

Fall 2022

Analysis Of Public Sentiment of Covid-19 Pandemic, Vaccines, And Lockdowns

Devinesh Singh
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects



Part of the [Artificial Intelligence and Robotics Commons](#), and the [Other Computer Sciences Commons](#)

Recommended Citation

Singh, Devinesh, "Analysis Of Public Sentiment of Covid-19 Pandemic, Vaccines, And Lockdowns" (2022).
Master's Projects. 1107.

DOI: <https://doi.org/10.31979/etd.42du-5923>

https://scholarworks.sjsu.edu/etd_projects/1107

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Analysis Of Public Sentiment of Covid-19 Pandemic, Vaccines, And Lockdowns

A PROJECT

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By Devinesh Singh

December 2022

The Designated Project Committee Approves on the Project Titled
Analysis Of Public Sentiment of Covid-19 Pandemic, Vaccines, And Lockdowns

by
Devinesh Singh

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

December 2022

Dr. Fabio Di Troia

Department of Computer Science

Dr. Genya Ishigaki

Department of Computer Science

Dr. Faranak Abri

Department of Computer Science

ABSTRACT

CoV-2 pandemic prompted lockdown measures to be implemented worldwide; these directives were implemented nationwide to stunt the spread of the infection. Throughout the lockdowns, millions of individuals resorted to social media for entertainment, communicate with friends and family, and express their opinions about the pandemic. Simultaneously, social media aided in the dissemination of misinformation, which has proven to be a threat to global health. Sentiment analysis, a technique used to analyze textual data, can be used to gain an overview of public opinion behind CoV-2 from Twitter and TikTok. The primary focus of the project is to build a deep learning classifier to analyze user sentiment on TikTok. Several deep learning models were developed, including Convolutional Neural Networks (CNN), Long-Short Term Memory (LSTM), Attention Mechanism, and Bidirectional Encoder Representations from Transformer (BERT). CNN excels at local feature extraction, whereas LSTM can store sequential data without loss of information. BERT can overcome the issue of ambiguous sentences and phrases; specifically, it can differentiate between homonyms. Models were trained on Sentiment140, a Twitter dataset. Once these models were trained, the models with the best performance were then used to classify sentiment of the TikTok users from Mar 2020 to August 2021. Proposed models can be used by both government institutions and businesses to understand concerns surrounding the pandemic.

Keywords: COVID-19, LSTM, CNN, BERT, Sentiment Analysis, TikTok

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	RESEARCH OBJECTIVE AND MOTIVATION	3
III.	RELATED WORKS	4
IV.	HISTORY AND BACKGROUND	6
	4.1 WORD EMBEDDINGS	6
	<i>4.1.1 Word2Vec</i>	<i>7</i>
	i. Continuous Bag of Words	7
	ii. Skip-Grams	8
	<i>4.1.2 Glove2Vec</i>	<i>8</i>
	<i>4.1.3 BERT</i>	<i>9</i>
V.	DATASET AND DATASET PREPARATION.....	11
	5.1 DATASET A – SENTIMENT140	11
	5.1.1 DATASET A PREPROCESSING	11
	5.2 DATASET B – TIKTOK DATASET	13
	5.2.1 DATASET B PREPROCESSING	14
VI.	DEEP LEARNING (DL) MODELS	16
	6.1 CONVOLUTIONAL NEURAL NETWORKS	16
	<i>6.1.1 Convolutional Layers</i>	<i>16</i>
	<i>6.1.2 Pooling Layer</i>	<i>17</i>
	<i>6.1.3 Fully Connected Layer</i>	<i>17</i>
	<i>6.1.4 Dropout Layer</i>	<i>18</i>
	6.2 LONG SHORT-TERM MEMORY	18
	<i>6.2.1 Forget Gate</i>	<i>19</i>
	<i>6.2.2 Input Gate</i>	<i>19</i>
	<i>6.2.3 Output Gate</i>	<i>19</i>
	<i>6.2.4 Sigmoid Function</i>	<i>20</i>

6.2.5 <i>Hyperbolic Tangent (Tanh) Activation Function</i>	20
6.3 CNN-LSTM.....	21
6.4 CNN-LSTM WITH ATTENTION MECHANISM	22
6.5 BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT).....	22
VII. PROPOSED MODELS.....	26
7.1 NEURAL NETWORKS WITH PRE-TRAINED GLOVE EMBEDDINGS.....	26
7.2 NEURAL NETWORKS WITH PRE-TRAINED WORD2VEC EMBEDDINGS	26
7.3 NEURAL NETWORKS WITHOUT PRE-TRAINED WORD EMBEDDINGS	26
7.4 BERT TRANSFORMER.....	27
VIII. EXPERIMENTS AND MODEL RESULTS.....	28
8.1 EXPERIMENT ENVIRONMENT.....	28
8.1.1 <i>Hyperparameters for Directional Models</i>	28
8.1.2 <i>Hyperparameters for BERT Model</i>	28
8.2 MODEL TRAINING.....	29
8.3 RESULTS	29
IX. RESULTS ON TEST DATASET.....	33
X. CONCLUSION.....	35
REFERENCES.....	36

LIST OF TABLES

Table 1: Average Runtime and Accuracy Scores of Different Embeddings 29

Table 2: Average Runtime and Accuracy Scores of Different NN Models 30

Table 3: Neural Network Models trained with GloVe Results 30

Table 4: Neural Networks trained with Word2Vec Results 31

Table 5: Neural Networks trained with Embedding Layer Results 32

Table 6: Average Test Accuracy of Different Embeddings..... 33

Table 7: Neural Networks trained with Embedding Layer Results 34

Table 8: Neural Networks trained with Word2Vec Embeddings Results 34

Table 9: Neural Networks trained with GloVe Embeddings Results 34

LIST OF FIGURES

Figure 1: Comparison of CBOW and Skip-Gram Architectures [8]	8
Figure 2: Architecture of Glove2Vec [9].....	9
Figure 3: BERT Embeddings Architecture [24]	10
Figure 4: Raw Sentiment140 Dataset	11
Figure 5: Data Cleaning Process.....	13
Figure 6: Raw Emotions Towards Covid-19 on Reddit Dataset.....	14
Figure 7: 5x5 Image and Filter with Resulting Output Array [10].....	17
Figure 8: Max Pooling vs Average Pooling [11]	17
Figure 9: Schematic Diagram of a Basic CNN Architecture [12]	18
Figure 10: Neural Network without Dropout vs Neural Network with Dropout [13]	18
Figure 11: Schematic Diagram of LSTM Architecture [14].....	20
Figure 12:Schematic Diagram of CNN-LSTM Architecture [15].....	21
Figure 13: BERT Input Data Transformation.....	23
Figure 15: LSTM without Pre-trained Embeddings Training Curve.....	31

I. INTRODUCTION

CoV-2 has upended most aspects of daily life; with nationwide lockdowns imposed, people across the United States have had to adjust to new ways of life, work, and school. Although these preventive measures aimed to contain the spread of the infection, they caused a detrimental effect on public mental health. Several studies conducted worldwide have shown the negative impact that lockdowns had on an individual's psychological well-being [1].

Governments worldwide implemented some form of social distancing to comply with public health sectors and the World Health Organization (WHO) and combat the spread of the infection; however, many reports have shown a high non-compliance rate to social distancing. Other studies have identified differences in opinions between compliant people and those who are not compliant. These studies rely on studying public sentiment surrounding CoV-2 and government preventive measures to understand the reasoning behind an individual's choice to comply or not to comply [2].

Researchers often resort to social media as a source to understand public opinion on various aspects of the environment in which people live, especially concerning controversial issues that people routinely face. Social media makes up a large part of everyone's life; platforms like Twitter, Instagram, and TikTok are commonplace for people worldwide to communicate, share and express opinions on every topic. With over a billion posts every day, businesses and government agencies can effectively utilize social media platforms and user posts to address business problems and understand product concerns and challenges the public face.

Due to the massive volume and diversity of the information available through social media applications, businesses cannot manually process and analyze the data; instead, businesses seek automated tools like machine learning to help process and analyze the massive datasets. For

example, businesses use Artificial Neural Networks (NNs), a technique that mimics the neural structure of the human brain, to learn about the input values iteratively and to classify social media posts by sentiment as positive, negative, or neutral.

Several NNs can be used to classify input data: Convolutional Neural Network (CNN), Long Short-term Memory (LSTM), and a combination of the CNN and LSTM, among other NNs architectures; each algorithm has its advantages and disadvantages. Businesses must choose an algorithm that minimizes the misclassification rate while generalizing to non-training data. Performance metrics like accuracy score, recall, precision, and F1-Score could be used to compare the characteristics and performance of the algorithms. These metrics allow businesses to choose the algorithm that performs best through different experiments.

II. RESEARCH OBJECTIVE AND MOTIVATION

The proposed research paper will analyze sentiment emoted through the tweets and TikTok posts and comments since the beginning of the CoV-2 pandemic; the paper's objective is to answer the following questions – i) Can we develop a comprehensive classifier that can generalize to different input datasets? and ii) How do pre-trained models affect the performance of a classifier?

Social Media platforms – Twitter, Instagram, Reddit, and TikTok – became a primary source to share information in recent years. TikTok, a social media platform where people share short-form videos between 15 seconds and 10 minutes, attracts over a billion users monthly [25]; user comments can be extracted to analyze user's sentiment toward the pandemic and other topics.

III. RELATED WORKS

Social media has become the most influential virtual space to not only network with friends, family, and colleagues, but also for businesses to advertise their brands and products online; businesses can reach many people within minutes, which can help reduce costs. Additionally, businesses can harness user feedback and engagement on social media to determine interest and/or dissatisfaction with products and services; however, with billions of posts every day from multiple sources, businesses need automated tools to filter through the data.

Sentiment analysis can be performed using two approaches: lexicon-based analysis and machine learning-based analysis. Rule-based classification relies on human-crafted set of rules to determine sentiment; specifically, lexicons are compared to the pre-defined set of rules to determine user sentiment. Lexicons that do not appear in the limited vocabulary are labelled neutral [24]. Alternatively, machine learning-based classification learns to associate lexicons from patterns in the text automatically and predict future behavior using past behaviors.

[25] proposed a lexicon-based approach to classify documents. Hu et. al. achieved an accuracy of 72.9%; however, the proposed model did not account for pronoun resolution. Hoffman et. al. introduced phrase-level analysis that accounted for pronoun resolution; particularly, the researchers used WordNet, a database of semantic relations between words, to determine the subject of pronouns [26]. They achieved an accuracy of 81.5%, an 8% increase from Hu et. al. [27] extended previous works to apply lexicon-based approaches to Twitter. Proposed solution achieved an average score of 90% [28]. Gilbert et. al. proposed a rule-based model designed specifically for sentiment analysis on social media text. Valence Aware Dictionary for sEntiment Reasoning (VADER) combines qualitative analysis and empirical validation to compute sentiment scores; specifically, the combines a dictionary to map lexical

features to emotion intensity, and five heuristics to encode the text. VADER outperformed other classifiers with an accuracy of 96% [29].

Tripathi et. al. proposed classical machine learning techniques – logistic regression, decision trees, and random forest vector machines to classify Twitter data; it was found that decision tree performed best among the classical machine learning techniques with an accuracy of 88.51% [30]. Saad et. al proposed the use of regression and classical machine learning techniques – logistic regression, decision trees, and random forest on social media data; it was found that decision trees performed best, with an accuracy score of 71.39% [31].

Feizollah et. al. developed neural networks – CNN, RNN, and LSTM – to analyze Twitter data; it was found that a bidirectional LSTM outperformed other deep neural networks with an accuracy of 84.6% [32]. Gedupudi et. al. extended the research of Feizollah et. al. to develop neural networks on analyze data on both Twitter and Reddit and to compare skip-gram and CBOW embeddings; it was found that CNN-LSTM trained with both Word2Vec embeddings performed best with an accuracy of 83.7% [19]. Tibrewal et. al. compared classical machine learning techniques – naïve bayes, random forest, decision trees, logistic regression, and XGBoost – to BERT transformers using a Twitter dataset; it was found BERT outperformed the classical machine learning techniques with an accuracy of 95.12% [33].

IV. HISTORY AND BACKGROUND

Natural Language Processing (NLP), a subfield at the intersection of artificial intelligence, computer science, and linguistics, is concerned with the interactions between computers and human language [6]; mainly, NLP focuses on automatic computational processing of natural languages to perform tasks like translations and information extraction. Sentiment Analysis, a process used to comprehend emotion behind the text, combines NLP and machine learning techniques to process, identify, extract, quantify, and study affective states and subjective information [7]. For example, businesses can effectively gather customer feedback and utilize sentiment analysis to understand public opinion towards products or a brand; the feedback can improve existing and new products and provide a better experience to customers. However, before a business can understand the sentiment behind customer data, the dataset must first be pre-processed. Text pre-processing is essential for natural language; it transforms the data into a more digestible form that can be fed into classifiers and can improve classifier performance. Machine Learning (ML) models require textual inputs to be vectorized – convert strings to numbers; word embeddings are the most popular method approach to represent words and documents as numeric vector input. Methods and procedures are discussed in detail below.

4.1 Word Embeddings

Numerical representations are a prerequisite for most ML models. Embeddings represent words in a corpus as meaningful numbers; these numbers – weights – capture the context of the sentence and underlying semantic relationships and similarities between words. Specifically, models learn weights to represent semantic relationships between words and related words are positioned closely in the embedding space. Different embedding frameworks vary in their approach to represent contextual relationships:

4.1.1 Word2Vec

Word2Vec, introduced by Mikolov et al., is a family of neural network-based models used to create a numerical representation of words and find local contextual and semantic information [17]. Essentially, Word2Vec accepts a text corpus as input and outputs a vector representation for each word in the corpus. Word2Vec employs a two-layered NN to examine neighboring words in a corpus and determine whether the current word(s) have any semantic relationship; if pairs of words are determined to have a semantic relationship, they are given similar embeddings [8]. Word2Vec is not a single algorithm but rather a combination of techniques – continuous bag of words (CBOW) and skip-gram.

i. Continuous Bag of Words

CBOW aims to predict the probability that a target word occurs given its context. First, the target word and context words are encoded using one-hot encoder. Second, the model is trained to predict the target word given N context words, where N is a user-specified window size; the window size limits the number of words considered in the evaluation. Finally, the model outputs a matrix of weights representing the word embeddings. Fig. 1 provides a general overview of the CBOW architecture; context words are given as input to the model, which are then used predict the target word(s). CBOW then outputs a vector of probabilities, which indicates the similarity between the target word and the context words. For example, CBOW with a window size of 2 would convert the sentence ‘I would like some juice’ as follows: ([I, would], like), ([would, like], some), ([like, some], juice). Then, CBOW would predict the target – ([context], target) – using the context words.

ii. Skip-Grams

Skip-Gram architecture aims to achieve the reverse of CBOW; mainly, Skip-Gram predicts the context words given a target word. First, the words in the corpus are assigned a corresponding vector using one-hot encoding; vectors are of length $[1, v]$, where v is the vocabulary length. Next, the vectors are passed into the hidden layer, where the vectors are multiplied by the weights of the hidden layer to produce an output vector. Finally, the softMax function is applied to the output vector to convert the vector of numbers into a vector of probabilities. Each component in the vector represents the probability of the word appearing within the window size of the target word. Fig. 1 illustrates a high-level view of the skip-gram architecture.

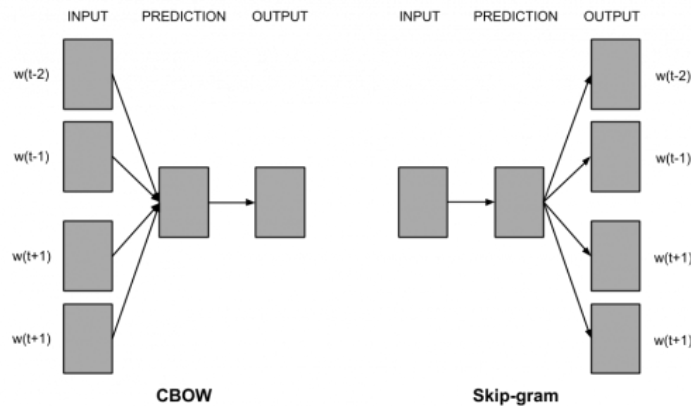


Figure 1: Comparison of CBOW and Skip-Gram Architectures [8]

4.1.2 Glove2Vec

Glove2Vec, introduced by Pennington et al., is an unsupervised machine learning algorithm that captures global contextual information and derives vector representations for words [18]. Unlike Word2Vec, GloVe2Vec computes global co-occurrence statistics whereas Word2Vec computes local co-occurrence statistics. GloVe takes a corpus of text and transforms each word into a position in a low-dimensional space; it constructs a co-occurrence matrix and computes a

co-occurrence probability using the word's frequency in the corpus in representations of linear substructures of the word vector space. Finally, vectors are factorized to yield a lower-dimensional matrix, where each row is a vector representation for the corresponding word as seen in Fig. 2.

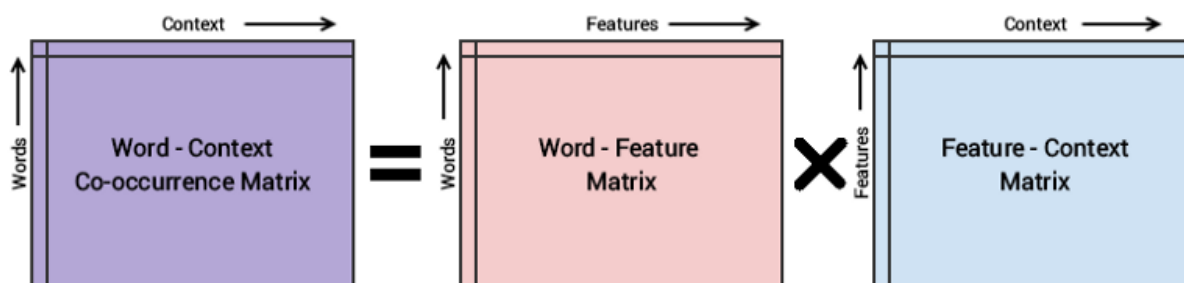


Figure 2: Architecture of GloVe2Vec [9]

4.1.3 BERT

BERT, introduced by Devlin et al., is a transformer-based architecture that learns contextual relations between all words in a sentence, regardless of their respective position; it is centered on the WordPiece model, with the vocabulary limited to 30,000 words and all characters in the English alphabet [19]. Words that do not appear in the vocabulary are decomposed into the largest possible sub-word contained in the vocabulary; as a last resort, BERT decomposes the word into individual characters. Decomposition allows the sub-words to maintain some contextual information, which avoids an overloaded vocabulary – ex. homonyms assigned the same definition. Fig. 3 illustrates the BERT embeddings. For example, the input word is ‘mistakenly’, which does not appear in the BERT lookup table; the word is broken up into ‘mistake’ and ‘nly’, such that ‘mistake’ and its vector representation can be found in the lookup table.

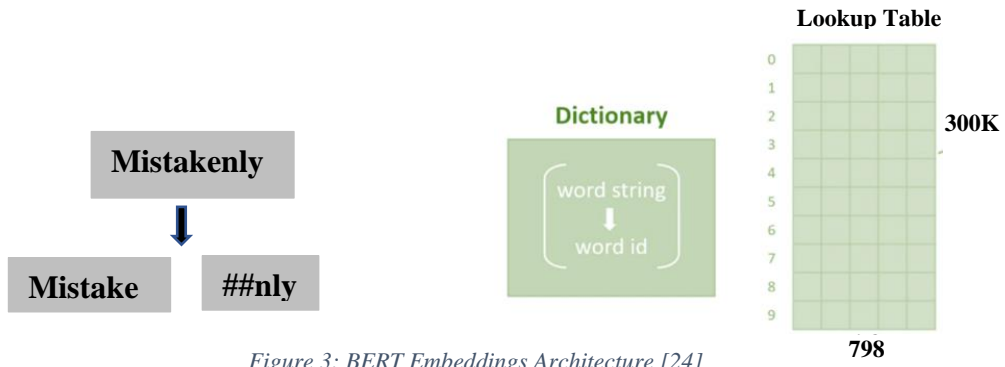
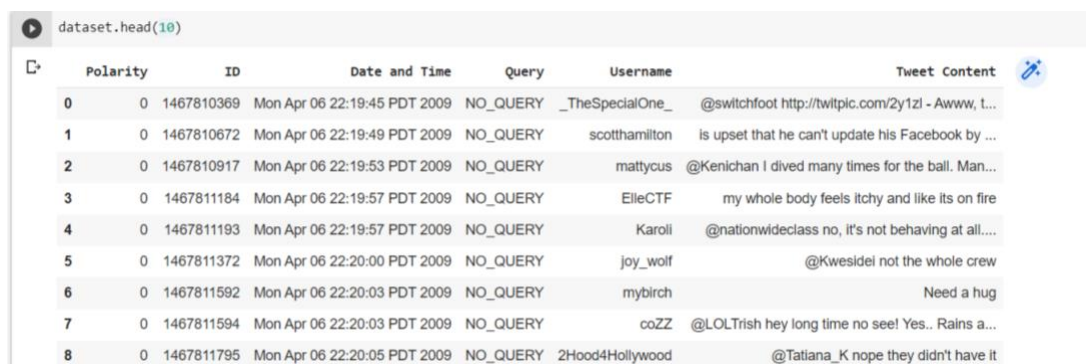


Figure 3: BERT Embeddings Architecture [24]

V. DATASET AND DATASET PREPARATION

5.1 Dataset A – Sentiment140

Models were trained and validated using the Sentiment140 dataset, curated by graduate students at Stanford University. Note, the Sentiment140 dataset is not related to pandemic. It contains 1,600,000 tweets with the following six fields: tweet polarity, tweet ID, date and time, query flag, username, and tweet text. A subset of the dataset is shown in Figure 4. We were only interested in the tweet polarity and tweet text fields. Polarity is divided into three classes: negative, neutral, and positive, represented by numeric values 0, 2, 4, respectively; however, upon further review, the dataset did not contain any tweets that were labeled as neutral. Sentiment140 target classes were balanced, with 800,000 entries per class.



	Polarity	ID	Date and Time	Query	Username	Tweet Content
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all...
5	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew
6	0	1467811592	Mon Apr 06 22:20:03 PDT 2009	NO_QUERY	mybirch	Need a hug
7	0	1467811594	Mon Apr 06 22:20:03 PDT 2009	NO_QUERY	coZZ	@LOLTrish hey long time no see! Yes.. Rains a...
8	0	1467811795	Mon Apr 06 22:20:05 PDT 2009	NO_QUERY	2Hood4Hollywood	@Tatiana_K nope they didn't have it

Figure 4: Raw Sentiment140 Dataset

5.1.1 Dataset A Preprocessing

Preprocessing the dataset can dramatically improve the performance of the models. The following procedure was used to clean the dataset:

1. The `value_counts` function from the Pandas library is a useful function that sums the number of unique values. It was found that the dataset contained 1,581,466 unique tweets; this implies the dataset contains approximately 18,500 duplicate tweets. Duplicate tweets were removed via the `drop_duplicates` function from the Pandas library.

2. Remove user mentions '@', hashtags '#' symbol, and retweets 'RT'; these symbols do not provide any useful information towards user sentiment. Note, the actual tag that followed the hashtag symbol was not removed as these may contain useful information.
3. Punctuation can be used to express emotion in some situations – exclamation points can either indicate strong emotion towards a subject or emphasize a statement; however, other punctuation, like commas and periods, can add noise to the dataset. All punctuation were removed to remain consistent with previous works.
4. URLs do not convey any user emotion; we can remove them from the dataset.
5. Stop-words such as 'I', 'to', 'are', etc. and contractions ('isn't', 'we're', etc.) were removed from the dataset; these words do not hold any meaning nor add any useful information towards user sentiment. Python provides a built-in library with a list of stop-words from the English language.
6. Convert all words to lowercase; this is a common practice used for consistency, simplicity, and to avoid case insensitivity. For example, 'Good' and 'good' would be considered different words if the two words are not converted.
7. Encode the 'positive' sentiment numerical value from 4 to 1.
8. Remove irrelevant columns from the dataset: tweet ID, date and time, query, and username. Neither of these columns help us evaluate user sentiment; rather they provide sensitive user identification information.
9. Tweets must be tokenized before the text can be effectively used by a classifier. Tokenization divides a large text into smaller units, like words or lines; individual words, separated by whitespace, are tokenized.

Data cleaning procedure summary is shown in Figure 5.

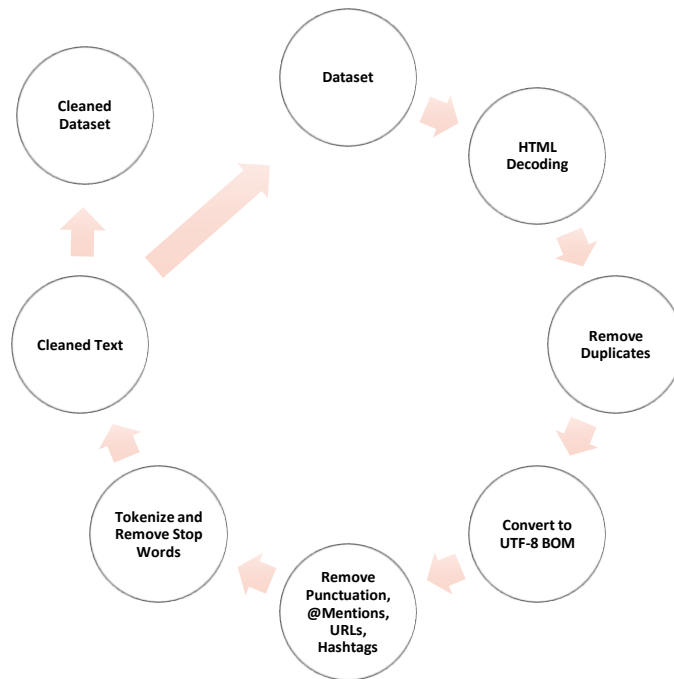


Figure 5: Data Cleaning Process

5.2 Dataset B – TikTok Dataset

Models were tested using the TikTok Comments dataset, curated by Bansal et. al. [20]. It contains 646,473 posts, scraped from July 2021 through April 2022, with the following fields: created_utc, author, app_version, body, posted_date, sentiment_polarity, sentiment_subjectivity, sentiment, preview, score, and month. CoV-2 related posts were scraped using the following queries words: “pandemic”, “social distancing”, and “stay at home”, among others. A subset of the dataset is shown in Figure 6. We were only interested in the polarity and text fields. Polarity is divided into three classes: negative, neutral, and positive, respectively; however, upon further

review, only 1% of the dataset contained Reddit posts that were labeled as neutral. Neutral entries were removed from the dataset.

index	likes	dislikes	created_utc	author	subreddit	body	score	permalink	date	sentiment_polarity	sentiment_subjectivity	sentiment	preview	month
0	71951	0	156579617	trame9	UnethicalAP/ops	catch coronavirus for free too!	84	/r/unethicalAP/ops/comments/foqkxvot_a_lot_of_samer_citizens_are_going_to_catchthecorv	2020-04-01	0.5	0.8	positive	catch coronavirus for free too!	4
1	71952	1	156589558	chracpht	Coronavirus	Americans are so retarded that they think wearing a mask means you have coronavirus, and that having coronavirus means they should avoid you. This is why we can't have nice things!	79	/r/Coronavirus/comments/foqkxvot_what_woman_punished_rebbed_for_wearingthecorv	2020-04-01	-0.1	0.9	negative	Americans are so retarded that they think wearing	4
2	71953	2	1565790288	chracpht	Coronavirus	Agf: So last year "17,000 Americans died from the common flu." It averages between 27,000 and 73,000 per year. Nothing is shut down. The economy goes on. At this moment there are 548 confirmed cases of Coronavirus, with 22 deaths. "I think about them!" - Donald J. Trump, May 9, 2020 (https://twitter.com/realDonaldTrump/status/127272582148987912) "I think about that!"	58	/r/Coronavirus/comments/foqkxvot_how_projects_100_to_200_coronaviruscases1st	2020-04-01	0.04236111111111111	0.4838888888888889	positive	Agf: So last year "17,000 Americans died from the	4
3	71954	3	1565792180	ReadTheNewsArticle	Coronavirus	Agf: As the night showed that he had, could not eliminate the coronavirus with regular therapy and that he might still have been infected, the patient was treated with a plasma transfusion from recovered Covid-19 patients. Agf: The man's status turned negative two days later. "Silver lining. I guess."	38	/r/Coronavirus/comments/foqkxvot_coronavirus_patient_mild_symptoms_but_longhealy9	2020-04-01	-0.1	0.15887436874355	negative	Agf: As the night showed that his body could not e	4
4	71955	4	1565790887	Prinob47	r8a	Prayers up for coronavirus	35	/r8a/comments/foqkxvot_hesher_boston_saltic_guard_marsuc_smat_glamth2hdwv	2020-04-01	0.0	0.0	positive	Prayers up for coronavirus	4

Figure 6: Raw Emotions Towards Covid-19 on TikTok Dataset

5.2.1 Dataset B Preprocessing

The following procedure was used to clean the dataset:

1. Binary Classification models trained on a dataset with an unbalanced distribution of classes will lead to biased models; biased models will impact its ability to predict the minority class. Dataset B had unbalanced classes – negative class outweighed the positive class by approximately 100,000 entries. Undersampling – a process where observations are randomly deleted from the majority class to match the number of observations from the minority class – was used to balance the two classes.
2. Removed irrelevant columns from the dataset: created_utc, author, subreddit, date, sentiment_polarity, sentiment_subjectivity, preview, score, permalink, and month. Neither of these columns help us evaluate user sentiment; rather they provide sensitive user identification information and information used by the author to label the dataset.
3. It was found that the dataset contained 646,473 unique posts; this implies the dataset does not contain duplicate posts.
4. URLs do not convey any user emotion; they can be removed from the dataset.
5. Punctuation can be used to express emotion in some situations – exclamation points can either indicate strong emotion towards a subject or emphasize a statement; however,

other punctuation, like commas and periods, can add noise to the dataset. All punctuation were removed to remain consistent with previous works.

6. Stop-words such as 'I', 'to', 'are', etc. and contractions ('isn't', 'we're', etc.) were removed from the dataset; these words do not hold any meaning nor add any useful information towards user sentiment. Python provides a built-in library with a list of stop-words from the English language.
7. Removed words of length 3 or less and non-English words using the Brown corpus; particularly, the dataset contained response body encodings – gt and lt – that added noise to the dataset.
8. Convert all words to lowercase; this is a common practice used for consistency, simplicity, to avoid case insensitivity. For example, 'Good' and 'good' would be considered different words if the two words are not converted.
9. Encode both the 'negative and 'positive values to 0 and 1 respectfully to be consistent with the Sentiment140 dataset.
10. Reddit posts were tokenized before the text was by a classifier. Tokenization divides a large text into smaller units, like words or lines.

VI. DEEP LEARNING (DL) MODELS

6.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN), introduced by LeCun et al. in the 1980s, is an artificial neural network (NN) architecture commonly applied in various computer vision tasks — image processing, image classification, object detection, and classification [21]. Recently, CNN's have become increasingly popular for NLP applications, like translations and sentiment analysis. CNN consists of three layers: convolutional layer, pooling layer, and fully connected layers, described in detail below.

6.1.1 Convolutional Layers

Convolutional layer is composed of a series of feature maps and filters, called convolutions, that extract patterns from the input data; each feature map iterates through the input with a sliding window – filter matrix – and applies a dot product between the filter matrix and input represented as a matrix to extract high-level features, such as edges and shapes, from images or text. The matrix that results from the dot product represents the importance of the detected feature(s). Fig. 7 illustrates the arithmetic combination between a 5x5 input image and a 3x3 filter (convolved feature) matrix to form a set of features in the output array. CNN learns the values of these filters on its own during the training process; the more filters in the model, the more features get extracted, and the better CNN becomes at recognizing patterns. An activation function is applied to the output to provide a non-linear relationship for our output.

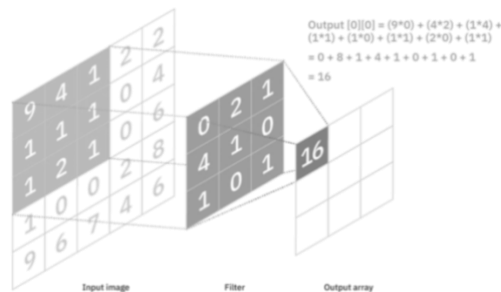


Figure 7: 5x5 Image and Filter with Resulting Output Array [10]

6.1.2 Pooling Layer

Pooling progressively reduces the dimensionality of the feature maps; dimension reduction increases the model's efficiency. Additionally, it extracts the dominant features while eliminating features that contribute modestly to the model. There are several pooling functions:

- (a) Average Pooling: compute the average of the values from the map.
- (b) Max Pooling: take the largest value from each feature window.
- (c) Sum Pooling: sum values in the feature map.

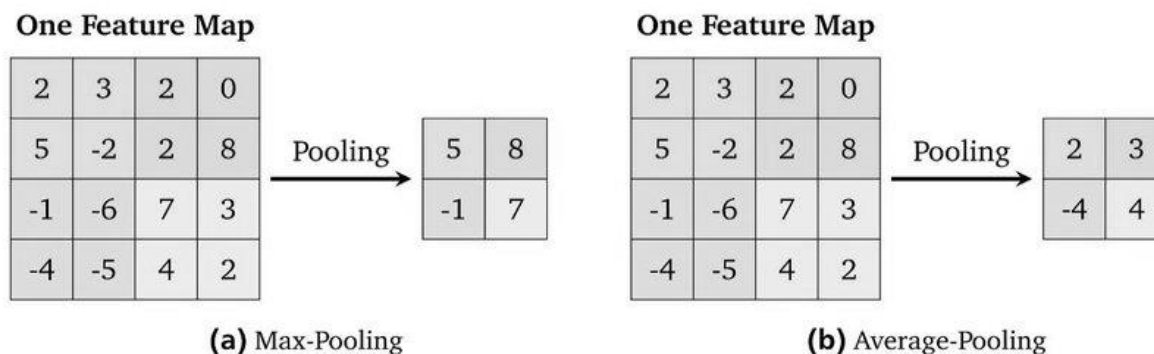


Figure 8: Max Pooling vs Average Pooling [11]

Figure 8 illustrates the difference between max-pooling and average-pooling; a 2x2 filter is applied to the feature map to reduce the dimensionality. Max-pooling takes the maximum value within the 2x2 filter, whereas average-pooling computes the average. Generally, an activation function – ReLu, Sigmoid, or Softmax – is applied to the output of pooling layers.

6.1.3 Fully Connected Layer

Fully Connected Layer is a simple, feed forward NN that forms the last layers of CNN; it uses the reduced feature map weights to flatten the final output, connect the neurons between different layers, and feed it to the CNN to train for a series of epochs, seen in Figure 9. During

the training process, output from the convolution layers begins to classify the textual input as either positive or negative based on the features from previous steps.

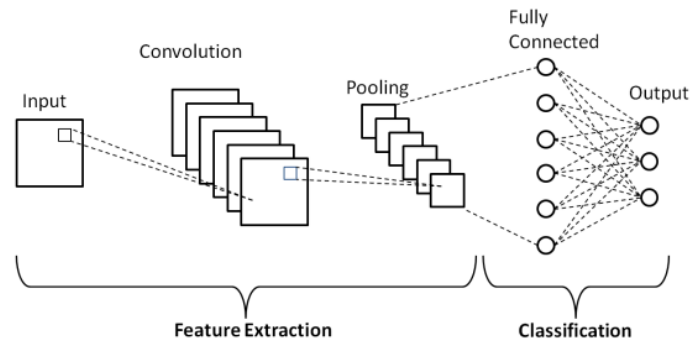


Figure 9: Schematic Diagram of a Basic CNN Architecture [12]

6.1.4 Dropout Layer

Occasionally, the features chosen by the previous layers may result in overfitting in the training dataset; this occurs when the model memorizes the training data, and the model cannot be generalized to new input data. A dropout layer prevents the issue of overfitting. Dropout layer randomly selects and drops a set of nodes and edges during the training process to add noise, illustrated in Figure 10; noise necessitates other nodes to learn a sparse representation of the data.

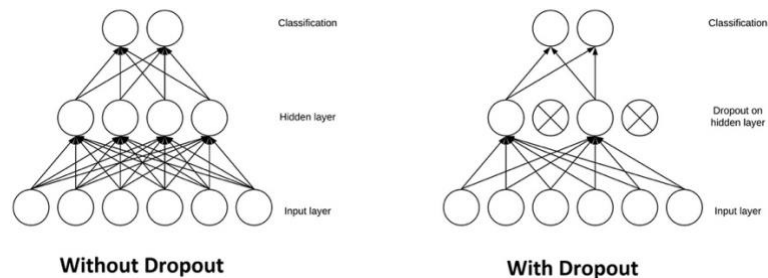


Figure 10: Neural Network without Dropout vs Neural Network with Dropout [13]

6.2 Long Short-term Memory

Long Short-term Memory (LSTM), introduced by Schmidhuber et al. in 1997, is an artificial recurrent neural network (RNN) architecture used to model sequential data; it was designed to overcome the vanishing and exploding gradient problem present in RNN [22].

LSTM is widely used in machine translation and speech recognition, among other applications.

A typical LSTM comprises three gates: input gate, output gate, and forget gate that controls the information that flows in and out at the current time step.

6.2.1 Forget Gate

LSTM must first decide which bits of information should be removed or kept from a cell given both the previous hidden state and new input data. At each time step, the LSTM network combines the hidden state (i.e., the text) from the previous time-step and input from the current time-step to produce a new output. Sigmoid function takes the output and returns values between 0 and 1; values closer to 1 are kept while values closer to 0 are removed. Output values are pointwise multiplied with the previous cell states. Note, components multiplied by values near 0 will be deemed irrelevant and thus removed by the network.

6.2.2 Input Gate

LSTM must then decide which bits of information to add to the cell state. First, the network combines the hidden state from the previous time-step h_{t-1} with input from the current time-step x_t ($h_{t-1} + x_t$). Note, the previous hidden state and current time step for the input gate are the same as those used in the forget gate; inputs are then passed through a second sigmoid function to determine whether to add or ignore information from the current input. Next, the same input is passed through a tanh activation function to regulate the network and reduce bias; tanh outputs a vector of values between -1 and 1. Finally, output from the sigmoid function is pointwise multiplied with output from the tanh activation function to determine what information should be added to the cell state and what information should be discarded.

6.2.3 Output Gate

Finally, the output gate determines the value of the next hidden state. First, the previous hidden state and current time step are passed through a third sigmoid function. Next, the output

from the sigmoid function is passed through a tanh. Finally, the results from both the sigmoid function and tanh function are pointwise multiplied to decide which bits of information the new hidden state should carry to the next time step; the new hidden state is used for prediction.

Figure 11 illustrates the high-level architecture of the LSTM network.

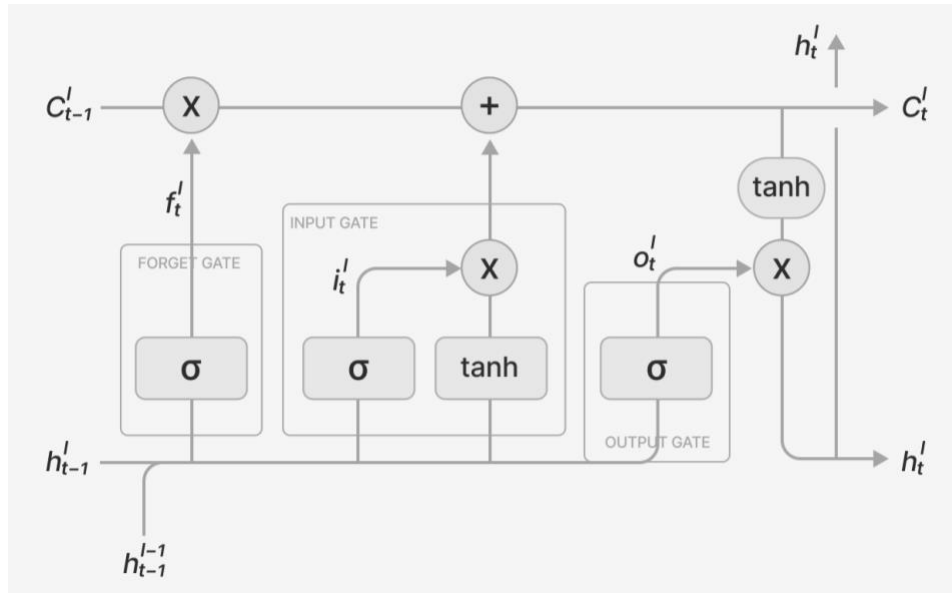


Figure 11: Schematic Diagram of LSTM Architecture [14]

6.2.4 Sigmoid Function

Sigmoid function is a non-linear activation function, contained by the gates, used to maintain values that flow within a model between 0 and 1; it helps the LSTM network learn important information and forget extraneous information. If the product is 0, information is omitted, whereas information is kept if the product is 1.

$$\phi(z) = \frac{1}{1 + e^{-x}}$$

6.2.5 Hyperbolic Tangent (Tanh) Activation Function

Tanh Activation function is a non-linear activation function that regulates the values that flow through the network. Numerous transformations on the input vectors may cause values to explode, causing other values to seem insignificant; the tanh function ensures that the values

remain between -1 and 1 so the function avoids the exploding gradient problem and remains differentiable.

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

6.3 CNN-LSTM

CNN-LSTM architecture, proposed by Shi et al., was developed for temporal sequence and visual time-series prediction problems; it combines and leverages the two architectures to build a model to tackle more sophisticated tasks like text classification and video conversion [23]. CNN architecture is used to automate feature extraction from the data. LSTM is used to analyze and evaluate features and feature dependencies extracted by CNN. Thus, the CNN-LSTM model can combine feature extraction of CNN with LSTM's capability to learn dependencies in sequential data.

CNN-LSTM architecture comprises the following layers: convolution layer, pooling layer, LSTM layer, and fully connected layer. A series of convolutions are applied to the input sequence to produce a feature map; the pooling layer reduces the dimensionality of the feature maps. LSTM layer captures long-term dependencies over window features respectively [3]. Finally, the fully connected layer predicts whether the sentence has positive or negative sentiment. Figure 12 illustrates the architectural structure of the network.

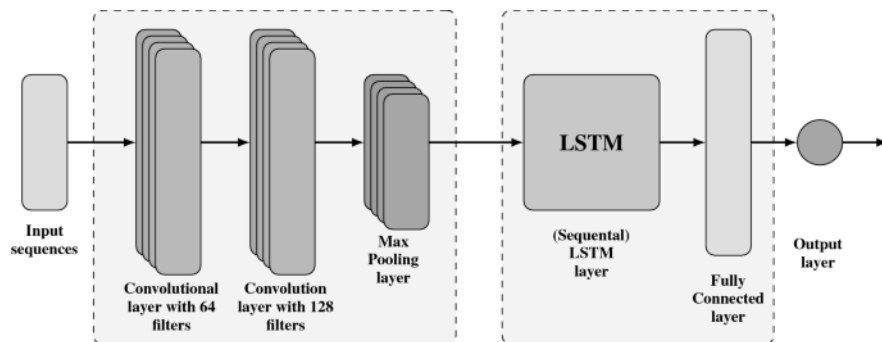


Figure 12: Schematic Diagram of CNN-LSTM Architecture [15]

6.4 CNN-LSTM with Attention Mechanism

Attention Mechanism is a technique that enhances the essential portions of the input data while it diminishes other portions of the input data; it was introduced to address the inability of encoder-decoder models to retain longer sentences. Recurrent Neural Networks (RNNs) tend to forget relevant information from previous time steps of a sequence before processing the entire sentence. Additionally, not all words contribute equally to representing the sentence meaning. Therefore, an attention mechanism was introduced to extract words that are important to the sentence's meaning and aggregate the representation of the informative words to form a sentence vector. In essence, the attention mechanism assumes that the next word in a sentence depends on the current word being processed and assigns a weight to words based on relevancy to the current and previous word. Finally, a weighted sum of the output vectors is taken and sent through dense layers and SoftMax for text classification.

6.5 Bidirectional Encoder Representations from Transformers (BERT)

BERT, developed by Devlin et al., is a transformer-based architecture commonly applied in various NLP tasks – sentiment analysis, semantic role labels, and word predictions; specifically, BERT adopts attention mechanisms to deal with long-term dependencies between words in a sequence and learn contextual relations between all words in a sentence, regardless of their respective position [19]. For example, the word ‘bank’ can have several interpretations: a financial institution or land alongside a river. Given the sentence: ‘Bob arrived at the bank across the river’, attention mechanism can learn to immediately heed ‘river’ and conclude ‘bank’ refers to land alongside a river.

Contrastingly, directional models – CNNs and RNNs – process sentences sequentially to learn contextual relationships between words that precede a term; however, this method may

produce the wrong output, as context learning is limited. For example, if a directional model was given the sentence: ‘Bob arrived at the bank across the river’, the model would neglect the latter half of the sentence, which provides context for ‘bank’; without the context ‘... across the river.’, the model may be misled to believe Bob arrived at a financial institution because directional models are limited to condition a term on previous words in a sentence and cannot reach future input information.

BERT removes the unidirectionality constraint, which enables the model to condition and learn information from both the left and right side of a term; it leverages two unsupervised tasks – Masked Language Model (MLM) and Next Sentence Prediction (NSP) to pre-train the model and perform tasks in a bidirectional manner. Nevertheless, input sequences must be transformed into a specific format that BERT can use to train the MLM and NSP.

Preprocessing the dataset is essential to transform raw input data into a format BERT can recognize and process; the pre-process procedure is broken down into three stages: tokenization, segmentation, and word sequencing. Identifier Tokens [cls] and [sep] are applied to the sentence pairs to indicate that the model will perform classification and indicate separation between sentences respectfully. Let us understand the procedure through an example in Figure 13.

Consider the following two sentences: “Paris is a beautiful city. I love Paris”. Tokenization would convert the two sentences as follows: [[cls], Paris, is, a, beautiful, city, [sep], I, love, Paris, [sep]].

Input	[CLS]	Paris	is	a	beautiful	city	[SEP]	I	love	Paris	[SEP]
Token embeddings	$E_{[CLS]}$	E_{Paris}	E_{is}	E_a	$E_{beautiful}$	E_{city}	$E_{[SEP]}$	E_I	E_{love}	E_{Paris}	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

Figure 13: BERT Input Data Transformation

Segmentation Embedder receives the tokens from the previous step and returns whether the token is from sentence A or sentence B. Finally, since BERT processes all words in parallel, the model requires information on word order; positional embedder returns the position of the word in the original sequence, which can then be fed into the BERT transformer to pre-train.

MLM is one half of the pre-training process behind the BERT model; it essentially optimizes weights inside BERT to obtain the original sentence. First, the text is split into smaller units – words – that can be assigned meaning. Second, the tokens are copied to create an identical vector which can be used to compute loss and optimize the model. Third, a random selection of tokens – 15% of the terms – are masked so the model can be trained to predict the masked words using context of the words that surround it. For instance, the sentence ‘It is now autumn’ would be represented as ‘It is now [MASK]’. BERT is then trained to predict [MASK] as being ‘autumn’; however, masked words are not always replaced by the [MASK] token, because the token does not always appear during fine-tuning. Instead, 80% of the masked words are replaced with the token, 10% are replaced with random words, and 10% are left unaffected. Finally, a loss is computed as the difference between the predicted output probability distributions for each masked token and the true value of the masked token. MLM is repeated until the loss function converges.

NSP is the other half of the training process behind the BERT model, which allows the model to understand long-range dependencies across pairs of sentences; essentially, NSP is a binary classifier in which two sentences are fed as input to BERT and the model predicts whether the two sentences have a logical, sequential relationship or whether their relationship is simply random. 50% of the pairs will be true consecutive sentences and the rest of them would have random sentences as its pair. BERT is then trained to recognize whether the sentence pair is

sequential or not sequential as follows: the embeddings generated during the pre-processing are fed into the BERT transformer which results in a matrix of weights and biases; the weights are fed into the SoftMax function, which returns the probability of the sentence pair being isNextSentence or NotNextSentence.

Note: BERT trains both MLM and NSP in parallel to minimize the combined loss function of the two strategies; the models are trained using the Toronto BookCorpus and English Wikipedia datasets, which are readily available.

VII. PROPOSED MODELS

7.1 Neural Networks with Pre-Trained GloVe Embeddings

Individually, the four models discussed above – CNN, CNN-LSTM, CNN-LSTM with Attention Mechanism, and LSTM – will be trained using the pre-trained GloVe 6B embeddings; each model will produce word embeddings of 100-dimensional vectors. Embeddings from the GloVe 6B model are retrieved and assigned to each word; if a word does not appear in the pre-trained model, we assign it with the value zero. Finally, the embeddings are passed to the embedding layer sequentially.

7.2 Neural Networks with Pre-Trained Word2Vec Embeddings

Individually, the four models discussed above – CNN, CNN-LSTM, CNN-LSTM with Attention Mechanism, and LSTM – will be trained using the Google News pre-trained Word2Vec model; each model will produce word embeddings of 100-dimensional vectors. Embeddings from the Google News Word2Vec model are retrieved and assigned to each word; if a word does not appear in the pre-trained model, we assigned the word with value zero. The embedding layer is seeded with the Word2Vec weights.

7.3 Neural Networks without Pre-Trained Word Embeddings

Individually, the four models discussed above – CNN, CNN-LSTM, CNN-LSTM with Attention Mechanism, and LSTM – will train and use the embedding layer in its respective model to learn word embeddings. Words in the tweets are tokenized and assigned a random vector representation; the model progressively learns semantic relationships between the words and updates the vectors through backpropagations.

7.4 BERT Transformer

BERT will be trained using the BERT-base pre-trained model; each model will produce word embeddings of 300-dimensional vectors. Embeddings from the BERT-base model are retrieved and assigned to each word; if a word does not appear in the pre-trained model, BERT will compute and assign an embedding on a sub-word level.

VIII. EXPERIMENTS AND MODEL RESULTS

8.1 Experiment Environment

Python 3.7 was used to conduct the experiments. PyCharm, a Python Integrated Development Environment (IDE) was used to train and validate results for the CNN-LSTM, CNN-LSTM with Attention, LSTM, and LSTM with Attention models. Google Colab Pro, a cloud based Jupyter Notebook environment, was used to train and validate results for the BERT transformer. Models were implemented using the PyTorch library.

8.1.1 Hyperparameters for Directional Models

Proposed models can be variously modified according to the type and parameter adjustment of the layers that constitute the network. The default dropout value of 0.5 was used for all four models; half of the nodes and edges were dropped from the fully connected layer. Learning rate was initialized to 0.00001 to regulate the update of weights at the end of each batch. Adam Optimizer, a stochastic gradient descent procedure used to iteratively update network weights, was used to handle complex problems, such as the categorization of tweet's sentiment. Embedding Size, a way to represent each word in a text as a real-valued vector, was initialized to 100 for each of the proposed models. Hidden Size, the number of features of the hidden state, was initialized to 128. Cross Entropy Loss Function was used to minimize the loss.

8.1.2 Hyperparameters for BERT Model

BERT has minimal, task-specific parameters that can be fine-tuned. The default dropout value of 0.3 was used; thirty percent of the nodes and edges were dropped from the fully connected layer. Learning rate was initialized to 0.00001 to regulate the update of weights at the end of each batch. Adam Optimizer, a stochastic gradient descent procedure used to iteratively update network weights, was used to handle complex problems, such as the categorization of

tweet’s sentiment. Embedding Size, a way to represent each word in a text as a real-valued vector, was initialized to 300 for each of the proposed models. Hidden Size, the number of features of the hidden state, was initialized to 128. Cross Entropy Loss Function was used to minimize the loss.

8.2 Model Training

The experiment was performed by splitting the dataset into train, validation, and test sets with a split ratio of 0.8, 0.1, and 0.1. The validation and test sets are of equal sizes. The best method is to take the whole dataset and shuffle it before splitting, as shuffling the dataset makes the model more robust. Models were trained using a batch size of 128 and 50 epochs.

8.3 Results

NN models trained with either GloVe or Word2Vec took significantly less time to train compared to NN models trained without a pretrained embedding; models trained with Word2Vec had the shortest runtime overall, with an average runtime of 6,662 seconds, seen in Table 1. BERT model was prohibitively slow to train with CPU alone; thus, BERT was trained with GPU to reduce the runtime. Without GPU, BERT’s runtime was nearly forty-eight hours. GPU allowed BERT to complete computation within 39,132 seconds.

Embedding Type	Average Accuracy Score (%)	Average Validation Accuracy Score (%)	Average Runtime (sec)
GloVe	90.33	89.18	28,753
Word2Vec	91.22	89.39	6,662
Embedding Layer	90.79	83.87	120,922
BERT (with GPU)	97.34	92.57	39,132

Table 1: Average Runtime and Accuracy Scores of Different Embeddings

Generally, the LSTM model performed marginally worse than the other models, regardless of the embedding type used as seen in Table 2; LSTM had an average training accuracy score of 86.4% which is a slight improvement from the accuracy score seen in [5].

LSTM consistently had accuracy scores that plateaued around 90% regardless of the number of epochs, whereas the other models continued to see an increase in accuracy score as the number of epochs increased. LSTM with Attention achieved the highest average accuracy, followed by CNN-LSTM with Attention; the two models differed by 0.12%; however, with a average runtime 5,600 seconds shorter than CNN-LSTM with Attention, LSTM with Attention would be the better choice.

Model Architecture	Average Accuracy Score	Average Validation Accuracy Score	Average Runtime (sec)
LSTM with Attention	91.34	87.62	36,233
LSTM	89.64	86.40	17,264
CNN-LSTM with Attention	91.22	87.91	41,854
CNN-LSTM	90.54	87.97	41,102

Table 2: Average Runtime and Accuracy Scores of Different NN Models

CNN-LSTM performed better compared to [1] regardless of the embeddings used. NN models trained without a pre-trained embedding performed marginally better than [1]. CNN-LSTM architecture trained with pre-trained embeddings achieved an accuracy score of approximately 89% compared to 83% from [1] trained with Word2Vec, seen in Table 4.

Model Architecture	Train Accuracy	Validation Accuracy	Loss	Runtime (sec)
LSTM with Attention	90.1647	89.4941	0.25993	17,264
LSTM	89.4794	88.8196	0.293203	14,793
CNN-LSTM with Attention	91.9079	88.9971	0.180607	41,854
CNN-LSTM	89.7726	89.4116	0.35549	41,102

Table 3: Neural Network Models trained with GloVe Results

NN models trained with GloVe embeddings generally performed similarly to the NNs trained with Word2Vec, seen in Table 3; models trained with GloVe had an average accuracy score of 90.33% whereas models trained with Word2Vec embeddings had an average accuracy score of 91.22%, results seen on Table 1. Nevertheless, Word2Vec, on average, took approximately one-fourth of the time to complete at 6,662 seconds compared to 28,753 seconds

it took GloVe to train. Additionally, although models trained without pre-trained embeddings achieved a relatively high train accuracy, the runtime of the models trained with the embedding layer makes it inferior to the models trained with pre-trained embeddings.

Model Architecture	Train Accuracy	Validation Accuracy	Loss	Runtime (sec)
LSTM with Attention	93.1201	89.6749	0.251897	6,033
LSTM	90.2673	88.5163	0.122854	3,064
CNN-LSTM with Attention	90.9280	89.7184	0.237278	6,097
CNN-LSTM	90.5520	89.6377	0.157389	11,453

Table 4: Neural Networks trained with Word2Vec Results

CNN-LSTM trained without pre-trained embeddings performed significantly better than [7] with an accuracy of 91.91% compared to 81.57%; however, the validation accuracy for all models is notably worse compared to the training accuracy. Although the validation accuracy is expected to be lower than the training accuracy, the validation accuracy plateaued around 20 epochs while the training accuracy continued to increase with each epoch; this indicates overfitting – models are unable to generalize to non-training data – has occurred. Fig. 14 illustrates the LSTM without pre-trained embeddings overfitting to the training dataset; the model is unable to generalize to the validation dataset after 20 epochs.

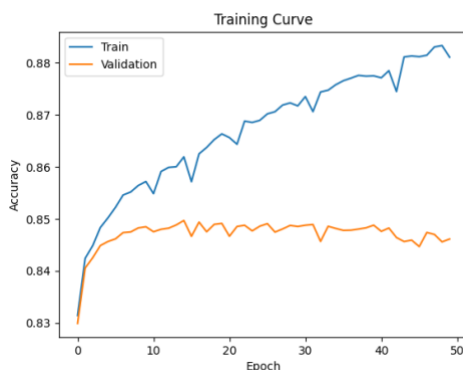


Figure 14: LSTM without Pre-trained Embeddings Training Curve

Additionally, discrepancies between the model’s performance are more apparent when the models are trained without a pre-trained embedding, whereas the training accuracies vary

marginally with pre-trained embeddings. Particularly, NN models trained with a pre-trained embedding have training accuracies that differ by approximately 2%, whereas the train and test accuracies differ by approximately 6% with the models trained on the embedding layer, seen on Table 5.

Model	Train Accuracy	Validation Accuracy	Loss	Runtime
LSTM with Attention	90.6895	83.6947	0.223075	106,328
LSTM	89.1620	81.8649	0.229848	90,843
CNN-LSTM	92.1176	84.7931	0.224059	137,636
CNN-LSTM with Attention	91.1851	85.1120	0.245582	148,882

Table 5: Neural Networks trained with Embedding Layer Results

IX. RESULTS ON TEST DATASET

TikTok Comments dataset, curated by Bansal et. al. was used to predict the sentiment for each comment; the five networks – CNN-LSTM, CNN-LSTM with Attention, LSTM, LSTM with Attention, and BERT – were used to test user sentiment toward the pandemic.

NN models trained with pre-trained embeddings took significantly less time to train compared to NN models trained without a pretrained embedding; models trained with Word2Vec had the shortest runtime. BERT was prohibitively slow to run on CPU alone; thus, K80 GPU with 12 GB of RAM, provided through Google Colaboratory, was used to reduce the runtime.

Word2Vec performed considerably better than NNs trained with either GloVe embeddings or the embedding layer; it averaged a test accuracy of 79%, whereas GloVe and the embedding layer averaged 75% and 72% respectfully, seen on Table 6. Nevertheless, BERT accomplished a test accuracy of 84%, approximately 5% better than Word2Vec.

Embedding Type	Test Accuracy	Average Loss
Word2Vec	79.6799	0.192355
GloVe	75.2976	0.272308
Embedding Layer	72.6839	0.230641
BERT	84.0445	0.0846

Table 6: Average Test Accuracy of Different Embeddings

Generally, NN models trained with the embedding layer underachieved compared to the models trained with pre-trained embeddings; while these models achieved an average train accuracy of 90.79%, the test accuracy plummeted to 72.68%, which is an approximate 18% drop-off. LSTM attained the best test accuracy of 75.14% compared to 78.84% in [5]. CNN-LSTM performed worse compared to [4] with an accuracy score of 71.79% compared to 79.8%, seen in Table 7. BERT outperformed all models, with an accuracy score of 84.04% compared to 79.68% of that seen in Word2Vec models.

Model Architecture	Test Accuracy	Loss
LSTM with Attention	73.6947	0.223075
LSTM	75.1357	0.229848
CNN-LSTM	71.7931	0.224059
CNN-LSTM with Attention	70.1120	0.245582

Table 7: Neural Networks trained with Embedding Layer Results

Overall, NN models trained with Word2Vec achieved the highest test accuracy compared to the models trained with GloVe and the embedding layer; Word2Vec is shown to achieve a test accuracy of 79.62% compared to 75% and 72% from GloVe and the embedding layer. CNN-LSTM trained with Word2Vec outperformed [4], seen in Table 8; it improved upon [4] by 6%.

Model Architecture	Test Accuracy	Loss
LSTM with Attention	79.4941	0.251897
LSTM	78.7607	0.122854
CNN-LSTM	79.9971	0.237278
CNN-LSTM with Attention	80.4675	0.157389

Table 8: Neural Networks trained with Word2Vec Embeddings Results

GloVe embeddings, Table 9, resulted in subpar outcomes despite its relatively high training accuracy; it can be shown that there was an approximate 15% drop-off between the training accuracy and test accuracy. Drop-off may have resulted from the models overfitting to the training dataset.

Model Architecture	Test Accuracy	Loss
LSTM with Attention	77.2849	0.25993
LSTM	78.5494	0.293203
CNN-LSTM	73.7184	0.180607
CNN-LSTM with Attention	71.6377	0.35549

Table 9: Neural Networks trained with GloVe Embeddings Results

X. CONCLUSION

Since 2020, the fear of infection, combined with nationwide lockdowns contributed to more online presence than seen in the past; internet engagement grew 61% during the first wave of the pandemic [28]. Social Media platforms – Twitter and TikTok – saw millions of new user flock to their platforms for news, entertainment, and to share their opinions; particularly, both platforms have seen a considerable number of COVID-19 related posts researchers can use to measure public awareness and the impacts the pandemic has had on people’s lives.

Throughout the research, we discussed and presented various proposed approaches to analyze public sentiment – LSTM, CNN, CNN-LSTM, Attention Mechanism, and BERT – along with different word embeddings – Word2Vec, GloVe, and embedding layer – to determine which combination was best suited for the task. Generally, BERT outperformed the other four NN models with a training accuracy of 97.34% and test accuracy of 84.04%; however, BERT required additional computational resources to reduce runtime. Without GPU, BERT took nearly two days to complete 50 epochs compared to 10.9 hours with GPU.

Word2Vec proved to be the most viable embedding solution, in terms of performance; its average test accuracy score, 79.68%, was comparative to that of BERT. Additionally, compared to both GloVe and the embedding layer, Word2Vec generally had a significantly shorter runtime, which makes it ideal for those who want results in a timely manner. CNN-LSTM with Attention trained with Word2Vec embeddings proved to be the best combination and it provided businesses with an additional tool to study sentiment.

The design and flexibility of neural networks allow them to be applied to any task within text classification. Potential improvements can be made by analyzing the combination of CNN, LSTM, and BERT architectures or additional transformer architectures, like ALBERT.

References

- [1] A. Budhiraja, "Learning less to learn better-dropout in (deep) machine learning," *Medium*, 15-Dec-2016. [Online]. Available: <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5>. [Accessed: 01-Nov-2022].
- [2] A. E. Guissous, "9: Example for the Max-pooling and the average-pooling with a filter ...," *ResearchGate*, Nov-2019. [Online]. Available: https://www.researchgate.net/figure/Example-for-the-max-pooling-and-the-average-pooling-with-a-filter-size-of-22-and-a_fig15_337336341. [Accessed: 01-Nov-2022].
- [3] A. Khorram, M. Khalooei, and M. Rezghi, "End-to-end CNN + LSTM deep learning approach for bearing fault diagnosis - applied intelligence," *SpringerLink*, 27-Aug-2020. [Online]. Available: <https://link.springer.com/article/10.1007/s10489-020-01859-1>. [Accessed: 01-Nov-2022].
- [4] A. Krouska, C. Troussas, and M. Virvou, "Deep learning for twitter sentiment analysis: The effect of pre-trained word embedding," *Learning and Analytics in Intelligent Systems*, pp. 111–124, 2020.
- [5] Bforblack, "Understanding the bert model," *Medium*, 05-Sep-2021. [Online]. Available: <https://medium.com/analytics-vidhya/understanding-the-bert-model-a04e1c7933a9>. [Accessed: 01-Nov-2022].
- [6] C. Nicholson, "A beginner's guide to word2vec and neural word embeddings," *Pathmind*. [Online]. Available: <https://wiki.pathmind.com/word2vec>. [Accessed: 01-Nov-2022].

- [7] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM Neural Network for Text Classification," *arXiv*, 30-Nov-2015. [Online]. Available: <https://arxiv.org/pdf/1511.08630.pdf>. [Accessed: 05-Sep-2021].
- [8] D. Dhami, "Understanding Bert- word embeddings," *Medium*, 05-Jul-2020. [Online]. Available: <https://medium.com/@dhartidhami/understanding-bert-word-embeddings-7dc4d2ea54ca>. [Accessed: 01-Nov-2022].
- [9] D. Sarkar, "Implementing deep learning methods and feature engineering for Text Data: The glove model," *KDnuggets*, 25-Apr-2018. [Online]. Available: <https://www.kdnuggets.com/2018/04/implementing-deep-learning-methods-feature-engineering-text-data-glove.html>. [Accessed: 01-Nov-2022].
- [10] "Defining a neural network in pytorch¶," *Defining a Neural Network in PyTorch - PyTorch Tutorials 1.12.1+cu102 documentation*. [Online]. Available: https://pytorch.org/tutorials/recipes/recipes/defining_a_neural_network.html. [Accessed: 01-Nov-2022].
- [11] I. E. Livieris, E. Pintelas, and P. Pintelas, "A CNN–LSTM model for gold price time-series forecasting," *Neural Computing and Applications*, vol. 32, no. 23, pp. 17351–17360, Apr. 2020.
- [12] IBM Cloud Education, "What are convolutional neural networks?," *IBM*, 20-Oct-2020. [Online]. Available: <https://www.ibm.com/cloud/learn/convolutional-neural-networks>. [Accessed: 01-Nov-2022].
- [13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North*, vol. 1, no. Proceedings of the 2019 Conference of the North

- American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171–4186, Jun. 2019.
- [14] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Jan. 2014.
- [15] K. Chakraborty, S. Bhatia, S. Bhattacharyya, J. Platos, R. Bag, and A. E. Hassanien, “Sentiment analysis of covid-19 tweets by deep learning classifiers—a study to show how popularity is affecting accuracy in social media,” *Applied Soft Computing*, vol. 97, p. 106754, Dec. 2020.
- [16] K. Mohamed Ridhwan and C. A. Hargreaves, “Leveraging twitter data to understand public sentiment for the COVID-19 outbreak in Singapore,” *International Journal of Information Management Data Insights*, vol. 1, no. 2, p. 100021, Nov. 2021.
- [17] N. W. Bourmistrova, T. Solomon, P. Braude, R. Strawbridge, and B. Carter, “Long-term effects of COVID-19 on mental health: A systematic review,” *Journal of Affective Disorders*, vol. 299, pp. 118–125, Nov. 2021.
- [18] P. Baheti, “The Essential Guide to Neural Network Architectures,” V7, 21-Oct-2022. [Online]. Available: <https://www.v7labs.com/blog/neural-network-architectures-guide>. [Accessed: 01-Nov-2022].
- [19] P. Gedupudi, “Analyzing Public Sentiment on COVID-19 Pandemic,” *SJSU Scholar Works*, 2021. [Online]. Available: https://scholarworks.sjsu.edu/etd_projects/1031/. [Accessed: 16-Nov-2021].

- [20] S. Bansal, "3.5m Tiktok mobile app reviews," *Kaggle*, 23-Sep-2021. [Online]. Available: <https://www.kaggle.com/datasets/shivamb/35-million-tiktok-mobile-app-reviews>. [Accessed: 01-Nov-2022].
- [21] S. Hochreiter and J. Schmidhuber, "Long short-term memory.," *Neural Computation*, vol. 9, no. 8, pp. 1733–1780, 1997.
- [22] S. Padidar, S.-may Liao, S. Magagula, T. A. Mahlaba, N. M. Nhlabatsi, and S. Lukas, "Assessment of early COVID-19 compliance to and challenges with public health and social prevention measures in the Kingdom of Eswatini, using an online survey," *PLOS ONE*, vol. 16, no. 6, Jun. 2021.
- [23] "Sentiment on Twitter data set using recurrent neural network - long short term memory," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 11S, pp. 1206–1211, 2019.
- [24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv.org*, 07-Sep-2013. [Online]. Available: <https://arxiv.org/abs/1301.3781>. [Accessed: 01-Nov-2022].
- [25] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 8, no. 1, pp. 216–225, 2014.
- [26] J. Singh, P. Tripathi, "Sentiment analysis of Twitter data by making use of SVM, Random Forest and Decision Tree algorithm," 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT), 2021, pp. 193-198, doi: 10.1109/CSNT51715.2021.9509679.

- [27] A. F. Hidayatullah, S. Cahyaningtyas, and A. M. Hakim, "Sentiment analysis on Twitter using neural network: Indonesian presidential election 2019 dataset," *IOP Conference Series: Materials Science and Engineering*, vol. 1077, no. 1, p. 012001, 2021.
- [28] A. I. Baqapuri, "Twitter sentiment analysis," *arXiv.org*, 14-Sep-2015. [Online]. Available: <https://arxiv.org/abs/1509.04219>. [Accessed: 04-Nov-2022].
- [29] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *MIT Press*, 01-Jun-2011. [Online]. Available: <https://direct.mit.edu/coli/article/37/2/267/2105/Lexicon-Based-Methods-for-Sentiment-Analysis>. [Accessed: 04-Nov-2022].
- [30] K. Lee, "Sentiment analysis — comparing 3 common approaches: Naive bayes, LSTM ...," *Towards Data Science*, 29-May-2021. [Online]. Available: <https://towardsdatascience.com/sentiment-analysis-comparing-3-common-approaches-naive-bayes-lstm-and-vader-ab561f834f89>. [Accessed: 04-Nov-2022].
- [31] M. Z. Asghar, A. Khan, S. Ahmad, M. Qasim, and I. A. Khan, "Lexicon-enhanced sentiment analysis framework using rule-based classification scheme," *PLOS ONE*, vol. 12, no. 2, Feb. 2017.
- [32] Y. E. Karaca, S. Aslan, "Sentiment analysis of covid-19 tweets by using LSTM learning model," *Computer Science*, 2021.
- [33] V. Singh, V. Tibrewal, C. Verma, Y. R. Singh, T. Sinha, and V. K. Shrivastava, "A Bert Model-based sentiment analysis on covid-19 tweets," *Soft Computing: Theories and Applications*, pp. 539–549, Jun. 2022.