

Fall 2021 STAT 542 — Project 3 Report

1 Overview

Goals In project 3, we work with the IMDB movie reviews data set, where each review is labelled as positive or negative. In more details, the data set contains 50,000 sample data points and 5 variables. These variables are: the identification number (variable "id"); the categorical variable "sentiment", where 0 stands for negative and 1 for positive; the variable "score", where the 10-point score assigned by the reviewer: here scores 1-4 correspond to negative sentiment, scores 7-10 correspond to positive sentiment, and this dataset contains no reviews with score 5 or 6; lastly the variable "review" containing strings of comments associated with the IMDB reviews. Our goal is to build a binary classification model to predict the sentiment of a movie review.

Input/Output/ Process Given 50,000 data points, we will create 5 different data sets. Each of these data sets contains training and testing data, where 25,000 data points are assigned to test data. This test data contains 2 parts: one is test with 2 columns ("id" and "review"), and the other one is the same test data, but with our label "sentiment" and variable "score". The first part of the test data (review) is processed and fed to our model to output probability prediction of the sentiment of the associated movie review, and then we use this result to compare with the labels in the second part of the testing data. The rest is assigned to training data of the particular split, and we note that the training data do not contain the "score" column to make sure that it is not used as an input feature to our model.

Evaluation The model performance will be evaluated using Area under the ROC Curve (AUC). Note that an ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds (where we threshold the probability prediction). This curve plot two parameters: True Positive Rate (TPR), and False Positive Rate (FPR). These are defined as $TPR = \frac{TP}{TP + FN}$ and $FPR = \frac{FP}{FP + FN}$, where TP = true positive, FP = false positive, and FN = false negative. For our model to qualify, we need the models to achieve an AUC of at least 0.96 on all 5 data sets.

Contribution I work on project 3 individually and there is no other team member. My NetID is xxx

2 Technical Details

2.1 Overview of training/ testing process

In this section, we briefly go over the steps for building our models. The process consists of 3 major steps:

- Step 1: The first step is to create the vocabulary that will be used to train our model in later steps. The process of creating this vocabulary are detailed in the supplemental HTML file that goes with the submission. This part basically follows two main steps: the review texts from the original big data set are stemmed, tokenized, and vectorized to create a DocumentTerm matrix (consisting of maximum 4-grams), where the vocabulary size is the

number of columns of this matrix. The vocabulary size is further reduced to 976 words using Lasso with logistic regression. This smaller vocabulary is saved to use to the next steps.

- **Step 2:** In this step, we used the previously created vocabulary of size 976 to train our model. Here we first go over the details of models we use, and we will include the details of the data pre-processing in Section 2.2. Note that this processing procedure is used for both train and test data, and the process is the same for all 5 splits of data.

First, we load in the customized vocabulary of size 976. Then the train data is pre-processed, and we apply the vocabulary on the use the train data ("reviews") to create a DocumentTerm matrix for the train data.

Using this DocumentTerm matrix as input, we train a logistic regression (family = 'binomial') with Ridge regression ('alpha' = 0) and cross-validation over a range of lambda values. The chosen lambda value to use for testing is the **lambda.min** from the built-in lambda sequences of glmnet. We then go through Step 3 and output the probability predictions (for movie review sentiments: positive or negative) together with their associated ids for the test data.

- **Step 3:** This step is dedicated to our testing process. For each split data, the first part of test data (containing variables "ids" and "reviews") is loaded in and processed like train data (2.2). Then the customized vocabulary is applied on this processed test data to create the test DocumentTerm matrix. The trained logistic regression (with Ridge and lambda.min) is used to output probability prediction for movie review sentiment for the test data. The second part of the data (with label of movie review sentiment) is then loaded in for AUC calculation.

2.2 Common Data Pre-processing

In this section, we go over some pre-processing steps on both train and test data that we use.

- **Stop-words removal:** This step is done only in step 1 (above) where we process the whole original data set to create the customized vocabulary. Basically we choose common words such as 'a', 'the', 'i', 'me' that does not contribute much to the sentiment of the review and remove them to create a better, more meaningful vocabulary.
- **Removing special symbols and punctuation:** Here, we remove brackets and punctuations from the movie reviews.
- **Tokenization:** We then use the 'itoken' function in text2vec library from R to convert all reviews into lower cases and perform tokenization by specifying tokenizer to be word tokenizer.
- **Vectorize texts from reviews** Next, we use the customized vocabulary (loaded in in the beginning) and create_vocabulary function from text2vec package to generate n-grams (with maximum being 4-gram) from each of the review text and use vocab_vectorizer function to vectorize this. Then we use this vectorized text and the tokenized texts from reviews (iterator) to get the corresponding DocumentTerm matrix.

3 Results

3.1 Computational results

In this section, we include the testing results of AUC for each of the 5 splits of data sets and the average over all of them in Table 3.1. Using vocabulary of **size 976**, we obtain the results that meet the benchmark of AUC being at least 0.96 for all 5 sets.

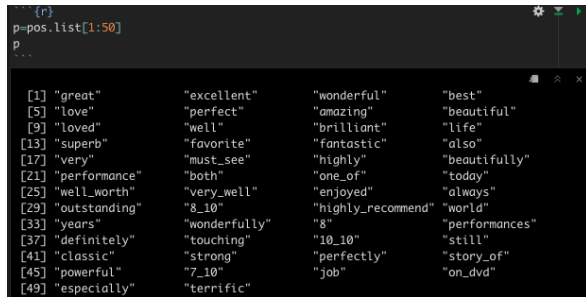
Split (S)	S1	S2	S3	S4	S5	Average over 5 splits
AUC	0.9693	0.9685	0.9688	0.9694	0.9682	0.9688
Runtime (in seconds)	49.32	45.68	43.45	44.02	44.39	44.372

Table 3.1: AUC results for 5 splits

To get this result, the computer system we used is Macbook pro with 2.6 GHz 6-Core I7, 32GB memory.

3.2 Interpretability/ Error Analysis

For the model that we have built with this condensed vocabulary, it's relatively easy and intuitive to understand how the model makes its prediction using the vocabulary. We first include the top 50 positive versus negative words (from our customized vocabulary) ordered in decreasing order using the t-statistics in Table 3.1 and 3.2.



[1]	"great"	"excellent"	"wonderful"	"best"
[5]	"love"	"perfect"	"amazing"	"beautiful"
[9]	"loved"	"well"	"brilliant"	"life"
[13]	"superb"	"favorite"	"fantastic"	"also"
[17]	"very"	"must_see"	"highly"	"beautifully"
[21]	"performance"	"both"	"one_of"	"today"
[25]	"well_worth"	"very_well"	"enjoyed"	"always"
[29]	"outstanding"	"8_10"	"highly_recommend"	"world"
[33]	"years"	"wonderfully"	"8"	"performances"
[37]	"definitely"	"touching"	"10_10"	"still"
[41]	"classic"	"strong"	"perfectly"	"story_of"
[45]	"powerful"	"7_10"	"job"	"on_dvd"
[49]	"especially"	"terrific"		

Figure 3.1: Top 50 positive words



[1]	"bad"	"worst"	"waste"	"awful"	"terrible"
[6]	"worse"	"boring"	"stupid"	"no"	"nothing"
[11]	"poor"	"horrible"	"minutes"	"just"	"so_bad"
[16]	"even"	"crap"	"acting"	"poorly"	"supposed"
[21]	"at_all"	"script"	"plot"	"why"	"ridiculous"
[26]	"worst_movie"	"not_even"	"lame"	"avoid"	"annoying"
[31]	"wasted"	"don't"	"dull"	"2"	"mess"
[36]	"money"	"pointless"	"cheap"	"pathetic"	"oh"
[41]	"any"	"redeeming"	"laughable"	"thing"	"couldn't"
[46]	"badly"	"1"	"unless"	"than_this"	"sorry"

Figure 3.2: Top 50 negative words

For these positive/negative words, we include a small sample from one model that we fit on split 5 to show how it estimate the coefficients in Table 3.3. To understand better how these words help with the prediction, we can look at several examples included in Table 3.2. It seems that the more positive words the review has, and especially if it's in the top 50 words, the more likely it's predicted to be positive and vice versa. Moreover, the presence of top 50 positive or negative words usually indicate better and has more predictive power for the sentiment of the movie review. For example, in example id = 14191, although we have more negative words than positive words, none of these words come from the top 50 words, and thus we can see that it doesn't give a strong classification of the sentiment and a threshold value of 0.5 could lead to mis-classification. We can observe these behaviors in all examples below and also throughout most of the data sets.

There are also cases when the model makes mistakes. For example, for movie review with id = 39320, the reviews use a lot of words in the positive words list to describe certain things/plot of the movie, but only to give a not very positive review, and thus misleads our model. The review of such case is included briefly here:

... *Young brides drop dead at the altar after saying do: Their corpses are stolen by a renowned horticulturist Dr. Lorenz(Lugosi) and a couple of his freakish minions as his aging wife (Elizabeth Russell) needs injections of the glandular fluids of the young virgins to **remain forever young...forever beautiful. An eager local cub reporter (Luana Walters) bride wore the same rare orchid to the altar; an orchid in which Dr. Lorenz would be most knowledgeable. A typical horror movie....***

id	# of Positive words	# Negative Words	True Label	Probability Prediction
48625	11 (2 from top 50)	7 (1 from top 50)	1	0.939
14191	8 (0 from top 50)	10 (0 from top 50)	0	0.253
29766	12 (5 from top 50)	19 (11 from top 50)	0	0.00052
36354	3 (0 from top 50)	4 (0 from top 50)	0	0.33
39320	8 (2 from top 50)	7 (1 from top 50)	0	0.516

Table 3.2: Examples showing intepretability of our model

Positive Words	Coefs	Negative Words	Coefs
"great"	0.2452	"bad"	-0.2258
"excellent"	0.47	"worst"	-0.5156
"wonderful"	0.396	"waste"	-0.652
"best"	0.258	"awful"	-0.5385
"love"	0.1105	"terrible"	-0.378

Table 3.3: Sample of 5 positive/negative words with coefficients from our model

4 Conclusion and Reference

In this report, we follow the approach that is suggested by Prof. Feng Liang from her Campuswire post. However, one weakness of the model is that we are building the vocabulary from the whole data set which has both test and train data. There are cases in real practice where we don't have this luxury of having the test data in advanced, and thus this model might not perform well in those occasions. Through going over the data, we have some observations:

- Some of the top negative words seem ambiguous and can potentially mis-classify some reviews, for example: "supposed", "thing", etc.
- Although some words are neutral and ambiguous, pairs of consecutive word can provide more complex meaning that can't be captured by single words, and thus n-grams help improve our model.