Multilayer Perceptron Neural Network for Classifying Coronavirus NLP Data

Morgan L. Ford
fordm2@rpi.edu
Rensselaer Polytechnic Institute
Troy, New York, USA

ABSTRACT

Many kinds of information has been circulated on the internet during the coronavirus pandemic, especially on Twitter. It is important to be able to get a sense of how the population is feeling about the current situation of the pandemic and current events as the status of the pandemic continuously changes. By examining tweets, I have created a Multilayer Perceptron Model that can correctly classify tweets as being positive, negative, or neutral in sentiment with 80.4% accuracy.

KEYWORDS

datasets, neural networks, natural language processing, principal component analysis

ACM Reference Format:
Morgan L. Ford. 2018. Multilayer Perceptron Neural Network for Classifying Coronavirus NLP Data. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 EXECUTIVE SUMMARY

Many kinds of information has been circulated on the internet during the coronavirus pandemic, especially on Twitter. It is important to be able to get a sense of how the population is feeling about the current situation of the pandemic and current events as the status of the pandemic continuously changes. By examining tweets, a classification model can be created to classify tweets as being of positive, negative, or neutral sentiment.

The data was provided already split into testing and training sets. It contained metadata about the tweet, the tweet itself, and the sentiment. The sentiments were split into 5 levels: Extremely Negative, Negative, Neutral, Positive, and Extremely Poisitve. I examine both using all 5 levels and combining the extreme and regular sentiments together to create 3 levels. For the purpose of creating my models, only the tweet and the sentiment are utilized.

The Multilayer perceptron was the model used to classify the NLP data. This is a feedforward neural network model that uses multiple layers, with each layer fully connected to the following one. The nodes of the layers use nonlinear activation functions, with some hidden layers in between the input and output layers.

Many options of models are examined. I examine the difference between two different methods of creating the bag of words. I examine the difference between changing all words to be lower case or keeping all caps words. I examine the possibility of using dimensionality reduction to improve processing time. I also examine if it is possible to classify the sentiments based on the number of all caps words and hastags alone.

I have created a Multilayer Perceptron Model that can correctly classify tweets as being positive, negative, or neutral in sentiment with 80.4% accuracy. This model was chosen over other models for its high accuracy, though it has a long processing time. As an alternative to this model, I examine using PCA for dimensionality reduction in order to reduce the processing time by reducing the number of factors. The best of these models using 1000 factors instead of 10,000. It performs with 74.9% accuracy.

2 BENCHMARKING OF OTHER SOLUTIONS

The first model I looked at used MLP Classifier. This model ran with 81% accuracy. While pre-processing the data, they mapped the sentiments to numerical values. They then cleaned the text of the tweets, by removing urls, html tags, digits, and hashtags. They first created a bag of words model. Then they used a neural network called MLP Classifier - Multi-layer Perceptron classifier. It relies on an underlying Neural Network to classify. Another model also used the MLP Classifier. This model also ran with 81% accuracy. They first cleaned the tweets to remove the links, usernames, hashtags, and audio/video tags.

The next model I look at compared different types of neural networks. First, they convert the sentiment labels to numerical labels. Then they embed the data, which turns the positive integers into dense vectors. First, they use Gated recurrent units (GRUs), which are a type of neural network. It performs with 40.75% accuracy. Then, they use bidirectional LSTM, which uses two models. It performs with even less accuracy at 38.9% accuracy. Then they use Bidirectional GRU, which performs with similar accuracy at 39.28%. Then they use convolutional neural networks, which performs with 38% accuracy.

The next model I looked at used naive bayes. First, again, they clean the tweets. First they remove stop words and correct spelling errors. Then, they convert to lowercase and remove mentions and tags. They use the naive bayes classifier which produces an accuracy of 73.4%.

Table 1 shows the accuracy of the different models benchmarked.

All of the models used the same features, with slight differences depending on how they cleaned the text. Some models mapped

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

Model	Accuracy
MLP Classifier	81%
Naive Bayes	73.4%
GRU	40.75%
Bidirectional GRU	39.28%
Bidirectional LSTM	38.9%
Convolutional NN	38%

Table 1: Benchmarking Other Solutions

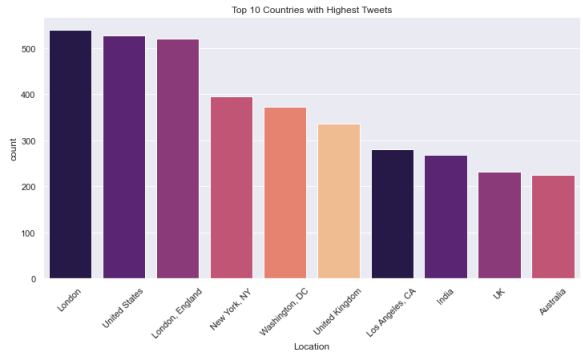

Figure 1: Top 10 Counties with Highest Tweets

the sentiments to numerical values and some didn't. I think this depended on what kind of model they were using. I think the MLP Classifier is the most successful because of the neural network model it uses. However, the Naive Bayes model also performs with similar accuracy. I think I will try both and see which performs better. Something that I would like to take away from all of these models is the way that they pre-processed the text, including removing stop words and fixing spelling mistakes.

3 DATA DESCRIPTION AND INITIAL PROCESSING

The data was provided already split into training and testing sets. the data provided contained the following columns: UserName, ScreenName, Location, TweetAt (the date of the tweet), OriginalTweet (the text of the tweet), and Sentiment (the classification label). Sentiment contained five levels: Extremely Negative, Negative, Neutral, Positive, and Extremely Positive. Figure 2 shows the distribution of these levels. Most tweets are negative or positive, with some neutral tweets an fewer extremely positive and extremely negative tweets. Figure 1 shows the top 10 countries with the highest number of tweets, though the location data will not be used.

For classifying NLP data, only the strings of the tweets and the labels will be used. Figure 3 shows the distribution of the most common words, besides stop words, as well as sorted by sentiment.

As part of the initial processing, the tweets needed to be cleaned in order to create the bag of words to create the classification models. The tweets had urls, html tags, video tags, and audio tags. User names that appeared from re-tweeting or tagging another user were also removed. Digits and double spaces were also removed. The hashtags were kept, but the "#" symbol was removed.

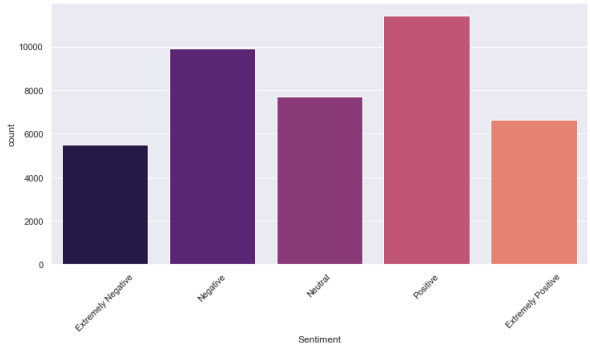

Figure 2: Distribution of Tweet Sentiments

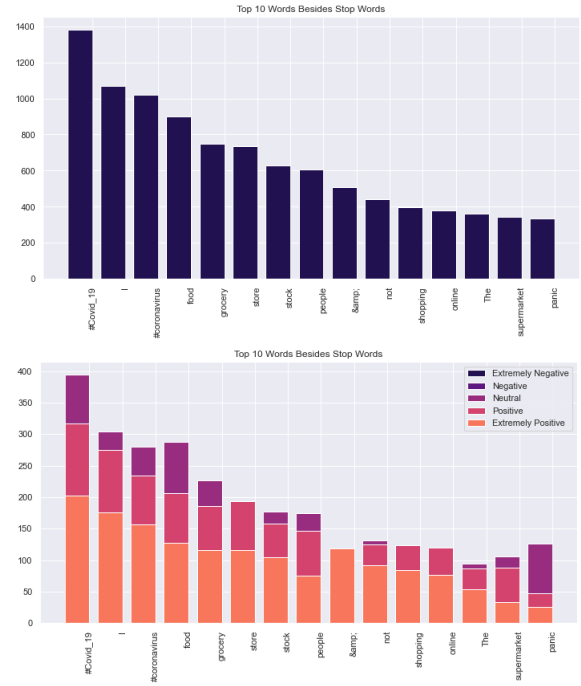

Figure 3: Top 10 Words Besides Stopwords

Next, the words in each tweet were lemmatized. Lemmatizing, as opposed to stemming, examines the root of the word in relation to the grammatical form instead of just choping off the end of the word [1]. For example, if the word "groceries" was stemmed, it would become "groceri". With lemmitization, it becomes "grocery" instead. This lead to better results since the words were more accurate to their root word.

As an example of cleaning the tweets, let's examine an example tweet. "All month there hasn't been crowding in the supermarkets or restaurants, however reducing all the hours and closing the malls means everyone is now using the same entrance and dependent on a single supermarket. #manila #lockdown #covid2019 Philippines https://t.co/HxWs9LAnF9". This tweet became "month crowd supermarkets restaurants however reduce hours close malls

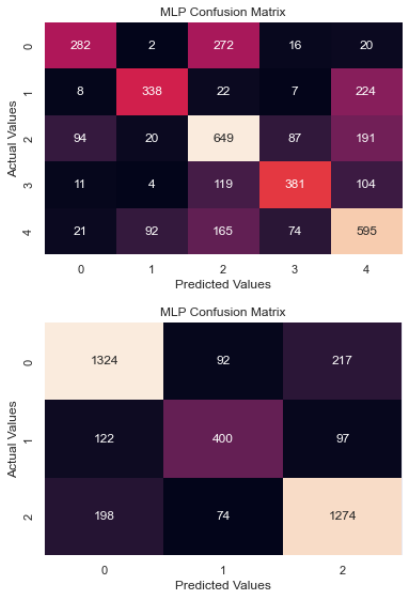

Figure 4: TfidfVectorizer

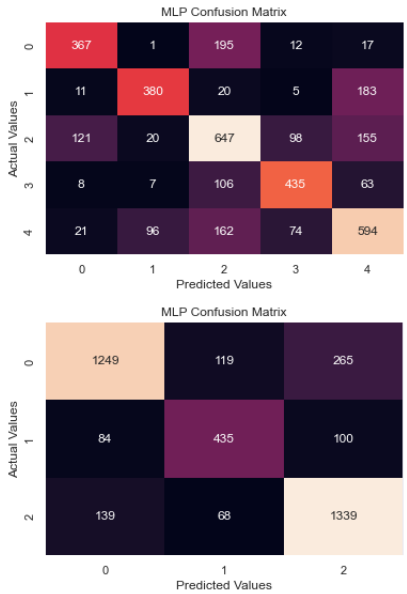

Figure 5: Best Model - using MLP and CountVectorizer with no PCA

mean everyone use entrance dependent single supermarket manila lockdown covid philippines".

4 MODELING

The Multilayer perceptron was the model used to classify the NLP data. This is a feedforward neural network model that uses multiple layers, with each layer fully connected to the following one [2]. The nodes of the layers use nonlinear activation functions, with some hidden layers in between the input and output layers. In the following models, 3 layers were used. The default neurons per layer of 100 was used. It used the Adam solver for weight optimization with relu activation for the hidden layer.

The bag of words model needs to be created for any NLP problem. Initially, I tried TDfidfVectorizer to create the bag of words. However, this created a model with very long run time, greater than 250 seconds, and slightly reduced accuracy. The confusion matrices for this model can be seen in Figure 4. As such, I used CountsVectorizer instead. Where TDfidfVectorizer uses the weights of the number of occurrences, CountsVectorizer is a simple version that only includes the number of occurrences. Once CountsVectorizer was utilized, the matrix was made dense and could be used to develop the model. Using the max features setting, 10,000 features were found.

I found that using 3 layers returned the best results out of what I tried. As such, many of the default settings for sklearn's MLP [3] function were utilized. I utilized early stopping to keep the model not too computationally taxing. This checked if the increase in the validation score for each epoch was significant. The original model that used all five levels performed with 63.3% accuracy. Using only 3 labels performed with 80.4% accuracy. Figure 5 shows the confusion matrices.

I was curious to see if keeping all caps words as separate from the lowercase version of the same word would increase the accuracy, as all caps words tend to have a different meaning and tone than lower case words. As such, when cleaning the strings I converted all of the words to lower case expect words that were all caps. Then, a new bag of words was created and an MLP model was created. This model performed with roughly the same accuracy and the same F1 Score as the original model, so it is hard to say if this made a difference or not. Figure 6 shows the confusion matrices for this model.

Creating these models was very computationally taxing, as one would imagine for 10,000 factors. Next I will examine using PCA to reduce the dimensionality of the bag of words matrix in order to improve processing time while maintaining accuracy. Using the explained variance ratio to find the number of features to have 20% explained variance, I first examined using 22 features. While the actual PCA process took quite a while, I was able to reduce the MLP processing time to less than 7 seconds for 5 levels and less than 20 seconds for 3 levels. It did, however, greatly reduce the accuracy.Figure 7 shows the confusion matrices. Figure Figure 8 shows the confusion matrices Next, I tried 100 components. This was actually faster than 22 principal components. It increased the accuracy slightly, but not by a significant amount. Last, I tried 1000 components. This took a little longer but performed with accuracy more similar to the model without dimensional reduction. I believe that this model would be worthwhile as it has similar accuracy to the original model with greatly reduced processing time. Figure 9 shows the confusion matrices.

Out of curiosity, I decided to examine just the instances of words that are hashtags and number of all caps words. I was interested to see if it would be at all possible to classify the sentiments of the tweets based on these factors alone. They both performed poorly,

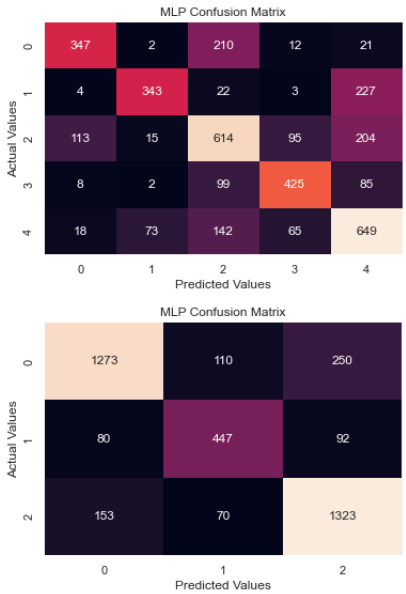

Figure 6: Including all caps words

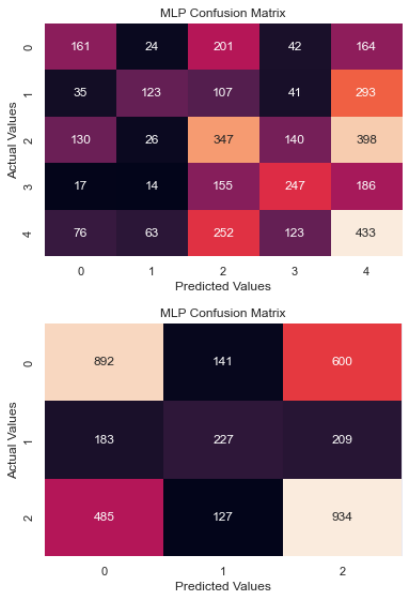

Figure 7: PCA 22 Components

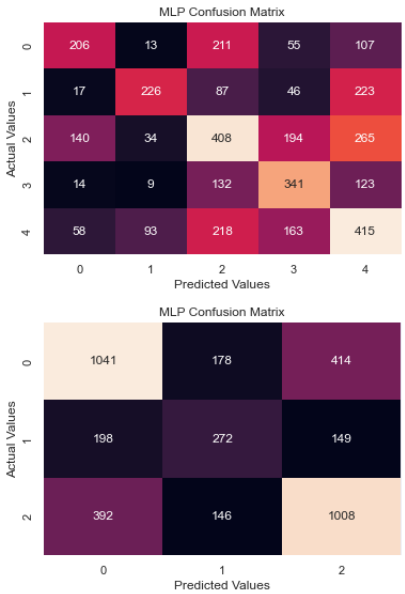

Figure 8: PCA 100 Components

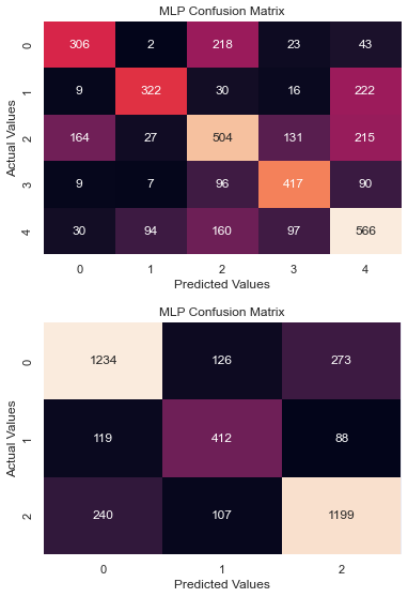

Figure 9: PCA 1000 Components

but it was interesting to see the results. They did performed better than simply guessing, which was a fun surprise. Interestingly, the model classified almost every data point as being positive in sentiment for both cases. Figure 10 and Figure 11.

Table 2 shows the processing time, accuracy, and F1 score errors for each model, as well as comparing between the 5 label version and the 3 label version. The original model clearly outperforms all the other models at 80.4% accuracy, but is relatively computationally taxing. For a faster alternative, the 1000 factor PCA model performs at 74.9% accuracy in only 14.3 seconds. I would choose the PCA model in most circumstances, unless I had access to a stronger computer. Otherwise, especially with larger datasets, the orginal model, while high performing, takes too long to be useful.

REFERENCES

[1] Prabhakar Raghavan Christopher D. Manning and Hinrich Schütze. 2008. Stemming and lemmatization. Retrieved Apr 26, 2022 from https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html#:~:text=Lemmatization%20usually%20refers%20to%

	5 labels			3 labels		
Model	Time (seconds)	Accuracy	F1 Score	Time (seconds)	Accuracy	F1 Score
Original (counts vectorizer)	76.99	0.633	0.63	77.42	0.804	0.80
TDfidf Vectorizer	262.12	0.597	0.60	262.19	0.787	0.79
Keeping all caps	83.71	0.629	0.63	92.47	0.802	0.80
Number of Caps	1.809	0.247	0.11	1.332	0.409	0.24
Hashtags only	72.92	0.255	0.14	90.48	0.413	0.35
22 Principal Components	6.677	0.345	0.34	17.17	0.541	0.54
100 Principal Components	3.859	0.420	0.42	6.444	0.611	0.61
1000 Principal Components	13.60	0.557	0.56	14.30	0.749	0.75

Table 2: Processing Time and Acccuracy of Models

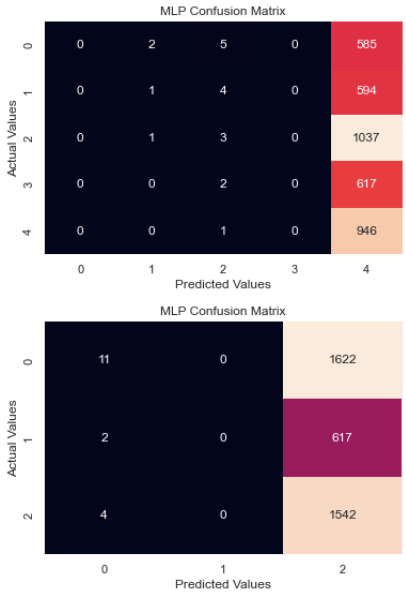

Figure 11: Number of All Caps words

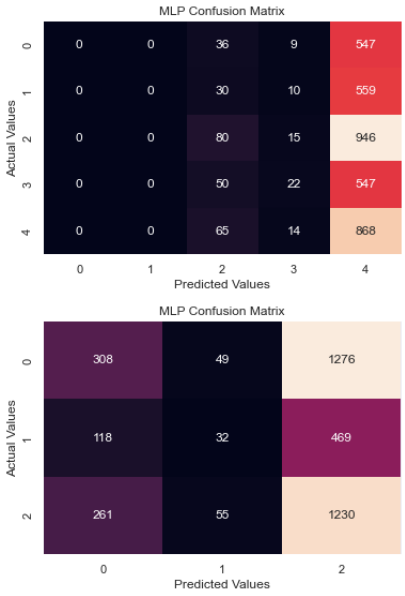

Figure 10: Hashtags

20doing,is%20known%20as%20the%20lemma%20

[2] Michael Fuchs. 2021. NN - Multi-layer Perceptron Classifier (MLPClassifier). Retrieved Apr 26, 2022 from https://michael-fuchs-python.netlify.app/2021/02/03/nn-multi-layer-perceptron-classifier-mlpclassifier/#mlpclassifier-for-binary-classification

[3] scikit learn developers. 2022. sklearn.neural$_network.MLPClassifier.RetrievedApr26, 2022from$