Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Classification of COVID-19 Tweets Using Bidirectional Encoder Representation for Transformers(BERT)

Master's Thesis in Computer Science

by

Samuel Opare

Internal Supervisors

Martin G. Skjæveland

Magnus S. Book

June 15, 2021

Abstract

The outbreak of COVID-19 in the later part of 2019 caused a lot of panic and led to the loss of millions of lives. Much of the chaos could have been avoided if the spread was detected in the early stages of the outbreak. Also had there been adequate information about its mode of transmission, prevention measures and symptoms, the outbreak could have been controlled. In this thesis we perform a classification of COVID-19 tweets using BERT.

BERT is a deep learning algorithm that is designed using transformers. It is broadly used on text data in natural language processing. We modify the BERT architecture for COVID-19 tweet classification. We also show how to train the algorithm to identify our tweets using biomedical literature abstracts as an alternate data source.

We discovered that the BERT model performs unsatisfactorily in our tests in comparison to our baseline model, logistic regression. We also learned that the BERT model requires a large amount of data for training. This is despite the fact that it has been pre-trained. We also discovered that training the model with a combination of tweets and literature abstracts improves its performance as opposed to training it with only literature abstracts.

Acknowledgements

I would like to thank all my professors at the Department of Electrical Engineering and Computer Science of the University of Stavanger. I would also like to thank my supervisors, Martin Skjæveland and Magnus Book for their guidance throughout the process of writing.

Also, special thanks to Mr. Philip Boateng, Eric Narh and my mother, Henrietta Narh.

Contents

Abbreviations

Acronym	What (it) Stands For
NLP	**N**atural **L**anguage **P**rocessing
BERT	**B**idirectional **E**ncoder **R**epresentation for **T**ransformers
WHO	**W**orld **H**ealth **O**rganization
MLM	**M**asked **L**anguage **M**odel
LSTM	**L**ong **S**hort **T**erm **M**emory
RNN	**R**ecurrent **N**eural **N**etwork
ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork

Chapter 1

Introduction

In 2020, the world came to a standstill as many countries were affected by the COVID-19 virus. It was a phenomenon that affected both businesses and individuals. Many businesses had to be shut down while many employees were laid off. Public gatherings were banned and lock-downs were imposed by governments in a bid to control the spread of the virus. In December 2019, the first report of COVID-19 infection came from Wuhan, China. On January 20, 2020, it was proclaimed a health emergency. The World Health Organization reported it a pandemic soon after. The virus had spread from its epicentre, Wuhan to every continent in the world in less than three months. The speed with which the virus spread was alarming and led to the loss of millions of lives. Many governments were overwhelmed by the situation and were failing to protect their citizens despite their painstaking efforts to control the outbreak.

In the early stages of the COVID-19 outbreak, not much was known about its symptoms, mode of transmission, risk factors, treatment and ways to protect oneself from contracting it. This made it difficult to control. For instance, about 80% of the carriers were asymptomatic [1], but due to inadequate information, they went about their normal lives. Thereby, spreading the disease unwittingly. At the time of writing in May 2021, the virus has infected 129 million people and killed 3.76 million people worldwide. This experience highlights the need for public health surveillance in detecting an outbreak in its early stages. Undoubtedly, technology plays a major role in such strategies especially through the use of data.

In this project, we investigate an approach to health surveillance that would enable us to determine and predict patterns of a possible disease outbreak. This would enable intervention at an early stage and prevent devastating effects. We propose the use of textual data from Twitter to fine-tune and train a deep learning model to classify tweets related to COVID-19 into five categories. These categories are symptom, treatment,

health risk, transmission and prevention. Categorising the data into these groups broadly aids in identifying and understanding a disease. It would also provide insight into preventive measures and available treatments for the disease. The objective of this task is to create a deep learning model which can be applied in the detection of future outbreaks of diseases with similar symptoms. It may also be beneficial in cases where much is not known about the disease. This would be possible through transfer learning where features or information learnt by a machine learning model in performing a task can be applied by the same model in performing another task of a similar nature.

Due to the scarcity of relevant tweets about COVID-19 in the early stages of the outbreak, we consider making use of biomedical literature abstracts in addition to tweets as suggested by Khatua et al. [2] for training our model. In the event that the disease of concern has some historic data from previous research or outbreaks, biomedical literature about it can be used in building our model. We use both old and new literature about COVID-19 to perform this task. In addition to solving the problem of scarcity of data, the biomedical literature provides quality content that is useful in training the model. The combination of data from the two sources could potentially improve the performance of our model significantly.

Also, the challenge of finding large amounts of labelled tweets serves as a hindrance to research work. The approach of implementing an unsupervised learning model (such as K-means clustering [3]) on the data to group and label it before performing the main task at hand is cumbersome and would rather be avoided. We hope that this model can serve the purpose of labelling data for similar tasks.

This project aims to explore effective methods of improving public health surveillance. We make use of a state of the art deep learning model to predict disease outbreak through the classification of tweets using biomedical literature abstracts, tweets and Natural Language Processing (NLP).

The primary contributions of this thesis are:

- Fine-tuning of a deep learning model to perform classification of tweets related to COVID-19 disease outbreak.
- Investigating the efficiency of using biomedical literature for classifying tweets to learn new information during a disease outbreak.
- How a combination of data from different sources can be used to train a deep learning algorithm to improve its performance.
- Comparison of the performance between a deep learning model and a machine learning model.

- Provision of a collection of labelled tweets related to COVID-19 in the categories of symptom, health risk, prevention, treatment and transmission.

This paper is divided into six main chapters: In the second chapter, we introduce some fundamental concepts used in this project. The third chapter focuses on some related works, the fourth chapter focuses on our solution approach of using Twitter and literature data for the classification of COVID-19 tweets, In the fifth chapter, we evaluate our results. We conclude and offer suggestions on future work in the sixth and final chapter.

Chapter 2

Preliminaries

In this section, we deal with the theoretical foundations of the project. We discuss Natural Language Processing(NLP), language models, Artificial Neural Networks (ANN), Transformers and more.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a growing field of research in statistics, computer science, and artificial intelligence that focuses on giving computers the ability to process and interpret natural language in the form of text or speech. It is applied in tasks that require a computer to analyse the contents of a document and extract some useful information from it. It is used across many domains including Robotics for speech recognition, in machine translation for translation of languages, sentiment analysis and also in medical informatics and bioinformatics for tasks like hospital record management and classification of protein sequences in DNA and RNA.

NLP deals with data that is represented in sequences, for instance, audio, video, and textual data (sentences). Due to the vast amount of textual data available, NLP tends to be a very useful area of research for many applications. The data used in this project is unstructured text data. We apply some Natural Language Processing techniques to process and analyse it, this provides us with some useful information for categorization. A common problem in NLP is the problem of polysemy. Polysemy refers to the existence of multiple meanings for the same word or phrase. A popular Python library used for tasks of this nature is the Natural Language Toolkit (NLTK)[4].

2.2 Language Models

A language model learns to predict the probability of words occurring within a sequence. This ability to model the rules of natural language using probabilities gives NLP some leverage to perform certain high-level tasks. To explain why probabilities are used, let us assume for example that we have two texts "going man home" and "the man is going home". We would expect the second text to have a higher probability of being the right sequence than the first. This approach provides a basis for grammatically correct language models.

Language models can be divided into two categories:

1. Statistical language models: These use statistical models such as the Hidden Markov model, n-grams and Bayes' theorem to learn the probability distribution of words. They work by generating features from an input of large corpora(collection of terms in textual data) and make probabilistic decisions by assigning real value weights to the generated features. For example, based on the frequency of the occurrence of a term it may be penalised or rewarded by applying a weight to it. These models do not hold any contextual information.

2. Neural language models: These leverage neural networks to model language. Neural networks refer to a collection of algorithms used in Artificial Intelligence to learn patterns in data to solve classification problems. We discuss neural networks more in Section 2.3, on neural network models. They are a significant improvement over statistical-based methods since these neural networks learn semantic and syntactic relationships among words in a text and therefore employ contextual understanding in performing high-level tasks like question answering.

2.2.1 Word Embeddings

Word embedding is a method by which words are transformed into real number vectors that the computer can understand. Methods like Word2vec [5] were introduced to replace one hot encodings which are a scarcity matrix(matrix consisting mostly of zeroes, representing the absence of a term from the sequence) and inefficient for natural language processing. Word embeddings led to improved results but failed to integrate the context of words within a sentence into its embedding. For instance, the word 'bank' will have the same embedding in the sentence 'I am going to the river bank' as 'I am going to create a bank account' even though they have different meanings in both sentences. It's also said to have a short dependency range. Meaning that it failed to consider how the

words surrounding a particular word in a sentence relates to that word. Especially when it is part of a long sentence or document.

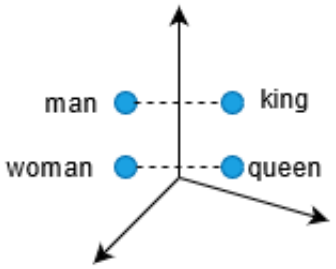


Figure 2.1: Word embeddings can model gender relationships in words

2.2.2 Data Processing

Processing data before passing it to a deep learning model is a recommended practice in NLP. Removing irrelevant terms, converting different words with the same root to their root word among other things leads to better results. In this section, we discuss some of these text pre-processing steps.

- Tokenization: This is the process of separating words from a text string into individual parts called tokens. Tokenization also removes punctuation marks from the text. For example a sentence such as 'we are the world.' will have four tokens which are 'we', 'are', 'the', and 'world'.
- Stemming: The practice of removing prefixes and suffixes from a word is known as stemming. This allows words to be shortened into their base form. This is to prevent words that are derivatives of other words from being interpreted as a word with a completely different meaning. For instance, the word running will be converted to run.
- Stopwords Removal: This involves the removal of grammatic constructs such as articles, pronouns and conjunctions. For instance, the words and, or, to, and on are removed. This is because these words contribute little to no value to the overall context of the text.
- Lemmatization: This is the process of converting words to their root form. It also groups different derivates of a word as a single word. For example, the words 'are', 'been', 'be', 'was' and 'is' are converted to their base form, 'be'.

2.2.3 Logistic Regression

Logistic regression is a statistical-based classification algorithm used for binary and multi-class classification tasks [6]. It is based on linear regression which is used to model the correlation between variables by fitting the data to a linear function. It uses a logit function to predict the probability of an observation belonging to a class. Logistic regression is mostly used in classification tasks where the predicted variable is categorical in nature or a discrete value. It is usually used as a base model for testing other classification models because of its simplicity.

linear regression function:

$$y = b_0 + b_1 \cdot x_1 + \ldots + b_n \cdot x_n \tag{2.1}$$

Apply a sigmoid function to the linear regression to obtain a probability value.

$$P = \frac{1}{1 + \exp -y} = \frac{1}{1 + \exp -(b_0 + b_1 \cdot x_1 + \ldots + b_n \cdot x_n)} \tag{2.2}$$

It is estimated using Maximum Likelihood Estimate(MLE). MLE is the process of using a likelihood function to compute parameters that are likely to yield the expected output.

2.3 Neural Networks

Neural networks refer to a collection of algorithms used in Artificial Intelligence to learn patterns in data and solve complex problems. The techniques in artificial intelligence that make use of neural networks are generally known as deep learning.

The human brain serves as a model for Artificial Neural Networks(ANN). It is built up of layers of nodes that are interconnected (also called neurons). A perceptron is the most basic type of neural network [7]. The input layer and the output layer are the only layers that make up a perceptron. A feed-forward neural network refers to a neural network which functions by passing data into the network through a single layer (the input layer). The data is passed on to the next layer in a specific direction until the output is given from the output layer. An example of a basic feed-forward neural network is shown in Figure 2.2 below. Hidden layers are additional layers between the input and output layers in more complicated neural networks. Each node in the hidden layer is connected to all nodes in the preceding layer and all nodes in the following layer, and information is passed through these connections. The more complicated the network is, the more complicated the tasks it can complete. A network may have many hidden layers and

many nodes within a layer. This is where the term deep learning emerges from. An example of a complex neural network is the Convolutional Neural Network(CNN) [8] which is used in object recognition.

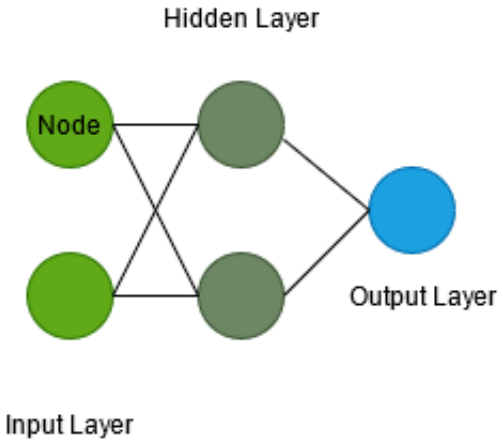


Figure 2.2: A simple feed-forward Neural Network with 1 hidden layer

The neural network is trained to perform tasks such as object recognition by computing weights which are applied to a function to predict the output of a given input. Using the example of an image recognition task, we describe how a neural network functions.

If we wanted a neural network to classify a given image into one of the following categories; cap, car or cat, we would train it by providing it with many images of these items. The neural network then learns the patterns and details of the various images with their labels. Hence when the neural network is provided with an image that can be classified into one of these categories it can do so accurately. Supervised learning is the name for the process of training the neural network. We discuss the difference between supervised and unsupervised learning in section 2.3. The following steps illustrate the learning process:

1. Each node within the input layer of the neural network represents one pixel of the image. Hence, if the input image has a dimension of 15 x 15, the input layer will have 225 nodes. Each node holds a value that represents the intensity(colour) of the pixel. Each node passes this value to every node in the next layer after multiplying it with a random weight. Each node sums up these weighted values. This weighted sum looks like this:

$$w_1a_1 + w_2a_2 + w_2a_2 + ... + w_na_n \tag{2.3}$$

where a_i represents the pixel value from node i in the previous layer and w_i is a randomly assigned weight.

2. An activation function is used to set a threshold of values that are acceptable to be passed onto the next layer. The sigmoid function is an example of a widely used activation function. This changes the weighted sum's value to a number between 0 and 1. We discuss activation functions later in Section 2.3. After applying the activation function we have:

$$\sigma(w_1a_1 + w_2a_2 + w_2a_2 + ... + w_na_n) \tag{2.4}$$

where σ is the activation function.

3. A bias is added to determine how high the weighted sum needs to be for an activation function to be applied:

$$\sigma(w_1a_1 + w_2a_2 + w_2a_2 + ... + w_na_n - b) \tag{2.5}$$

where b is the bias

The complete formula for computing the output of node 0 in layer 1 is as follows:

$$a_0^{(1)} = \sigma(w_{0.0}a_0^{(0)} + w_{0.1}a_1^{(0)} + w_{0.2}a_2^{(0)} + ... + w_{0.n}a_n^{(0)} + b_0)$$

Figure 2.3 illustrates it further. The values $a_0^{(o)}$ to $a_n^{(o)}$ represent the values of all nodes in layer 0, the weights $w_{0.0}$ to $w_{0.n}$ represents the weights applied to each node value in layer 0 respectively. A weighted sum of these values plus the bias b_0 yields the value $a_0^{(1)}$ which is passed to node 0 in layer 1 after applying an activation function. The values of each node in every layer are computed in the same manner.

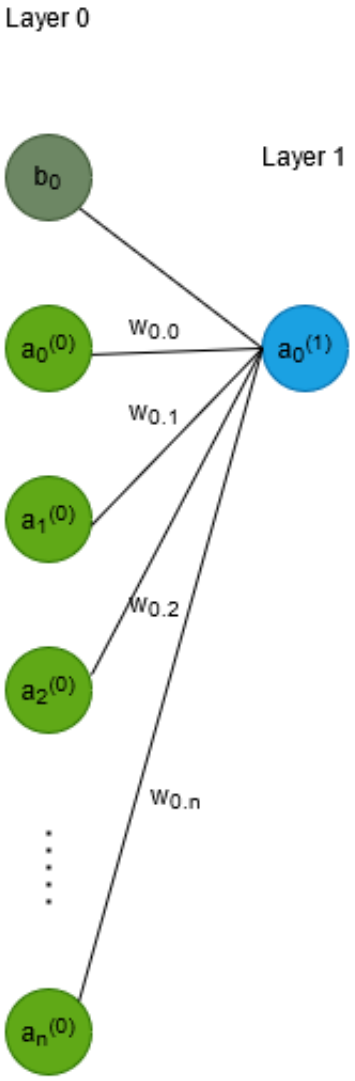


Figure 2.3: Computing the output of node 0 in layer 1

The complete process can be represented as a matrix operation as follows:

$$\sigma\left(\begin{bmatrix} w_{0.0} & w_{0.1} & \dots & w_{0.n} \\ w_{1.0} & w_{1.1} & \dots & w_{1.n} \\ . & . & \dots & . \\ w_{k.0} & w_{k.1} & \dots & w_{k.n} \end{bmatrix} \begin{bmatrix} a_0^{(i)} \\ a_1^{(i)} \\ . \\ a_n^{(i)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ . \\ b_n \end{bmatrix}\right)$$

4. Steps 1-3 are done recursively to readjust the weights through a process called backpropagation(Section 2.3) to minimize the error until the predicted output matches the expected output. The error, in this case, refers to the discrepancy between the expected value and the value outputted by the model.

Activation Functions

Activation functions are used to determine whether a node's value should be passed on or not. Activation functions map a resulting value to a value between 0 and 1 or -1 and 1 or other ranges depending on the activation function used. There are two types of activation function, the linear and the non-linear. Non-linear activation functions are mainly used in deep learning. Hence, we introduce the most common ones here.

1. Sigmoid function: The sigmoid function is an activation function that transforms an input value into a number between 0 and 1. As a result, it is used in tasks where the probabilities of outputs must be computed. It is best used for binary classification tasks (classification into one of 2 categories).

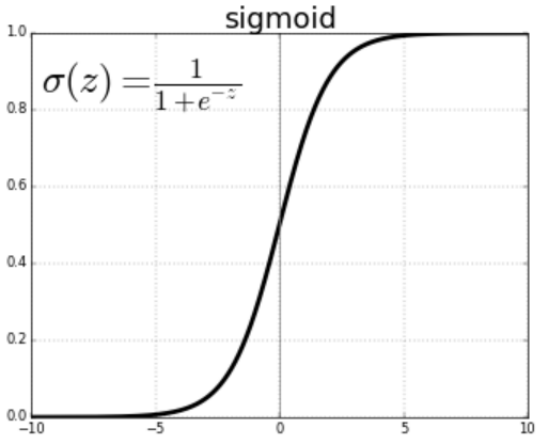


Figure 2.4: Sigmoid function

2. Softmax: Softmax activation function is similar to the sigmoid function in shape. It is used primarily in multi-class classification tasks (classification involving more than 2 categories)

$$S(z)_i = \frac{exp^{z_i}}{\sum \exp^{z_i}}$$

3. ReLU: The Rectified Linear Unit(ReLU) function maps the output value to a value from 0 to infinity. Used most often as the default activation function in multi-layer perceptrons.

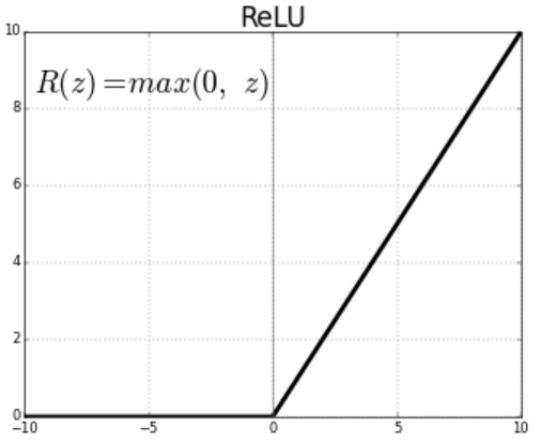


Figure 2.5: ReLU function

4. Tanh: Hyperbolic tangent(tanh) function maps the output value to a value from -1 to 1. Mostly used for classification tasks involving two classes.

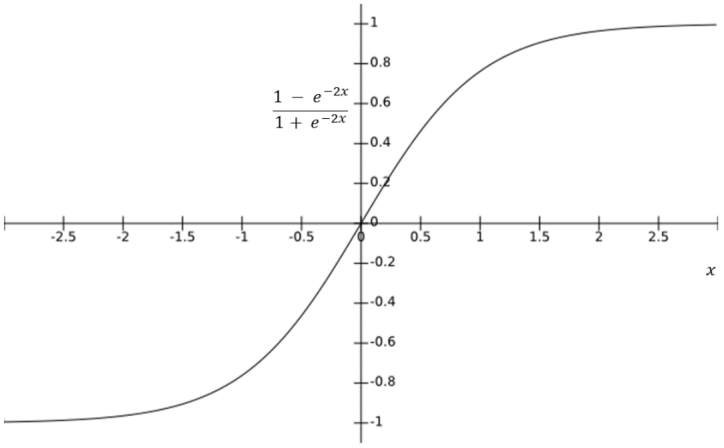


Figure 2.6: Tanh function

backpropagation and Optimization

Backpropagation is a technique for correcting the weights and biases of a model to produce the expected output. Gradient descent is a technique used in backpropagation for optimizing a function by finding its local minima. A local minimum is a point on a function where its value is smaller than nearby points but may be larger in comparison to distant points. The gradients of the cost function determine how much the weights and biases of the neural network have to be adjusted. The gradient or slope of a function shows how much a change in its argument x affects its output y. That is, it lets us know which direction the function is moving and allows us to know how much the parameter x needs to change to minimize the output of the function. This is how the local minimum

is computed by adjusting the parameters of the cost function. During backpropagation, the error may increase or decrease exponentially as a result of the distance of the neural network(a large number of layers). This is termed the vanishing gradient problem. Other loss functions used in optimization include cross-entropy [9].

Metrics and Evaluation

In performing a classification task, the predictions made by the model fall into 4 main groups which are as follows:

- True Positive (TP): When an observation is predicted to belong to a class that it actually belongs to.
- True Negative (TN): When an observation is predicted to not belong to a class, and it actually does not.
- False Positive (FP): When an observation is predicted to belong to a class, and it actually does not.
- False Negative (FN): When an observation is predicted to not belong to a class, but it actually does.

Accuracy, precision, recall, and f-measure(f1-score) are the metrics used to assess a model's performance.

- Accuracy is a measure of the percentage of correctly predicted values. It is calculated by dividing the number of correct predictions by the total number of predictions.

$$accuracy = \frac{correct\_predictions}{total\_number\_of\_predictions}$$

- Precision is a measure of the fraction of values that are predicted to belong to a class to the total number of values that are predicted to belong to that class. It is measured by dividing all true positive predictions by all positive predictions.

$$precision = \frac{TP}{TP + FP}$$

- Recall is a measure of the fraction of values that are predicted to belong to a class to the total number of values that actually belong to that class. It is measured by dividing all true positive predictions by a sum of true positives and False negatives.

$$recall = \frac{TP}{TP + FN}$$

- F-measure is a measure of accuracy that balances the precision and recall values. It is a form of accuracy measure that takes the weights of the class labels into account.

$$f - measure = 2 * \frac{(precision * recall)}{(precision + recall)}$$

Supervised and Unsupervised Learning

Supervised learning refers to a technique of training a deep learning model where inputs and desired outputs are fed to the model. During training, the estimated output is corrected using a cost function when it deviates from the desired output. A cost function is used to correct the error made by a model during training. A commonly used cost function for supervised learning is the Mean-Squared Error(MSE). Gradient descent is applied to minimize the MSE of the model. Examples of tasks that use this method are classification and regression algorithms.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(y_i - \hat{y}_i)^2}$$

where y_i is the expected value and $\hat{y}_i$ is the predicted value and n is the number of data samples.

Conversely, unsupervised learning is a method of learning where a model is given a set of inputs without the desired output. The model is trained by penalizing it by using a cost function that minimizes error. Examples of tasks that use this method of training is clustering.

Overfitting

Overfitting refers to the situation where our deep learning model fits almost perfectly to the data during learning. It results in high training and validation accuracy. However, when the model is tested on new data it performs woefully because it is unable to generalise to new data. In training a deep learning model many parameters have to be considered to moderate and control the training to prevent overfitting. Some of these parameters are learning rate, number of layers, number of nodes and activation function.

2.3.1 Recurrent Neural Networks

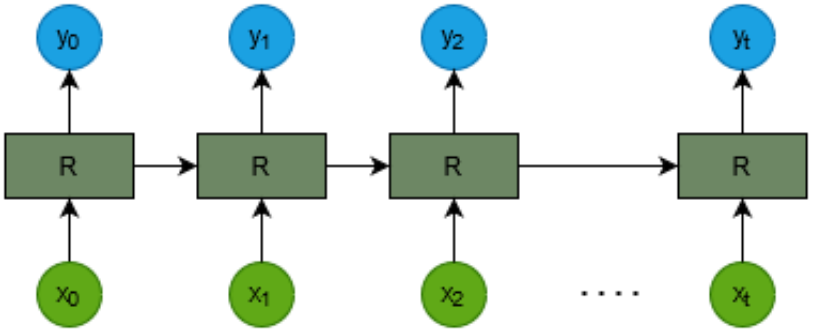


Figure 2.7: Architecture of a recurrent neural network

A Recurrent Neural Network(RNN) functions similarly to a feed-forward neural network. However, the main difference between the two is that the RNN has an internal loop that loops over each element of the input sequence, unlike the feed-forward network that processes all inputs in a single pass. It maintains information from each element encountered during each pass of the loop. Therefore, previously encountered information influences outputs. As a result, the order in which input is passed matters. passing the word 'gold' before 'pen' as input would produce different results from if it were passed in the reverse order. Each node of the RNN passes information about itself and other nodes to the next layer during each pass.

It is well suited for tasks that represent data as a sequence. For instance, text data. The problem of vanishing gradient is a fundamental shortcoming of RNNs.

2.3.2 LSTM: Long-Short Term Memory

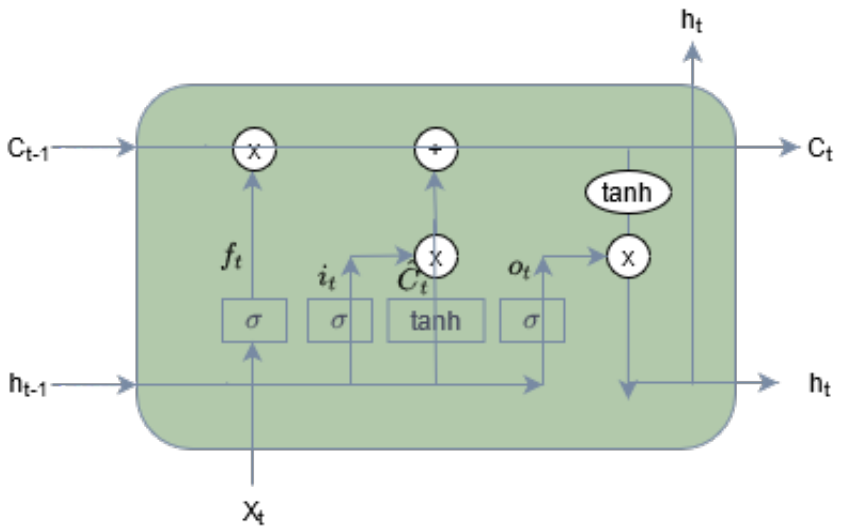


Figure 2.8: Architecture of an LSTM node

This is a special type of RNN that was created to solve the problem of vanishing gradients and helps the neural network to learn long-term dependencies.

LSTM has 4 gates within each node as opposed to one in the RNN, the tanh gate. The gates consist of three sigmoid gates and a tanh layer. The gates are the forget gate, input gate and output gate [10]. Another key component is the cell state through which information is passed between nodes. The gates determine which information is added to or removed from the cell state [11]. It functions as follows:

1. When values are received from the previous layer/state, the forget gate determines which values should be removed from the cell state by returning a value of 0 or 1 for each value received from the previous state. 0 meaning forget and 1 meaning keep.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.6}$$

2. Next, the value to be stored in the cell state is determined by first determining which values to update. The input layer is in charge of this. Following that, the tanh layer generates new candidate input values that will be added to the cell state.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.7}$$

$$\hat{C}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{2.8}$$

3. The old cell state is then updated with the new cell state values.

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.9}$$

4. The output values are determined by passing the cell state values through a sigmoid layer and then by applying a tanh function to the cell state values to convert the values to values between -1 and 1. The values from the sigmoid and tanh are then multiplied to obtain the final output.

$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.10}$$

$$\hat{C}_t = O_t * tanh(C_t) \tag{2.11}$$

2.3.3 Autoencoder

The autoencoder has a similar architecture to the feed-forward network. However, it is shaped like an hourglass with fewer nodes in the middle layer than the input and output layers. It is used for compressing information. This network is symmetrical about the middle and is composed of three parts. The encode, the decoder and the code. The encoder, portion before the middle, compresses the input data. The middle portion, the bottleneck or code, contains the smallest dimension of the compressed input. The decoder produces an output that is as close to the input as possible. One use of this architecture is removing noise from an image.

2.4 Transformer

The transformer is a sequence and transduction model designed to function without a recurrent or convolutional neural network. Transduction is the conversion of a sequence of input of type A into a sequence of output of type B. For example, Translation of a sentence from English to Norwegian. A sequence refers to any string of data. For instance, a sequence of text, a sequence of numbers or a sequence of video frames. The transformer uses the attention mechanism, which is integral in the modelling of the long-range dependencies between input and output. This means that the relationship among words in a long sentence is learned. Thereby enabling the model to understand the context of words within the sequence during training. An attention mechanism is a technique used to model long-term dependencies by storing network states independently and switching attention between them. First, the hidden layer states of each iteration in the encoders are stored. The decoding layers then receive the information from the

encoders after being filtered by an attention context. This process of filtering adds context to the decoder by highlighting certain features(words). It works by using self-attention, which has been proved to be effective in jobs like reading comprehension and question-answering systems. Self-attention is an attention mechanism used in the transformer to simulate various positions of a single sequence to model its representation [12].To put it another way, for each input in the sequence, self-attention tries to figure out how it relates to the other words in the sequence to gain a better understanding. The advantage of the transformer over recurrent and convolutional neural network-based sequence and transduction models is that it is designed to make parallelization of the task possible and it can model long-range dependencies with ease [12].

2.4.1 Transformer Architecture

The transformer follows the encoder-decoder structure. It uses stacked self-attention and pointwise layers in both the encoder and decoder.

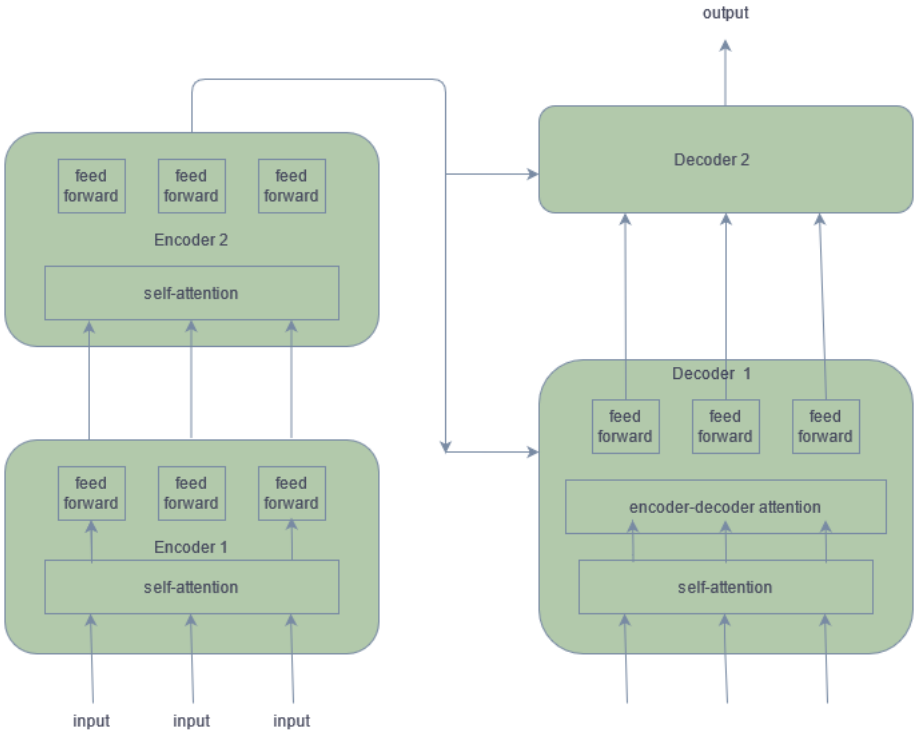


Figure 2.9: Architecture of a transformer

- Encoder: The encoder is made up of N identical layers stacked together. The multi-head self-attention layer and the location wise simple feed-forward network are both sub-layers of each encoder. The output of each sublayer is normalized by a normalization layer before it is passed on to the next sublayer.

- Decoder: The decoder also consists of the same number of N identical layers as the encoder. Each having three sublayers. Along with the two sublayers of the encoder, the decoder has a third layer which is the masked multi-head attention layer. The decoder also employs the normalization layer for each sublayer's output. The decoder also has an encoder-decoder layer that helps it focus on the appropriate part of the sequence.

2.4.2 Transformer Encoder and Decoder Relationship

First, the input sequence is passed to the first encoder within the stack, these inputs are processed and passed on to the next encoder in that order until it reaches the final encoder within the encoder stack. The last encoder's output is sent to all of the decoders in the decoder stack. The decoder then finalizes the task. For instance, it converts the English input into Norwegian output.

2.4.3 The Self-Attention Mechanism

To compute attention, three vectors called query (q), key (k), and value (v) vectors are created from the input embedding. To illustrate the process of calculating self-attention, let us consider the sentence "Love is war".

1. The first step is to compute a score for each word in the sequence in relation to the word "Love" (the first word in our sequence). This score determines the level of importance of each word in relation to other words when encoding the sequence. The dot product of the query vector(q1) of the word "Love" and the key vectors of words in the sequence are computed to determine the score [13] of each word.

$$s_1 = q_1 \cdot k_1$$

$$s_2 = q_1 \cdot k_2$$

$$s_3 = q_1 \cdot k_3$$

2. The next step is to divide the scores by the square root of the dimension of the key vector.

$$s_1/d$$

$$s_2/d$$

$$s_3/d$$

3. The results are then normalized using the softmax activation function.

$$x_1 = softmax(s_1/d)$$

$$x_2 = softmax(s_2/d)$$

$$x_3 = softmax(s_3/d)$$

4. Next, the normalized score of each word is multiplied by the value vector(v_1, v_2, v_3) of that word.

$$x_1 \cdot v_1$$

$$x_2 \cdot v_2$$

$$x_3 \cdot v_3$$

5. After that, the scores are added together to get the result vector$(z1)$, which is the self-attention vector for the word "Love".

$$z_1 = x_1 \cdot v_1 + x_2 \cdot v_2 + x_3 \cdot v_3$$

6. The output$(z1)$ of the self-attention layer is served as input to the feed-forward network.

7. Repeat steps 1-6 to compute self-attention for other words in the input sequence in the order in which they appear. That is, next, “is” and then “war”.

This process of computing self-attention is done multiple times in a parallel and independent fashion. The outputs are then transformed linearly and concatenated before being passed on to the next layer. This is termed multi-head attention.

2.4.4 Limitations of the Transformer

Despite its advantages, the transformer has one major drawback which affects its performance. The transformer can only work on a fixed length of input. In effect, a sequence must be truncated into chunks if the input length is greater than the expected length. This leads to context fragmentation because sentences have to be truncated. Hence, leading to information loss.

The transformer-XL was introduced to remedy this situation[14]. The architecture of the transformer-XL functions such that the hidden state computed for a segment is passed on to the next segment as part of its input. This ensures that the context from

the previous segment is maintained during the process of training the current segment. Hence, mitigating the drawback of fragmented context resulting from the truncation of sequences into segments or chunks[14].

2.5 BERT: Bidirectional Encoder Representation from Transformers.

BERT is a language model from Google AI and is designed based on the transformer architecture. However, all the layers are encoder only. It uses an attention mechanism for training. Also, it operates in two directions as its name implies. The self-attention layer of the model computes the self-attention of the tokens in both directions of the sequence. That is, it learns from both the left and right contexts of a token within the sequence [15]. This aids in the resolution of the issue of polysemy.

For example, In the sentences "I can lift a light weight" and "turn on the light", the word "light" has a different meaning in both. This is easy for humans to interpret. However, for a machine, this is not easily interpreted. Bidirectional training enables the machine to learn the contexts of the word in the two sentences and distinguish between the two. Hence, preventing errors or poor performance. Language models like word2vec fail in this regard. In a word2vec embedding, the word light will have the same vector representation in any sequence. Therefore, lacking context.

BERT functions through pre-training and fine-tuning. Pre-training and fine-tuning are concepts from transfer learning where a model is trained for specific tasks and then the same model is fine-tuned to perform other tasks. One good reason for using this technique of fine-tuning a pre-trained model is when you do not have a lot of data for training.

The BERT framework is available in two versions:

1. BERT BASE: It has 12 transformer layers, 12 attention heads and 110 million parameters.
2. BERT LARGE: It has 24 transformer layers, 24 attention heads and 340 million parameters.

2.5.1 How it Functions

Pre-training of BERT is done by training the model on two tasks. Namely:

1. Masked Language Model (MLM)
2. Next Sentence prediction

We explain these tasks further in section 2.5.3. You will learn how these tasks enable the BERT model to be pre-trained to perform well on other NLP tasks after some fine-tuning.

2.5.2 Data Pre-processing

Input to the BERT model is pre-processed by converting the input sequence into three types of embeddings. Namely:

1. Position Embedding: These embeddings encode information about the position of tokens within the sequence.
2. Segment Embedding: These are embeddings for sentence pairs. Each sentence within the pair has a unique embedding which enables the model to distinguish between the two. These embeddings are important for tasks like a question answering system.
3. Token Embedding: These are embeddings for each token within the sequence.

These embeddings are summed up and fed to the model as input. This complex embedding enables the BERT model to be pre-trained for multiple NLP tasks.

2.5.3 Masked Language Model (MLM)

The masked Language model trains the model to predict a missing token within the sequence. This is accomplished by substituting a [MASK] token for the token to be estimated. This language model allows the BERT framework to be truly bidirectional. For the masked language model to be successful some caveats are introduced in using the [MASK] token. These are:

1. During training, 15% of the words were hidden at random.
2. To prevent the model from always expecting the [MASK] token as the missing word to be predicted, the masked word was not always replaced with the [MASK] token. This is because the [MASK] token would not appear in other tasks. Instead, 10% of the time the hidden word was replaced with random words and another 10% of the time it was not replaced at all. It was replaced with the [MASK] token 80% of the time.

The masked language model is important because it helps the model understand the relationship among words. This helps the model put words and sentences into context.

2.5.4 Next Sentence Prediction

In this task, the model is trained to predict the sentence that comes next within a sequence. That is, given two sentences A and B, is sentence B the sentence that comes after sentence A in our sequence or not? This is a classic binary classification task. The task is performed using large text data. The data is split into pairs of sentences in the following manner.

1. For 50% of the sentence pairs, the second sentence is the exact sentence that follows the first. This half of the data is labelled 'IsNext'.
2. The other half of the sentence pairings are as follows: the second sentence is a random sentence selected from the corpus. This half of the data is labelled 'NotNext'.

Pre-training on next sentence prediction allows the BERT model to be used for tasks like text summarization and auto-completion like on google search. With the model pre-trained on the two tasks discussed above, BERT can perform very well on multiple tasks and achieve state-of-the-art results.

Chapter 3

Related Works

Twitter is a social media platform that has large user-generated data that can be used in investigating and understanding relevant issues facing the population. This can be done through the use of Natural Language Processing and machine learning. Myslin et. al (2013) [16] demonstrates the utility of Twitter in investigating health issues through their research on the sentiments of Twitter users towards emerging tobacco products like shisha and e-cigarettes. Their analysis provided an insight into how these products were viewed and hence gained insight into appropriate means of intervention if it was deemed necessary. The availability and abundance of Twitter data is also leveraged to investigate the utility of Twitter in real-time health surveillance [2]. It was posited that real-time monitoring of tweets could be useful in the early detection of an outbreak. They approached this by designing a classification model to group tweets related to the Zika and Ebola virus into various categories. They approached this task using an extra tree classifier algorithm [17] and word2vec embeddings [18].

Though Twitter is a good source of data for investigating a wide range of issues due to the size of data and its availability, the problem with Twitter is that it is fraught with fake news and propaganda [19]. In a bid to investigate the validity of this problem, Chen et. al (2020) [20] identified Twitter accounts that were determined to be bots, that is, automated accounts controlled by computer software rather than a human, by analysing tweets posted about the coronavirus. These social media bots are created with the intention of propagating conspiracies and unfounded information. Another challenge of using Twitter for health surveillance is location bias. That is due to the low rate of internet penetration in third world countries as compared to the western world or the population of people in urban cities as compared to rural villages, data from certain locations may be more readily available than others which brings about some bias in the data collected. Also, tweets may be posted in different languages which

makes data collection more difficult. In predicting the correlation between the volume of tweets and infection rates of influenza based on geographical location [21], it was discovered that tweets from urban cities far outnumbered tweets from rural areas thereby creating a bias. It was also noted that using NLP in addition to various location detection modules yielded a higher correlation with the gold standard data (weekly reports from the Infectious Disease Surveillance Center (IDSC)).

3.1 BACKGROUND

In this section, we provide a brief background about the project. We talk about our data sources and the importance of public health surveillance for the early detection of an outbreak and present some examples of public health surveillance projects.

3.1.1 Coronavirus Disease 2019 (COVID-19)

Coronavirus disease 2019 is a contagious disease caused by the new severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) [22]. The disease is believed to have originated from Wuhan, China where its first case was identified in December 2019[23] and has become a significant health concern in the world. The virus is named SARS-CoV-2 because after the virus genome was sequenced it was discovered that it was genetically related to the cause of the SARS pandemic in 2003[24]. The origin of the virus is unclear. However, it is speculated that it may have originated from bats due to genome sequence similarity of 96% between SARS-CoV-2 and another coronavirus dominant in bats[22].

From here onward we will refer to the virus as COVID-19 in this text. At the time of writing in May 2021, the virus has infected 129 million people and killed 3.76 million people worldwide. COVID-19 can be transmitted among people through respiratory droplets, physical contact or particle suspension in the atmosphere [22]. Airborne transmission is the main mode of transmission and is considered highly dangerous. Hence, the use of face masks and social distancing are the most efficient preventive measures to be adopted. About 80% of the carriers of the virus are asymptomatic [22], that is, they show little to no signs of illness. This is perhaps due to strong and healthy immune systems. However, there is evidence that they can transmit the disease to others [22]. Symptomatic patients experience very severe symptoms that may result in death [24]. Some symptoms shown by infected patients include coughs, discomfort in the chest and difficulty breathing[24]. Avoiding contact with the virus is the most efficient strategy to avoid infection. Hence, recommended preventive measures include covering of the mouth and nose with a mask, washing and sanitizing hands regularly, covering the

mouth and nose when one coughs or sneezes and disinfecting frequently touched surfaces regularly [22]. A better understanding of transmission methods, virus infection rate, replication and incubation period are crucial for developing treatments and interventions for COVID-19. Viral replication is the process during infection where the virus gets into a host cell. It then generates many clones of its genome which further infect other cells of the host [25]. Incubation refers to the time from virus infection to the time symptoms begin to show in a patient[26]. From chest radiography, it is seen that the majority of COVID-19 patients lungs are affected [22]. This results in pneumonia in some cases. As a result, clinical management of the virus has been confined to measures associated with respiratory support and ventilation. Though a small percentage of patients suffer severe complications, the aged and persons with underlying illnesses or prior medical disorders, are at a higher risk of experiencing severe symptoms. [24].

Medications associated with other diseases are being investigated as viable options to treat and manage the coronavirus. This is based on clinical data collected from affected patients. Fortunately, after many months of research and isolating COVID-19 samples from patients to understand it through genomics sequencing and other experiments, medical experts have been able to create vaccines with a high percentage of potency [22].

3.1.2 Twitter

Twitter is a micro-blogging and social media networking service where users communicate with one another by sending messages called tweets. Tweets can be posted, retweeted or liked by registered members of Twitter depending on the privacy settings of the user. Tweets posted by a user are mostly visible to people who follow the user's account. These are called followers. Tweets were originally limited to 140 characters in length, but were increased to 280 characters in November 2017. Tweets may include audio, video or images.

Figure 3.1: A tweet from the Norwegian Health Department about COVID-19.

Figure 3.1 above shows an example of a tweet related to COVID-19. This is a tweet posted by the Norwegian Health Department about COVID-19. From the image, the date and time the tweet was posted can be seen. The accounts username, helse_og_omsorg, is also visible. The number of retweets, 2, is also visible.

We define some frequently used terminologies on Twitter below:

- Retweet: A retweet is equivalent to a share. By retweeting a post a user shares the tweet to their followers without editing it. This is accomplished by pressing the retweet button on a tweet.
- Quote Retweet: A quote retweet is the act of captioning a tweet from another user with your comment. This is done by clicking the quote retweet button on a post and adding your comment.
- Like: A user may like a tweet from other users by clicking the like button on the tweet.
- Mention: A mention refers to the act of directing a tweet to a specific person by mentioning them using their username. If we should post a tweet with helse_og_omsorg's username mentioned. Their account would receive a notification that they've been mentioned. They can then respond as needed. To mention a user, their username must be prefixed with the @ symbol. Example, @helse_og_omsorg.
- Hashtag: A hashtag is a mechanism used by Twitter to group tweets related to a specific topic. A hashtag is prefixed by the hash symbol followed by a word or phrase. Figure 3.1 contains the hashtag *#korona*. *Korona* is Norwegian for Corona. This lets us know the tweet is about the coronavirus. When a user searches for *#korona* on Twitter, all tweets with this hashtag will show up including the tweet in Figure 3.1.

3.1.3 Biomedical Literature

Our second dataset is biomedical literature abstracts. This data is gathered from Kaggle and the COVID-19 open research dataset challenge (CORD-19) [27]. Research results and discoveries in the biomedical industry are presented in these biomedical papers. Biomedical literature abstracts have been used as a source of data for performing classification tasks on Twitter data [2].

3.1.4 Public Health Surveillance

Syndromic surveillance as defined by the World Health Organization (WHO) is 'the continuous, systematic collection, analysis, and interpretation of health-related data needed for the planning, implementation, and evaluation of public health practice' [28]. This continuous monitoring of the health status of the population is strategically carried out to detect an outbreak in its early stages. This process plays an instrumental role in efforts to prevent potentially devastating outbreaks like the COVID-19 pandemic. Syndromic surveillance is assisted by data for real-time disease monitoring. It involves the use of statistical tools and analysis for detecting unusual health activity. This process is carried out to detect disease outbreak even before a diagnosis is made of a specific disease. Data for such surveillance is usually collected by health professionals during consultations. Hence it is mostly based on symptoms shown by patients. This data is then fed to surveillance systems for analysis [29]. Large organisations like the Centre for Disease Control and Prevention (CDC) have systems for monitoring diseases such as SARS and Influenza. However digital methods for public surveillance relies heavily on queries made on search engines and social media feeds. An example of such digital methods includes Google Flu Trends which provided estimates of influenza infections through the use of web search query data analysis [30].

Google flu Trends was found to overestimate the number of infections as its estimates were double that of visits to the hospital. Lazer et al.(2014) [31] cited two reasons; big data hubris and algorithm dynamics. Data from Twitter is also considered highly useful for health surveillance for many reasons. For one, The richness of the content of tweets allows for useful data to be collected. Also, Twitter's data access policies allow for open access to tweets through the Twitter API. It is also useful for real-time monitoring of contagious diseases because certain keywords can be monitored. Also, hashtags make it easy to group and monitor topics of interest. Tweets were analysed during the 2009 H1N1 outbreak to learn about the opinions and experiences of people [32], Twitter is also used as a low-cost alternative for researchers to track and detect trends in public health concerns over a period of time. As a result, Twitter is the most widely utilized social media site for monitoring public health [29]. Parker et al.(2013) [33] used frequent terms sets from health-related tweets as queries for text mining and information retrieval from Wikipedia articles to determine which articles are frequently accessed. The changes in the terms sets and articles give an indication of what the health concerns of the public are.

Chapter 4

Solution Approach

In this chapter, we present the experimental setup, describe the dataset and processing methods. We also describe our implementation of the BERT model and tokenization of input data. Finally, we train and test our model. The code for this project is available on Github [1].

4.1 Experimental Setup and Data Set

We consider two data sources for this project. Due to the scarcity of relevant biomedical information on Twitter on a disease during the early stages of an outbreak, we consider training our model with a combination of biomedical literature abstracts and tweets. To determine which approach helps us retrieve useful information from Twitter through classification.

4.1.1 Twitter Data

Our tweets are collected based on certain keywords and hashtags alluding to COVID-19 pandemic. Examples of these keywords are pandemic, coronavirus, and lockdown. Similarly, examples of hashtags used to retrieve tweets from the Twitter Application Programming Interface (API) were *#pandemic* and *#lockdown*. The data is put together by a related project on tracking social media discourse about the COVID-19 pandemic [20]. The data consists of tweet_ids from January 2020 to date. The tweet_ids are then used to access the tweet data through a process called hydration. The python library, twarc [34] is used for this.

[1] https://github.com/samOpare/covid-twitter-classification

```
def hydrator(id_file):
    print('hydrating {}'.format(id_file))

    gzip_path = id_file.with_suffix('.jsonl.gz')
    if gzip_path.is_file():
        print('skipping json file already exists: {}'.format(gzip_path))
        return

    num_ids = raw_newline_count(id_file)

    with gzip.open(gzip_path, 'w') as output:
        with tqdm(total=num_ids) as pbar:
            for tweet in twarc.hydrate(id_file.open()):
                output.write(json.dumps(tweet).encode('utf8') + b"\n")
                pbar.update(1)
```

Listing 4.1: Function for hydrating tweets with twarc

Hydrating the tweets is time consuming since Twitter limits the hydration to 900 requests with 100 tweet ids in a 15-minute window per user. Therefore, we are able to hydrate about 7 million tweets per day. It took about 3 months to completely download tweets from 6 months of 2020 mainly due to network interruptions.

Data Processing

The tweets are stored in a json format. The json formatted files contain fields such as created_at, id, full_text, hashtags, friends_count and user_mentions for each tweet. Our project is concerned with processing the tweet text itself, so we process the json file to obtain our tweets. The tweets are located in the key fields full_text or retweet_status of our hydrated json files. We check to see if one of the two fields exists. We then append our tweet to our list of tweets. Before we append the tweet, we make sure it is English by calling the function, isEnglish, which checks that the tweet is not in a different language.

```
# determines if text is english or not
def isEnglish(s):
    try:
        s.encode('utf-8').decode('ascii')
    except UnicodeDecodeError:
        return False
    else:
        return True
```

Listing 4.2: Function to filter english terms

Tweets are very unstructured and contain very unique characters which do not play any important role in the NLP task. We perform some pre-processing of the tweets before

passing them to the model for classification. Twitter-specific pre-processing enables us to identify and remove URLs, hashtags, RT and mentions. We also perform stemming, lemmatization, and the removal of stop words from our tweets. We perform the Twitter-specific pre-processing using the library tweet-preprocessor [35]. We also use NLTK [4] and genism[36] library for the removal of stopwords, stemming, and lemmatization.

Limitations

Though many tweets were collected, we observed after preliminary analysing that many of these tweets were non-English. While another percentage of the tweets were unrelated to health, which is our focus in this project. Most of the tweets were political in nature and had to be removed from our list of tweets. We attempted removing all non-English tweets using a function, but this worked partially because it is based on ASCII characters. Tweets in languages like Chinese which use special characters not found in the English alphabet were successfully removed. Many other languages that write using English based alphabets were not removed.

Labelling the Tweets

For our classification model to perform successfully we label our data since this is a supervised learning task. Data labelling is done manually by reading through individual tweets to determine which category they fall under. We have 2000 labelled tweets in total with 5 categories. Our categories are symptom, treatment, prevention, transmission, and health risk. A sample of tweets that fall under each category is shown in Table 4.1 below.

Tweet	Class
New data from a preliminary trial suggests that the drug remdesivir has a positive effect on coronavirus recovery. "Everything that we've seen so far gives a lot of optimism and hope," says Dr. Babafemi Taiwo, who is leading part of the study on the drug. https://t.co/ZpOyuVK0Fs https://t.co/jxrtMxYmiO	treatment
3/4 Case reports suggest that this virus is highly contagious. Infections possible ... * in the incubation phase * from asymptotic patients * in a very short period of time #coronavirus	transmission
hi, public health student chiming in with some friendly reminders during this coronavirus outbreak - WASH YOUR HANDS - carry around hand sanitizer - practice the vampire sneeze - wear surgical/cloth masks - did i mention to wash your hands?	prevention
I literally got so sick I would wake up every night choking and not able to breathe for like 2 weeks https://t.co/AbMq7iUkpf	symptom
This includes the elderly & people with pre-existing medical conditions (such as asthma, diabetes, heart disease). Stay safe stay home.	health risk

Table 4.1: Sample tweet labels

4.1.2 Biomedical Literature

We use biomedical literature abstracts as alternative data for training our model in the absence of relevant tweets. Our csv formatted file consists of the columns 'title', 'journal', 'abstract', 'authors' ,'doi' ,'publish_time' ,'sha' and 'full_text_file'. It also has 414,019 rows.

Processing the Data

We prepare the data for the task by dropping all duplicates and filling missing fields in the title and abstract with 'no text available'. We observe that not all these papers are directly related to covid, hence, we write a function that looks through the title and abstract to find papers that contain words such as covid, -cov-2, ncov and corona. This reduces our data to 70,814 rows. We then perform further processing such as lemmatization, stemming and stop word removal.

```
#reduce the dataset to literature discussing covid specific topics
def find_covid_lit(df):
    df1 = df[df['abstract'].str.contains('ncov')]
    df2 = df[df['abstract'].str.contains('corona')]
    df3 = df[df['abstract'].str.contains('covid')]
    df4 = df[df['abstract'].str.contains('-cov-2')]
    df5 = df[df['abstract'].str.contains('cov2')]


    data =[df1,df2,df3,df4,df5]
    df = pd.concat(data)
    df=df.drop_duplicates(subset='title', keep="first")
    return df
```

Listing 4.3: Selecting corona specific abstracts

Labelling the Data

The data is labelled programmatically through the use of search terms in the abstract and title. Since some of these papers deal specifically with symptoms or transmission or prevention among others, we are able to label them by searching through the title and the abstract for relevant keywords for each category. The data is then labelled as such. After this, we have 4046 labelled data in total. With 2464 labelled symptoms, 1149 labelled as treatment, 239 labelled as transmission, 189 as prevention and 8 as health risk. Due to the highly uneven number of records per class, we compute weights that are used in training the BERT model and also perform some scaling before using our baseline model, logistic regression.

4.2 Design and Implementation

In this section, we discuss the fine-tuning of a BERT model and use it for our classification. BERT has been proved to produce cutting-edge results. Due to it being pre-trained, it is considered to perform well even when trained with little data.

4.2.1 Defining the Model Architecture

In implementing the BERT model, we use the keras API [37] with a tensorflow [38] backend. Keras is a library that simplifies the implementation of deep learning models. It is a much simpler alternative to tensorflow which has a steeper learning curve.
To fine-tuning the BERT model to classify tweets to elicit information about a disease

outbreak, we use the base BERT model which has 12 transformer layers, 12 attention heads and 110 million parameters. We then modify it with two more dense layers and an output layer for training and fine-tuning the BERT model with features from our training data.

4.2.2 Tokenizing the Input Data

As mentioned in Section 2.5 about BERT, the model requires input in a specific format made of three different embeddings. We write a function to encode the data in this format. Each row of data is loaded into the *bert_encoder* function where the string is split based on the dot symbol. We then tokenize each sentence using the BERT *tokenization* class method, *tokenize*. The output of tokenization is shown below in Figure 4.1.

```
['by', 'defining', 'the', 'three', 'metadata', 'token', '##s', 'created', 'from', 'the', 'input', 'data', '.', 'token
', 'em', '##bed', '##ding', '##s', ',', 'position', 'em', '##bed', '##ding', '##s', 'and', 'sequence', 'em', '##bed',
'##ding', '##s']
```

Figure 4.1: Result of tokenizing input sequence

The tokenize function returns tokens of words with suffixes and prefixes separated and represented as tokens starting with a double hash symbol. We then add the [CLS] and [SEP] tokens to signify the beginning and end of each sequence as previously described in Section **??**, on pre-training a BERT model. The *convert_tokens_to_ids* function converts the tokens into numeric ids which can be passed on to the model for training. We then perform padding of the ids with zeros based on the *max_len* parameter specified. The zeros indicate to the model that these inputs can be ignored during training. For example, we specify the maximum length of the input sequence as 512. Therefore, we add a padding of 0s to the sequence for sequences with length lesser than 512. The function returns 3 arrays. These are the input embeddings used in fine-tuning the BERT model.

```
def bert_encoder(lines, tokenizer, max_len=250):
    token_embeddings = []
    mask_embeddings = []
    segment_embeddings = []

    #tokenizing input sequences
    for sentence in lines:
        sentence = sentence.split('.')

        for line in sentence:
            line = tokenizer.tokenize(line)

            line = line[:max_len-2]
            input_seq = ["[CLS]"] + line + ["[SEP]"]
            pad_len = max_len - len(input_seq)
```

```
            tokens = tokenizer.convert_tokens_to_ids(input_seq) + [0] * pad_len
            pad_masks = [1] * len(input_seq) + [0] * pad_len
            segments_id = [0] * max_len

            token_embeddings.append(tokens)
            mask_embeddings.append(pad_masks)
            segment_embeddings.append(segments_id)

    return np.array(token_embeddings), np.array(mask_embeddings), np.array(segment_embeddings)
```

Listing 4.4: The bert_encoder function

4.2.3 The Model Architecture

The model can be modified by adding one or more layers to it depending on the use case at hand. We attach three dense layers. With one serving as an output layer. Dense layers are layers with fully connected nodes. To curb and prevent overfitting of the data during training, we include drop out layers between the dense layers. We apply the Rectified Linear Unit (ReLU) activation function to each node of the layers to map their outputs to a value between 0 and infinity. A value of 0 means this node will not be activated. Any other value activates this node as described in Section 2.3. For our output layer, the softmax activation function is used. The softmax activation function returns a number between 0 and 1 which specifies the probability of each tweet belonging to one of the five classes. For this reason, we have five output nodes in the output layer. Each node, belonging to one specific class and outputs a value which is the probability of the observation belonging to that class. The text sequence is predicted to belong to the class which yields the highest probability.

There is no way of determining how many nodes each layer within the hidden layers should have except through trial and error. We settled on using 64 nodes and 32 nodes on the first and second layers attached.

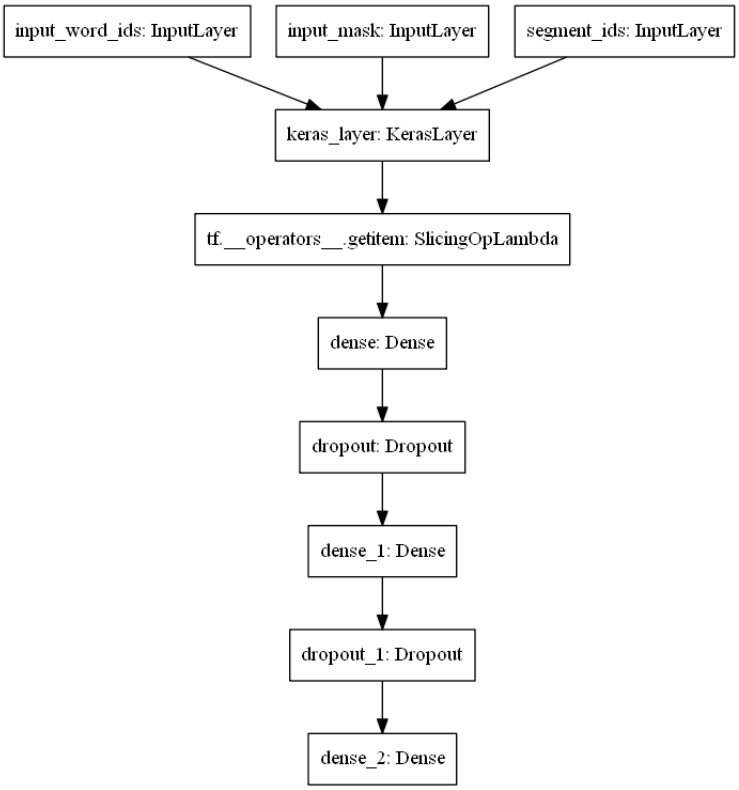


Figure 4.2: The model Architecture

Before training the model, we need to specify other parameters. We do this via the compile method of the model. We specify the loss function to minimize the error during training. For this task, we use the categorical cross-entropy loss function. We also specify an optimization function, adamW [39]. Optimization is discussed above in the portion regarding gradient descent. We also specify a metric for monitoring the training process. We select accuracy.

```
def build_bert_model(bert_layer, max_len=512):
    input_word_ids = tf.keras.Input(shape=(max_len,), dtype=tf.int32, name="input_word_ids")
    input_mask = tf.keras.Input(shape=(max_len,), dtype=tf.int32, name="input_mask")
    segment_ids = tf.keras.Input(shape=(max_len,), dtype=tf.int32, name="segment_ids")

    _, sequence_output = bert_layer([input_word_ids, input_mask, segment_ids])

    clf_output = sequence_output[:, 0, :]
    layer = tf.keras.layers.Dense(64, activation='relu', kernel_regularizer='l1')(clf_output)
    layer = tf.keras.layers.Dropout(0.5)(layer)
    layer = tf.keras.layers.Dense(32, activation='relu', kernel_regularizer='l1')(layer)
    layer = tf.keras.layers.Dropout(0.5)(layer)
    out = tf.keras.layers.Dense(5, activation='softmax')(layer)

    model = tf.keras.models.Model(inputs=[input_word_ids, input_mask, segment_ids], outputs=out)
    model.compile(tf.keras.optimizers.Adam(lr=1e-5), loss='categorical_crossentropy', metrics=['accurac
```

```
    return model
```

Listing 4.5: Function for creating the model

4.2.4 Training the Model

Training the model involves feeding it with data that consists of text sequences along with their labels. The model learns features or patterns from the data and can associate certain features to specific labels. We then test the performance of the model by testing its ability to predict the class of new data points accurately. Dropouts are attached to each layer to check for overfitting. The dropout layer randomly sets some of the values of the output vector from a layer to 0. The dropout rate can be set to 0.2 or 0.5. During testing, output values are scaled down to account for this dropping out which is not applied during testing. We also compute and apply class weights during training to account for the class imbalance.

Chapter 5

Result Evaluation

We perform a test by training the BERT model with various data sets to determine how well it performs in each circumstance. We also compare the results of the BERT model to a logistic regression model to see which one performs better under these circumstances. We train the two models with only tweets; we also train them with only biomedical literature abstracts and finally train them with a combination of both biomedical literature and tweets.

Table 5.1 shows the results of labelling the test data, the BERT model performs less satisfactorily compared to the baseline model. The logistic regression model outperforms the BERT model in two out of three tests. The accuracy of the BERT model when trained with Twitter data alone is 0.21 while that of the logistic regression is 0.64. When trained with only biomedical literature the accuracy of the BERT predictions is 0.28 while that of the logistic regression model is 0.20. Finally, when trained with a combination of the two datasets, the accuracy of the BERT model is 0.30 while that of the logistic regression model is 0.58.

The BERT model performances moderately when trained with the biomedical literature alone. On the other hand, the logistic regression model performed the worst in this scenario. This may be because the amount of data available for training with biomedical literature was significantly more than the quantity of data available for training with tweets alone. Deep learning models tend to require very large amounts of data for training to avoid overfitting and perform well. The logistic regression model may have performed worse in this case because of the difference in the language used on Twitter and language used in writing biomedical literature. Tweets are normally informal in nature while biomedical literature is very formal and contains terms that are technical and directed towards a niche group of people who understand these terms. As a result, in training, the logistic regression model would encounter words which it would not encounter in the

Model	Training Data	Class	F1	Recall	Precision	Accuracy
Logistic regression	tweets	health risk	0.00	0.00	0.00	
		prevention	0.75	0.79	0.71	
		symptom	0.29	0.20	0.50	
		transmission	0.61	0.72	0.52	
		treatment	0.63	0.50	0.85	
						0.64
	biomedical lit.	health risk	0.00	0.00	0.00	
		prevention	0.19	0.11	0.75	
		symptom	0.21	0.96	0.12	
		transmission	0.23	0.14	0.64	
		treatment	0.18	0.12	0.36	
						0.20
	combined	health risk	0.27	0.18	0.50	
		prevention	0.76	0.67	0.87	
		symptom	0.20	0.28	0.16	
		transmission	0.60	0.72	0.51	
		treatment	0.48	0.38	0.65	
						0.58
BERT	tweets	health risk	0.21	0.17	0.29	
		prevention	0.08	0.20	0.05	
		symptom	0.29	0.24	0.36	
		transmission	0.05	0.03	0.14	
		treatment	0.09	0.18	0.06	
						0.21
	biomedical lit.	health risk	0.34	0.30	0.39	
		prevention	0.25	0.43	0.18	
		symptom	0.31	0.33	0.28	
		transmission	0.00	0.00	0.00	
		treatment	0.00	0.00	0.00	
						0.28
	combined	health risk	0.50	0.42	0.61	
		prevention	0.18	0.80	0.10	
		symptom	0.02	0.01	0.33	
		transmission	0.00	0.00	0.00	
		treatment	0.00	0.00	0.00	
						0.30

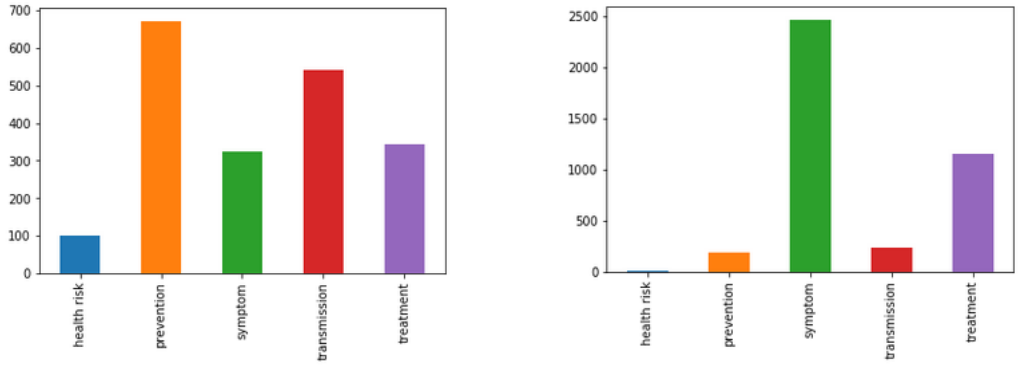
Table 5.1: Table of results

test set of tweets. This may be the cause of its poor performance. This however shows the BERT model tends to generalise much better with different tasks.

The BERT model performs worse when trained with only Twitter data while the logistic regression model obtains its best result. This is because unlike in training with biomedical literature alone, in training with tweets the logistic regression model encounters words that it also encounters during testing. However, the BERT model does not perform very well again due to the quantity of tweets available for training. This may have resulted

in overfitting of the model. Training with a combination of both datasets results in a moderate performance in the logistic regression model and the best performance from the BERT model. This shows that both models do much better with a combination of datasets, especially, the BERT model. This may be due to the increase in the quantity of data for training as well as its ability to be fine-tuned.

In general, the BERT model performed much worse than expected. The major contributing factor for this may be the insufficient amount of data available for training. Deep learning models require substantial amounts of data for training, however, due to time constraints we were unable to label more data. This is because to correctly label the data it requires attention and time. Figure 5.1 shows the distribution of our data.



(a) (b)

Figure 5.1: Bar plot showing the distribution of our data. (a) represents tweets and (b) is biomedical literature abstracts

We also notice that, after training with tweets only, BERT had its best precision score in predicting health risk and symptoms. With values of 0.29 and 0.36, respectively. It also shows the same trend with the F1 and Recall values. This is withstanding the fact that these classes have the fewest observations. On the other hand, the baseline model shows a higher F1-score for the classes with the most data. That is, prevention, transmission, and treatment. The F1-score shows that these classes were predicted with high accuracy. The precision score of treatment was very high at 0.85. but had an F1-score of 0.63 which shows that many observations were falsely classified to it. The class prevention had consistently high scores across all three metrics particularly because of its quantity. Tweets about prevention were in large quantities because people kept tweeting about staying home, wash hands and other preventive measures. Very little information was being tweeted about things like transmission or treatment because it is usually professionals that have this information.

When trained with only biomedical literature, the BERT model does a better job of predicting the classes accurately. The base model falsely predicts most of the classes to

belong to prevention and transmission from a comparison of the precision and F1-scores. What may have contributed to the poor performance is the quality of the data and the labels. Upon training with combined data, the base model does a good job of accurately predicting prevention and transmission with F1-scores of 0.76 and 0.60. These classes have the most data. The BERT model is more accurate on the least populated category; health risk.

We also notice that our data overfits especially when we train with only tweets. Figure 5.2. shows a plot of train and validation accuracy and loss. From the figure, we notice the accuracy increases with each epoch of training. The loss also reduces in both the train and validation. However, when we test the model on our test data, we see an opposite effect. This is a sign of overfitting. This was the trend during training with the three different data sets.

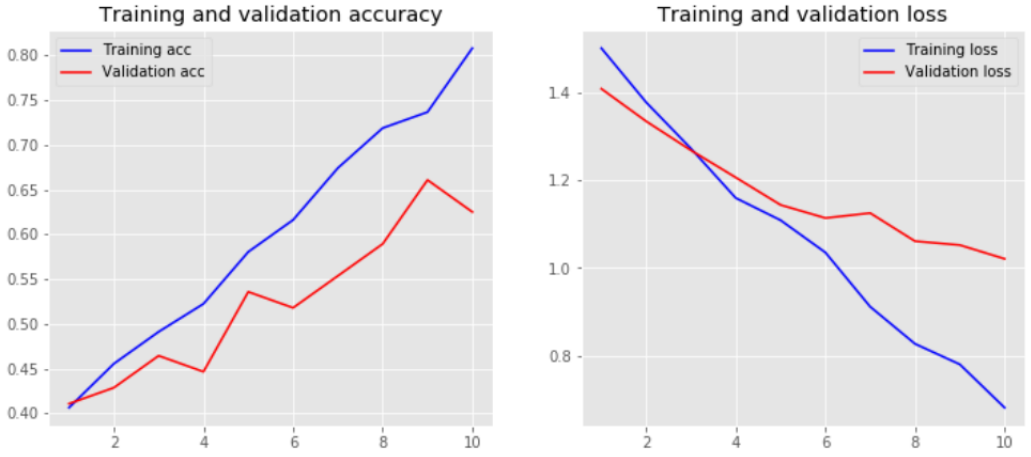


Figure 5.2: Plot of training and validation accuracy and loss

We try to remedy this by implementing early stopping so that the training stops earlier if there is no increase in the accuracy for a number of epochs. We also vary the parameters mentioned in section 2.3 on overfitting, but the model still performs poorly. Due to time constraints, we were unable to label substantial amounts of data to carry on with the experiments.

Chapter 6

Conclusion and Future Directions

The theoretical part of this thesis covers some basic concepts of NLP such as language models and data processing. It also covers some advanced concepts such as Artificial Neural Networks, Transformers and BERT. We also discussed the process of pre-training a BERT model using Masked Language Model and Next Sentence Prediction [15].

We implemented a BERT model to aid in public health surveillance using Twitter data. We collected and hydrated tweets related to COVID-19 from the Twitter API. We group the data into 5 categories (symptom, prevention, health risk, treatment, and transmission) by labelling them. We also labelled biomedical literature abstracts of papers related to COVID-19. Using these datasets, we attempted to fine-tune and train the BERT model to classify tweets into these categories. We fine-tuned the BERT model by adding 2 hidden layers and apply ReLU activation functions and an output layer with a softmax activation function. To counter overfitting, we attached a dropout layer to each of the hidden layers. We also implemented the tokenization and encoding of the input for BERT by applying the [SEP] and [CLS] tokens.

We trained our model with tweets only, biomedical literature abstracts only, and a combination of biomedical literature and tweets to determine how well the model performs in each circumstance. We also perform these tests using, a logistic regression model as our baseline model. We observed that the BERT model does not perform as well as expected. Compared to the logistic regression model and given the same quantity of training data, the BERT model performed poorly. This emphasizes the importance of having a vast amount of data while training a deep learning model. Even if BERT has been pre-trained, fine-tuning still necessitates a considerable amount of data.

Training our models with biomedical literature abstracts alone yielded bad results in both models but was worse in the baseline model. Training with a combination of tweets and biomedical literature abstracts leads to improved results in both models.

In future works, more attention has to be taken to deal with overfitting. Some enhancements that can be made to improve performance include:

- Freezing the BERT layer when fine-tuning the model. So that the BERT layer is not retrained. Only the additional layers attached should be trained.
- Training the BERT model with a substantial amount of data will improve its performance and prevent overfitting.
- Adding more layers and nodes when building the model.
- When encoding the data split longer texts into smaller text at the full stop before adding the [CLS] and [SEP] tokens.

List of Figures

List of Tables

Bibliography

[1] Nelson Aguirre-Duarte. Can people with asymptomatic or pre-symptomatic covid-19 infect others: a systematic review of primary data. page 2020.04.08.20054023, 2020. doi: 10.1101/2020.04.08.20054023. URL https://www.medrxiv.org/content/medrxiv/early/2020/04/16/2020.04.08.20054023.full.pdf.

[2] Aparup Khatua, Apalak Khatua, and Erik Cambria. A tale of two epidemics: Contextual word2vec for classifying twitter streams during outbreaks. *Information Processing and Management*, 56(1):247–257, 2019. ISSN 03064573. doi: 10.1016/j.ipm.2018.10.010.

[3] Guan yong Shi Na, Liu Xumin. Research on k-means clustering algorithm, 2010.

[4] 2021. URL https://www.nltk.org/#natural-language-toolkit.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[6] Chao-Ying Joanne Peng, Kuk Lida Lee, and Gary M. Ingersoll. An introduction to logistic regression analysis and reporting. *The Journal of Educational Research*, 96 (1):3–14, 2002. doi: 10.1080/00220670209598786.

[7] ROSENBLATT F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958. doi: https://doi.org/10.1037/h0042519.

[8] A. Krizhevsky, I. Sutskever, and E. Hinton. Imagenet classification with deep convolutional neural networks. 2012.

[9] Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels, 2018.

[10] Christopher Olah. Understanding lstm networks, August 27, 2015 2015. URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[11] Chase Brandon, Michael Holloway, and Micah Silberstein. Reading comprehension on squad usingtensorflow.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. attention is all you need. 2017.

[13] Bodacious Blog. Rewrite of 'understanding transformers in nlp: State-of-the-art models', 2020. URL https://mullikine.github.io/posts/review-of-understanding-transformers-in-nlp-state-of-the-art-models/.

[14] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[15] Mohd Sanad Zaki Rizvi. Demystifying bert: A comprehensive guide to the groundbreaking framework, September 25, 2019 2019. URL https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/.

[16] M. Myslin, S. H. Zhu, W. Chapman, and M. Conway. Using twitter to examine smoking behavior and perceptions of emerging tobacco products. *J Med Internet Res*, 15(8):e174, 2013. ISSN 1438-8871 (Electronic) 1438-8871 (Linking). doi: 10.2196/jmir.2534. URL https://www.ncbi.nlm.nih.gov/pubmed/23989137.

[17] Thomas G. Dietterich. Ensemble methods in machine learning. 2000.

[18] Omer Levy Yoav Goldberg. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. 2014.

[19] Cody Buntain and Jennifer Golbeck. Automatically identifying fake news in popular twitter threads, 2018.

[20] Emily Chen, Kristina Lerman, and Emilio Ferrara. Tracking social media discourse about the covid-19 pandemic: Development of a public coronavirus twitter data set. *JMIR Public Health Surveill*, 6(2):e19273, May 2020. ISSN 2369-2960. doi: 10.2196/19273. URL http://publichealth.jmir.org/2020/2/e19273/.

[21] S. Wakamiya, Y. Kawai, and E. Aramaki. Twitter-based influenza detection after flu peak via tweets with indirect information: Text mining study. *JMIR Public Health Surveill*, 4(3):e65, 2018. ISSN 2369-2960 (Print) 2369-2960 (Linking). doi: 10.2196/publichealth.8627. URL https://www.ncbi.nlm.nih.gov/pubmed/30274968.

[22] W. G. Dos Santos. Natural history of covid-19 and current knowledge on treatment therapeutic options. *Biomed Pharmacother*, 129:110493, 2020. ISSN 1950-6007 (Electronic) 0753-3322 (Linking). doi: 10.1016/j.biopha.2020.110493. URL https://www.ncbi.nlm.nih.gov/pubmed/32768971.

[23] K. G. Andersen, A. Rambaut, W. I. Lipkin, E. C. Holmes, and R. F. Garry. The proximal origin of sars-cov-2. *Nat Med*, 26(4):450–452, 2020. ISSN 1546-170X (Electronic) 1078-8956 (Linking). doi: 10.1038/s41591-020-0820-9. URL https://www.ncbi.nlm.nih.gov/pubmed/32284615.

[24] N. Zhu, D. Zhang, W. Wang, X. Li, B. Yang, J. Song, X. Zhao, B. Huang, W. Shi, R. Lu, P. Niu, F. Zhan, X. Ma, D. Wang, W. Xu, G. Wu, G. F. Gao, W. Tan, Investigating China Novel Coronavirus, and Team Research. A novel coronavirus from patients with pneumonia in china, 2019. *N Engl J Med*, 382(8):727–733, 2020. ISSN 1533-4406 (Electronic) 0028-4793 (Linking). doi: 10.1056/NEJMoa2001017. URL https://www.ncbi.nlm.nih.gov/pubmed/31978945.

[25] K. Shirato, M. Kawase, and S. Matsuyama. Wild-type human coronaviruses prefer cell-surface tmprss2 to endosomal cathepsins for cell entry. *Virology*, 517:9–15, 2018. ISSN 1096-0341 (Electronic) 0042-6822 (Linking). doi: 10.1016/j.virol.2017.11.012. URL https://www.ncbi.nlm.nih.gov/pubmed/29217279.

[26] Franck Touret, Magali Gilles, Karine Barral, Antoine Nougairède, Etienne Decroly, Xavier de Lamballerie, and Bruno Coutard. 2020. doi: 10.1101/2020.04.03.023846.

[27] 2019. URL https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge.

[28] World Health Organization. public health surveillance, 2021. URL https://www.who.int/immunization/monitoring_surveillance/burden/vpd/en/.

[29] Allison E. Aiello, Audrey Renson, and Paul N. Zivich. Social media– and internet-based disease surveillance for public health. 41(1):101–118, 2020. doi: 10.1146/annurev-publhealth-040119-094402. URL https://www.annualreviews.org/doi/abs/10.1146/annurev-publhealth-040119-094402.

[30] E. H. Chan, V. Sahai, C. Conrad, and J. S. Brownstein. Using web search query data to monitor dengue epidemics: a new model for neglected tropical disease surveillance. *PLoS Negl Trop Dis*, 5(5):e1206, 2011. ISSN 1935-2735 (Electronic) 1935-2727 (Linking). doi: 10.1371/journal.pntd.0001206. URL https://www.ncbi.nlm.nih.gov/pubmed/21647308.

[31] D. Lazer, R. Kennedy, G. King, and A. Vespignani. The parable of google flu: Traps in big data analysis. 2014. ISSN 1203-1205.

[32] C. Chew and G. Eysenbach. Pandemics in the age of twitter: content analysis of tweets during the 2009 h1n1 outbreak. *PLoS One*, 5(11):e14118, 2010. ISSN 1932-6203 (Electronic) 1932-6203 (Linking). doi: 10.1371/journal.pone.0014118. URL https://www.ncbi.nlm.nih.gov/pubmed/21124761.

[33] J. Parker, Y. Wei, A. Yates, O. Frieder, and N. Goharian. A framework for detecting public health trends with twitter. 2013. doi: https://doi.org/10.1145/2492517.2492544.

[34] 2021. URL https://twarc-project.readthedocs.io/en/latest/twarc2/.

[35] 2021. URL https://pypi.org/project/tweet-preprocessor/.

[36] 2009. URL https://radimrehurek.com/gensim/parsing/preprocessing.html.

[37] keras, 2021. URL https://keras.io/.

[38] tensorflow, 2021. URL https://www.tensorflow.org/.

[39] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.