to the next function in the `customDTM` function. The next function is the `create_vocabulary` function, which does exactly what you would expect it to based on its name. With respect to arguments, it takes in the iterator, the stop words, and n-gram, which specifies "the lower and upper boundary of the range of n-values for different n-grams to be extracted". We set our range as one to four words long as anything longer than four words is practically a sentence. The result of this function is put into an argument into `prune_vocabulary`, which reduces the size of the vocabulary by specifying the number of times the word has to appear in the document, as well as specifying a minimum and maximum proportion of occurrences. We set our values to be between 0.001 and 0.5 as anything lower is too rare to be useful and anything higher just adds white noise decreasing our prediction accuracy. Finally, the reduced vocabulary is put into the `create_dtm` function along with the iterator to create the training vocabulary.
The next step involves using the other custom function that we wrote, `conform`. This function takes in the result over the previous step and the vocabulary made in the `myvocab.Rmd` file. It adds the values of the column to a new design matrix if there is a matching column name in both the `myvocab` and `dtm_train`, and a zero column if not. We then put this new design matrix into our model. The only model that we tried and ended up sticking with was a five fold cross validation model using ridge regression in the binomial family of models as it reached the benchmark on the first attempt. (To get a ridge regression model using `cv.glmnet`, we set the value of the argument `alpha = 0`.) Once we had trained the model, we applied our two custom functions to the testing data, exactly as we did to the training data. We then used the `predict` function to make predictions on the testing data. One of the important arguments that we inputted into the `predict` function was setting `s = lambda.min`, which is the lambda value with the minimum mean cross validated error. Finally, we used the `auc` function from the `pROC` package to get the AUC. This process was used for each of the five splits.

Model Validation

We were quite pleased with the performance of the model on each of the five testing tests, as our lowest AUC on any of the test sets was **0.9654**, which is comfortably above the bar for full credit of 0.96. However, this means that out of every 10,000 predictions, on average, 400 out of them would be incorrect classification. One of the potential reasons for this is that the model put more focus on certain words and thus made a wrong prediction for the overall sentiment. Another reason might be that the key words in the reviews we are using for prediction did not use the words that are captured in the vocabulary, which led the model made a poor prediction based on the limited words it has. There are still certain limitations on the model. For instance, the vocabulary is built specifically based on the data set for this project alone. Thus, it might give unreasonable or lower AUC results for other data sets or situations. Also, there is no magnitude considered by the model, meaning the result is binary and the middle ground is ignored by this model. However, those are rather common grading that people tend to give. Plus, people don't tend to assign the same emotional value to the same grade. These two factors combined could potentially reduce the value of the prediction that the model produced. For a more refined sentiment analysis, a more complex model is needed to differentiate the magnitude of each classification.
To improve our model, we could first focus on the dictionary. We found there are some redundancy within he dictionary, such as 8, 8_out and 8_out_of, and 8_10 are all included, Thus, we could make the current one more concise to include additional words within 1,000 word limit. For instance, we could check the correlation between tokens, then decide if we can replace certain negate words with it antonym, or potentially treat different tends of verbs as the same and including additional different words to increase the accuracy. Also, since the goal of this project is to identify if the sentiment is positive or negative, rather than assigning the magnitude of emotion, we could potentially treat different level of adjectives (i.e. bad, worse, worst) as the same and see if this would give us more room to fit additional words within the 1,000 limit for the improvement in AUC. Alternatively, while it would exceed the dictionary wording limit's top grading level for the purposes of the project, we could have increased the size of the vocabulary and see if that would in turn give us a higher AUC. Additionally, rather than using lasso to train the model, we could use other methods like XGBoost, Naive Bayes, and so on and see if those would give us a boost in the AUC score. Some interesting findings include there are words that don't make sense from a glance but were selected by `R`, such as zombies, become, vhs, keeps, doing and so on. This is also one of the reason we believe improving

the dictionary building process or having some post-adjustment methods could be one of the most helpful methods to improve the model.

Interpretability

Our model is rather simple to interpret as we used cross-validated Ridge regression. We simply tried to find the correlation between a certain token (either a word or a short phrase) and the binary emotional classifications, which is negative and positive. Then we could find and use these key tokens in future reviews and predict the overall sentiment. There isn't any contextual information, such as the race or gender of the reviewer, any previous historical reviews this person published, or the overall sentiment for the movie that's being reviewed, build within the model building. Thus, it is able to generate unbiased predictions from this perspective. However, it might give biased result if the key words in the review captured by the dictionary is opposite of the overall sentiment. However, with a sufficient dictionary, we would decrease the possibility of such event.

Table 1: AUC and Run Time For Each Split

Split Number	AUC	Run Time
1	0.966	21:16.23
2	0.966	22:09.60
3	0.9658	18:05.67
4	0.9661	18:05.03
5	0.9654	21:42.76

Computer Specifications

- Computer: 13-inch MacBook Air; Early 2015
- Processor: 1.6 GHz Dual-Core Intel Core i5
- Memory: 8 GB 1600 MHz DDR3
- Graphics: Intel HD Graphics 6000 1536 MB