CS598 - Project 4

Xiaoming Ji

Computer System

Hardware

- Dell Precision Tower 5810
- CPU: Intel Xeon E5-1607 @ 3.10GHz
- Memory: 32GB
- GPU: Nvidia GeForce GTX 1080 (2 cards)

Software

- OS: Windows 10 Professional 64bit
- R: 3.5.1
- R Packages:
  - text2vec
  - tokenizers
  - xgboost_0.71.2
  - glmnet_2.0-16

Overview

The goal of this project is to build a binary classification model to predict the sentiment of IMDB movie review.

The given movie review dataset file `Project4_data.tsv` has 50,000 rows (i.e., reviews) and 3 columns. Column 1 "new_id" is the ID for each review (same as the row number), Column 2 "sentiment" is the binary response, and Column 3 is the review.

The given dataset file `Project4_splits.csv` is used to split the 50,000 reviews into Three sets of training and test data. The dataset has 3 columns, and each column consists of the "new_id" (or equivalently the row ID) of 25,000 test samples.

The required performance goal is is to achieve minimal value of AUC over the three test datasets. For model interpretability, the vocabulary size should less than 3000.

For this project, I used logistic linear/lasso regression model to select vocabulary and logistic linear/ridge regression model for training/prediction. Such approach achieved `0.965` AUC with only `1000` vocabulary size.

Build Vocabulary

Building vocabulary is basically a feature selection process. I first build the vocabulary using all 50,000 reviews with tokenization process as:

- Filter out HTML tag and non-alphanumeric characters.
- Remove stop words:

i, me, my, myself, we, our, ours, ourselves, you, your, yours, their, they, his, her, she, he, a, an, and, is, was, are, were, him, himself, has, have, it, its, of, one, for, the, us, this

- Extract 1 to 4 ngram terms

Then I prune this vocabulary with conditions:

- minimum 5 occurences over all documents.
- minimum 0.1% of documents which should contain this term.
- maximum 50% of documents which should contain this term.

This gives me a vocabulary size of 29035.

I have tried several approaches to reduce the size of the vocabulary:

1. Normalized Difference Sentiment Index as described in A Quick R Demonstration on Sentiment Analysis and Textmining
2. Discriminant Analysis for Two-sample T-test as illustrated by Professor. Feng Liang.
3. Use (XG)Boosting to build a model on all review data and select features/terms using `xgb.importance()` ordered by Gain.
4. Use logistic linear regression model with Lasso regularization to fit the review data, then select the none-zero cofficient term (3200+). For these terms, sort by the absolute value of coefficient, then select the 1000, 2000, 3000 terms as vocabulary candidates.

My testing shows approach #3 and #4 work better than #1 and #2. #4 works extremely well and I can achieve above 0.965 AUC using 1000 terms. Adding more terms results in bigger AUC (see the performance number in next section).

Note: I also tried tokenization with word stemming but got lower AUC with same vocabulary size. Thus, no word stemming is used.

Classification Model: type of algorithms/models used/explored, pre-processing, training process, tuning parameters, etc.

For classification model, I used logistic linear regression with Ridge regularization. `cv.glmnet` is used to select the minminal λ to build the model.

Three datasets are tested on vocabulary of size 1000, 2000 and 3000 respetively. And I get the performance matrix as:

	VocabularySize:1000	VocabularySize:2000	VocabularySize:3000
Performance for Split 1	0.9657573	0.9776277	0.9805633
Performance for Split 2	0.9660925	0.9782075	0.9810840
Performance for Split 3	0.9653049	0.9782085	0.9811342

Computation time: 411 seconds

Discussion on model limitations, explanation on errors, and future steps you could take to improve your model.