Assignment1: Text Classification

CSE256: Statistical NLP: Spring 2022
University of California, San Diego

Zhaoxing, Lyu
A59011303
zhlyu@ucsd.edu

1 Introduction

In this project, a supervised text classifier was built to classify the sentiment of user reviews into negative or positive reviews. There are two sections in this project. The first one is Guided Feature Engineering by changing the basic CountVectorizer to a vectorizer utilizing Term Frequency Inverse Document Frequency (TFIDF), and the second one is finding other ways to perform feature engineering on this model. These two parts will be illustrated in this report.

2 Guided Feature Engineering

In this part, CountVectorizer will be changed to Term Frequency Inverse Document Frequency (TFIDF) to further alter the N-gram from 1-gram to 2-gram (a bigram) and 3-gram(trigram). Additionally, the regularization parameter, C, in the logistic regression will be modified to improve the result.

2.1 N-gram

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech, where items are words in this project. During this project, TFIDF was implemented to change 1-gram to 2-gram and 3-gram and the result is shown in the table1.

It can be found that the change of N-gram does not improve the result of the baseline which is 1-gram. Additionally, if just considering 2-gram and 3-gram, the result in dev set become worse.

2.2 Regularization Parameter C

In this part, the regularization Parameter C in the logistic regression. To have the better result in this part, (1,2) ngrams were chosen. C was changed from the default 1 to 5, 10 and 20 to see the difference and the result is illustrated in table 2.

This result indicates that with the increase of the C, the result of both train accuracy and dev accuracy were improved. C is the inverse of regularization strength. Larger the C values, the weaker regularization, which allows the model to become more flexible.

Table 1: Different accuracy on train and dev sets on different ngram range

ngram_range	Accuracy	
	train	dev
(1,1)	0.9821038847664775	0.777292576419214
(1,2)	0.957005674378009	0.7707423580786026
(1,3)	0.9801396769969446	0.7707423580786026
(2,3)	0.9975993016150153	0.7205240174672489

Table 2: Different accuracy on train and dev sets on different regularization Parameter C

C	Accuracy	
	train	dev
1	0.9821038847664775	0.777292576419214
5	0.9978175469227412	0.7816593886462883
7	1.0	0.7882096069868996
10	1.0	0.7903930131004366

Table 3: Different accuracy on train and dev sets on different independent feature engineering

Feature Engineering	Accuracy	
	train	dev
baseline	1.0	0.7903930131004366
basic preprocessing	1.0	0.7882096069868996
stemming	1.0	0.7925764192139738
lemmatization	1.0	0.8078602620087336

* the n-gram is (1,2) and regularization Parameter C is 10

3 Independent Feature Engineering

Feature engineering is the process of improving the results from a machine learning process by using domain knowledge to extract features from raw data. Apart from guided feature engineering, three methods were proposed and implemented in this section, including text preprocessing, lemmatization, and stemming. The result of these methods will be illustrated and evaluated.

3.1 Basic Text Preprocessing

Text preprocessing is an important step in natural language processing (NLP) tasks. It cleans the text to allow the machine learning algorithm to have a better outcome. In this project, removing numbers, punctuation and lowercase are three achieved basic preprocessing steps. The result is illustrated in table 3

However, the test result shows no improvement compared to the baseline, which illustrates that the text data used in this project is clean enough, and preprocessing, including the three steps above, does not affect the performance.

3.2 Stemming

There are words with the same meaning in a text but have some variations, which will confuse machine learning training and prediction. For example, a stemmer of English Stemming is implemented in this step to solve this issue. In linguistic morphology, stemming is the process of reducing inflected words to their word stem, base, or root form. In Natural Language Processing, stemming is the process to normalize the words with the same meaning but have some variations. For example, a stemming algorithm can reduce the word fishing, fished, and fisher to stem fish.

The result of this implementation is illustrated in table 3. It can be found that there is an improvement compared to the baseline, indicating this implementation has a positive effect on the result. Though it is easy for a human to find the words with variations, these variations create ambiguity in machine learning training and prediction. Therefore, to improve the accuracy of the NLP algorithm, using stemming from recognizing and changing these words to a uniform type is essential.

3.3 Lemmatization

Apart from stemming, there is another implementation called lemmatization, which is achieved in this project. In linguistics, lemmatization is the process of grouping together the inflected forms of a word so that they can be analysed as a single item. Unlike stemming, the purpose of lemmatization is to

correctly identify the intended part of speech and the meaning of a word in a sentence. For example, the base form 'walk' is the called the lemma for the word, which may appear as 'walked', 'walks', or 'walking'.

The result of lemmatization is shown in table 3, indicating that this implementation had the best accuracy. This experiment further illustrates that lemmatization is a more robust operation. By cutting the suffix from the word directly, stemming is a general operation, while lemmatization is an intelligent operation where the proper form will be looked at in the dictionary. Therefore, lemmatization forms the better features in machine learning.

4 Conclusion

This report implemented guided feature engineering, which changed the 1-gram to n-gram, but there was no improvement. Changing the regularization parameter C to more significant improved the result significantly as the model became more flexible. Further, independent feature engineering methods were implemented, including introductory text preprocessing, stemming, and lemmatization. The results indicated that basic text preprocessing like removing numbers, punctuation, and lowercase did not have a good effect. Stemming and lemmatization improved the accuracy to varying degrees. Lemmatization also had a better outcome than stemming, indicating it is a more intelligent operation in improving machine learning features.