Predicting Sentiments of Twitter Tweets by Training and Testing Machine Learning Models

James Condos

1. Introduction

For this task, we created Machine Learning models to predict the sentiment of tweets through training data and testing data (Rosenthal, 2017). It was seen that the training data provided has 28002 entries of tweets. Each row contained a unique id and sentiment value consisting of one of three discrete values (negative, neutral or positive). It also contained the tweet text itself.

This has been accomplished through training models on this data and testing on our unlabeled data sets. These unlabeled data sets consisted of 6100 data entries, with unique id's and a tweet associated with each one. This can be seen as an artificial split of 70:30. As such, this required supervised learning models to make use of the labeled training data. The training data consisted of 3715 negative tweets, 5428 positive tweets and 12659 neutral tweets. One could argue that the data is extremely skewed, which was evident in the analysis.

We implemented the following machine learning models as part of our statistical analysis: Naive Bayes, KNN (K-Nearest-Neighbour), Random Forest Classifier, SVM (Support Vector Machines), and SGD (Stochastic Gradient Descent). These statistical models use probabilistic outcomes, similarity of features grouping and distinctions between multiple classes. In addition to these models, we used a baseline (control) model Zero-R. A simple model that predicts a constant value; the most common sentiment from the training dataset. It allows for us a point of comparison against the other models, providing additional reliability and enabling us to further validate our results.

The analysis has shown that there are more ideal models used to scrape twitter data to make predictions on unlabelled data, which will be further discussed in this paper. We do this by applying machine learning practices such as preprocessing, and parameterization. Our aim being: to make the most accurate predictions for sentiment analysis, and to establish which models are most useful/accurate in the pursuit of this goal. We found through various trials that ultimately, certain features and certain models are more congruous with better accuracy scores.
It must be noted that there are more advanced techniques such as neural networks to optimise the predictions of these sentiments, however, this is **out of the scope** of this research paper.

1.1 Hypothesis

From the data sets (Rosenthal, 2017), our hypothesis seemed to be that the models created would give accuracy scores that are not congruent with good machine learning models. This could be accounted for by the fact that the sentiment data for the training module is extremely skewed, with the majority sentiments being neutral. Hence, we would expect accuracy scores ranging from 60 to 70 from, as seen from (Suchdev, Kotkar, Ravindran, Swammy, 2014). As such, we would expect our models to predict neutral more often than the other discrete variables in our domains, while the model would predict positive and negative labels at about the same rate.

2. Literature Review

Much of the discussion revolving around these papers has been focused on various literature discourse. Before starting our research, we delved into other academic techniques for sentiment analysis and other similar tasks. Below are some of the other papers and results we found. As such, it was clear that the results discussed in these papers could be used as a basis and starting point for our investigation.

(Vishal, Sonawan, 2016) had shown that a SVM (Support Vector Machine) model was almost **exactly as accurate** compared to the use of their Näive Bayes model (Accuracy Scores of 74.56 Vs 76.68 Respectively). This gave a good baseline to work around, and to expect our accuracy scores for these two models to have a percentage difference of +/- 5%.

3. Methods

Our method involved a variety of stages. Initially we brainstormed different possible features that could be gathered from the twitter data. Then preprocessed the twitter data by stripping unnecessary information and segregating useful information into its different features. We then conducted statistical analysis of features and their influence upon sentiment, then analysis of six different models and comparison of their performance results. Being sure to note the highest performing models.

3.1 Feature Brain-Storming

The figure below indicates some of the possible features that are available for analysis in twitter tweets (punctuation, number of mentions, number of hashtags, tweet length, number of urls, average word length) were some of the features that we used. In addition to those features we suspected that capitalisation, or capitalisation percentage would have been highly useful in predicting sentiment, regrettably the training data was devoid of a letters case.

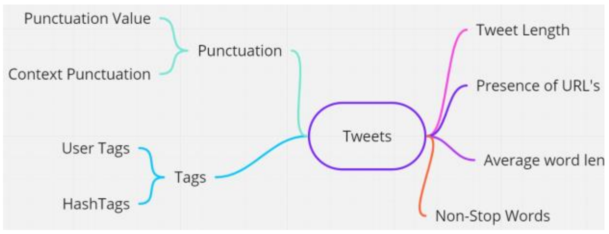


Figure 1- Figure includes feature relevance for tweets through and brainstorming for model evaluation.

3.2 Pre-Processing and feature Segregation

It was evident that the tweets had lots of information that was distracting from sentiment analysis or at least the word analysing techniques such as TFIDF. Thus the first task was to strip the input data into different features, then analyse whether any of the features were useful for sentiment analysis later. The features removed from the text data was the following: tweet length, number of links, number of mentions, number of hashtags, the tags themselves (through tfidf), the punctuation, number of non ascii characters, as well as post preprocessing features such as average word length and tfidf of the left over words.

It was highly necessary that the preprocessing and cleaning of the input data was done in a correct order to limit information loss. Such as stripping links, and hashtags and mentions before stripping punctuations.

The remaining text was purely words which was turned into tf-idf data so as it could be analysed by the machine learning models. Another categorical feature that was also converted to tf-idf was hashtags.

tf-idf reflects how important words are in a corpus and enables words to be weighted before entering the models. The sparse tf-idf matrix acted as the primary/most important feature for our model.

After these features were separated, the training data was analysed to see if any of the features contained were predictive or reflective on what the overall sentiment of the tweet would be. If any feature had a significantly different distribution to the training data: ie 58% neutral 24.9% positive and 17% negative then it may have predictive properties.

3.3 Feature analysis

A surprisingly significant feature was the number of links present. Any number of links present increased the likelihood of the tweet being labelled as neutral, this further increased with the number of links until at more than 2 links (possibly due to the limiting number of characters a tweet can include) 100% of tweets are labelled as neutral. These are the statistics.

Base frequency in tweet data: Neutral- 58.06%, Positive- 24.90%, Negative- 17.04%

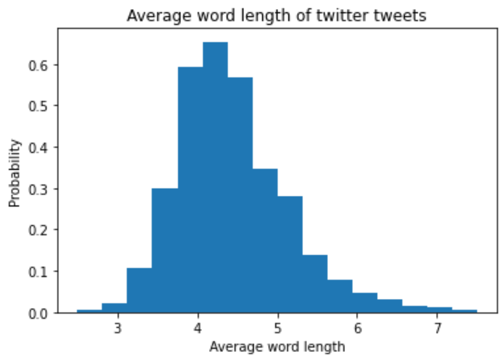
Feature links are present: Neutral- 63.10%, Positive- 22.11%, Negative- 14.79%

Feature one link is present: Neutral- 61.96%, Positive- 22.67%, Negative- 15.36%

Feature two links are present: Neutral- 73.16%, Positive- 17.22%, Negative- 9.62%

Feature >2 links are present: Neutral- 100%, Positive- 0%, Negative- 0%

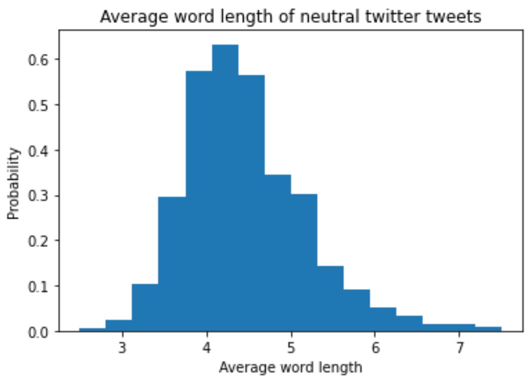
A feature that had much less information value than had been predicted was tweet mean word length. It was also somewhat counterintuitive in that negative tweets have the highest average word length, with positive tweets having the smallest. The distributions are indicated below.



Mean- 4.438

STD- 0.7544

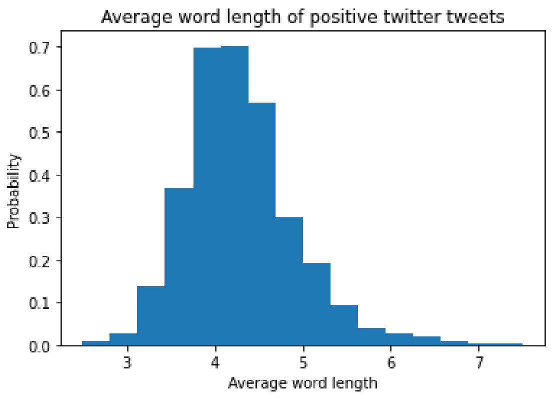
Figure 2- Figure includes a distribution of the average word length of twitter tweets.



Mean- 4.473

STD- 0.8021

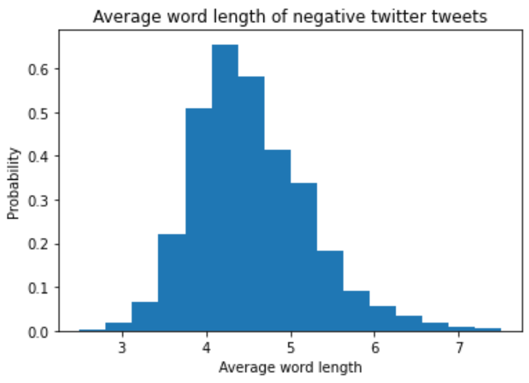
Figure 3- Figure includes a distribution of the average word length of neutral twitter tweets.



Mean- 4.295

STD- 0.6477

Figure 4- Figure includes a distribution of the average word length of positive twitter tweets.



Mean- 4.530

STD- 0.7017

Figure 5- Figure includes a distribution of the average word length of negative twitter tweets.

It was evident that in figures 2, 3, 4, 5, they are all approximately normally distributed, giving no rise to any fluctuation in predictions based on these word lengths. All with approximately similar means ranging around 4.4 and 4.5.

While we analysed the distribution and predictive abilities of many features it occupies too much space to include them all. as such they are attached at the end of this report.

4. Machine Learning Models

The six models/classifiers we implemented as part of our analysis were Bayesian (multinomialNB), random forest, support vector machine, K nearest neighbour, stochastic gradient descent classifier, as well as our base model zeroR. The best performing models were then paramatised.

4.1 ZeroR

ZeroR acted as our baseline model, as the training data was mostly neutral at 58%, we felt that a 58% predictor for 3 possible outcomes was quite good, unfortunately though, the kaggle data seemed to be much less neutral at about 34%.

This model required no input data.

4.2 Bayesian

The bayesian classifier applies naive bayes theorem and assumes independence between all features.

The Bayesian model was tested on a variety of input data, from purely numeric features to just tfidf, as well as a stacked model containing both.

Purely on the numeric features (mean word length, tweet length etc) the bayesian classifier gave a performance of 59%, while this is slightly higher than the base model, it was a disappointing result for the number of features that were fed into it.

A Bayesian model was also made purely on a sparse tf-idf word matrix. It performed better than the model that was based on the linear features, with a score of 63%.

A bayesian model was performed on the stacked matrix combining the linear features and sparse tf-idf matrix, it performed worse than the other models with a score of 61%.

4.3 Random Forest Classifier

The random forest classifier is an estimator that fits decision tree classifiers on subsamples of datasets and uses averaging to improve predictive accuracy.

The training accuracy for the RFC was 58.33%. As it was dramatically lower than the other models, we decided against conducting further investigation of this model.

4.4 SVM

The SVM SVC classifier is a support vector classifier that repeats infinitely until the model improvement after iterations reaches near 0.

The training accuracy of the SVM SVC classifier on the sparse tf-idf matrix was 66.57%.

It was seen from our literature review in section 2, that our SVM and SGD models would hover around 75% accuracy score (though the literature was only positive or negative sentiments). Although our models did not reach this, their differences were both ~1%, which was expected.

The base parameter for the SVC has it so that it will run infinite iterations until it stops improving. This was why SVM's parameters were not optimised as it took nearly 30 minutes to run 1 training model.

Even so, SVM was the highest scoring submission on kaggle at ~57%. (all kaggle results were dramatically lower due to the difference in number of neutral values.

4.5 K-Nearest-Neighbour

The KNN classifier is an implementation of the K-nearest neighbours algorithm (predicting values based off of "k" nearest values).

The KNN performed a 60.15% accuracy based on the tf-idf matrix. Even with changes to K, this value never breached over 63%.

4.6 SGD classifier

The SGD classifier is an estimator that implements regularized linear models with SGD learning. The gradient of loss is updated with a decreasing learning rate.

By default it fits a linear support vector machine, which also explains why it had such similar scores to the SVM SVC model.

The SGD classifier performed 65.1% accuracy on the tf-idf matrix.

Due to the speed of SGD it was chosen as the primary model for analysis.

SGD was also tested with the stacked features sparse matrix but like the other models this only reduced its accuracy to 61%.

4.6.1 parameter optimisation

There were no improvements to sgdclassifier with different max_iter values, only deprovements when max_iter reached below

5000. The best loss parameter was Modified Huber performing slightly better than Hinge.

Through repeated testing it was found that there was very limited difference between the performance of different alpha values.

0.00027 was found to be the best with 68% on the training data.

The best set of parameters found through parameterization were: Modified Huber, Max_itr = 1000, alpha = 0.00027, class weight = balanced.

While some of these values may be overfitted, random sampling and shuffling of train test/split hopefully mitigated some of this.

5. Conclusions

Through this research paper, we have discussed how various methods can predict sentiment analysis for twitter tweets, and how it can be improved via feature analysis. It has shown that there are various features regarding tweets that are more congruous to a higher accuracy scores for our machine learning models.

After preprocessing our data, we use vectorization methods to stack our features, including hashtags and tags. It was seen that our best model was the SGDclassifier, as it was fast and enabled parameter optimisation, as well as SVM being quite reliable, despite taking thousands of iterations and many minutes to run. These had the highest accuracies of 65.1% and 66.57% respectively, giving a ~1% difference in accuracies. This was expected as it allowed us to optimise the labelled data with appropriate smoothness properties, which is evident from our results.

Our hypothesis concluded that neutral tweets did get predicted more often than positive and negative tweets, however, our accuracy scores were quite a bit lower than expected, resulting in a 3% point difference below our threshold of 60%.

From this, this could be improved more in future iterations of this data and machine learning analysis. This could be done by implementing neural networking trees and deep learning to further extend this sentiment analysis, as it has the ability to work with insufficient information, and is quite robust.

6. References

Zahoor, S., & Rohilla, R. (2020, August). Twitter sentiment analysis using machine learning algorithms: a case study. In *2020 International Conference on Advances in Computing, Communication & Materials (ICACCM)* (pp. 194-199). IEEE.

Kharde, V., & Sonawane, P. (2016). Sentiment analysis of twitter data: a survey of techniques. *arXiv preprint arXiv:1601.06971*.

Suchdev, R., Kotkar, P., Ravindran, R., & Swamy, S. (2014). Twitter sentiment analysis using machine learning and knowledge-based approach. *International Journal of Computer Applications*, *103*(4).

Rosenthal, S., Farra, N., & Nakov, P. (2017, August). SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)* (pp. 502-518).

Gelbart, M. Mike Gelbart. (2020, December 1). CPSC 330 binus segment: ‘SGDClassifier’ and ‘SGDRegressor’. Youtube. https://www.youtube.com/watch?v=nnzltuniT_E

Brownlee, J. (2021, April 27). Stacking Ensemble Machine Learning With Python. https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/

7. Appendix

Attached below is all the statistical analysis that was performed on features.