Sentiment Analysis with Support Vector Machine (SVM)

Nyoman Wikananda Santana (its50022201043)

30 April 2024

1 Introduction

User sentiment holds valuable insights for product reviews and improvements. One of the key aspect in machine learning called *sentiment analysis* which is a branch of *natural language processing* (NLP) used to detect the user sentiment. Throughout the time, the implementation of NLP methods has been used to analyze and categorize user sentiment review. One popular model is a simple machine learning model called *Support Vector Machine* (SVM). SVM is a popular machine learning model in classification area which shown promising result in classification task, including text classification.

The use of traditional machine learning model such as SVM supported by the small data available. Deep learning model tends to overfit, unable to generalize on a small dataset. This work will utilize SVM to deal with sentiment analysis task given a quite small dataset of 1500 user reviews. This work also shows that a simple SVM model without any feature engineering could outperform SVM model with feature engineering.

2 Methodology

This work implements SVM and utilizing some feature engineering techniques to try maximize the result. The whole experiment is done by analyzing the data, applying some pre-processing, choosing feature engineering technique, applying SVM and finally testing the result.

2.1 Dataset

The dataset used is movie review dataset from Kaggle that consists only 1500 reviews for training set and 500 reviews as the test set. The training dataset will be split further into training set, evaluation set. The evaluation set takes 10% of the whole training dataset and leaving the remaining 90% for training.

The next step will be looking further into the data. Figure 1 shows that the dataset is quite balanced in term of distribution. The dataset itself contain 752 positive reviews and 748 negative reviews, which clearly tells that the dataset is balanced.

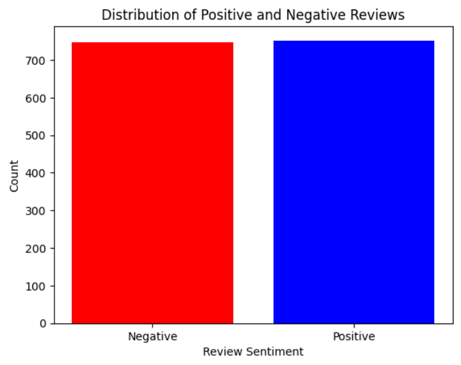


Figure 1: Number of positive reviews and negative reviews

Most of the reviews written in around 4000 characters with the longest one consists of more than 12000 characters. More reviews consists around 6000 characters and around 2000 characters.

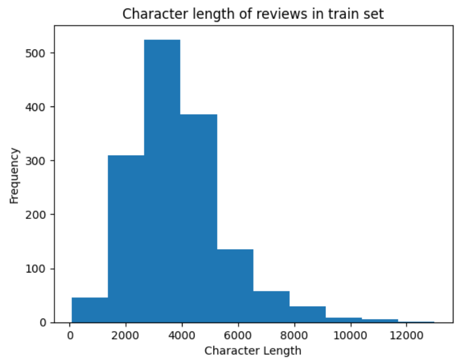
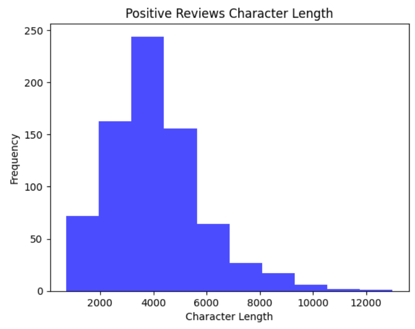


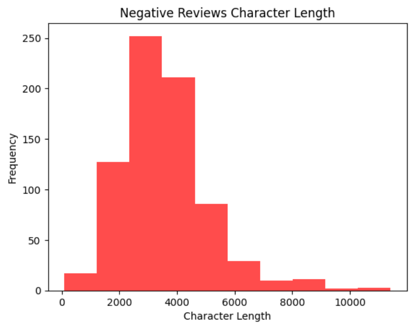
Figure 2: Character length of reviews in train set

With such distribution, we can plot for each positive review's character length and negative review's character length. As the result in Figure 3, the distribution shape is quite similar with overall character length of the data. Moreover, the distribution shape is also identical each other although we can see negative reviews has more data with around 4000 characters.

When it comes to word counting, It can be seen in Figure 4 that the word counts distribution shape also identical with the character length distribution shape. Moreover, in Figure 5 clearly show that both word count distribution in positive reviews and negative reviews offer identical distribution shape. This emphasize how well balanced the whole dataset is.



((a)) Positive reviews' character length



((b)) Negative reviews' character length

Figure 3: Plot of character length distribution in each positive reviews and negative reviews

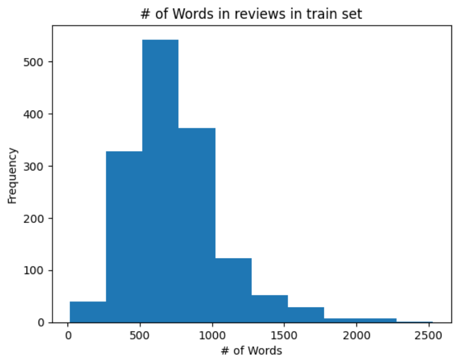


Figure 4: Number of words in train set

2.2 Pre-processing

The pre-processing applied to the dataset is as follows:

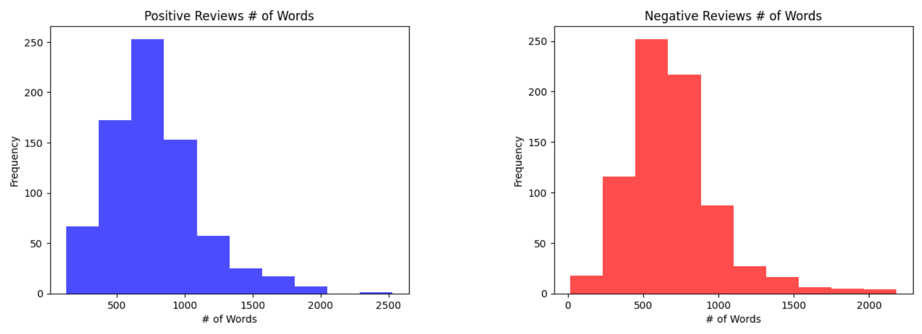
1. Lowercase

   Lowercase used to make all characters in the training and test set to the lower case to reduce vectorization complexity.

2. Convert Emoticons, Emojis, and Chat Words

   The dataset could contain some emoticons, emojis or chat words. Those unicodes could contain meaningful emotions, so it is necessary to convert them into actual text.

3. Stopword Removal



((a)) Positive reviews' words count ((b)) Negative reviews' words count

Figure 5: Plot of character length distribution in each positive reviews and negative reviews

Stopwords in english such as 'the', 'a', 'is' does not contain enough emotion to the text. Removing it would still allow the model to capture overall emotion and reduce text dimensionality.

4. HTML Tags Removal

   The texts contain some html tags such as '<br>', '</br>' and other tags such as '\n'. These tags will be removed to reduce the text dimensionality as well.

In this work, stemming or lemmatization did not performed as tested that doing so would reduce the SVM performance.

2.3 Feature engineering

This works tried to utilize feature engineering to try boost the model performance. Some ideas used for feature engineering is word count, positive word counts, negative word counts, and finally the sentiment score. For the positive and negative word counts, we use `afinn` package library to count them as `afinn` has a lexicon of positive and negative words which is useful for sentiment analysis. As the sentiment score, we calculate based on the word count, positive word counts, and negative word counts, following formula:

$$\text{sentiment} = \frac{\#\text{-positive} + \#\text{-negative}}{\#\text{-words}} \tag{1}$$

Doing all of these giving us 4 extra new features that will be used as another experiment using SVM. The result of the feature engineering could be seen from Table 1:

Preprocessed	total_len	pos_count	neg_count	sentiment	category
airplane ! con...	686	18	24	-0.01	positive
got love dis...	1336	52	30	0.02	positive
characters jon...	1129	38	26	0.01	negative
vikings v ...	687	29	32	-0.00	negative

Table 1: Feature engineering result

2.4 Preparing SVM Model

Since SVM model accepts vector, the text data need to be transformed or embed into vector form. TF-IDF will be used to do the task. Vectorizing using TF-IDF giving us the vector representation of the training set and the test set. As for the extra features, we will normalize them first using Z-norm technique which is provided by `sklearn.StandardScaler` library. After normalizing, the extra features will be combined into the vector representation of the training set and the test set.

The SVM model used is a linear SVM which initialized with parameter `C=1` and `gamma=0.001`. Other type of SVM like polynomial, rbf, and sigmoid was experimented, but didn't give favorable result as much as linear SVM does. The model will be tested using the data with feature engineering and without any feature engineering to see how the feature engineering affects the model performance.

2.5 Evaluation

After training, the model will be evaluated using the evaluation set. The evaluation metric used is the standard accuracy score following formula:

$$\text{Accuracy} = \frac{y_{\text{pred}}}{y_{\text{true}}} \tag{2}$$

where y_{pred} is the total correct guess and y_{true} is the total label to be guessed.

3 Results

As mentioned, this work experimented applying SVM on the dataset with and without feature engineering. The result of the prediction can be shown in Table 2:

As it can be seen, the result without any feature engineering outperform the result with feature engineering, breaking our expectation that feature engineering might be helping us boost the model performance.

There are several things that could cause such result. From Table 1, we can see for the first text giving us more negative words than positive words,

	With feature engineering	Without feature engineering
Accuracy	0.8467	**0.8933**

Table 2: Accuracy result of SVM model with feature engineering and without feature engineering.

resulting in negative sentiment. But the category of the first text is positive. The same thing happen in the third text, where there are more positive words than negative words resulting positive sentiment, but the actual sentiment label is negative. The positive and negative word count using `afinn` is not really effective to calculate the actual positive and negative words, hence causing misalignment in the sentiment score and the actual sentiment label. Such things could affect and cause confusion to the SVM model, hence lowering its performance.

4 Conclusions

This work use SVM to categorize the sentiment of user review, given a small training set. This work also utilize feature engineering to be experimented against the data without any feature engineering. The result show that SVM model perform better given the data without any feature engineering with accuracy 89.33% while using the data with feature engineering only giving accuracy of 84.67%. For further work, another lexicon of positive and negative words can be explored to improve the sentiment score result, hence might boost the model performance with the feature engineering.