Project 3: Movie Review Sentiment Analysis

CS598: Practical Statistical Learning

Naomi Bhagat - nbhagat3, Michael Miller - msmille3, Joe May - jemay3

16 November 2023

Assignment Data

Program: MCS-DS Assignment post: campuswire

Team contributions:

Person	Contribution
Naomi Bhagat	myvocab.txt/Vocab Generation
Michael Miller	mymain.R/Coding
Joe May	Report/Writing, Quality Assurance

Overview

Section 1: Technical Details

Our model is a sentiment analysis of text data for imdb movie reviews. It constructs a document-term matrix out of a custom vocabulary, uses a vectorizor to match vocabulary between the test and train data, and predicts the sentiment of the review.

Data Pre-processing:

Our implementation begins with a few short pre-processing steps. We begin by installing the text2vec, glmnet, and pROC packages, unless they're already installed, in which case we load the library. Then we establish whether or not the code is in debugging mode, initialize the seed, read in the vocab file 'myvocab.txt', and create place holders for the auc scores.

Model Implementation:

The model is implemented in a slightly different fashion, depending on if it's in debugging mode or now. If it is in debugging mode, the run time is tracked, and the evaluation metrics are printed out. If it is not in debugging mode, the model is run virtually the same, but predictions are written on a text document. Since the difference is minimal between the cases when debugging mode is and is not on, the following is a description of the model implementation both have in common, with additional detail provided for the differences in modes at the end.

First our model reads in the training data from tab separated value files. The HTML tags are all then removed from the 'review' column. Next a vocabulary is created with the itoken() iterator. At this point and using this iterator, the text is all made lower case, and tokenized. A custom vocabulary is then used to make the document-term matrix (DTM) for training the model. Further description of the vocabulary can be found in the vocab file, but the (DTM) uses the custom vocabulary and a n-gram range between 1 and 4.

Afterwards we train the model using lasso regression. More specifically, we use a binary classification for 'glmnet', an alpha of 1, and 'auc' as the measure type. The alpha score of one is that which selects the lasso regression. We chose to use lasso regression because of it's higher performance than alternative options.

Finally. we obtain the predicted scores, output them to a data frame, and output the results in some manner as specified by the debugging flag. In debugging mode, we obtain the ROC score, record the time it took to run, and print it out. Out of debugging mode, we output everything to mysubmission.csv.

Section 2: Evaluation Metrics

The computer system this was run on was a Dell Inspiron 3501, 1.00 GHz with 8.00 GB of installed RAM for all 5 splits. Below is a summary of the split #, the AUC score, and the time (in seconds) it took to train and evaluate each split:

Split	AUC	Time (s)
1	0.963809603950088	1.08598738114039
2	0.965047149295919	58.6640400886536
3	0.964181995790127	54.9420478343964
4	0.964553103713986	1.09632801612218
5	0.963925343978638	59.117889881134