Text Classification using Count/TFIDF Vectorizer and Word2Vec

Name: Wenyuan Chen
PID: A15516589
Course: CSE156

Abstract

While the baseline of the text classification with white-space tokenization achieves 0.7773 test accuracy, this report aims to explore better feature engineering methods while using the same classifier – logistic regression. This report is divided into 2 parts. The first part compares baseline CountVectorizer with TFIDFVectorizer. The second part also explores a new prepossessing method and explore word2Vec method. **Keywords:** TFIDFVectorizer, CountVecotrizer, Word2Vec, Feature Engineering

Introduction

The baseline model without tuning ngram range and C regularizer within the logistic regression has achieved train accuracy 0.982 and test accuracy 0.777. The overfitting problem exists as the difference between train and test accuracy is too big to neglect. Thus, Part 1 and Part 2 aim to increase the test accuracy and better generalize the model. In the first part, TFIDFVectorizer beats the baseline by 2%. In the second part, before tokenization, all input text is turned into lower case and all the punctuation and stop words is removed. Using this cleaned input data, performance of baseline model doesn't improve while TFIDFvecotrizer outcompetes the baseline by 2.1%. Word2Vec has worse performance than baseline, 12% less test accuracy. The details leading to the result will be discussed in the method and result section.

Method and Result 2.1 Guided Feature Engineering

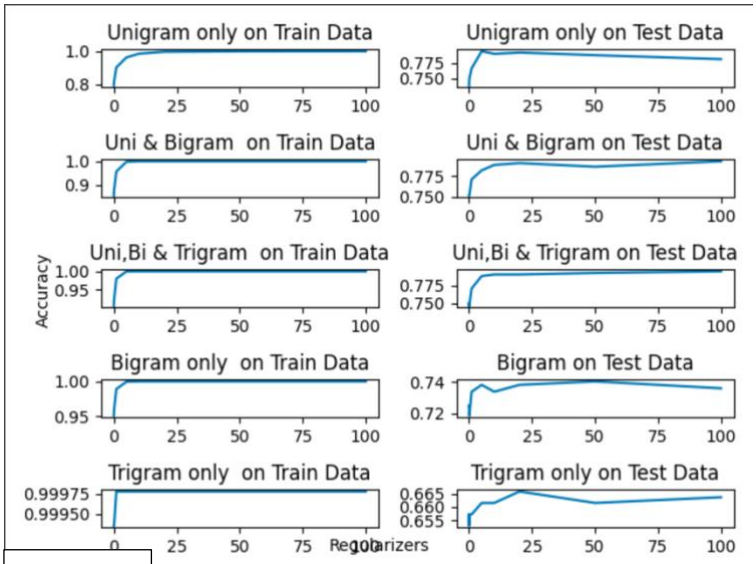
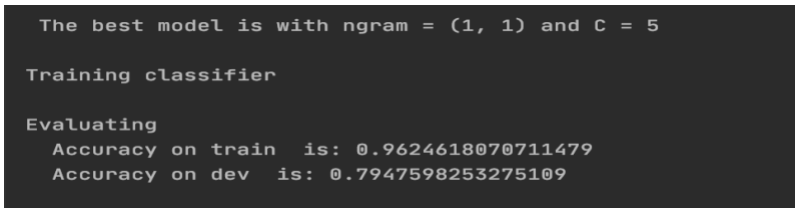


Figure 1

The model explored is TFIDFVectorizer. The tokenization is white-space and no features are added or removed. I grid searched through 5 different ngrams ([1,1],[1,2],[1,3],[2,2],[3,3]) and 10 different regularizers ([0.001, 0.01, 0.1, 1,5,10,20,50,100,1000]) and obtain Figure 1 on the left.

From the figure 1, we can see that the overfitting issue persists across different ngrams and regularizers since the gaps between train and test accuracy isn't small enough. Also, overall, combination of different grams perform better than just one type of gram. For example, Uni & Bigram model has better test accuracy than Bigram only. Similarly, Uni,Bi & Trigram model better than Trigram only. The greater the regularizer value, the less regularization; the model thus can get very complicated and thus the training accuracy can always achieve 100%. From figure 1, large regularization value assoiates with better test accuracy. However, once passes certain values, the test accuracy doesn't change as much, as we can see for the 5 models after approximately value 12 for regularizer, the test accuracy plateaued.

Figure 2



From figure 2, we can see that TFIDFVectorizer with unigram(1,1) model and regularizer =5 for logistic regression has the best performance, which is improved rom 0.777 to 0.795. The overfitting problem is slightly mitigated as we can see that the difference between train and test accuracy is smaller compared to the baseline model(0.962–0.794 vs 0.982–0.777). I think the performance improved because we changed from CountVectorizer to TfidfVectorizer. Usually the most frequent words such as article "the" don't have the most significance meanings but they have the most weights in CounVectorizer model whereas Tfidf model copes with this issue and thus the performance improved.

Method and Result 2.2 Individual Feature Engineering

I experiment with 2 prepossessing ways of removing text features. For both ways, all the input texts are lower cased, but one removes both stopwords and punctuations while the other removes only the punctuations. After both ways, I vectorize the cleaned data with 1) CountVectorizer 2) TFIDFVectorizer 3) Word2vec. My rational for converting all input texts into lower case is that the model will treat the same words the same way regardless of the capitalization, which will better generalize. My rational for removing punctations and stopwords is that by removing unrelated information, the model when training will reduce noises.

1.CountVectorizer

a. Remove both stopwords and punctuation

In this section, I use the same setup as baseline model but prepossessing the text feature as the subtitle above suggests. The result is in figure 3. The test accuracy decreases from 0.777 to 0.742 while the train accuracy also decrease from 0.982 to 0.977. This change indicates that some feature I removed actually is indicative, which explains the decrease in accuracy. To confirm my conjecture, I ran the same model with 9 different regularizers [0.01, 0.1, 1,5,10,20,50,100,1000]

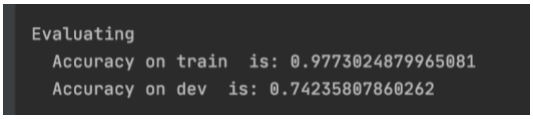


Figure 3

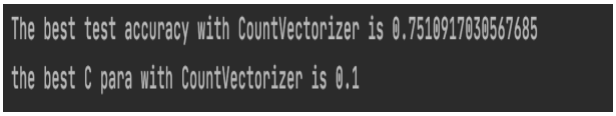


Figure 4

The result from figure 4 suggest that no matter how I tune the hyperparameters the highest accuracy is 0.751 which is still less than the baseline (0.777), which confirms my conjecture that I removed some indicative information.

b. Remove only punctuation

In this section, I ran the same model with 9 different regularizers [0.01, 0.1, 1,5,10,20,50,100,1000] as above but engineering the text feature by removing only punctuation.

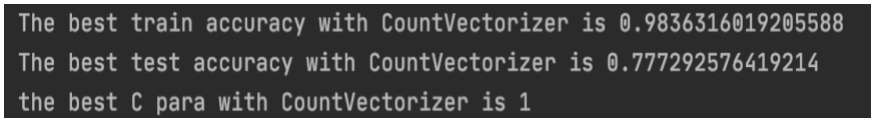


Figure 5

From figure 5, we can see that after feature engineering, compared to baseline, test accuracy remains the same with the same parameters but train accuracy increase . Thus, we can conclude that 1) "stopwords" include

some indicative information for text classification because by removing stopwords our accuracy decreases. 2) Our model overfits more compared to the baseline. The reason why we see such changes in accuracy may be removing punctuation and lowercase don't affect each token's frequency, thus don't help with better generalization. However, it reduces noise for the training data and thus contributes to the overfitting problem. Given the results, for the following two models, TFIDFVec and Word2Vec, I will only remove punctuation and do lower case.

2.TFIDF Vectorization

a. Remove only punctuation

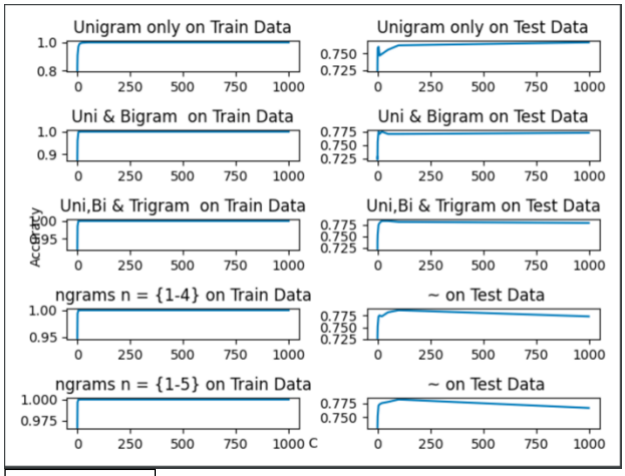


Figure 6

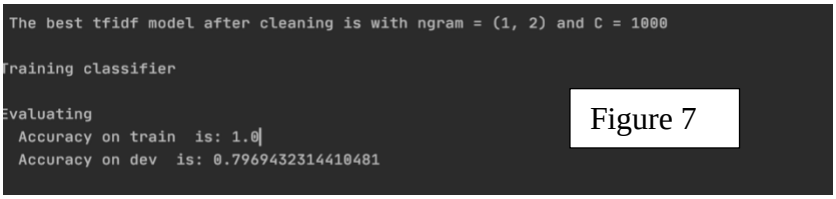


Figure 7

In this section, the model explored is TFIDFVectorizer. The tokenization is white-space and punctuation removed and lower cased. I grid searched through 5 different ngrams ([1,1],[1,2],[1,3],[1,4],[1,5]) and 9 different regularizers ([0.01, 0.1, 1, 5, 10, 20, 50, 100,1000]) and obtain Figure 6 on the left. We can see that the overfitting issue persists across different ngrams and regularizers since the gaps between train and test accuracy isn't small enough(22%).

The result is in figure 7. We can see that compared to baseline, train accuracy increase from 0.982 to 1.0 while test accuracy increase from 0.777 to 0.797. This increase might come from the fact that removing punctuation and lower case allows the model to only consider the key feature rather than punctuation that used to separate sentences or phrases.

3.Word2Vec

a. Remove only punctuation

In this section, the model explored is Word2Vec where each word has its own vector representation. I grid searched through 3 vector sizes [50,150,300] and 9 different regularizers [0.01, 0.1, 1, 5, 10, 20, 50, 100,1000] and obtain Figure 6 on the left. However, it's a classification task for each review. According to reference 1, I averaged the vector representation for each review to generate review level vectors.

Figure 8

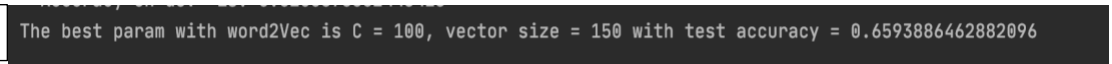
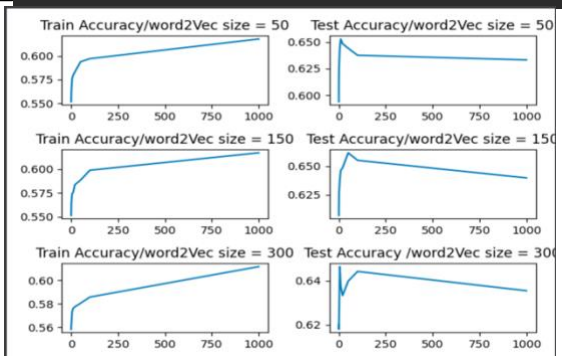


Figure 9



From Figure 8 and 9, we can see that Word2Vec has worse test and train accuracy than baseline regardless of the hyperparameters. I think word2Vec fails to work because 1) we don't have large enough data sets to gather informative embeddings. 2) it can't handle unseen words well. If there is any unseen word in the test data, I have to give all zeros for that vector. Thus word2Vec might not be generalize well for this specific classification task.

Conclusion: Stop words carry indicative information and TFIDF vectorization model with removing punctation and lower case prepossessing outcompete

Reference

1. *Table of contents*. word2vec_text_classification. (n.d.). Retrieved April 11, 2022, from http://ethen8181.github.io/machine-learning/keras/text_classification/word2vec_text_classification.html