Sentiment Analysis of Coronavirus Tweets

Gerrid La Sala
Faculty of Science University of Western Ontario
London, Canada
glasala@uwo.ca

Zak Hodgson
Faculty of Engineering
University of Western Ontario
London, Canada
zhodgson@uwo.ca

Sean Friedrich
Faculty of Science University of Western Ontario
London, Canada
sfriedri@uwo.ca

Priyank Patel
Faculty of Engineering
University of Western Ontario
London, Canada
ppate244@uwo.ca

Abstract-Social media has become a routine part of many lives. It allows people to share ideas, thoughts, and emotions on an unprecedented scale. This has led to social media, such as twitter, becoming an accurate representation of the population's views on any given topic. However, the sheer number of opinions on social media have made it hard to aggregate and analyze in the past. By using machine learning models, we can more effortlessly predict the emotions of a tweet., allowing for a better understanding of the public's current opinions on a given topic or keyword. This paper analyzes the accuracy of machine learning models on a dataset of COVID-19 related tweets to predict the author's emotions and allow governing bodies to better understand public opinion. The models tested included simple Naive Bayes and Linear SVM classifiers to act as a baseline and low-computation estimations, and an LSTM classifier. The Linear SVM model yielded the best results of the linear/Naive Bayes classifiers, with a 54% weighted accuracy. The more complex LSTM model achieved an accuracy of 64%.

Index Terms—Tweet, COVID19, N Gram, TF-IDF, Naive Bayes, SVM, LSTM

I. Introduction

The increasing use of social media has opened up a new avenue for observing people's behaviour. During times of crisis, the volume of social media posts tends to increase exponentially, vastly increasing the amount of information being spread. Twitter has been the one of the most popular social media sites, with over 187 million daily active users. During times of crisis, important information is often being posted. The problem is that computers can have a difficult time separating crisis related tweets from those that are not crisis related.

One way that computers can better understand human language is through natural language processing (NLP). NLP can be used to process large amounts of text data, allowing computers to do things like interpret meaning and measure sentiment. This technique is useful to organizations for crisis events as it allows them to determine how a crisis is affecting public sentiment. The COVID-19 pandemic has created an unprecedented state throughout the world. People have had to adapt to a whole new way of life. This drastic change means people have inevitably formed opinions on COVID-19, but these sentiments can be hard to track. The purpose of the project is to create a model through machine learning that can classify the sentiment of COVID-19 related tweets.

II. Related Works

The value of analyzing tweets during times of crisis is becoming increasingly known, which can be seen by the number of research studies being done on the subject. One study using twitter sought to group tweets based on what type of natural disaster they related to [2]. The study was performed in Ecuador with the purpose of increasing the awareness for relief agencies of harmful events. They used NLP to analyze tweets in real-time for keywords to find and classify tweets that may be mentioning natural disasters, and alerting the proper authorities of this information.

NLP is needed to solve problems like these because of the noise and variability of data obtained from social media. Another group used NLP to identify hurricane related tweets and classify user evacuation behaviour [3]. Using machine learning to process social media is becoming increasingly popular and there are numerous studies exploring this idea. Another similar classification problem was performed through NLP to identify misogynistic language in tweets [4]. In the paper they used machine learning models to detect and classify misogynistic language.

Since the COVID-19 pandemic has been declared, there has been an increasing number of studies done collecting twitter data with the purpose of understanding how the public has reacted to the changes the pandemic has brought. One such study analyzed 4 million tweets related to COVID-19 by hashtag [5]. The researchers used a latent Dirichlet allocation, which is a NLP modelling method. They used this model to classify popular unigrams, bigrams, and sentiments in the tweets they collected. This differs from our study because of the type of model used to classify the data. It also differs because we classify the unigrams and bigrams by sentiment, which they did not do. The overall theme of the study was similar, but they go different directions with regard to how the data is used and classified.

Another COVID-19 related twitter study was performed using NLP and machine learning frameworks to detect tweets containing COVID-19 symptoms that users were self-reporting [6]. They trained a BERT-based classifier. They grouped tweets into negative and positive classes. Tweets classified as positive mean the model thinks the tweet contains self-reported symptoms of COVID-19. They had good results as well, with an F1 of 0.96 for negative (non-COVID-19) tweets, and 0.71 for tweets that potentially have COVID-19 symptoms.

III. DATA

The problem that this project looked to solve is how to identify the author's emotions to tweets relative to the CoronaVirus pandemic. The data set was accessed from Kaggle, where it was presented as a challenge. The data was created by Aman Miglani [1] . The dataset is composed of approximately 45000 tweets with 6 variables. There are 3798 observations within the test set and the remaining in the training set. The response variable of the dataset is called “Sentiment”, which is a quinary response composed of (Extremely Negative, Negative, Neutral , Positive, Extremely Positive). The variable dictionary is displayed in Table 1.

Table 1
Data Dictionary

Variable	**Variable Explanation**
Tweets	holds the text of the tweet
Sentiment	denotes the sentiment of the tweet ranging from Extremely Negative to Extremely Positive
Location	identifies the location of the tweet *future work*
Date	when the tweet was posted by the author *future work*
User Name	identifies user
Screen Name	identifies screen name

IV. EXPLORATORY DATA ANALYSIS

Exploratory data analysis is a critical step where initial investigations on the original dataset are performed. This is done to discover patterns and prepare for the correct direction of the incoming data cleaning process.

A. Target

In the classification task, imbalanced data typically refers to a problem where the classes are unequally distributed. This can result in a variety of problems, for instance making the model more inclined to classify the data into majority categories during the training process. This could significantly reduce the actual generalization ability of the model. From Figure 1, it can be shown that all 5 labels are well represented in the dataset. Overall, negative tweets comprise 37.4% of the dataset, positives comprise 43.9%, and neutral tweets make up the remaining 18.7%.

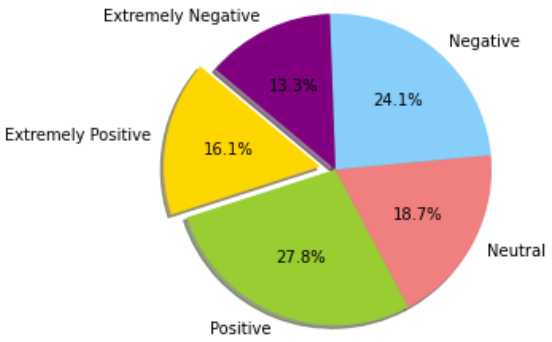


Fig 1. Target Distribution on Training Data.

B. Most Common Words

Figure 2 below shows the 10 most common words in the dataset and how often they occur. This serves as a baseline for our n-gram analysis as we look to determine the word to sentiment relationship.

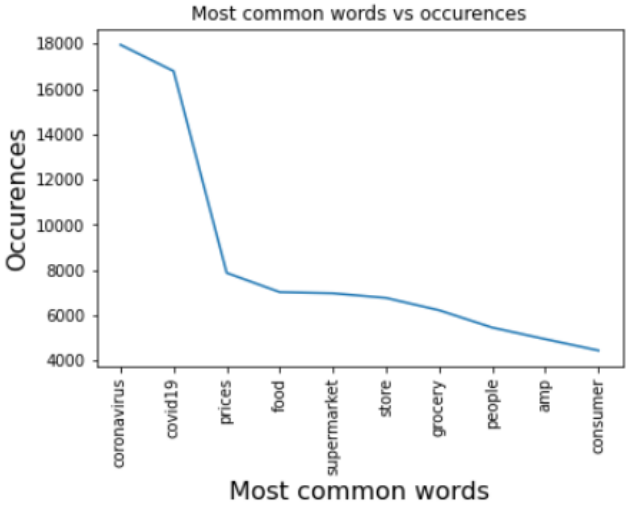


Fig.2 Most Common Words

V. DATA PREPROCESSING

Before the models can be tested, the data needs to be preprocessed and converted into a consistent and easily digestible format to ensure the text variation can be reduced. Some of the techniques performed in this step include:

A. Punctuation Removal

The first technique used in data preprocessing was to remove all punctuation from the tweets. This helps convert the data to a more consistent format, which will increase the models’ abilities to correctly categorize tweets. Figure 3 shows a string including all the characters which were removed from the tweets during this phase.

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

Fig 3. Characters removed from tweets.

B. Stopwords Removal

"Stop words" is a term which is commonly used to refer to the most common list of words in a language. Most commonly, they convey little to no emotional value. Therefore, removing them from the tweet dataset is crucial for ensuring our data is digestible and compressed.

C. Lowercase Conversion

To maximize the consistency of our data, all words were converted to lower-case. Additionally, this ensures that capitalized words are not treated differently than lower-case versions of the same word. This will increase the accuracy of our models' analysis.

D. Spelling Correction

As tweets often contain a variety of typos and informal phrases, it is necessary to reduce the effects. This will allow the model to gain more understanding from what would appear as jargon before. As a result, a spellchecker was used on our dataset to eliminate a large majority of misspelled words.

E. Length Analysis

To increase the consistency of our data, a length analysis was performed on the data. Figure 4 below shows the results of this analysis. The study has shown that most tweets are between 20 and 30 words in length. As a result of our findings, tweets are later padded with zeros to a minimum length of 30. This helps maintain our data's consistency and digestibility. In addition, tweets longer than 30 words are removed to reduce error as sample sizes were too low for an effectively trained model.

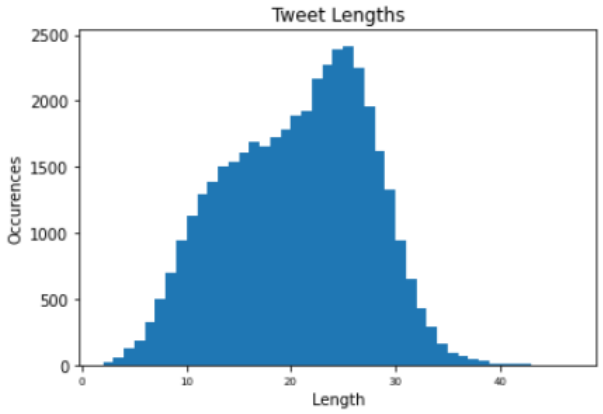
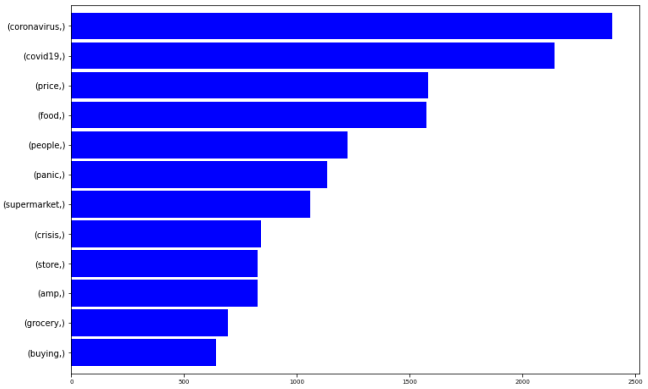


Figure 4: Length Analysis of Tweets

VI. N-GRAM ANALYSIS

An N-gram analysis is defined as a contiguous sequence of *n* items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. In our case we performed both a One-gram and Bigram analysis over a given set of words for each of the five sentiment possibilities. Figures 5 and 6 present the distribution results of One-gram and Bigram of the tweets after preprocessing the data. With that you can identify that within a One-gram analysis of the data there are many similar words but those that vary are key words such as "panic" and "help" . Within the Bigram analysis there is a greater variation as words like "panic", "buying", and "price" are more commonly found in Negative tweets. Whereas "stay safe" and "social distancing" are more likely to be used in positive tweets regarding the Coronavirus. This is very useful in constructing and training the model to distinguish between them. The remaining models for each sentiment can be located in the appendices.

Top 12 One-grams in Extremely Negative Tweets



Top 12 One-grams in Extremely Positive Tweets

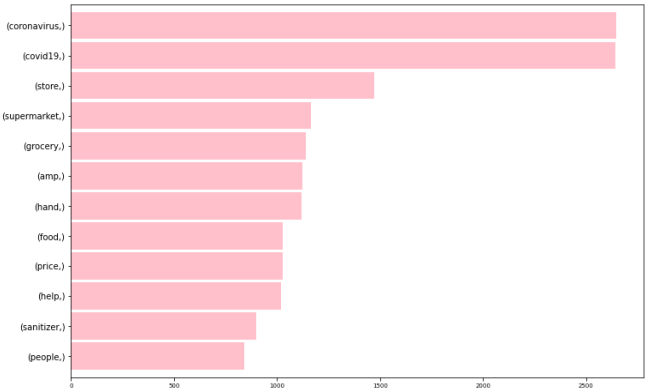
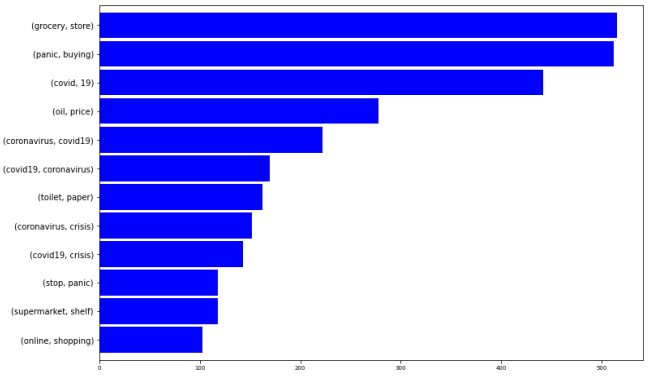


Fig.5 One Gram on Training Data (Extremely Negative/Positive)

Top 12 Bigrams in Extremely Negative Tweets



Top 12 Bigrams in Extremely Positive Tweets

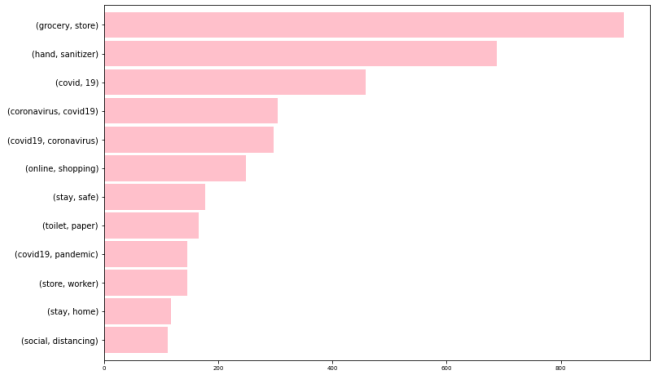


Fig.6 Bigram on Training Data (Extremely Negative/Positive)

VII. METHODS

The goal of the model is to classify a tweet as an emotional output. Given the large amount of tweets the model would have to go through in a practical scenario (both in training and as an implemented application), simple models were first trained to attain a baseline for the predictions. These models included Naive Bayes and Linear SGD variants. A Long Short Term Memory (LSTM) network was then implemented as a more complex model to attain better accuracy.

The linear and Naive Bayes models both accept alpha values to control the laplace smoothing. As the models are designed to take in tweets, there will constantly be new data and words that did not appear in the training dataset. For example, “Coronavirus” was rarely used before 2019, and “Pfizer” until recently. The lack of training on these unfamiliar words can lead to issues with zero probability. To mitigate this, the classifiers were cross-validated using different alpha Laplace smoothing constants. The constants ranged from 2 to 0, with 0 omitting all smoothing.

The models also all accepted tokenized strings for inputs. Tokenizing the tweets preprocesses, tokenizes, and filters the input. The implemented tokenizer determines the frequency of all the top 1000 words and removes all those outside of the dictionary. The main benefit of tokenizing is that by breaking the text into small, known fragments the model is more easily able to extract meaning from the text [11] . This process outputs a dictionary of features that can be trained on. The index of the tokens in the dictionary are directly linked to their frequency in training, which can then be used to identify the significance of a particular token in a tweet.

A. *Naive Bayes with Tokenizing*

Tokenizing produces a “bag of words” to represent the tweet alongside the occurrence frequency of each word. These frequencies are then used to classify tweets based on the words within a particular tweet and their frequency. A multinomial Naive Bayes was used as opposed to a regular Naive Bayes. Naive Bayes typically assume that data follows a normal (Gaussian) distribution. In natural language, the frequencies and features will follow a multinomial distribution; therefore multinomial Naive Bayes will perform better for this model which processes English tweets[8].

B. *Naive Bayes and TF-IDF*

Tokenizing allows for a model to make predictions based on term occurrence, however it fails to account for the frequency of insignificant words dominating the training algorithm. These would normally be stripped away with perfect tokenization, removing any words that would not affect the outcome. In this model, inputs can vary greatly and hard-coded stop words cannot fully account for the insignificant yet frequently occurring tokens. This issue is fixed by downscaling the weights of certain terms based on their frequency in the document (tweet list), otherwise known as Term Frequency times Inverse Document Frequency (TF-IDF).

C. *Linear SVM and TF-IDF*

Linear support vector machines (SVM) are regarded as one of the best text classifications algorithms[9]. This is largely attributed to text data being linearly separable; that is the value of a text (the word itself) has a significant impact on its classification. Slightly different words by value (such as hate and hats) are strongly linearly separable in their classifications. On top of this, there are large amounts of text data in the document for training, as well as many features. Linear kernels can handle this scaling of features and data efficiently, as opposed to models mapped to a higher dimensionality which do not tend to improve performance and significantly hurt computation time with model complexity [10].

D. *Bidirectional LSTM with Tokenizing*

A Bidirectional LSTM is an extension of a traditional LSTM model that can improve performance for sequence classification problems such as this. A Bidirectional recurrent neural network (“RNN”) duplicates its first recurrent layer in order to prove the input sequence as is for the first layer and a reversed copy into the second. [12]

Long Short Term Memory networks more commonly referred to as LSTM is a special RNN which can learn long-term dependencies [13]. As opposed to a general neural network an RNN has the ability to take in and process sequence data while also dealing with sequence change. An example of dealing with sequence change would be how a word will have different meanings relative to its context which is where an RNN excels. Though if the sequence is too long RNN has problems dealing with long-term dependencies as the gradient explosion and gradient disappearance could occur during the training. This problem is solved by LSTM adding three gate units in the hidden layer, including ”forget gate”, input gate’ and ’output gate’ As demonstrated in the figure below.

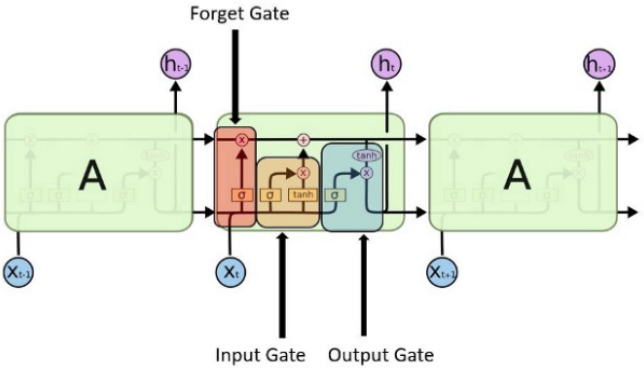


Fig.7 LSTM Structure

An LSTM network follows the same form of chain repeating as the model above. The transmission is controlled through its internal gates removing irrelevant information and remembering the long-term memory. The LSTM model constructed for this analysis with the tweet text after embedding the input layer. The structure of the network model for the project is shown in Figure 8.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 30, 16)            160000
_________________________________________________________________
bidirectional (Bidirectional (None, 30, 40)            5920
_________________________________________________________________
bidirectional_1 (Bidirection (None, 40)                9760
_________________________________________________________________
dropout (Dropout)            (None, 40)                0
_________________________________________________________________
dense (Dense)                (None, 5)                 205
=================================================================
Total params: 175,885
Trainable params: 175,885
Non-trainable params: 0
```

Figure 8. LSTM Model

The parameters used in the model are implemented as follows:

- The number of words per tweet within the training set is from 2 to 30. Each tweet is converted to a sequence of size 30. The embedding layer receives the tweet sequences and transforms it to a matrix 30 x 16 where 16 is the number of space dimensions embedding a single word.
- A bidirectional LSTM is implemented over two layers of the model. The first bidirectional layer trains on the as-is input sequence with 20 hidden memory units with a return sequence. Where the second runs on a reversed copy also over 20 hidden layers. This aims to reduce parameters to avoid over fitting.
- The dropout layer is run at a coefficient of 0.2 thereby excluding one in five inputs from each update cycle.
- A sigmoid function is used as an activation function for the output layer.
- The learning rate is initialized with Keras Adam optimizer which is defaulted to 0.001 along with a beta1 of 0.9, beta2 of 0.999 , epsilon of 1e-08 and decay=0.0.

The model was trained over an epoch of 50. A patience of three was implemented over the fitting process. If the validation accuracy had not improved over 3 rounds the process was terminated.

Figures 9 and 10 show the accuracy and loss in respect to epoch on both training and validation sets.

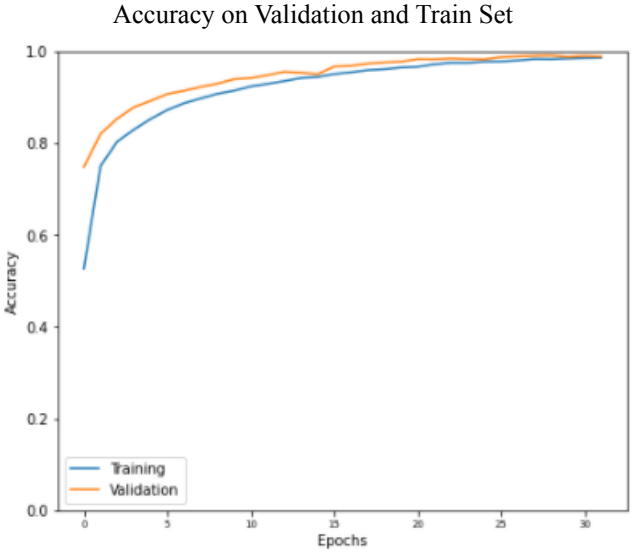


Figure 9. Accuracy with Epoch on Validation and Training Set.

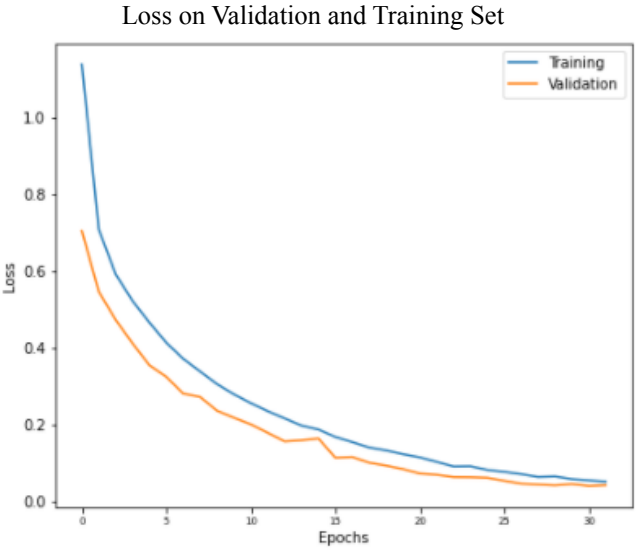


Figure 10. Loss with Epoch on Validation and Training Set

VIII. EXPERIMENTAL RESULT

Each model was trained on over 40000 relevant tweets, cross-validated with 5 folds, and then tested on 4000 tweets.

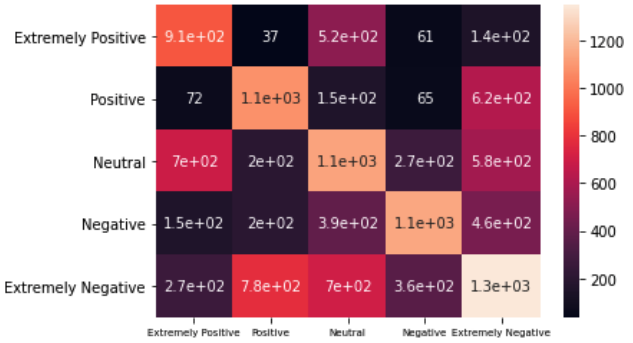
A. *Naive Bayes with Tokenizing*

Figure 11 shows the precision, recall, F1 score, and accuracy on the test dataset using the Naive Bayes model with just tokenization. The F1 score was 0.45, significantly higher than a model randomly guessing the classification (~0.2 accuracy). The confusion matrix after training indicates that many tweets are misclassified, with seemingly random relationships. An alpha smoothing value of 0.1 was chosen with cross-validation. The model achieved its goal of consuming low computational resources while predicting classifications to a higher degree of accuracy than blind

guessing. This model also acts as a baseline for other models, given its simplicity and accuracy.

Figure 11. Results of Naive Bayes Classifier

	precision	recall	f1-score	support
Extremely Negative	0.43	0.55	0.48	1671
Extremely Positive	0.47	0.54	0.50	1994
Negative	0.39	0.39	0.39	2878
Neutral	0.60	0.49	0.54	2344
Positive	0.43	0.39	0.41	3461
accuracy			0.45	12348
macro avg	0.46	0.47	0.46	12348
weighted avg	0.46	0.45	0.45	12348



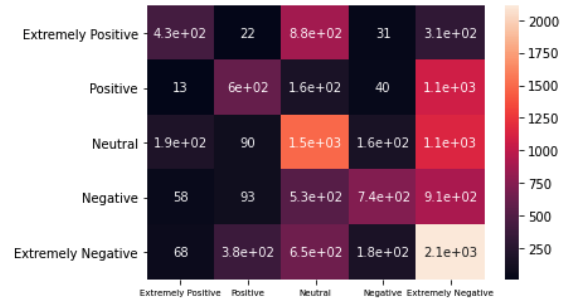
B. *Naive Bayes and TF-IDF*

To improve the Naive Bayes classifier, TF-IDF was also performed on the inputs. This was expected to reduce the impact frequently appearing words would have on the classification. As seen in figure 12, the recall has dropped significantly, indicating that the amount of correctly identified positives has dropped. However, the F1 score remains almost constant and the confusion matrix indicates that mislabeled classes have some form of predictability (many tweets are incorrectly labelled as 'Extremely Negative'). This consistency makes the model arguably better than the previous Naive Bayes, but by a small margin if at all. An alpha value of 0.1 was chosen from cross-validation.

Figure 12. Results of Naive Bayes Classifier with TF-IDF

	precision	recall	f1-score	support
Extremely Negative	0.56	0.26	0.35	1668
Extremely Positive	0.51	0.32	0.39	1899
Negative	0.40	0.48	0.44	3075
Neutral	0.64	0.32	0.42	2327
Positive	0.38	0.62	0.47	3379
accuracy			0.43	12348
macro avg	0.50	0.40	0.42	12348
weighted avg	0.48	0.43	0.43	12348

clf__alpha: 0.1



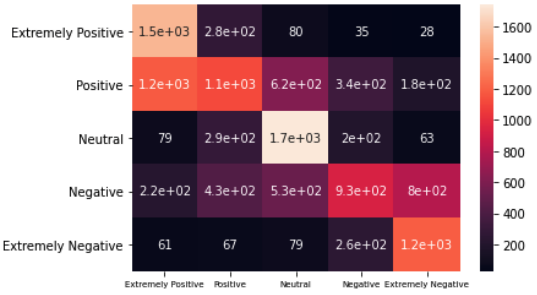
C. *Linear SVM and TF-IDF*

The Linear SVM was implemented to improve on the Naive Bayes model without the time complexity of more advanced models such as LSTM. The model had a significantly better recall and F1 score, indicating that it performed as expected. The higher recall reflected that most of the positive classes were correctly identified, which is again supported by the confusion matrix indicating that less classes were mislabeled. The alpha value chosen from cross-validation was 0.01. The F1 accuracy of 0.53 is ~10% higher than the Naive Bayes counterparts.

Figure 13. Results of Linear SVM Classifier with TF-IDF

	precision	recall	f1-score	support
Extremely Negative	0.53	0.72	0.61	1655
Extremely Positive	0.50	0.78	0.61	1964
Negative	0.53	0.32	0.40	2905
Neutral	0.57	0.73	0.64	2374
Positive	0.51	0.33	0.40	3450
accuracy			0.53	12348
macro avg	0.53	0.58	0.53	12348
weighted avg	0.53	0.53	0.51	12348

clf__alpha: 0.001



D. *LSTM with Tokenizing*

Figure 14 shows the precision, recall and F1 score. Figure 15 shows the confusion matrix.

- The weighted average precision increased from 53% to 64%
- The weighted average recall increased from 53% to 64%
- The weight average F1 score increased from 51% to 64%

Compared with the baseline model the performance of the LSTM model with Tokenized embeddings increases across all metrics.

Figure 14. Precision, Recall, F1-Score of LSTM Model on Test Set.

	precision	recall	f1-score	support
0	0.62	0.61	0.62	947
1	0.67	0.64	0.65	592
2	0.70	0.67	0.68	619
3	0.73	0.65	0.69	599
4	0.56	0.64	0.60	1041
accuracy			0.64	3798
macro avg	0.66	0.64	0.65	3798
weighted avg	0.64	0.64	0.64	3798

Figure 15. Confusion Matrix of LSTM Model on Test Set

IX. CONCLUSIONS

This project aimed to analyze various learned models and their performance on classifying the emotions of tweets related to the Coronavirus. The computationally inexpensive models implemented such as Naive Bayes and Linear SVM achieved F1 accuracies of 40-55%. A more computationally expensive LSTM network was trained and achieved a 64% F1 accuracy. The LSTM model is an effective model for assessing the emotions of a Coronavirus tweet and can help organizations, such as governing bodies, better understand public perception of a current event or topic.

To expand on this research, models could be trained to achieve a balance between the computation time of Linear models and the accuracy of the LTSM. With performance suitable for practical applications, machine learned models have significant potential to help classify public opinion and emotions on not only Coronavirus, but any key topic.

X. APPENDICES

Figure 16.Onegram Analysis of Extremely Negative Tweets

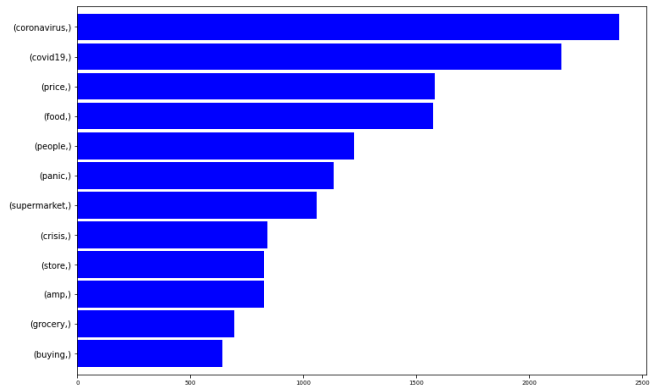


Figure 17.Onegram Analysis of Negative Tweets

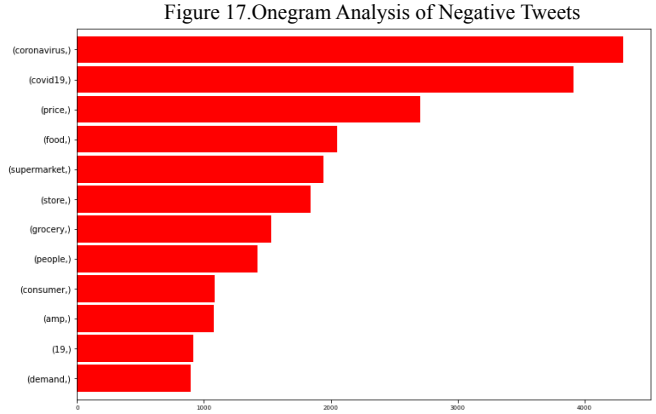


Figure 18.Onegram Analysis of Neutral Tweets

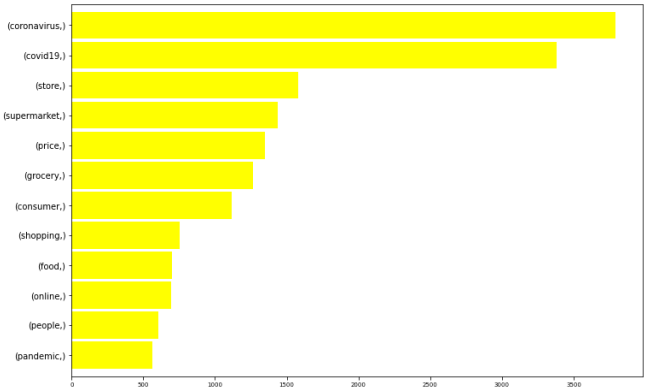


Figure 19.Onegram Analysis of Positive Tweets

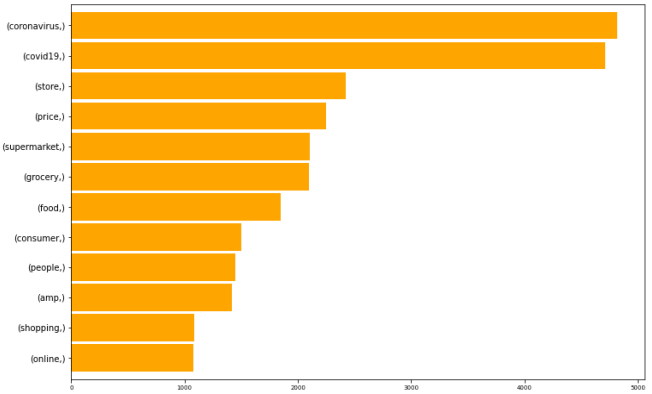


Figure 20.Onegram Analysis of Extremely Positive Tweets

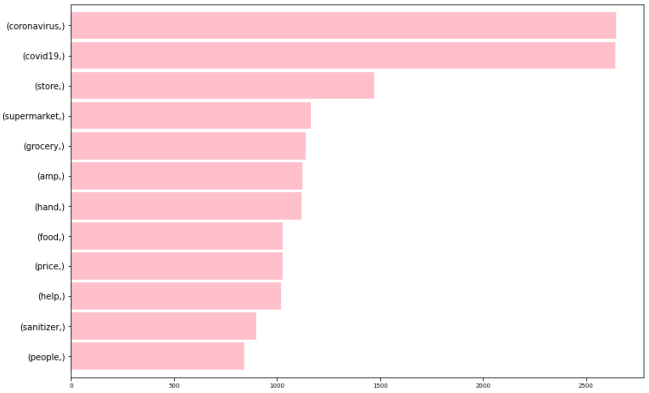


Figure 21.Bigram Analysis of Extremely Negative Tweets

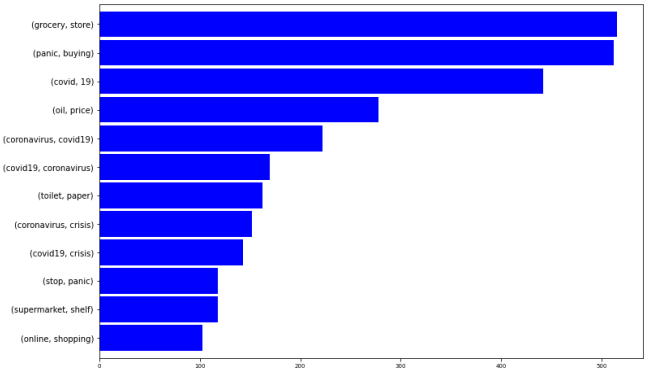


Figure 22.Bigram Analysis of Negative Tweets

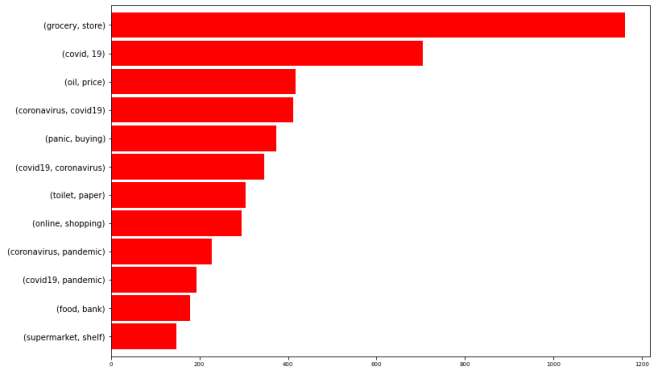


Figure 23.Bigram Analysis of Neutral Tweets

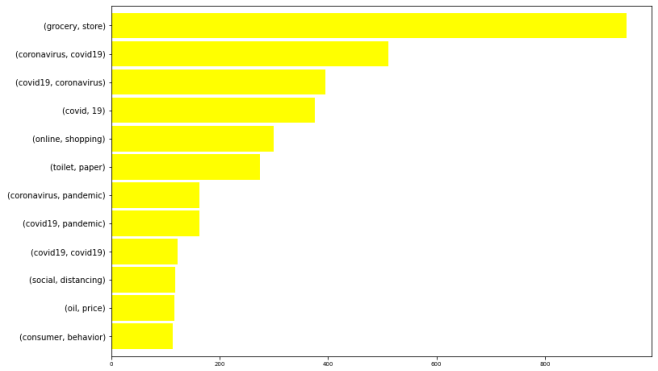


Figure 24.Bigram Analysis of Positive Tweets

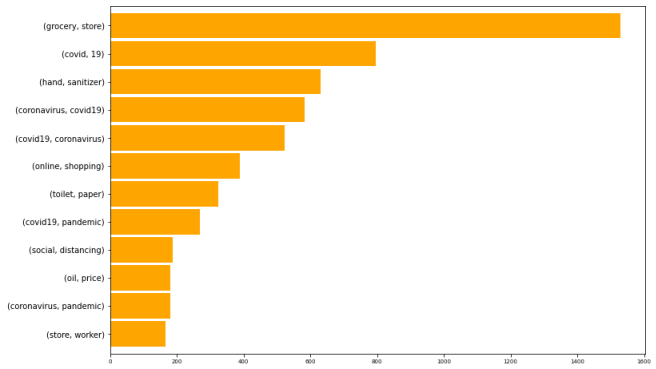
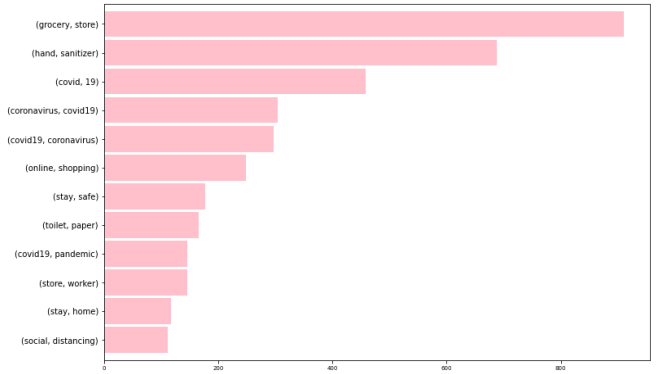


Figure 24.Bigram Analysis of Extremely Positive Tweets



REFERENCES

[1] Miglani, Aman. (n.d.) Retrieved April 1, 2021 from https://www.kaggle.com/datatattle/covid-19-nlp-text-classification

[2] Maldonado, Miguel & Alulema, Darwin & Morocho, Derlin & Proano, Marida. (2016). System for monitoring natural disasters using natural language processing in the social network Twitter. 1-6. 10.1109/CCST.2016.7815686

[3] K. Stowe, J. Anderson, M. Palmer, L. Palen and K. Anderson, "Improving classification of twitter behavior during hurricane events", Proc. 6th Int. Workshop Natural Lang. Process. Social Media. Assoc. Comput., pp. 67-75, 2018, [online] Available: http://www.aclweb.org/anthology/W18-3512.

[4] Anzovino M., Fersini E., Rosso P. (2018) Automatic Identification and Classification of Misogynistic Language on Twitter. In: Silberztein M., Atigui F., Kornyshova E., Métais E., Meziane F. (eds) Natural Language Processing and Information Systems. NLDB 2018. Lecture Notes in Computer Science, vol 10859. Springer, Cham. https://doi.org/10.1007/978-3-319-91947-8_6

[5] Xue J, Chen J, Hu R, Chen C, Zheng C, Su Y, Zhu T Twitter Discussions and Emotions About the COVID-19 Pandemic: Machine Learning Approach J Med Internet Res 2020;22(11):e20550 URL: https://www.jmir.org/2020/11/e20550 DOI: 10.2196/20550

[6] Al-Garadi, Mohammed Ali, et al. "A Text Classification Approach for the Automatic Detection of Twitter Posts Containing Self-reported COVID-19 Symptoms." (2020).

[7] Jayaswal, V. (n.d.). Retrieved April 05, 2021, from https://towardsdatascience.com/laplace-smoothing-in-naïve-bayes-algorithm-9c237a8bdece

[8] Multinomial distributions over words. (n.d.). Retrieved April 07, 2021, from https://nlp.stanford.edu/IR-book/html/htmledition/multinomial-distributions-over-words-1.html

[9] Joachims, T. (1998, April). Text categorization with support vector machines: Learning with many relevant features. In European conference on machine learning (pp. 137-142). Springer, Berlin, Heidelberg.

[10] Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification.

[11] Perry, Tal. (n.d.). Retrieved April 11, 2021 from What is Tokenization in Natural Language Processing (NLP)? - ML+ (machinelearningplus.com)

[12] Brownlee, J. (2021, January 17). How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras. Retrieved April 09, 2021, from https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/

[13] Mittal, A. (2019, October 12). Understanding RNN and LSTM. Retrieved April 08, 2021, from https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e