Sentiment Analysis of COVID-19 Tweets using Hybrid CNN-LSTM Implementation

Tan Yu Hoe
Diploma in Applied Artificial Intelligence and Analytics, School of Computing
Singapore Polytechnic

This technical paper aims to conduct a Sentiment Analysis of COVID-19 Tweets data, leveraging on the use of natural language processing. The implementation consists of a Hybrid CNN-LSTM architecture to extract and identify the affective states of the tweet. This implementation is done using Python programming and Keras API.

Keywords—Sentiment Analysis, COVID-19, Twitter, Natural Language Processing, Deep Learning, Convolutional Neural Network, Long-Short Term Memory, Keras

I. Introduction

The COVID-19 pandemic has impacted everyone's lives, and it has become such a big topic in an everyday context. With the pandemic is still ongoing and there is no clear sign of ending, there is a substantial increase of tweets with COVID-19 context, such that the data can be commonly found on the internet. Data in the present day comes in many forms, such examples are text, video, images, and sound. In this technical paper, I will implement a Bidirectional LSTM architecture to perform sentiment analysis on COVID-19 tweet data. Simple sentiment analysis would be extracting a piece of text and classifying the polarity of the given text at the document, paragraph, sentence, or feature level. [1] The dataset was sourced from Kaggle, with its main features being the tweet and the label – the general sentiment of the text. My main motivation for this paper will be to be a predictive neural network architecture that takes in a piece of tweet with COVID-19 context and predict with its respective sentiment.

II. Related works

I focused on looking for related works that had the following property:

1. Origination of sentiment analysis and its motivation
2. Recent works of sentiment analysis performed on tweets/corpus with COVID-19 context
3. Similar works with usage of hybrid CNN-LSTM architectures

For property 1, sentiment analysis is not a fairly new topic within the 21st century. It could be traced down to one of the earlier papers in 2002, where Bo Bang & Colleague discussed using machine learning methods to determine whether a movie review is positive or negative. [2] He mentioned the difficulty of using machines to properly determine the sentiments behind the movie reviews.

For property 2, I found a recent article written by Xiexie Zhou, who used different models and different embedding methods to achieve high accuracy in predicting the sentiments of tweets with COVID-19 related context. [3]

For property 3, I found an article worth reviewing, on fake news detection using a hybrid CNN-RNN approach written by Jamal Abdul Nasir. The author proposed an approach of using a hybrid model, making use of the CNN to extract local features and of the LSTM to learn long-term dependencies. Even though the predictive task is different, their objective and mine both falls under the topic of text classification. [4]

III. Dataset

The dataset was sourced from Kaggle, a subsidiary platform of Google for the data science community, originating from Aman Miglani. [5] The dataset comes as a train and test set, both having the same columns. There are 41,157 records from the train set and 3798 records from the test set, with each record containing attributes of a single tweet. For each tweet, we are provided with the username, screen name, location, time of the tweet, the original tweet, and the label. As stated by the author, the names and usernames have been encoded to avoid any privacy concerns.

Table 1: Attributes of the Data

Attribute	*Description*
UserName	Encoded Twitter Username
Location	Origination of the tweet
TweetAt	Date of the tweet
OriginalText	Tweet as a piece of text
Sentiment	General sentiment of the tweet represented as a polarity class

For data collection, the author of the dataset mentioned that he pulled the data from Twitter and he labelled each tweet manually with labels – "Extremely Negative", "Negative", "Neutral", "Positive", "Extremely Positive". This brings some scepticism to the data collection method as (1) there are doubts with the confidence of the sentiments, (2) the detailed source and collection process is not specified, and (3) there are some 45,000 over tweets to be labelled, manual tagging is not feasible and should be automated. Regardless, the dataset is has a 10.0 usability score on Kaggle, and is also

well received by the Kaggle community – 407 upvotes, 133 published notebooks, and 9 discussion forums.

Before doing any fancy stuff, I went through a process of Exploratory Data Analysis to get a better understanding of the dataset first. The main goal is to determine the issue within the dataset that requires a solution to ensure that our model will make better and accurate predictions. I will mainly focus on the tweet and the sentiment, as I would use these two columns for the predictive task later on.

Generally, the tweet and sentiment come as a string, being qualitative data. There are no missing values within these two columns, requiring not dropping of records.

```
My food stock is not the only one wh
ich is empty...\r\r\n\r\r\nPLEASE, d
on't panic, THERE WILL BE ENOUGH FOO
D FOR EVERYONE if you do not take mo
re than you need. \r\r\nStay calm, s
tay safe.\r\r\n\r\r\n#COVID19france
#COVID_19 #COVID19 #coronavirus #con
finement #Confinementotal #Confineme
ntGeneral https://t.co/zrlG0Z520j
```

Figure 1: Sample Tweet

Figure 1 shows a sample tweet from the training set, the contents mainly represented as ASCII (American Standard Code for Information Exchange) format. There are some dirty/noisy elements within the text such as unwanted punctuations, carriage return, line feed, hashtags, and hyperlinks, which are to be processed.

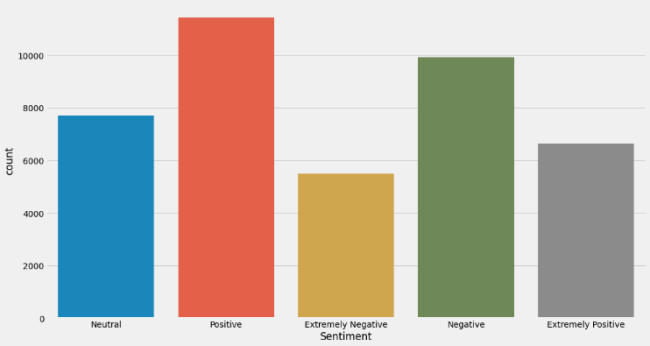

Figure 2: Class Distribution of Sentiments

Inferred from Figure 2, there is no strong imbalance of classes. If there were to have a class imbalance, random down-sampling would be the only solution as there are no quantitative columns. One issue with the labels being that there is "Extremely Positive" and "Extremely Negative" within the classes, the author did not specify the definition/criteria to make a sentiment to be extreme on its polarity; thus, it is hard to infer what exactly is "extremely". As such, I would convert "Extremely Negative" and "Extremely Positive" to "Negative" and "Positive".

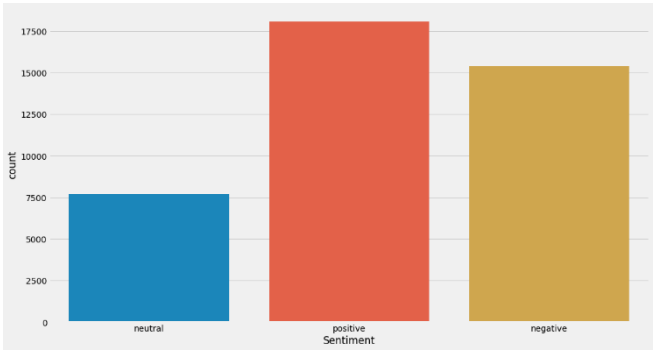

Figure 3: Class Distribution of Sentiments after mapping of classes

IV. METHODOLOGY

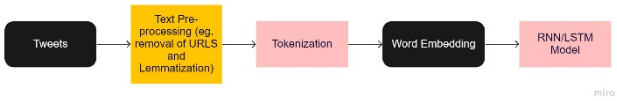

Figure 4: Proposed Methodology

Figure 4 shows the framework of the proposed methodology for sentiment analysis. This follows a series of steps with the major component that involves – 1. Tweet Pre-processing, 2. Tokenization, 3. Word Embedding, 4. Model Building and Training. Generally, this framework is a fairly common framework when it comes to a basic Text Classification task.

In Figure 1, the sample tweet shows an example of containing noisy features within the text, such as URLs, hashtags and ASCII contents. The issue with this is that as humans, we might be able to comprehend the meaning behind the text; however, for the machine, machine learning does not understand text corpus and even more so for the meaning of these noisy features. Thus we need to efficiently convert text to numbers for the machine to be able to learn and understand. For Tweet Preprocessing, the proposed method is to first remove noisy content within the text such as URLs, HTML, hashtags, and special characters which involves multiple regular expression operations. The second step is to perform lemmatisation, which is to reduce the inflected words to the root word. For example, "runs", "running", "ran" are all forms of the word "run", therefore "run" is the lemma of all these words.

```
my food stock one empty pleas
e panic there will be enough
food for everyone take need s
tay calm stay safe
```

Figure 5: Sample Tweet after Text Pre-processing

As earlier mentioned, machines are unable to understand text corpus, which comes a need to convert text to quantitative values. This process is called tokenization, which is to convert each text into a sequence of integer, where each integers corresponds to a unique word. This numerical rerpesentation are then padded with "0" ("0" is a reserved index that would not be assigned to any word) based on the

largest sequence to ensure every sequence would have the same shape.

```
array([112,   6,  31,  32, 153,  44,  22, 141, 397, 366, 180,   6, 167,
        80,  70,  14,  55, 885,  55, 103,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0])
```

Figure 6: Sample Tweet after Tokenization

Even though we converted each word to integers or tokens; these numerical representations do not provide any meaning or context to the text that could be used for inference. Rather than using these tokenised integers, we could convert these words into a dense vector representation that represent each word within the corpus to represent the entire vocabulary. This process is called word embedding many notable pre-trained words are embedding available, such as GloVe and Word2Vec, however, I will use the embedding layer provided by the Keras API.

```
<tf.Tensor: shape=(1, 281, 16), dtype=float32, numpy=
array([[[-0.00912071, -0.03297815,  0.00598743, ..., -0.04901484,
          0.01586849,  0.01724828],
        [ 0.01177664, -0.00482286,  0.01305325, ..., -0.00169401,
         -0.03977614,  0.0494034 ],
        [-0.02209679,  0.01709277, -0.01858161, ..., -0.02405515,
          0.02002939, -0.02905693],
        ...,
        [-0.0425273 , -0.03333119, -0.02027551, ...,  0.02737841,
          0.03019614,  0.01175817],
        [-0.0425273 , -0.03333119, -0.02027551, ...,  0.02737841,
          0.03019614,  0.01175817],
        [-0.0425273 , -0.03333119, -0.02027551, ...,  0.02737841,
          0.03019614,  0.01175817]]], dtype=float32)>
```

Figure 7: Sample Tweet after Embedding

V. Model Implementation

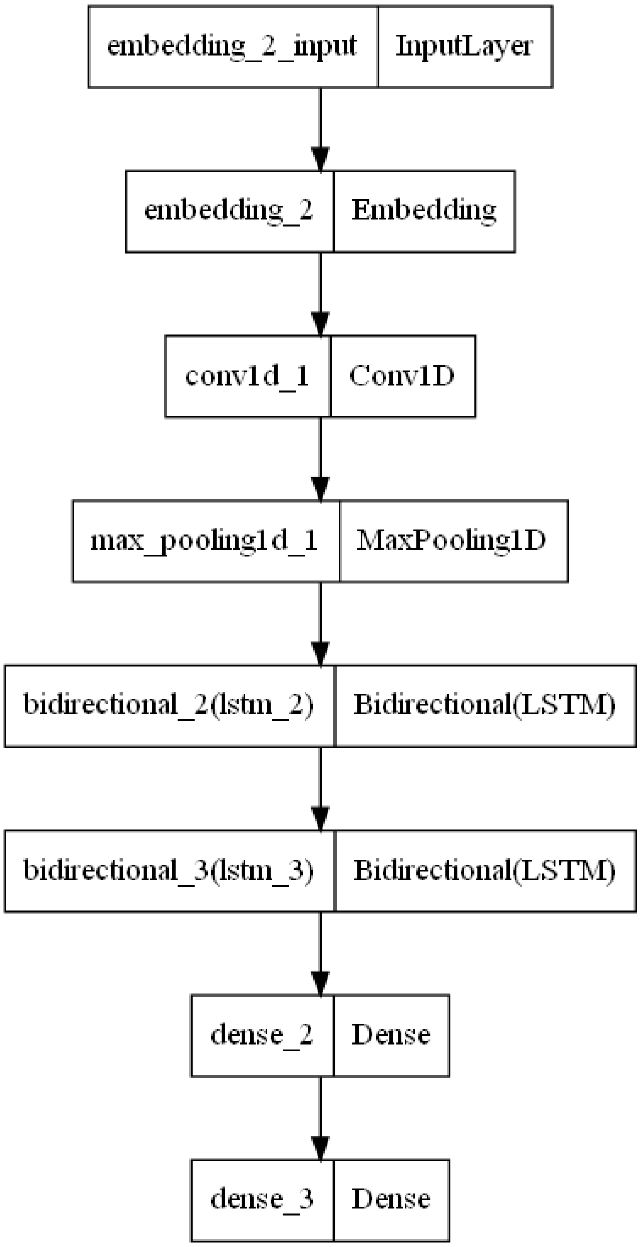

Figure 8: Hybrid CNN-LSTM Architecture

As seen in Figure 8, the proposed approach is to implement a hybrid CNN-LSTM network to make a prediction for the sentiments of the tweet with some COVID-19 context. Each major component has an important role to play within the architecture, CNN to perform feature extraction and LSTM to learn long term dependencies.

An LSTM (Long Short-Term Memory) layer consists of hidden states or cells that can process sequential inputs. The advantage of an LSTM unit over an RNN unit is that it consists of more gates, a long-term gate, and a short-term gate, that it could choose which information to keep and forget. This is a valuable asset for LSTM as it could better learn long-term dependencies in the training data. A bidirectional layer is added such that the LSTM cells can factor in the past and future context of the corpus, instead of just learning the past context.

However, an issue with LSTM is that it takes a very long time to train over large data due to the number of computations within the layer. This is where CNN (Convolution Neural Network) comes in. Earlier on, the word embedding has represented each word as a dense vector representation. A convolution layer could perform a number of matrix multiplication, which is in some sense extracting the features from the dense vector representation into a feature map. Max Pooling has then been added to down sample the feature map into a minimal representative element. This saves computation time as the LSTM structure can just learn from the down sampled feature maps.

The network is then ended with an output FC (Fully Connected) layer with SoftMax activation to output probabilities of the sentiments. The network is compiled with Adam (Adaptive Momentum) optimizer, sparse categorical log loss function. The model is trained with a batch size of 256, 10 epochs, average about 62.6s ± 4.64s per epoch with GPU acceleration and prefetching.

VI. Results and Discussion

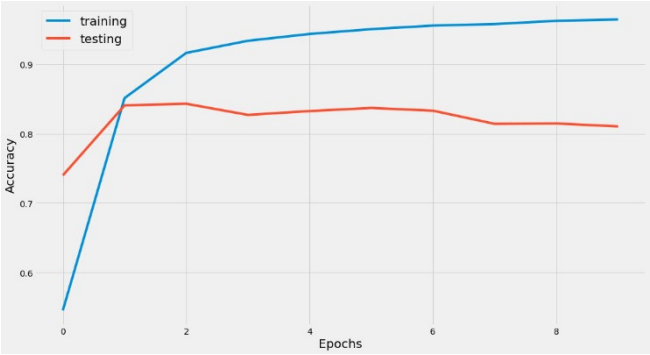

Figure 9: Accuracy Learning Curve (10 Epochs)

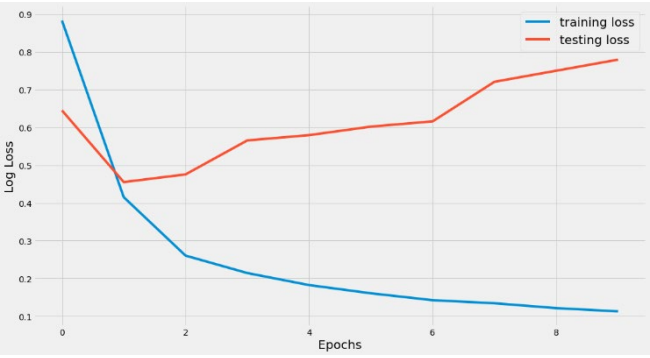

Figure 10: Log Loss Learning Curve (10 Epochs)

Table 2: Model Evaluation Results

Description	*Training Accuracy*	*Testing Accuracy*
Hybrid CNN-LSTM w/ 5 Epochs	94.38%	83.99%
Hybrid CNN-LSTM w/ 10 Epochs	96.45%	81.04%
Hybrid CNN-LSTM w/ 20 Epochs	97.88%	78.90%

The learning curves from Figure 9 and Figure 10 shows good performance in training data, but diminished results testing data as the network continues to train over the number of epochs. This indicates very clear signs of overfitting, due to the model being able to generalise very well with the training data but unable to generalise well new data, the testing data. Table 2 further supports these claims, as the number of epochs of model training increases, validation results start to diminish, a clear sign of overfitting.

With these issues, I went to look up more ways to further improve this framework that could be better implemented compared to the current method I've done. Below I listed down some ideas I could further expand my implementation.

- Use GloVe (Global Vectors for Word Representation) pretrained word embeddings for better reliability in results, as they are created based on statistical information and an unsupervised learning approach [6]
- A recent innovation in text augmentation, NLP Augmentation, could be used to add noise to the training data to better mitigate overfitting within the network [7]
- Implement Google's BERT language model for text classification [8]

VII. Results and Discussion

In this paper, we have explored the implementation of a hybrid CNN-LSTM model. Though multiple disadvantages are using the model such as overfitting and we did not achieve high test results, there is some effectiveness to a certain extent in the model learning from the tweet data and making predictions. For further improvement, perhaps expanding the current implementation could further impede better results.

Acknowledgment

I thank Dr Wilson and Dr Peter, who implemented this technical paper section into the assignment.

1. Nguyen, K. P.-Q., & Van Nguyen, K. (2020). Exploiting Vietnamese social media characteristics for textual emotion recognition in Vietnamese. 2020 International Conference on Asian Language Processing (IALP). https://doi.org/10.1109/ialp51396.2020.9310495

3. Zhou, X. (2021). Sentiment Analysis of COVID-19 Tweets. CS230 Deep Learning. Retrieved November 20, 2021, from http://cs230.stanford.edu/projects_spring_2021/reports/4.pdf.

4. Nasir, J. A., Khan, O. S., & Varlamis, I. (2021). Fake news detection: A hybrid CNN-RNN based Deep Learning Approach.

International Journal of Information Management Data Insights, 1(1), 100007. https://doi.org/10.1016/j.jjimei.2020.100007

5. Miglani, A. (2020, September 8). Coronavirus tweets NLP - text classification. Kaggle. Retrieved November 20, 2021, from https://www.kaggle.com/datatattle/covid-19-nlp-text-classification/.

6. Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). https://doi.org/10.3115/v1/d14-1162

7. Makcedward. (n.d.). Makcedward/NLPAUG: Data Augmentation for NLP. GitHub. Retrieved November 20, 2021, from https://github.com/makcedward/nlpaug.

8. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. CoRR, abs/1810.04805. Opgehaal van http://arxiv.org/abs/1810.04805