Sentiment Analysis of Tweets

Text Mining & Language Analytics

Fergus J. P. Walsh*

25th March, 2021

1 Introdution & Datasets

In this enquiry, a number of machine learning models were created to perform sentiment analysis on tweets, in which tagged an airline and expressed `positive`, `neutral` or `negative` opinion in the tweet. A training set of 10,248 tweets and a test set of 2,196 was supplied. No contextual information is given these tweets, but all the airlines mentioned have their operations based in the United States, while Virgin America was merged with Alaska Airlines in 2018, and the brand ceased to be used in April of that year.[1] It can therefore be inferred that these tweets were written before that date by travellers in the United States. Of the observations in both the training and the test set, around two-thirds have been classified as `negative`, a fifth `neutral` and the remainder `positive` (Table 1). The relative paucity of `positive` and `neutral` tweets in the training set will hamper any model's ability to correctly classify a test `negative` or `positive` tweet (see Section 5).

Likewise, there is a large disparity in the frequencies of airlines tagged in the tweets, (Table 2 and Figure 1), and as such this training set cannot be taken as rep-

*Word count: 1,503

[1] *Forbes* "Why Virgin America Will Soon Cease To Exist", 23 March 2017, `www.forbes.com/sites/laurengensler/2017/03/23/virgin-america-brand-retired-following-alaska-airlines-merger`; *Forbes* "The Official Final Day For Virgin America Flights", 10 January 2018 `www.forbes.com/sites/ericrosen/2018/01/10/the-official-final-day-for-virgin-america-flights`.

	Training Set		Test Set	
Class	Frequency	Percentage	Frequency	Percentage
`positive`	1,630	16%	361	16%
`neutral`	2,184	21%	455	21%
`negative`	6,434	63%	1,380	63%

Table 1: Distribution of labels in the two training sets. Total number of observations in training set: 10,248; test set: 2,196. Percentages rounded to the nearest whole number.

Airline	Frequency	Percentage
American Airlines	2,075	20%
Delta	59	< 1%
Jet Blue	1,601	15%
South West Air	1,727	16%
United Airlines	2,727	26%
U.S. Airways	2,078	20%
Virgin America	349	3%

Table 2: Frequency of airlines mentioned by users in training data. Percentages rounded to the nearest whole number.

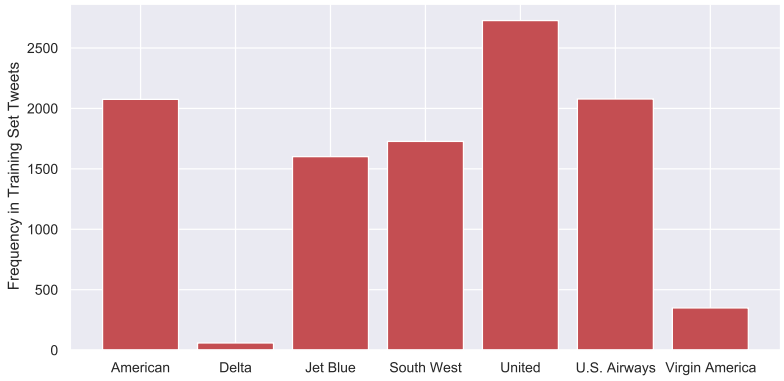


Figure 1: Frequencies of airlines mentioned in training data.

resentative of the airline industry as a whole. Note that the frequency column of Table 2 sums to 10,616, as in some tweets more than one airline is tagged.

2 Data Preparation

Firstly, the text and labels were read from the two .csv files and converted into lists. For the Naïve Bayes classifier, these were then tokenised. Although the Naïve Bayes classifier ultimately requires each text as a bag of words, and the Scikit-Learn `TfidfVectorizer` will re-tokenise each text, it is easier to remove punctuation and Twitter handles (e.g. "@Virgin America") at this stage.[2] Here too, any single token texts were removed and then the text is then reformatted as a bag of words, with each text being an element in a list.

For the neural networks, the labels and texts were likewise read and converted to lists; the labels were then encoded using the `LabelEncoder` and `OneHotEncoder`

[2] *Scikit-Learn Documentation* 2021, "sklearn.feature_extraction.text.TfidfVectorizer".

functions, in order to convert them into tensors.[3] `[1.0, 0.0, 0.0]` encodes `negative`, `[0.0, 1.0, 0.0]` encodes `neutral` and `[0.0, 0.0, 1.0]` encodes the `positive` label. The encoded labels are then again paired with the corresponding texts and saved as a new .csv, and then loaded from there as a PyTorch object. Here the texts are tokenised, and then split into training and validation folds.

It was chosen to use the Natural Language Toolkit `TweetTokenizer()` function for all tokenising operations when preparing the data. This function automatically tokenises user handles (e.g. "@SouthwestAir"), hashtags ("#notahappytraveler") and characters such as emoji and emoticons.[4] On Twitter, the hashtag can be used to tag a word or phrase to make a user's tweet easily searchable by other users; as such they often serve as a summary or description of the tweet's content. In a sentiment analysis setting, therefore, hashtags should be preserved as they may serve as significant predictors of sentiment. Otherwise, concatenated phrases, such as "#notahappytraveler", must be split up into their component elements in order to serve as predictors.

The disparity in the frequency of the different airlines may cause certain airlines to be associated by a model to a particular sentiment. For example, the token `@Delta` occurs only 52 times in the training set, of which 51 tweets are classified as `negative`, while the only `neutral` tweet is in fact praising U.S. Airways:

> 3096: `@USAirways Thank you, @USAirways! Your fare to from DTW to DCA was much lower than @Delta and @SouthwestAir! Thank you! You won me over!`

It was therefore chosen to remove user handles during the tokenisation process, to alleviate this issue.

3 Text Representation

For the Naïve Bayes classifier, text representation is handled by the Scikit-Learn `Tfidf Vectorizer` function. This function tokenises each text in the data set, creates a vocabulary for the data set, counts the occurance of each token in each text and then normalises these counts based on the token's frequency accross the whole data set (with Laplace smoothing).[5] These are the term frequency and inverse document frequency algorithms, which result in the count for each token representing its significance in the corpus.[6] In this case, tokens with greater inverse document frequency should serve as the most effective predictors of a text's sentiment. In addition, this normalisation process serves almost as a custom stop-word list based on the training data, as very common words recieve the smallest weighting. For this reason, stop word removal usually has little impact on Naïve Bayes classification performance.[7]

[3] *Scikit-Learn Documentation* 2021, "sklearn.preprocessing.LabelEncoder" and "sklearn.preprocessing.OneHotEncoder"; Brownlee 2016.

[4] *NLTK Documentation* 2020, "nltk.tokenize.casual module"; *SourceCodeQuery: "Python datasets.EMO.splits() Method Examples"* n.d., Example 2.

[5] *Scikit-Learn Documentation* 2021, User guide §6.2.3.1 "The Bag of Words representation".

[6] Jurafsky and Martin 2020, pp. 107–108.

[7] Jurafsky and Martin 2020, p. 60.

The PyTorch `TEXT.build_vocab()` and `LABEL.build_vocab()` functions index the vocabulary in the text and the number of labels, and add an index value for unknown words (`<unk>` and the padding word (`<pad>`, used for fitting texts to the batch size).[8] Then, the `nn.Embedding` function creates an embedding vector of the size of the vocabulary and the batch (the number of documents considered at once). By measuring the cosine distance between the dot product of each token vector, the semantic relationships between tokens can be computed, which allow the neural network to model semantic relationships.[9]

4 Machine Learning Models

4.1 Naïve Bayes

Two Naïve Bayes classifiers were used: a multinominal classifier and a Bernoulli or binary classifier. The Scikit-Learn multinominal classifier calculates the probabilty of token i being in sentiment class y, $\hat{\vartheta}_{yi}$ (equivalent to $P(x_i \mid y)$, as:[10]

$$\hat{\vartheta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where N_{yi} represents the the frequency of token i in class y in the co-occurence matrix of the training set N_y the frequency of all tokens in all tweets in the training data with class y, with α and αn representing smoothing parameters.

The Scikit-Learn Bernoulli Naïve Bayes classifier, however, treats the frequency of each token in each tweet as a binary variable; the frequency in the co-occurence matrix for each token therefore counts the number of texts the token occured in, rather than the total number of occurences across the whole data set.[11]

4.2 Neural Networks

Two Long-Short Term neural networks were created. The first (Model 3) used a batch size of 32, one hidden layer of 64 neurons and no dropout layer. The second (Model 4) had a batch size of 16, two layers of eight neurons each and a dropout rate of 0.5.

In both cases, the optimiser functon was `nn.CrossEntropyLoss()`, which provided a softmax output layer, necessary for multi-class classification, and so both networks returned dense outputs in order to provide the `nn.CrossEntropyLoss()` function with values for each three categories. As part of the optimisation process, a categorical accuracy function (`def categorical_accuracy(preds, y):`) takes these outputs and returns the predicted class from the maximum value for each element in the batch, and counting how many times this matches the true label.[12]

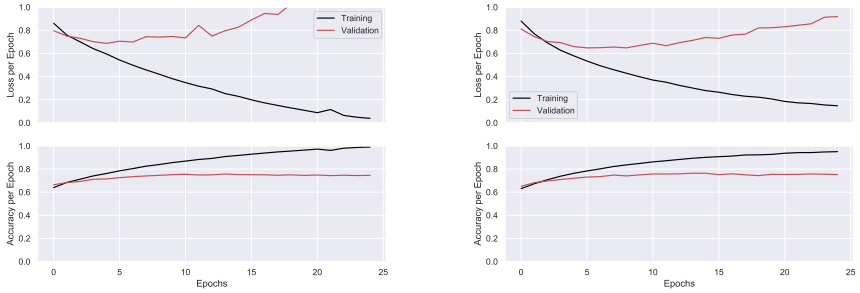
[8] *PyTorch Documentation* 2019, "torchtext.data".

[9] Jurafsky and Martin 2020, pp. 101–102, 105, 144; *PyTorch Documentation* 2019, "Word Embeddings: Encoding Lexical Semantics".

[10] Pedregosa et al. 2011, User Guide §1.9.2 Multinominal Naïve Bayes.

[11] Pedregosa et al. 2011, User Guide §1.9.4 Bernoulli Naïve Bayes.

[12] Trevett 2021, 5. "Multi-class Sentiment Analysis".



(a) Accuracy and loss values of Model 3 over 25 training epochs.

(b) Accuracy and loss values of Model 4 over 25 training epochs.

Figure 2

Figure 2 shows the accuracy and loss function against the training and validation folds during the training process. The sharp increase in the validation loss value after the fifth epoch clearly shows that Model 3 is overfit.[13] While the loss validation curve of Model 4 likewise trends upwards, it can be seen more and smaller hidden layers and a dropout layer guards against overfitting.[14]

5 Experimental Results

All four models were tested against the test data set. The text was tokenised using the `TweetTokenizer()` as before, and then the predicted labels of each model compared against the ground truth. These predictions are displayed in confusion matrices in Figures 3 and 4, while Table 3 gives performance statistics for each model.[15]

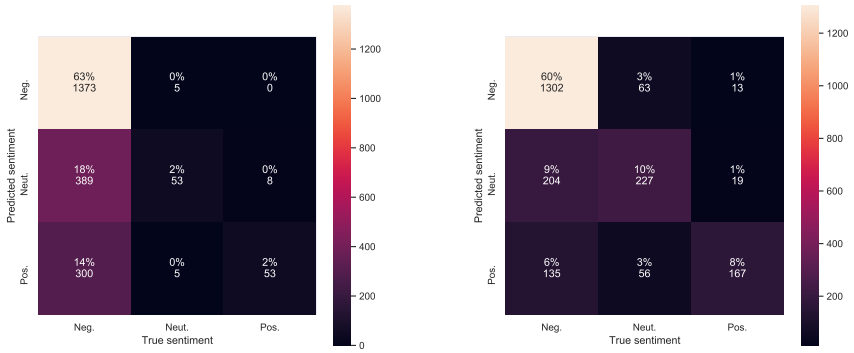
Firstly, the Bernoulli Naïve Bayes classifier seems at first to outperform the Multinominal classifier, with an overall accuracy of 68% comapred to 76%. However, accuracy does not function well as a performance metric in instances such as this, where the classes are unevenly distributed.[16] The precision of Model 1 is actually higher than Model 2, yet the recall statistics for the `neutral` and `positive` predictions of Model 2 are far more useful than Model 1's. It would seem, therefore, that if the aim of the classifier is to find positive tweets, then Model 2 is better than Model 1, and if the aim is to measure the number of negative tweets, then Model 1 is better. The two neural networks do not offer particularly significant improved performance over the Bernoulli Naïve Bayes classifier. As discussed above, both cannot take advantage of 25 epochs due to overfitting, while the lower recall of Model 3 may likewise be a result of overfitting.

[13] Chollet and Allaire 2018, p. 86.

[14] "the current collective wisdom suggests it is better to have an abundant number of hidden units, and control the model complexity instead by weight regularization," Efron and Hastie 2016, p. 361.

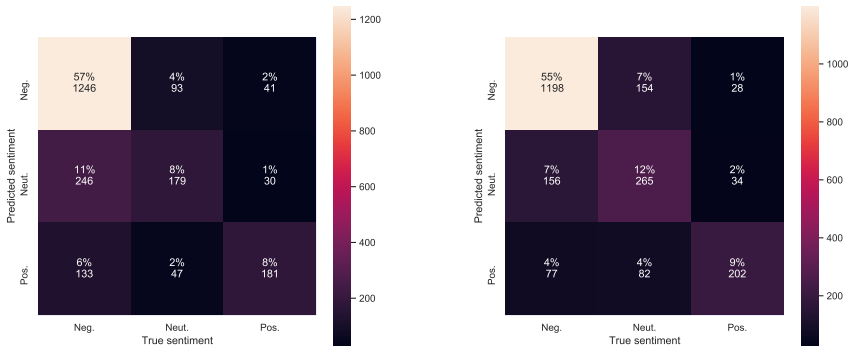
[15] Dennis T. 2019.

[16] Jurafsky and Martin 2020, p. 65.



(a) Model 1 confusion matrix. (b) Model 2 confusion matrix.

Figure 3



(a) Model 3 confusion matrix. (b) Model 4 confusion matrix.

Figure 4

	Multinominal: Model 1				Bernoulli: Model 2			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Total	0.68	0.79	0.42	0.42	0.76	0.76	0.64	0.68
`negative`		0.67	1.00	0.80		0.79	0.94	0.86
`neutral`		0.84	0.15	0.21		0.66	0.50	0.57
`positive`		0.87	0.15	0.25		0.84	0.47	0.60
	Single Layer RNN: Model 3				Deep RNN: Model 4			
Total	0.73	0.68	0.60	0.63	0.76	0.71	0.67	0.68
`negative`		0.77	0.90	0.83		0.84	0.87	0.85
`neutral`		0.56	0.39	0.46		0.53	0.58	0.55
`positive`		0.72	0.50	0.59		0.77	0.56	0.65

Table 3: Performance statistics of classifiers tested against the test data set. Figures given to 2 d.p.

6 Discussion

Unfortunately, due to the constraints of time, it was not possible to refine the models to achieve better classification results. Nevertheless, it is perhaps surprising how comparable Model 2 and Model 4 are, given that Model 4 takes advantage of word embeddings and thus uses context and semantic relations to construct its classification. It might be the case that the short length of tweets and the single subject matter of the data sets in this enquiry make such advanced techniques redundant.

Finally, a function (`tweet_predict`) was created that allows the end-user to enter a new tweet and select with which neural network to classify it. It was tested using some new tweets taken from Twitter, and one such example is shown below:[17]

```
   Please enter filename of classifier to be used (FJPW_Model3.pkl
or FJPW_Model4.pkl):
» FJPW_Model4.pkl
Please enter the tweet to be classified here:  @British_Airways
I was on the Edinburgh to London 9:25 flight (BA1435).  I
paid the £59 to upgrade.  I was then informed I that breakfast
may not be available .  I Not the case it, everyone got breakfast
apart from myself!  I paid for nothing - disappointed!
» This tweet has been classified as negative
```

[17] `twitter.com/DonnaHe82617024/status/1375166869531942918`

References

Brownlee, J. (2016). *Multi-Class Classification Tutorial with the Keras Deep Learning Library*. URL: machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library.

Chollet, F. and Allaire, J. (2018). *Deep Learning with R*. New York: Manning.

Dennis T. (2019). *Confusion Matrix Visualization*. URL: medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea.

Efron, B. and Hastie, T. (2016). *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Institute of Mathematical Statistics Monographs. Cambridge: Cambridge University Press. DOI: 10.1017/CBO9781316576533.

Jurafsky, D. and Martin, J. (2020). *Speech and Language Processing*. 3rd ed. Draft of 30 December 2020. URL: web.stanford.edu/~jurafsky/slp3.

NLTK Documentation (2020). Version 3.5. URL: nltk.org/index.html.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

PyTorch Documentation (2019). Version 1.8.0. Torch Contributors. URL: pytorch.org.

Scikit-Learn Documentation (2021). Version 0.24. Scikit-Learn Developers. URL: scikit-learn.org/stable/index.html.

SourceCodeQuery: "Python datasets.EMO.splits() Method Examples" (n.d.). URL: sourcecodequery.com/example-method/datasets.EMO.splits.

Trevett, B. (2021). *PyTorch Sentiment Analysis*. URL: github.com/bentrevett/pytorch-sentiment-analysis.