CS598 Project Report- Movie Review Sentiment Analysis (xxx)

1. Introduction & Goal

For this project we are provided with a dataset of IMDB movie reviews and its respective label as a positive or negative review. The goal is to predict the sentiment of 5 splits provided of IMDB review data such that AUC level for each of the split is greater than 0.96 and the common vocabulary count should be less than 1000. Output of this project is a file “myvocab.txt” containing vocabulary words less than or equal to 1000 and “mymain.r” which takes “myvocab.txt” as inputs and outputs “mysubmission.txt” (columns of “id” and “prob”) which is evaluated for accuracy using AUC. This report will outline method and models used to achieve this goal.

2. Data Overview and Pre-processing

The IMDB review dataset contains 50000 rows and 3 columns. Column 1 is “id” is the identification number for each of the review, column 2 is “sentiment” which contains 2 classes, 0 for negative and 1 for positive, column 3 contains the review.

2.1 HTML Tags removal

The review column in the dataset contain the HTML tags <> which were removed as they do not contribute to classifying a review.

2.1 Tokenization, capitalization and stop word removal

English language contain lot of common words (e.g. “is”, “was” etc)which for the purpose of analyzing and building models don’t add much value to the meaning of the document. Only split#1 was used to do this pre-processing and vocabulary build. List of 37 stop words were removed. Removal of these stop words reduce features used in document term matrix hence reduces time to train the model and also improves performance by reducing vocabulary. Each of the words were made lower case as case of the words do not contribute to sentiment of the movie. Words were then tokenized with phrases up to 4 words. Words/Phrases were further pruned by only including words which had minimum of 10 occurrences, and proportion of documents containing this term was set to be greater than 0.001 and less than 0.5. This reduced vocabulary size from ~10million to 28217 terms.

3. Model Building

3.1 Model#1: Elastic Reg. for Vocabulary Trimming & Ridge Reg for Train/Test & PostProcessing Vocabulary

Elastic Logistic Regression was used to trim vocabulary size to less than 1100. This was done by tuning the hyper parameter alpha to 0.1. This value was determined by iteration such the lowest vocabulary size is used which gave the highest AUC. The vocabulary size of less than 1100 was chosen by selecting 1056 nonzero coefficients (i.e df) from default of 100 lambda estimates. These coefficients were used as vocabulary to train and test the model for each of the 5 splits. With vocabulary size of 1056, AUC was >0.96 for each of the 5 splits. Post-processing was done to reduce vocabulary size further such that vocabulary size is <1000 and AUC to be >0.96. This post processing was done manually. Visual inspection of the vocabulary was done in addition to occurrence count of each of the term. From visual inspection it was observed there were several redundant usages of terms such as “1 out”, “1 out of” and “1 out of 10”. Since each of the word is a subset of another, all but one of the terms were removed from vocabulary. This was done multiple times reducing vocabulary size to ~1030. There were few noun words in the list such as “Gandhi”, “Segal” which did not contribute anything to sentiment were removed. Last, there were several weak learner words like “can”, “might”, “but” etc. which do not contribute to sentimental value. Removing these reduce vocabulary size to 999 with AUC of >0.96. Refer to Table 1 in results for breakdown of each of the splits. Although this meets project goal, but this required manual post processing which can be time consuming and is neither scientific nor exhaustive so alternate trimming by influential words using T-test was done next.

3.2 Model#2: Pre-screening with T-test, Elastic Reg. for Feature Selection & Ridge Reg for Train/Test

Since Model#1 required manual post-processing to reduce vocabulary alternative approach of using t-test to select 2000 most influential words first and then limit the vocabulary list to less than 1000 using elastic regression as done in model#1 was tried. This makes vocabulary easy to interpret as well. This screening method uses two-sample t-test to test whether X population and Y population have the same mean. Words are then ordered by magnitude of t-statistic and top 2000 words were chosen. After this pre-screening to 2000 words, feature selection of vocabulary was reduced to <1000 by Elastic Regression like Model#1 with alpha set to 0.1. Like Model#1, only split#1 was used to create this vocabulary. AUC for each of the split was evaluated using Ridge Regression and was >0.96 for each of the split. Refer to Table 1 in results for breakdown of each of the splits

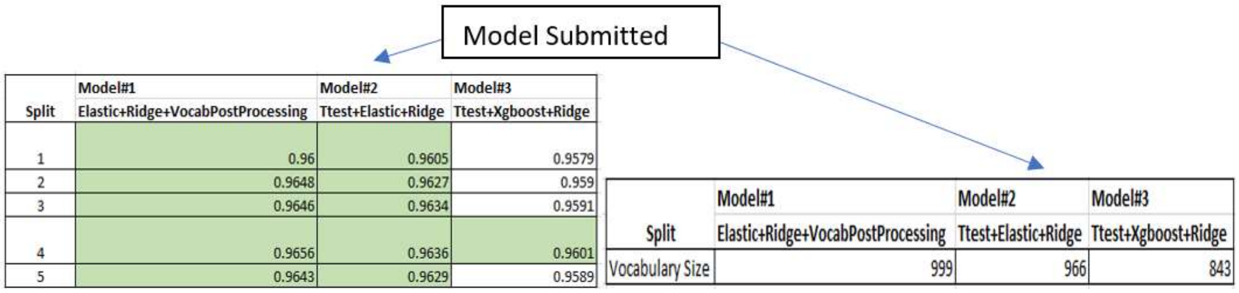
Use of this approach is desirable over Model#1 where manual post screening was required. Next, we investigate use of tree models for feature selection.

3.3 Model#3: Pre-screening with T-test, Xgboost for Feature Selection & Ridge Reg for Train/Test

For this approach, similar pre-screening with T-test was used but instead of Elastic Regression, a tree model approach was used. Document term matrix created after pre-screen to 2000 words was used to create 1000 trees with max depth of 2. Different number of trees/depth were tried with this approach but neither of them gave good results for all the splits as for AUC was <0.96. Refer to table 1 in the results for breakdown of each of the splits.

4. Results

Below is the table of results for the 3 models tried. Model#2 with T-test for pre-filtering with Elastic for feature selection to <1000 on split#1 to generate vocabulary and Ridge Regression to train/test each of split is submitted for this project. Note, vocabulary was generated based on split#1 only. This model provides adequate AUC of >0.96 for each of the split with vocabulary size of 966 which meets the target of less than 1000 words. There are some errors with this model which are discussed in detail in Error analysis.


Model Submitted

	Model#1	Model#2	Model#3
Split	Elastic+Ridge+VocabPostProcessing	Ttest+Elastic+Ridge	Ttest+Xgboost+Ridge
1	0.96	0.9605	0.9579
2	0.9648	0.9627	0.959
3	0.9646	0.9634	0.9591
4	0.9656	0.9636	0.9601
5	0.9643	0.9629	0.9589

	Model#1	Model#2	Model#3
Split	Elastic+Ridge+VocabPostProcessing	Ttest+Elastic+Ridge	Ttest+Xgboost+Ridge
Vocabulary Size	999	966	843

Table 1: AUC (left) Vocabulary size(right) for different models and splits

5. Interpretability & Error Analysis

Model#2 with T-test as prescreening was selected for this project as it provides some level of interpretability. Since coefficients of regression as based on joint modeling does not provide as much interpretability as T-test where features are based on marginal behavior. A brief visual inspection was done such that top 3 errors of each class from test data on split 1 was inspected. Below are the top 3 errors for each class complete text of review is omitted due to reviews being quite long.

id	prob	sentiment	score
1695	0.9976509	0	4
17907	0.9964032	0	3
24845	0.9930582	0	4
24568	0.0008952	1	8
11927	0.001808	1	7
26372	0.00405454	1	7

Table 2: Top 3 Misclassification for each class

The cases above were manually. All 6 of the reviews were long and 5 out of the 6 were very strongly worded positively/negatively. 1 out of the 6 was clearly a reviewer scoring error as possibly the reviewer thought scaling was 0 to 5 and gave a score of 4 indicating positive but with proper scoring this is a negative sentiment. 3 others appeared to have to similar reviewer scoring errors. In this case, it would be unlikely to detect this kind of human scoring error. The remaining 2 appears to be limitation of our simple bag or words approach. Partially, some of the weak learner words which are not adjective such as “and”, “just”, “or” etc. were part of the final vocabulary which contribute for a review to be incorrectly biased towards positive or negative as it is determined by distribution of each of these words. Below is a table summarizing the possible cause of errors for the top 3 misclassifications for each class:

id	# of positive words	# of Negative words	Possible Cause of Error
1695	21	23	Clearly, reviewer scoring error as he wrote 7/10 and highly recommended
17907	33	32	Worded Positively, possibly reviewer scoring error
24845	36	19	Worded Positively, possibly reviewer scoring error
24568	28	55	Contained Negative words but possibly caused by distribution of residual non-adjective words
11927	34	51	Worded Negatively, possibly reviewer scoring error

26372	28	58	Unclear if positive or negative from review.

Table 3: Top 3 misclassification and possible cause

To understand reviewer behavior, distribution of scores which were misclassified for split#1 was plotted and shown below. It can be observed most of the misclassification occur at border scores of 4 and 7. Also, it can be observed misclassification of both positive and negative is approximately the same and not just one sided.

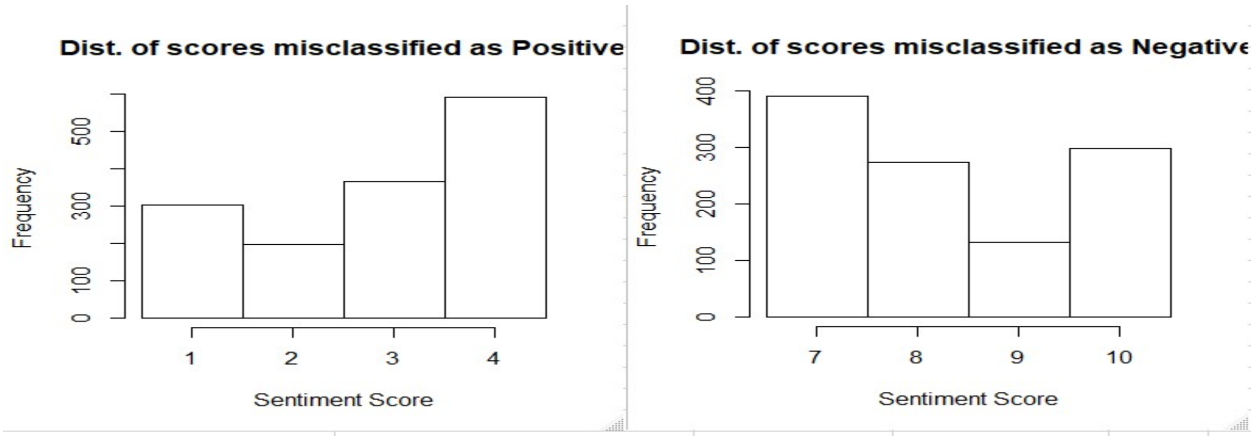


Figure 1: Misclassification Distribution for Split#1

6. Future Work

Since Bag of words just creates a set of vectors with count of words occurrences, it is recommended to try TF-IDF model since it contains information on the more important words and less important ones as well. Additionally, consider evaluating approach using deep neural network such as LSTM.

7. Acknowledgement

CS598 Professor Liang's Hints provided on piazza on what has been tried was extremely helpful

Kaggle Competition here .

8. Appendix- Model Runtime

All models were trained on a Windows PC using R(version 3.6.0) with library of tm, text2vec, glmnet, xgboost and slam. Detail computer configuration are as follows: Windows 8 64bit OS, Processor: Intel Core i5-4460 CPU @3.20GHz, 8GB RAM. Below is runtime for each of the models. Table below lists train/test time(in minutes) and vocabulary build time for each of the 3 models.

	Model#1	**Model#2**	**Model#3**
	Elastic+Ridge+VocabPostProcessing	**Ttest+Elastic+Ridge**	**Ttest+Xgboost+Ridge**
Split#1 Train+TestTime	2.03	2.18	2.22
Split#2 Train+TestTime	1.37	1.19	2.08
Split#3 Train+TestTime	1.37	1.21	1.64
Split#4 Train+TestTime	1.31	1.2	1.52
Split#5 Train+TestTime	1.27	1.19	1.51
Vocabulary Build Time	2.49	3.93	6.28

Table 4: Runtime (in minutes) for each the models and the splits