Project 3: Movie Review Sentiment Analysis

xxx[netid]

1. Introduction

In this project, we aim to do the sentiment analysis of movie reviews. The reviews have been labelled as positive or negative, so it can be considered as a binary classification problem. A subset of the data is taken from Kaggle competition: Bag of Words Meets Bags of Popcorn [1]. In this project set-up the data has been divided into five different splits each containing a training and testing set, each containing 25000 data points. Our aim is to achieve an AUC level of 0.96 or better while not exceeding the vocabulary size of 1000. This report, overall, has been structured in the following way and we skip the details of vocabulary creation as that has been mentioned in a separate document.

- Analysis of input data
- Model determination and Sensitivity Analysis
- Result Analysis and Interpretability
- Conclusion and Next Steps

2. Input Data Analysis

We performed some analysis of data using the hints provided by Professor Liang in Piazza [2] and were able to observe a few things which would be critical from modelling perspective.

i. The complete dataset has 50000 reviews with four columns: identity, sentiment with 0 indicating negative, and 1 positive, score, and review. The score column has been removed from training and testing to make sure that this doesn't get used by mistake.

ii. The complete dataset is equally divided between positive and negative sentiments.

iii. After creating the vocabulary of 2000 words using the "two-sample t-test" as discussed in piazza post [2], we do a quick pictorial analysis of words to see the importance of different words as shown in figure 1.

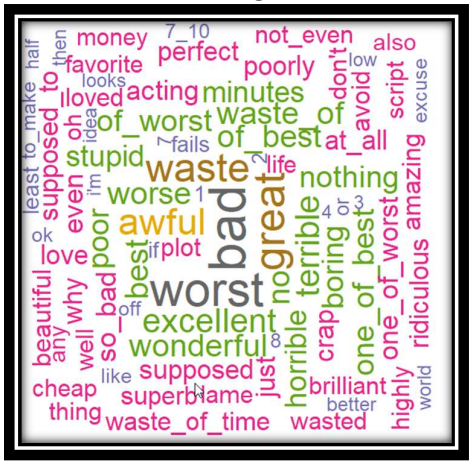

Figure 1: 2000 words

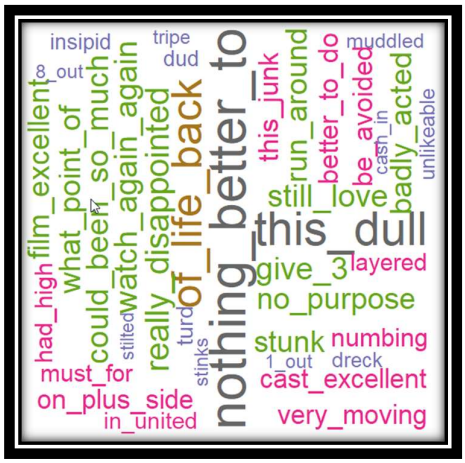

Figure 2: 959 words

iv. After using Lasso to reduce the size of vocabulary even further to 959 words, we see the change in the pattern of words with high frequency in figure 2. We see that the remaining words are of 2-gram.

3. Methodology and Parameter Sensitivity

We aimed to choose a strategy which would achieve the AUC target of 0.96 for each split with the vocabulary size of less than 1000. To achieve this, we took a stepwise approach of first meeting AUC target, and then slowly reducing the vocabulary size without violating the AUC target. The inspiration of these models was given by the professors in different piazza post [2]. As part of model preparation, we remove html tags using gsub as they don't add any value in the prediction. We also convert the tokens into lower case as we assume the case, although sometimes indicates extreme emotions, but wouldn't be highly reliable. While creating the model, we use the min of 1-gram and max of 2-gram. We tried 4 strategies as explained below.

i. **LVRC (Lasso Vocabulary Ridge Classification)**: In this model, we use the vocabulary found using Lasso and taking maximum number of words below 2000 limit. In this model, it came out to be 1956 words. We next used the ridge logistic regression with 10-fold cross validation to find the min lambda value which we later use in training using ridge logistic regression. The AUC for each split is given in figure 3.

Selected Model

Splits	TLVRC*	TLVRC	TLVXG	TVRC*	TVRC	LVRC
1	0.9602	0.9600	0.9544	0.9561	0.9624	0.9595
2	0.9632	0.9627	0.9546	0.9564	0.9626	0.9702
3	0.9629	0.9629	0.9553	0.9569	0.9631	0.9708
4	0.9634	0.9634	0.9551	0.9578	0.9635	0.9712
5	0.9625	0.9626	0.9542	0.9566	0.9624	0.9705

Figure 3: AUC of Models

ii. **TVRC (T-Test Vocabulary Ridge Classification):** In this model, the main difference is in the vocabulary creation where we chose the 2000 words directly using the two-sample t-test. The training model was same as 3.1. We see the results for each split using this approach in figure 3. The table also shows the results if we simply take top 1000 words from this list and call this as **TVRC***.

iii. **TLVRC (T-Test-Lasso Vocabulary Ridge Classification):** In this model, we improve the vocabulary by applying Lasso on 2000 words found using two-sample t-test and selecting top 959 words (max below 1000). We apply the same Ridge classification on this vocabulary as discussed in 3.1. This reduced the AUC of each split, but we still met 0.96 target. We tried to improve the result by further modifying the vocabulary using more Lasso results and manual filtering as discussed in the vocabulary generation report (1000 words). We call this model as **TLVRC***. We performed the parameter analysis by varying alpha and lambda as shown in figure 4. However, the best value was achieved with ridge regression and lambda min as calculated by 10-fold cross validation.

iv. **TLVXG (T-Test-Lasso Vocabulary XGBoost Classification):** In this model, we continued using the vocabulary used in model TLVRC* but changed the classification to XGBoost. We used this new model using the parameters settings as: maximum depth of trees as 6, learning rate as 0.06, number of rounds as 1000, and subsampling size as 0.5. We present the results in figure 3. The results were not better in comparison to the best model at this point TLVRC*, so we decided to vary the learning rate parameter but only on first split. However, it didn't help in improving anything further as shown in figure 5. Also, the run time of this model was much higher than previous ones, so we decided to not to pursue it further.

	Split				
Alpha	1	2	3	4	5
0.00	0.9602	0.9631	0.9629	0.9634	0.9625
0.05	0.9602	0.9625	0.9620	0.9626	0.9619
0.10	0.9597	0.9622	0.9615	0.9622	0.9615
0.15	0.9595	0.9620	0.9613	0.9619	0.9613
0.20	0.9592	0.9618	0.9611	0.9618	0.9612
0.25	0.9591	0.9617	0.9610	0.9617	0.9611
0.30	0.9589	0.9617	0.9609	0.9616	0.9610
0.35	0.9589	0.9616	0.9609	0.9615	0.9610
0.40	0.9590	0.9616	0.9608	0.9615	0.9609
0.45	0.9588	0.9615	0.9608	0.9615	0.9609
0.50	0.9587	0.9615	0.9608	0.9614	0.9609
0.55	0.9587	0.9615	0.9608	0.9614	0.9608
0.60	0.9586	0.9615	0.9607	0.9614	0.9608
0.65	0.9587	0.9614	0.9608	0.9614	0.9608
0.70	0.9585	0.9614	0.9607	0.9613	0.9608
0.75	0.9584	0.9614	0.9607	0.9613	0.9608
0.80	0.9586	0.9614	0.9607	0.9613	0.9608
0.85	0.9586	0.9614	0.9607	0.9613	0.9608
0.90	0.9585	0.9614	0.9607	0.9613	0.9608
0.95	0.9586	0.9614	0.9607	0.9613	0.9607
1.00	0.9585	0.9614	0.9607	0.9613	0.9608

	Split				
Lambda	1	2	3	4	5
1se	0.9595	0.9607	0.9626	0.9625	0.9616
0.01	0.9591	0.9620	0.9618	0.9625	0.9617
0.02	0.9596	0.9625	0.9624	0.9631	0.9623
0.03	0.9599	0.9627	0.9626	0.9633	0.9625
0.04	0.9600	0.9628	0.9628	0.9634	0.9626
0.05	0.9600	0.9628	0.9628	0.9634	0.9626
0.06	0.9600	0.9628	0.9629	0.9634	0.9626
0.07	0.9600	0.9627	0.9629	0.9634	0.9626
0.08	0.9600	0.9627	0.9629	0.9634	0.9626
0.09	0.9600	0.9626	0.9628	0.9633	0.9626
0.10	0.9599	0.9626	0.9628	0.9632	0.9625

Figure 4: AUC of Model TLVRC* at varying alpha and Lambda

eta	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10
Split1	0.9305	0.9445	0.9503	0.9527	0.9533	0.9539	0.9536	0.9534	0.9533	0.9526

Figure 5: Learning Rate Parameter Analysis of XGBoost for Split 1

4. Tech Specs

Operating System: Windows 10. **Processor**: Intel Core i7, 2.11GHz. **RAM**: 64GB

The time taken in vocabulary creation was around 1.8 minute. It included the file reading time, two-sample t-test, lasso analysis, as well as the post processing. Overall, the classification algorithm took ~6.21 minutes to run the complete program (5 splits) with each split taking almost 1.12 minute on average. This time includes reading the training file, creating DT-Matrix, 10-fold cross validation, logistic regression, as well as testing. We also looked at the running time of TLVXG model (XGBoost), and the time taken (7.5 minutes) was almost seven times more on average without giving better result.

5. Interpretability, Limitation and Error Analysis of the model

In this section we first try to defend our model for any racial discrimination while explaining the classification. As part of vocabulary selection, we have taken all the words in review, our stopping words include pronouns of both gender such as “his-her”, “he-she”, etc. None of these stopping words has any racial connotation. The selection is simply looking at statistical measures and the final filter has seven ("also", "always", "could", "both", "who", "thing", "got") words which are also gender and race neutral. We give one example each for positively, negatively and incorrectly predicted prediction. Just by analysing positive/negative sentiment words present in review, and the coefficients chosen by the ridge regression model, we can easily explain the sentiments in each as shown in figure 6. In the incorrectly predicted review, the number of positive and negative terms are evenly mixed, so it couldn’t

predict correctly. In some cases, even for positive terms the coefficients were negative due to collinearity. In one case, the review was written with many positive words written with negative connotation. We also observed that most of the incorrect reviews had the score of 4 or 7 and our model also scored these moderate reviews within the range of 0.4 to 0.6. So, this limitation of model should be explored as a next step for improvement.

Positive Prediction		Negative Prediction		Incorrect Prediction	
8_out	0.9372	mst3k	-0.6936	far_too	-0.2797
great_job	0.4649	laughable	-0.6327	definitely	0.2418
8	0.4588	worthless	-0.6113	simple	0.2368
surprisingly	0.4527	redeeming	-0.5998	release	0.1791
amazing	0.3721	worst	-0.5701	effort	-0.1698
save	-0.3080	hilarious	0.5321	bad	-0.1630
surprised	0.2517	solid	0.4683	animated	0.1418
great	0.2408	seconds	-0.4347	too_much	-0.1325
makes	0.1621	supposedly	-0.4334	once	0.1248
human	0.1569	coherent	-0.4180	looks	-0.0956

Figure 6: Top 10 coef of positively, negatively & incorrectly predicted review

6. Conclusion and Next Steps

The project gave us deep insights into the sentiment's analysis. We realized that how selected of vocabulary can be as critical as model itself as we cannot always afford to have all words due to complexity and overfitting. Using a single strategy, we were able to achieve the AUC target but with 2000 words as we trimmed the vocab size to 100, AUC deteriorated. So, a hybrid algorithm of two-sample t-test and Lasso, gave the best set of words to meet both targets. In our example, for classification, Ridge logistic regression performed better than XGBoost in accuracy as well as run-time for limited set of algorithm parameters analysed in this project.

As next step, we can do further parameter tuning for XGBoost as intuitively it should result into better performance. We can also analyse models such as neural networks, random forests at different parameter settings. In vocab creation, we can add more subject specific stop words such as "movie", "actors" etc and see the impact on performance. We can also apply stemming algorithms to avoid duplicate words such as "surprise", "surprisingly", "surprise" etc. We also need to improve the model limitation for predicting moderate reviews.

Another interesting observation was that R didn't set up the working directory correctly in for-loop after first split by passing partial path. So, we had to use the full path to make it work [3].

6. Acknowledgement

[1] https://www.kaggle.com/c/word2vec-nlp-tutorial

[2] Professor and TAs: Amazing help were provided in the office hours to understand concepts of different elements of the projects.

[3] http://stackoverflow.com/: Several pages on this website helped in some nuances of coding.