Twitter Sentiment Analysis and Prediction using Machine Learning in Python 3

University of Melbourne
Parkville, VIC 3010

1 Introduction

Twitter is a social media platform, with the main purpose of allowing people to share their opinions about ideas and issues worldwide through text and forms of multimedia known as "tweets". Sentiment analysis is the identification of the emotional tone behind text using natural language processing (NLP). Social media platforms often provide businesses and brands with honest and unbiased opinions about a product or service. Sentiment analysis can be utilized to gather this vast source of data to analyse individual emotions and overall public sentiment (Roldós, 2020).

The goal of this project is to build and analyse supervised Machine Learning techniques, to accurately predict the sentiment (negative, positive or neutral) behind Tweets.

The dataset we are using to construct and test our model is adapted from the data provided by the 2017 SemEval conference (Rosenthal, 2017). The dataset is split into a "Train" and "Test" set. We train and tune our model on the "Train" set, which is composed of tweets from 21802 authors. In this set, each row contains a tweet ID, the text from the tweet and the sentiment behind that tweet. For example,

"632977936187027000, seriously debating going to see paper towns on my own tomorrow, neutral"

We test our model on the "Test" set, which has a similar format except the 6099 rows do not include a sentiment label.

2 Method

In this section, we discuss the used features, our choice of vectorization algorithms, our ML model and the rationale behind them.

2.1 **Feature Engineering**

Before building a reliable model, we must first engineer "good" features to build a system on. This begins with selecting words from the tweets to use as features. These "good" features are a subset of all features which help us to achieve the following goals (A. Yousefpour, 2017):

1. Reduce computational cost
2. Reduce/avoid overfitting
3. Enhance classification accuracy of the model

We will refer to these goals as the feature "good" -ness goals.

Consider this example tweet from the training dataset:

"@bupaaustralia @gordypls why do you cover homeopathy? https://t.co/v5c2oxmrca"

It is evident that the mentions, "*@bupaaustralia"* and web-links do not provide any meaningful information in our sentiment analysis, so they are removed from all rows. Stop words are commonly occurring words that provide very little information to our model while also increasing computation cost and increasing overfitting. It would be nonsensical to include them. In our example, "why do you" is removed.

In fact, many parts of the tweets should not be considered in order to adhere to the rules we previously outlined. Numbers and non-letters (emojis, punctuation, hashtags) are removed. All capital letters have been turned to lower-case and accent marks are removed (é to e).

Another way we can build "good" features is through the use of lemmatization. Lemmatization is simply the reduction of words to their root words. For example, the root of "cats" is "cat" or the root of "better" is "good". This should enhance the classification accuracy of our model by standardising the language across our training and testing datasets.

To confirm our understanding, we can look to answer the question "do our chosen set of features make "good" features?". We will test the accuracy and speed of our model using the entire feature set vs the subset we have chosen in the "Results" sections.

2.2 **Word Vectorization**

Vectorization is simply the process of converting words into vectors so our model can judge the "distance" between them. This is the step where our features are created.

We considered the implementation of two vectorization techniques:

- Bag of Words (BoW)

- Term frequency-inverse document frequency (TF-IDF)

BoW is a naïve approach to vectorization which does not consider which words are more "important". This is not useful when we are not certain about the signal in the data, which is the case with our dataset. TF-IDF is an improvement in this regard, as it provides additional information on the importance of words by applying a weighting scheme.

BoW may be uncomplicated and require less overhead than TF-IDF, but the lack of information about the importance of words and the numerous other limitations make it irrational to implement here.

The hyperparameter we adjust from the defaults in TF-IDF is "ngram_range". By setting ngram_range to (1, 2), we are creating unigram and bigram features from the words. For example, the unigrams of "data science" are "data" and "science", while the bigram is "data science". By including both as features, we should gain some extra information about the semantics of the tweets which should improve the classifier accuracy.

2.3 Imbalances in Data

An issue that we stumbled upon while testing the performance of our classifiers was a low recall score for the "positive" and "negative" sentiments but an unreasonably high score for "neutral". The heavy imbalance towards the "neutral" sentiment was caused by 58% of the tweets in the training data having a "neutral" sentiment, with 25% and 17% having a "positive" and "negative" sentiment respectively. This issue was resolved by utilizing an over-sampling technique called Synthetic Minority Over-sampling Technique (SMOTE) to create new instances of the minority classes.

2.4 Machine Learning Classifiers

Many classification techniques can be used to predict the labels to our dataset, ranging from neural networks to boosting algorithms such as AdaBoost or XGBoost. However, some of these techniques can be difficult to implement, require extensive hyperparameter tuning, necessitate massive computational overhead or may simply not be viable for our circumstances.

We have chosen 4 algorithms that do not experience the previously mentioned deficiencies.

- Logistic Regression
- Linear Support-Vector Machine (SVM or SVC)
- K-Nearest Neighbour with K = 7
- Gaussian Naïve Bayes

K = 7 for KNN was chosen by comparing the recall against the accuracy. The recall for the positive and negative labels is too low for K < 7 and the accuracy is too low for K > 7.

We will investigate the question "which of these classifiers are appropriate to use here?" through studying performance metrics such as accuracy, precision, recall, Fscore, the confusion matrix of our respective models in the "Results" and "Discussion" sections. We have chosen these evaluation metrics as they thoroughly assess the performance of the classification models. A Zero-R method is also incorporated to further gauge the performance of the models against a baseline.

3 Results

In this section, we will present the results in terms of the evaluation metrics mentioned previously.

3.1 Performance of Features

Theoretically, the feature engineering our data underwent should have improved the "good" -ness of the resulting features. We will examine the resulting speed and accuracy of our feature set to determine if this is true.

```
Engineered features had accuracy 66.152% and took 5916.162 milliseconds.
Unprocessed features had accuracy 66.106% and took 6914.224 milliseconds.
```

FIGURE 1: PERFORMANCE OF "E VS U" FEATURE SETS

We fitted a logistic regression model to our engineered features, as discussed in sections 2.1- 2.2, and to an unprocessed feature set created from TF-IDF vectorizing all the words in the training data. We tested the accuracy and speed of both models, and the results can be seen in Figure 1.

3.2 Performance of Classifiers

We can investigate the performance of each classifier through the performance metrics previously mentioned.

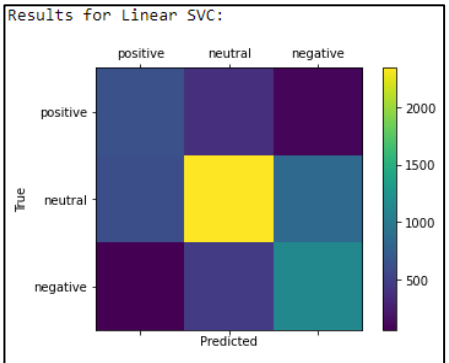

FIGURE 2: PERFORMANCE RESULTS FOR LINEAR SVC

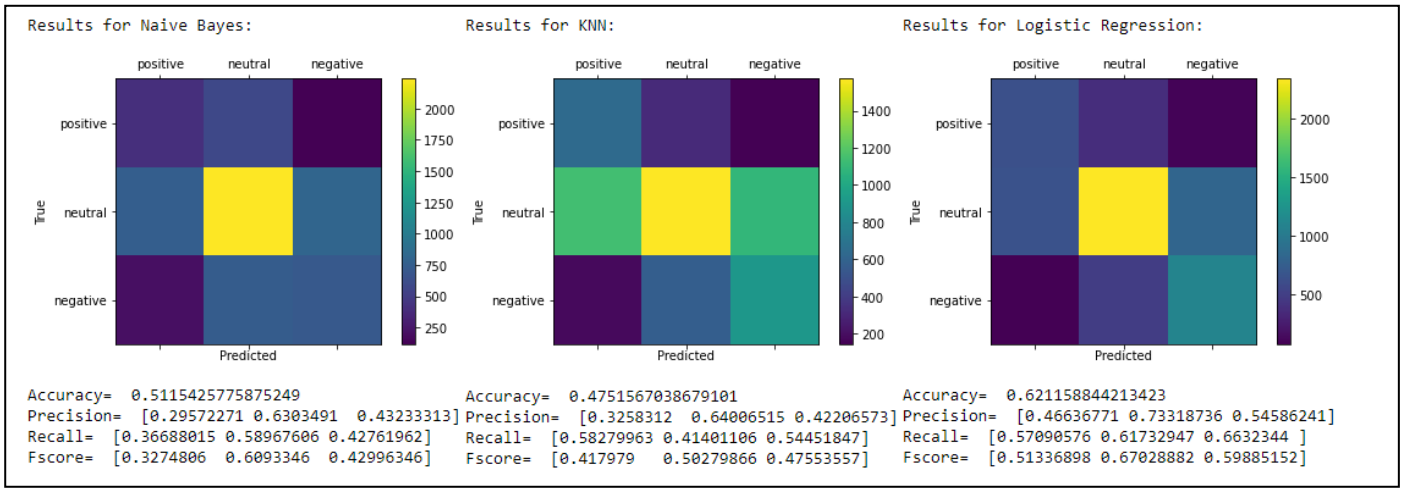

FIGURE 3: PERFORMANCE RESULTS FOR NAIVE BAYES, KNN AND LOGISTIC REGRESSION

From Figures 2 and 3 we can observe that logistic regression and linear SVC have similar performance metrics across all areas. These two are clearly the highest performing classifiers.

```
Accuracy=  0.5804922794679712
Precision=  [0.         0.58049228 0.        ]
Recall=  [0. 1. 0.]
Fscore=  [0.         0.73457148 0.        ]
```

FIGURE 4: PERFORMANCE OF ZEROR

In fact, KNN and naïve Bayes had lower accuracy than the zeroR baseline model, as can be seen in figure 4. We will discuss why this may be the case in the next section.

4 Discussion

In this section, we will discuss the possible reasons behind our results in section 3 and attempt to contextualise the systems' behaviour.

4.1 Analysis of Features

From the results in section 3.1, we can look to answer the question asked in section 2.1 "do our chosen set of features make "good" features?".

We observe that engineering the features has a positive impact on the accuracy of our model, as well as the computational overhead. This is the result of producing more important features that better capture the signals in the data, and by using a smaller model with fewer features, respectively. The method of feature engineering applied will always produce a smaller model as the engineered features are a subset of the entire feature set.

This leads us to deduce that the engineered features advance our models towards the first and third feature "good" -ness goals. In fact, the fewer features allow the model to reduce overfitting by increasing its ability to generalize.

Since the engineered features tick all the boxes of the feature "good" -ness goals, we can conclude that they are "good" features, or at least that they're more "good" than simply using all the words in the tweets as features.

4.2 Analysis of Classifiers

The accuracy for KNN and naïve Bayes is lower than that of logistic regression and linear SVC in section 3.2. This is relatively simple to explain.

The "naïve" in a naïve Bayes classifier indicates that the classifier assumes all attributes are independent when conditioned upon class labels. This means that this classifier requires all attributes to be independent to function effectively. Most real-world data sets do not have completely independent features and in fact, many carry similar signals towards a label. This is no less true for our testing dataset, where countless tweets could have homogenous signals.

The main weakness of the K-nearest neighbour classifier is the Curse of Dimensionality. This simply means that the number of data points required grows exponentially with the number of features. This is an obvious problem for our model as we have 166121 features and 21802 data points so we can understand why all performance metrics of the KNN classifier are so low.

This leaves two choices to answer the question in section 2.4, "which of these classifiers are appropriate to use here?". It is rather difficult to differentiate between SVC and logistic regression based on our performance metrics, however, SVC has slightly higher performance across most areas. This paired with (N. L. M. M. Pochet, 2006), which states that SVC is less sensitive to outliers, we can conclude that linear SVC is the best classifier for this problem. Logistic regression is also appropriate to deploy.

5 Conclusion

In closing, we have used NLP techniques to engineer features from thousands of tweets from a dataset and vectorized them using TF-IDF, tested their performance against the larger feature set, used 4 classification techniques to model the data and assessed their respective performances using a broad range of metrics. We can conclude that linear support-vector machine is the most effective classifier for the prediction of the sentiment behind tweets.

6 Bibliography

A. Yousefpour, R. I. (2017). Ordinal-based and frequency-based integration of feature selection methods for sentiment analysis. In *Expert Syst. Appl. 75* (pp. 80–93).

N. L. M. M. Pochet, J. A. (2006, May 19). *Support vector machines versus logistic regression: improving prospective performance in clinical decision-making.* Retrieved from Obstetrics & Gynaecology: https://obgyn.onlinelibrary.wiley.com/doi/10.1002/uog.2791

Roldós, I. (2020, April 10). *8 Applications of Sentiment Analysis*. Retrieved from MonkeyLearn: https://monkeylearn.com/blog/sentiment-analysis-applications/

Rosenthal, S. N. (2017). Retrieved from SemEval-2017 Task 4: sentiment analysis in Twitter. In Proceedings of the 11th International Workshop on semantic evaluation (SemEval ’17): http://alt.qcri.org/semeval2017/