Sentiment Analysis on COVID-19 Tweets

Submitted by

Shadman Sakib Ayon
18101395
Samira Ishrat
18101398
Sadia Afrin Mallick
18101396
Prodip Chandra Das
18101115

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
September 2022

Declaration

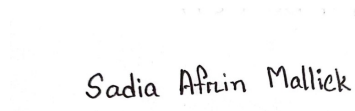
It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

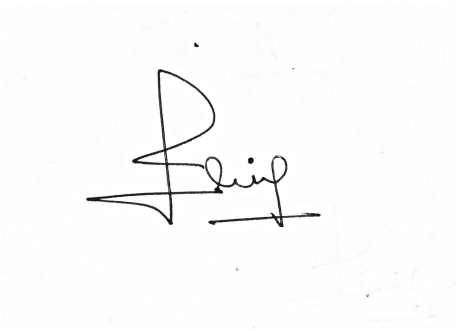
Student's Full Name & Signature:

Shadman Sakib Ayon
18101395

Samira Ishrat
18101398



Sadia Afrin Mallick
18101396



Prodip Chandra Das
18101115

Approval

The thesis titled "Sentiment Analysis on COVID-19 Tweets" submitted by

1. Shadman Sakib Ayon (18101395)
2. Samira Ishrat (18101398)
3. Sadia Afrin Mallick (18101396)
4. Prodip Chandra Das (18101115)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on September 20, 2022.

Examining Committee:

Supervisor:
(Member)

Faisal Bin Ashraf
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Dewan Ziaul Karim
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Co-ordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

The global spread of COVID-19, as well as the emergence of platforms as for many people a key source of information, has resulted in a wide range of reactions. But it is hard to keep up with this mass scenario. A significant number of individuals share their ideas and perspective on current events on social media, making it hard for a human to read and understand everything. There are a lot of information spreading through tweets. Using public comments available on Twitter, our study tries to do a sentiment analysis of the total conversation over COVID-19 in a document. We will try to improve the techniques and methods that were previously used in sentiment analysis. Our main focus is to look at tweets about COVID-19 from the previous year using natural language processing and neural network approaches. We have used a multiclass dataset and applied the same dataset to BOW, TF-IDF and One Hot Encoding. Furthermore, we tried to do a competitive analysis after training four different classifiers by applying these different pre-processing techniques in each classifier to find a better result. This way we tried to observe three different sentiment classes which are Negative, Neutral, and Positive in every methodology. However, we tried to generate a report of the best-performing combination of classifying algorithms and methods. Along the way, we tried to implement latest techniques to contributions on themes relating to Sentiment Analysis and compared the result with other techniques.

Dedication

This paper is dedicated to our educators and parents.

Acknowledgement

At the very beginning, we would want to convey our utmost appreciation and love to the Almighty for bestowing upon us the fortitude and calm necessary to complete our CSE courses and write this report on schedule.

Most importantly, this gives us the chance to extend our appreciation to our beloved supervisor, Mr. Faisal Bin Ashraf, for his immense guidance during this research work. During the process of organizing and developing this study work, he provided us with step-by-step oversight, support, guidance, encouragement, and inspiration, as well as constructive recommendations. Without these, we never would have finished our thesis. It is inconceivable that our thesis work could have been supervised and guided by a more qualified individual. We appreciate all his patience, his willingness to give his time so generously, supplying us with ideas, and offering all of the facilities that are required for the investigation.

Additionally, we would like to extend our gratitude to our Co-Supervisor Mr. Dewan Ziaul Karim who directly or indirectly has lent his hand in this venture, and our friends for encouraging, motivating, and believing in us all the time. We would like to use this chance to express our gratitude to our loved ones, beginning with our family for the unwavering love and support they have shown us throughout the years. We're grateful to our parents, who instilled a passion for science in us and encouraged us.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Natural language processing (NLP) is a branch of computer science, linguistics and artificial intelligence dealing with computer-human interaction. The process of designing a system which can create and process language as well as a person can is known as natural language processing. Text clarification is the process of categorizing a text turning to a collection of words. Natural language processing (NLP) is used to analyze text and assign specified tags or categories to it based on its context. The majority of this text data is unstructured, so classifying this data can be extremely useful. NLP is used to analyze sentiment, recognize topics, and detect languages. Predicting the emotion of tweets and movie reviews, as well as categorizing email as spam or not, are examples of text classification challenges. Deep learning algorithms are excelling in text categorization, producing state-of-the-art results on a variety of typical academic benchmark challenges. There are multiple approaches to classify a text. It can be divided into Rule-based System where a collection of basic linguistic principles is used to arrange texts into an orderly group. Using these customized linguistic rules, users construct a list of phrases that are categorized by categories. Machine-based classifiers are another option, as they learn to categorize data sets based on past observations. User data has both train and test data pre-labeled. It learns over time by accumulating categorization techniques from prior inputs. Machine-based classifiers employ a bag holding a word for feature extension. The final hybrid solution employs a rule-based system to construct a tag and machine learning for training and creating a rule. After that, the rule-based and machine-based rule lists are compared. If anything doesn't match the tags, a person can manually enhance it.

1.1.1 Motivation

Every day, the use of the internet and other technology grows exponentially. As a result, the use of social media is becoming more common. Most people with internet connections nowadays utilize a variety of online platforms such as Twitter, Facebook, Instagram, and YouTube. People can use such platforms to express themselves positively or negatively through postings, comments, conversations, and other means from the comfort of their own homes. During the outbreak, people are more likely to post comments and transmit news/tweets about COVID-19 on social media, both

positive and negative.

Here, thorough data pre-preprocessing of our dataset and implement several text classification models for sentiment analysis of covid-19 tweets to reach a better result is the major aim of this thesis. The motives for both of these goals will be discussed shortly.

1. Volume of Data: A lot is going on around us in covid situation and it is not possible to keep up to date with the current scenario but classifying massive volumes of textual content helps standardize platforms, improves search efficiency and relevance, and simplifies navigation, all of which contribute to a better user experience. Growing numbers of people have access to computers and the Internet, which has led to a rise in the quantity of information that can be found online. Keeping up with the ever-increasing deluge of data necessitates the classification of text and questionnaire information.

3. No Previous Investigation: To the author's knowledge, the production of text classifiers utilizing text classification approaches has not been extensively examined to determine which one should perform well in various contexts, and so further research is required. The comparison of several models in this paper will clear up any doubt. As a result, newcomers will not need to conduct any study before beginning their research.

4. Diversity of Implementation: Our major goal in this paper is to determine the overall review of the covid-19 influence by analyzing the feelings of covid-19 tweets. However, to finish our assignment, the comments and opinions must be organized into classes. These techniques may be used to detect any type of review, whether it's a virus or a calamity. The approach for determining the overall view is the same. As we are expecting to find the most efficient method for this task, this can also be useful in future studies.
With these constraints in mind, this thesis claims that it would be interesting to evaluate the performance of models like LSTM, RNN, RoBERTa , Naive Bayes Classifier to determine which model provides the most accuracy. This would be one approach to looking at the effect of words with clear meanings on the performance of these otherwise highly effective new models and procedures. More broadly, the techniques provided in this thesis will make it easy to leverage good text classification for subsequent studies without the need for analysis. It will lower the cost of analysis.

1.2 What is Sentiments analysis?

Traditionally, analyzing a piece of text to determine if it has positive or negative sentiment, as well as other polarities is called sentiment analysis. It is also known opinion mining which is the comparative analysis of how individuals feel about a certain thing. This feeling includes emotion, opinion, and attitude towards an entity. This entity might stand in for people, things, or things happening. All these topics are covered by reviews on social media. The terms Sentiments Analysis and Opinion Mining can both be used in the same sentence. They both convey the same notion.

However, some researchers disagree with this point. Sentiment Analysis working mechanism includes of finding the sentiments represented in a text. After finding the sentiment it completes the analysis part. Whereas Opinion Mining extracts and examines people's opinions about an entity. So, in this paper, we are focusing on Sentiment analysis particularly. Sentiment analysis aims to gather viewpoints, determines the feelings they convey, lastly, categorize their polarity into different sections. An example is given in Fig:1.1 of the sentiment analysis of product reviews for better understanding.

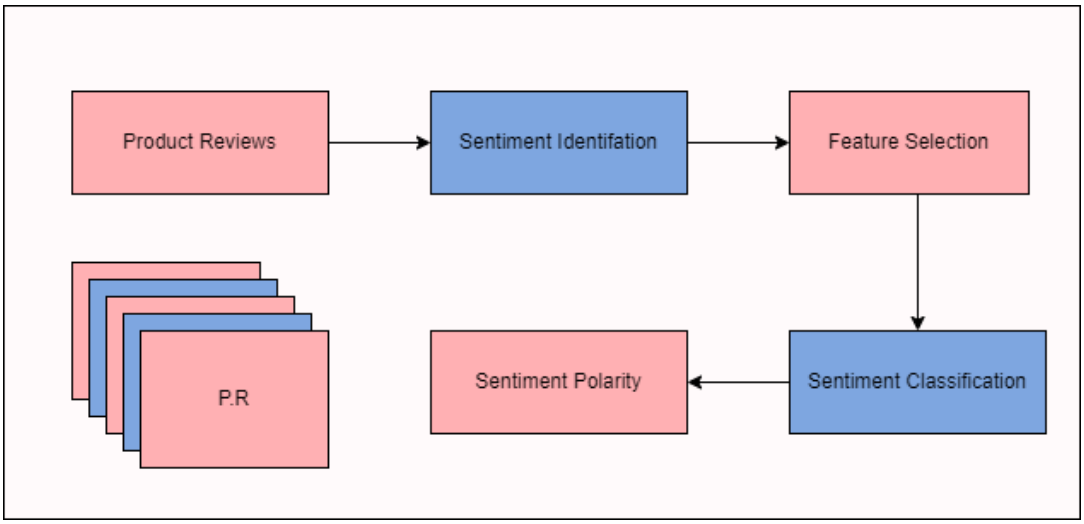


Figure 1.1: Product reviews Sentiment analysis

1.3 Sentiment Analysis Levels

Sentiment analysis uses three primary categorization levels. They are stated in the diagram of Fig 1.2.

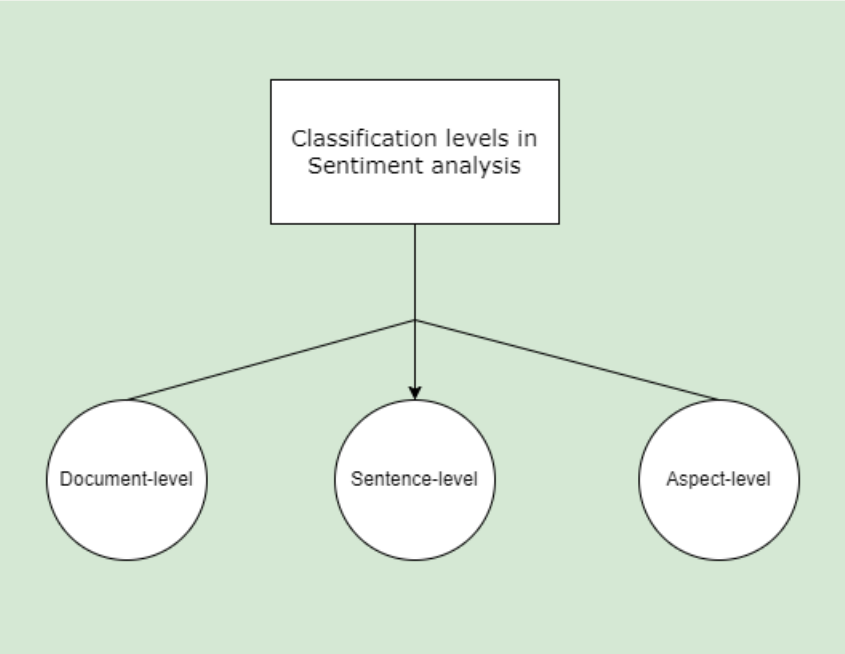


Figure 1.2: The classification levels in Sentiment analysis

With the use of document-level Sentiment Analysis, a document's opinion may be categorized as communicating a positive or negative view or sentiment. If we discuss a single subject, it views the entire article as a simple information unit. A good view is represented by the word favorable, whereas a negative one by the word unfavorable. Sentiment analysis at the phrase level seeks to classify the underlying emotional tone of written content. The first step is to establish if the statement under scrutiny is initially subjective or objective. This is the first phase. Because sentences are essentially little documents, conducting sentiment analysis at the sentence level does not substantially vary from conducting classification at the document level. Sentence level Sentiment analysis will evaluate whether a sentence reflects a favorable or unfavorable view if it is subjective. In many applications, classifying text in a document level does not provide the complementary details or perspectives on all aspects of the object. We must reach the aspect level to access these facts. The goal of aspect-level SA is to categorize the sentiment about the particular attributes of things.

1.4 How Can Sentiment Analysis Be Performed?

Three types of contemporary sentiment analysis techniques are categorized: knowledge-based, hybrid, and statistical. Here's how to perform sentiment analysis.

Knowledge-Based: This strategy categorized material based on words that convey emotion.

Statistical: To accurately detect sentiment, this method applies machine learning methods like deep learning and latent semantic analysis.

Hybrid: For accurate sentiment analysis, this method combines knowledge-based and statistical methodologies.

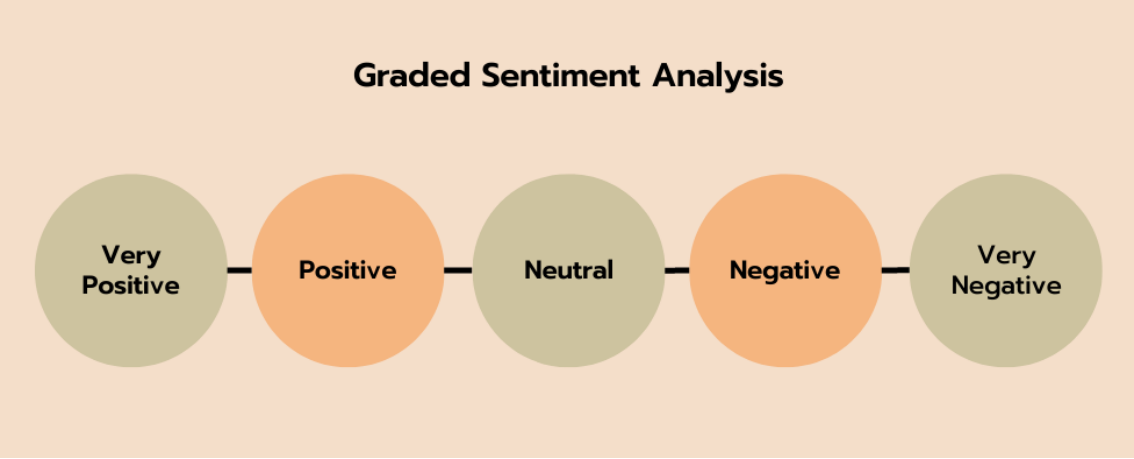
When sentiment analysis was first introduced, knowledge of machine learning, R, and Python was necessary. However, today's technology makes it possible to perform sentiment analysis with little to no technical expertise.

1.5 Types of Sentiment Analysis

Sentiment analysis concentrates on a text's polarity but extends beyond polarity to identify certain moods and emotions, urgency, and also intents. Positive, neutral, and negative are the polarities of a text. Emotions are considered as angry, happy, and sad. Furthermore, intentions can be categorized as interested and uninterested. The most popular types of sentiment analysis types are given below.

1.5.1 Graded Sentiment Analysis

You could think about increasing your polarity levels as necessary.
A generally used categorization could be,



Typically referred to as graded sentiment analysis, this can be used to assess 5-star evaluations, for example:

From the right we can consider Very negative as one star and sequentially moving towards left we can increase the amount of stars.

1.5.2 Emotion Recognition

Sometimes you might not be able to classify data in a manner of graded polarity, in that case you need to consider beyond polarity. And for that emotion recognition sentiment analysis is used for finding emotions like joy, disappointment, rage, and sorrow. In other words happiness and sadness. Lexicons or sophisticated machine learning algorithms are commonly used in emotion detection.

1.5.3 Aspect-based Sentiment Analysis

For achieving details, you need to dig down with the Aspect-based Sentiment Analysis. If the work is to find out some specific qualities or traits which people address in a favorable, neutral and unfavorable way then Aspect based sentiment analysis plays a vital role.

1.5.4 Multilingual sentiment analysis

Using a linguistic classifier, one could recognize the speech in messages automatically easily. The next step is to train a customized sentiment analysis model to classify texts according to the voice you've chosen. Language analysis of sentiment could be difficult. The preparation requires a significant investment of both time and effort. These materials may be found on the net, while some require the creation and can only be used if you know to code.

1.5.5 Intent Analysis

The intent analysis aids in determining if a buyer is intending to buy something or is only looking around. You may follow a consumer and promote to them if they are prepared to make a deal. You might just save time and money by avoiding advertising to customers who aren't ready to purchase. The intent analysis will do this for you.

1.6 Sentiment Classification Techniques

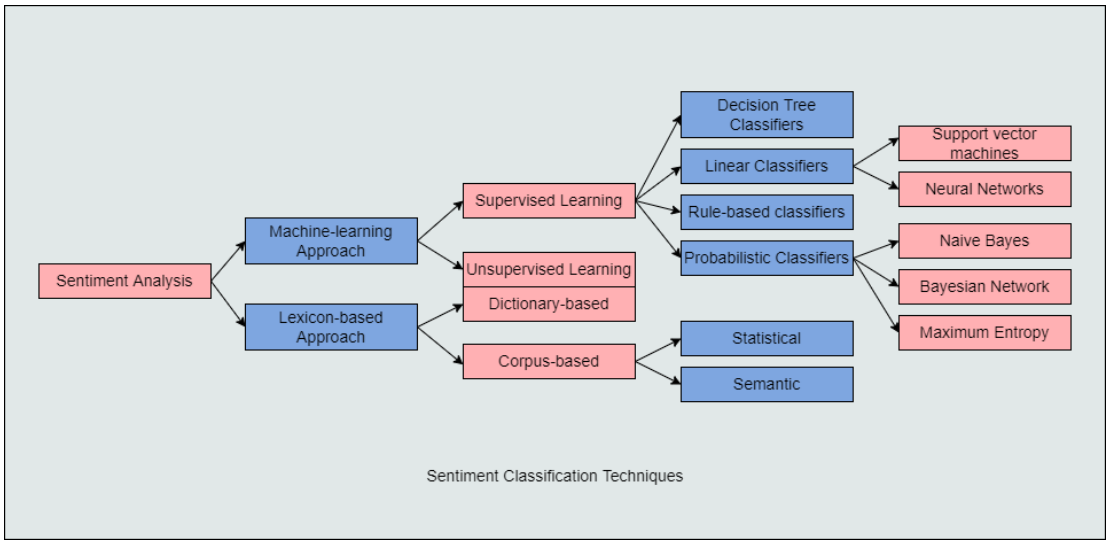


Figure 1.3: Techniques of Sentiment Classification

This diagram gives a general idea of the techniques of Sentiment analysis. Although, the diagram is made by using popular sentiment classification techniques. Sentiment analysis has two approaches. One is Machine-learning based approach and another one is a Lexicon-based approach. Unsupervised learning and supervised learning are further divisions of the machine-learning technique. Also, Neural Networks

and Naïve Bayes techniques are mostly used as popular sentiment classification techniques which are ML approaches.

1.7 Sentiments Analysis of Tweets

Tweet analysis might yield a substantial amount of sentiment data. These statistics help us realize how individuals feel about a range of issues.

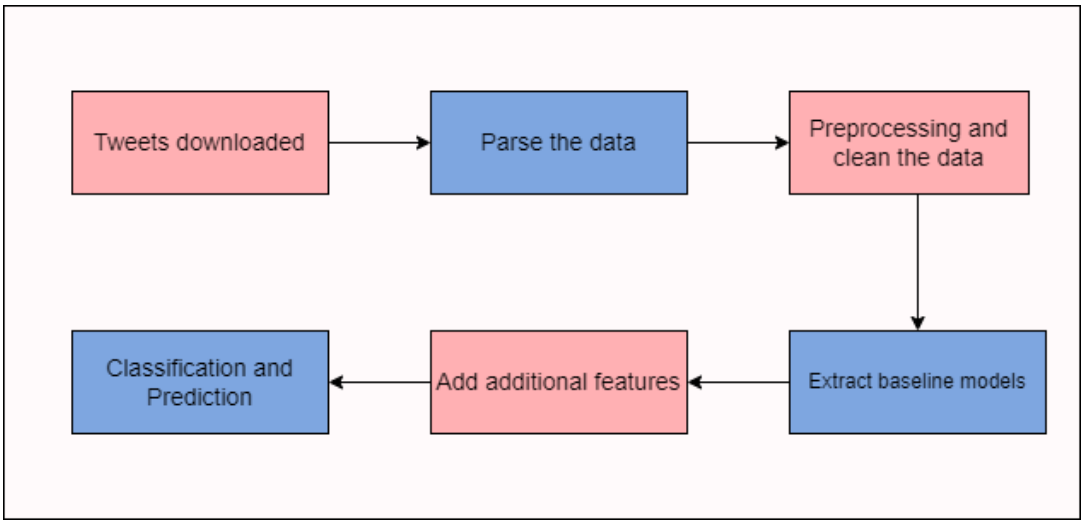


Figure 1.4: Sentiment Analysis of tweets

Modeling the data becomes challenging since they include both valuable and unusual characters and data. Preprocessing can solve this problem. Implementing desired models and adding additional features leads us to classification and prediction.

1.8 Problem Statement

With a growing popularity of social media in our daily life, it can be said that the current situation of millions of people can be comprehended by analyzing the mass data generated by the social media. As we already know covid-19 is a very significant event from the previous year and the information shared on social media about this matter is also huge. But from this mass information it is not possible for us to get any classified result. Also we can not determine anything from a random information. This problem provides us a great opportunity to use data mining and techniques of text classification and sentiment analysis to analyze the mass amount of data. Recently, one of the most popular social networks, Twitter allows to go through its API to get access to the data of users for free. From Medhat et al.[1] we can see that many researchers have already proposed many models to use this type of resource to extract information. But it cannot be said that which model is more suitable for this type of certain dataset.

1.9 Research Objective

Our goal is to make a final classification of text containing the vital information from the main text, which will let people save their time and successfully analyze the sentiments of covid tweets and understand them within a short period. Also, this is for the betterment of people to acquire the overall knowledge about the tweets. In order to determine the author's opinion on a certain subject, product, or other entity, computers may use a procedure called sentiment analysis. The objectives of this research paper are given below.

- Understanding the techniques of the previous classification methods and modify it, focusing on Covid Tweets.
- Acquiring knowledge about how pre-processing can improve accuracy and which methods are suitable for further research.
- Extracting the critical information from the text by using our novel machine learning algorithms.
- Studying the use of different text classification methods and compare the performance of various traditional models such as LSTM, SVM, Naive Bayes classifier, etc.
- After comparing various text classification models, we will be able to see which models perform with more accuracy to solve this problem. Furthermore, we will be producing a more simplified version.

Chapter 2

Literature Review

Sentiment analysis examines other people's opinions and categorizes them as either positive, negative, or neutral. Sentiment analysis can assess a huge number of opinions fast, allowing for analysis of public opinion. Using the strengths of both LSTM and CNN, the Bi-LSTM+CNN model is an attention-based hybrid that employs a second attention mechanism. For the purpose of gauging how well the proposed model works, we used information culled from the IMDB. The LSTM is an improved form of the RNN, with a gating system consisting of an input gate, a forget gate, and an output gate [2]. The algorithm extracts characteristics from various positions in a text using a CNN and the characteristics obtained from the convolutional layer are utilized to extract contextual information using LSTM. The attention mechanism enhances the weight distribution and applies a bias alignment across the inputs to process sequences of varying lengths. They also employed weights that had been pre-trained with a huge corpus using Word2vec, ensuring that the model was more accurate. In this study, they combine the strengths of LSTM and CNN to create a novel approach to sentiment classification that uses a CNN module for feature extraction from text and a Bi-LSTM module equipped with an attention mechanism. The study's ultimate goal is to capture the intricate relationships between nearby words in order to improve classification accuracy. They also want to boost the number of classification labels available for multi-class prediction. The accuracy of the suggested model improved as they increase the time to train data and quantity of epochs. To address the problems of long-term dependability and data loss that come with increasing amounts of training data, this approach may be preferable to current approaches.

CLSTM (Convolutional Long Short-Term Memory) for text categorization was developed by researchers at The University of Hong Kong [3]. This model combines a convolutional neural network and a long short-term memory network into a single structure. Here they have compared this C-LSTM model with a large number of baseline model and came up with satisfactory result. Their objective was to combine CNN and LSTM to find a new approach which will provide a better result than CNN models and RNN models. In this model [3], CCN is used to create a list of higher-level phrase representations. Then, to get the phrase representations, load them into a LSTM recurrent neutral network. By using this new model, they were able to use both local features and global and temporal sentence semantics. It is also mentioned in the paper that tensor-based operations or tree-structured convolutions

can be used in place of traditional convolution for further improvement.

In this study [4], the keyword expansion strategy was used and investigate the use of text from reputable sources, such as news sites, to augment twitter content. They test their hypothesis using four conventional classifiers on two Twitter datasets. The experimental results on several classifiers demonstrate that the proposed feature enrichment technique may overcome the sparsity limitation of short text with improved classification performance. Feature enrichment may be accomplished in three main ways: Through the use of external sources, the number of words in the feature set may be increased based on the interconnections in the source material (especially microblogs), Enrichment based on the dictionary: A lexical approach including smoothing may be used to address the issue of sparseness in the word.According to Kamath and Caverlee (2011) [5] collocation-based enrichment occurs when "two or more words together generate a phrase that represents the way people express things." Specifically, they investigate the idea of text enrichment by looking at several sources that cover similar ground to the tweet classes. Each category is then used to get a set of keywords from the tweets themselves, after which reliable online resources are searched. They utilized the Sanders (2011) [4] tweets (benchmark) dataset. To address the semantic gap and sparsity issue in tweets, the suggested technique enriches the class data by adding a collection of words taken from the relevant web sites. The suggested enrichment method has the potential to solve the problem of uncommon phrases in tweets while also improving categorization performance.
This paper [6] teaches how to do sentiment analysis and how to classify sentiment using the Naive Bayes classification approach. During the first quarter of the Covid-19 epidemic, data was gathered via Twitter. Several nations' governments and Covid-19 were mentioned in tweets. The Covid-19 first quarter epidemic, this study attempts to gauge public confidence in government. The findings of the study are likely to be utilized as a guide for public policymakers as they develop future policies. The goal of this research was to investigate the trust of mass in government at the time of Covid-19 outbreak. The major goal of this research is to utilize Twitter sentiment analysis to determine public trust in government response actions during a epidemic. Other features like as website URLs, special symbols and usernames are removed throughout the pre-processing. The tweets are classified using Naive Bayes and Bayesian Classifier. Prior to categorizing the training data, the tweets need to be sorted to positive results, bad, and neutral tweets. In the final stage, evaluation is used to assess the correctness of the algorithm that was used. The evaluation's outcome can be utilized to draw conclusions.

In this study [7] , we employ deep learning methods to develop NLP models for the goal of classifying the vast quantity of public tweets on the COVID-19 epidemic. The primary contribution of this work is the differentiation of multiple machine learning and deep learning models for conducting sentiment analysis on a collection of tweets related to the COVID-19 epidemic. This dataset was basically made up of a collection of tweets obtained for a Kaggle competition. Later on, the tweets were sensitively divided into five classes. In order to maintain input tweets more effectively with algorithms, some preprocessing actions were performed. First, they proposed some trendy algorithms of machine learning for text classification, using

several algorithms and classifiers. They then employed three distinct deep learning algorithms in this research which are LSTM, MLP and BERT. Better than the Gaussian Naive Bayes model, the BERT model achieved an F1 of 71%. However, the Gaussian Naive Bayes model has the worst accuracy. This study looked into a number of methods for categorizing Covid-19 tweets into multiple classes. These two factors might provide useful information and allow for additional inferences to be drawn about certain periods and geographical regions throughout the world.

This study [8] aims to do a sentiment analysis of the overall discussion with regard to COVID19 using public opinions available on Twitter. In addition to common Natural Language Processing techniques such as Bag of Words, in this work, they have used an LSTM network to perform the sentiment analysis task in a dataset of tweets related to COVID19.Miglani offered the dataset for the text classification model's training and testing, in which a multi-label convolutional neural network text classifier was built using SpaCy's new TextCategorizer component. Such dataset contains a lot of tweets, with a certain amount of tester tweets. Apart from the tweet text, it has the username, the screen name, the location, the date, and the sentiment. A rating of "Extremely Negative," "Negative," "Neutral," "Positive," or "Extremely Positive" is assigned to it. The following subsections describe the methodology followed to obtain a sentiment classification model. This proposed model's working module is pretty simple, A tweet sample converted to a numeric representation. The class is transformed to a categorical binary representation, and the text is turned to an INTS vector. Then, after preprocessing, it compares the stored words to determine the probable feelings. The classifier model was created using a deep neural network with numerous layers. Their suggested model was trained with Keras, a Tensorflow library, and a second model with character embedding. After training, the model was evaluated using the test dataset. The performance of the character-embedded model was very poor. Tokenization and count vectorization were chosen as methods for processing the input because they help characterize the data in a manner that the model can be trained, and so the goal of the project was to develop a neural network. The model achieves a high accuracy of 0.709 after comparison. When compared to the original data model, the neural network takes longer to train. A future topic to consider is increasing the quantity of data gathered and, in certain situations, using the Twitter API in order to have more up-to-date information or that can be updated when the code is executed.

To perform this study [9], they have used various types of models, and classification methods to analyze sentiment on Twitter data. NLP tweets from the Kaggle Coronavirus competition were used for text classification. As far the data is carrying a sentiment class, there is no need to re-annotate it. Their main aim is to compare the above models and find out which model is more suitable and preferable. With the Kaggle competition data, the purpose is to categorize this data into different sentiment classes. Now, SVM, LSTM and BERT are three categorization algorithms that are compared. Each categorization algorithm has different input properties. The Bag of Words (BOW) and TF-IDF vector space models are used as VSM methods. LSTM and word embeddings are the second technique. Word2vec and Glove are two approaches of word embedding which are used by them in this paper. BOW (Bag of Words) and TF-IDF were also used. Last but not least, BERT

is a model that takes a deep learning approach and has achieved state-of-the-art results on many different natural language processing tasks. BERT employs a bidirectional deep representation that can be tweaked by including one more outside layer. According to the results of the tests, BERT had the best results. BERT produced superior classification results than the other two techniques.

In this research, [10] they offered TextConvoNet, a CNN-based architecture that extracts and captures both intra-sentence and inter-sentence n-gram characteristics in input text data. It employs a two-dimensional multi-scale convolutional operation on the input and employs an alternate strategy for input matrix representation. We conduct an experimental investigation on five text categorization datasets to assess TextConvoNet's performance. The outcomes are evaluated using a variety of performance indicators. TextConvoNet beats deep learning and cutting-edge machine learning models for text categorization, according to the findings of the experiments. Using n-gram information in between distinct phrases to improve text classification outcomes using a convolution operation is still an open research subject for all academics. In addition, the input matrix structure is worth considering, as it might be updated to perform multidimensional convolution. TextConvoNet is a novel CNN-based architecture for text categorization shown in this study. Unlike previous work, the proposed architecture extracts intra-sentence and inter-sentence n-gram characteristics from text input using a 2dimensional convolutional filter. To begin, the text input is represented as a paragraph level (multi-sentence) embedding matrix, which aids in the application of 2-dimensional convolutional filters. The extract features are then subjected to a series of convolutional filters. But they need to encode input in higher dimensions. In order for convolutional methods to be able to accurately represent many facets of textual material.

Chapter 3

Methodology

3.1 Data Collection

For sentiment analysis, we need a massive quantity of social media data relating to COVID-19. Here we could use a Python library and get authentication from Twitter to filter the tweets to process our dataset. But such prepared data was already available on Kaggle. So the dataset we used was proposed by Miglani[11]. The tweets were pulled from Twitter and tagged manually.

3.2 Data Analysis

This dataset consists of total 41,157 tweets related to covid-19 from all over the world. The following attributes were prepared per tweet.

- UserName: Username was encoded to avoid privacy concerns.
- ScreenName: Screen name was encoded to avoid privacy concerns.
- Location: Name of the country from which the tweet was published.
- TweetAt: Day, month and year of the published tweet.
- OriginalTweet: Represent the exact tweet
- Sentiment: Sentiment classes, which represent the level of the tweet.

Here, the five sentiment class has been used and they are 'Extremely Negative', 'Negative', 'Neutral', 'Positive', and 'Extremely Positive'. The data set was split into a training set and a test set.

As this dataset contains tweets from all over the world, so we can represent the data in a plot to see the number of tweets based on their location; which is shown in Fig-3.2. From this, we can say that most of the tweets are from London, United States and London, England.

As the tweets in this dataset has been classified by five sentiment classes, From Fig-3.3, we can observe how often and where these classes appear in the training

dataset. It shows us that most of the tweets are placed in 'Neutral' category and the rate of 'Positive' tweet is greater than 'Negative' or 'Extremely Negative' tweet.

By using the original tweet and sentiment column from the dataset, we can see the most commonly used words in different sentiment classes in Fig-3.4. This will help us to visualize the type of words commonly used on positive, extremely positive, negative and extremely negative type of tweets.

3.3 Data Preprocessing

Before we start data preprocessing, we selected the features which we will need and simplify the dataset. We focused on the OriginalTweet and Sentiment columns in both the train dataset and test dataset as shown in Fig-3.1

	OriginalTweet	Sentiment
0	@MeNyrbie @Phil_Gahan @Chrisitv http://t.co/i...	Neutral
1	advice Talk to your neighbours family to excha....	Positive
2	Coronavirus Australia: Woolsworths to give elde...	Positive
3	My food is not the only one which us emp...	Positive
4	Me, ready to go supermarket during the #COV...	Extremely Nagative

Table 3.1: Example of interested features in dataset

The steps we used for data preprocessing is given below.

• Checking any null value: we checked for null values in both train dataset and test dataset and neither consists any null value.

• Drop duplicate and Nan value containing rows

• Class encoding and merging: Five sentiment classes were converted in three sentiment classes by merging 'Positive' and 'Extremely Positive' into 'Positive' class and merging 'Negative' and 'Extremely Negative' into 'Negative' class. Then each class was represented with an index. 'Negative' = 0, 'Neutral' = 1, 'Positive' = 2.

• Stop-word removal: In English words like a, an, the, as, in, on, punctuation symbols etc. and many more frequently used words are considered as stop-words. We removed stop-words from OriginalTweet by using nltk stopword list.

• Lower case conversion: Converting all the terms to lower case helps us to minimize the vocabulary size.
As our target was to apply several models for sentiment analysis, we followed different ways for a text representation.

3.4 Data Preprocessing Techniques

3.4.1 Bag of words:

Using natural language processing, the Bag of Word approach creates fixed-length vectors that track the frequency of each word in a given text. This process is referred to as vectorization. Every pixel in a picture is numbered during image processing in order to retrieve data from the image. In essence, a visual input cannot be understood by the system. It is transformed into a collection of numbers to make it more comprehensible. The natural language also has to be translated into numbers in order to be understood. And this method is known as vectorization. A vectorization method is bag of words. The text is referred to as a "bag" of words since it lacks information about the pattern or terms that have been deleted. The methodology primarily concentrates on whether or not the terms used in a text are recognized, rather than on their specific placement. In light of a few phrases, it will count the number of times each word appeared there and assign it a number.

When modeling text data using machine learning techniques, the bag-of-words model is one approach to encode it. This is the most basic way for converting texts into fixed-length vectors based on a word frequency. A text is seen as a bag of words containing itself, with grammar ignored. Although this feature generating approach is praised for its simplicity, it falls short when it comes to determining sentence contexts. Thus, the bag of words model was created by using CountVectorizer.

The model divides each word in the sentence first. The punctuation will all be removed. All capital words will also be changed to lowercase. Then a term's occurrences inside a single sentence will be tallied. For instance, if there are three sentences, the first one has three words, the second has five, and the third has eight. There are a total of 16 words in the bag. Afterward, the model assigns them a number based on whether or not that particular word was in the phrase. Each sentence gets its own vector. And it is the main concept that drives the bag of words. It basically employs the n-gram technique, which requires n numbers of words in order to keep away from a similar meaning misunderstanding. When two phrases with distinct meanings contain words that are similar, this happens. For instance, "This day is good. I won't miss it for anything" and "This is not good to me" both share the words "this" and "is not good". It might suggest that both have a bad significance. The model can provide a distinct result by taking successively 2 or more words at a time. But there is a problem. The length of the matrix and vector will be extremely long if word variation is really high. The computation will take significantly longer to complete. Otherwise, it is a pretty basic vectorization and tokenization approach.

3.4.2 One Hot Encoding:

One Hot Encoding is a method for transforming categorical data into a format that may be used to train machine learning algorithms for more accurate prediction. While certain machine learning algorithms, such as a decision tree, may directly operate on categorical data, the vast majority of these methods need all input and output variables to be numbers or numeric values. This necessitates the conver-

sion of all categorized information to numeric values. One hot encoding is a way to transform data in order to optimize its prediction by an algorithm. One-hot works by creating a new column for each category and giving each of those columns a binary value of 1 or 0. It uses a binary vector for each integer value. Using a one-hot encoding helps when the data being encoded have no connections to one another. Number order is considered important by machine learning algorithms. In a nutshell, the number of observable classes or variables determines the length of the vector generated by this approach.

3.4.3 Text Vectorization:

In natural language processing (NLP), word embedding is a method that transforms phrases or words from a lexicon into a matching vector of real numbers. These real numbers have the potential to be utilized in the process of determining word predictions as well as word similarity and semantics. Words are converted into numbers by a technique known as vectorization. Each word was converted into a token. Then a unique int value was associated with each token. In this way, every tweet was transformed into a vector of int using the token's index.

3.4.4 TF-IDF:

Full form of it is Term Frequency Inverse Document Frequency, which is one of the most fundamental vectorization methods. We are able to assess a word's degree of relevance inside a document with the use of the analytical measure TF-IDF. Regularity of Application For "Inverse Document Frequency," you might use the acronym TF-IDF. Useful for both NLP (which utilizes machine learning to assign numerical values to words in a language) and automated text analysis, this tool is indispensable (NLP). When TF-IDF was initially created, it was used to find and retrieve information from documents. In actual use, it increases in comparison to the frequency of a term and decreases in direct proportion to the frequency of documents that include the same phrase. Against assess the originality of a word, the occurrences in a document is compared to the quantity of papers in which it appears. As a result, despite their frequent appearances, words like this, what, and if are given a low score since they are rarely used in that specific text, despite their many appearances.

In TF-IDF, there are two terms. By measuring a word's frequency, TF enables us to ascertain how frequently it occurs in a document. The quantity of words used is calculated by dividing each one by its frequency. The second term, an IDF calculation takes the logarithm of the number of occurrences of a word in a set of documents and divides that result by the total number of reports that include those terms.

The term "term frequency" tracks how frequently a phrase appears in a text. Because the length of each piece of writing varies, there is a good chance that a particular word will show up a great deal more frequently in longer texts than in brief ones. This is because longer texts contain more words. In order to normalize the data, the

recorded frequency of each file is divided by the total amount of words present in the text. This is based on an approach that takes into account both word frequency and word importance. For each given piece of text, we can get the Term frequency ratio (TF ratio) of a particular word by dividing the number of occurrences of that word by the total number of words in the text.

TF = (How many times a word or phrase is used in a document)/ (Total number of words included inside the document)

Also, Inverse Document Frequency (IDF) approach can be used to determine a phrase's significance. All words are given the same weight and value when determining the TF. However, it is common knowledge that words like "is," "of," and "that" might be used frequently but have little actual meaning. In order to increase the frequency of the unusual terms while decreasing the frequency of the phrases. Prepositions or pronouns have little meaning in a sentence but are regularly employed for grammatical considerations, and there are also terms that are insignificant to the machine. IDF offers a solution to this particular issue.

IDF =Log [(#Number of documents)/(Number of papers with the term)]

For illustration, consider the following three sentences: "Good day," "Bad day," and "Day bad good." Finding the number of times the terms Good, Bad, and Day are used in each phrase and dividing that number by the total amount of words in each sentence will get the term frequency (TF) of Good, Bad, and Day in three sentences. For the IDF, we take a log of the total number of sentences, which in our instance is 3, and divide it by the number of sentences that include the particular word. Similar to the Good, it will be log (3/3) as there are a total of 3 sentences and the word good appears in each of them. We will now multiply the TF by IDF to get the result.

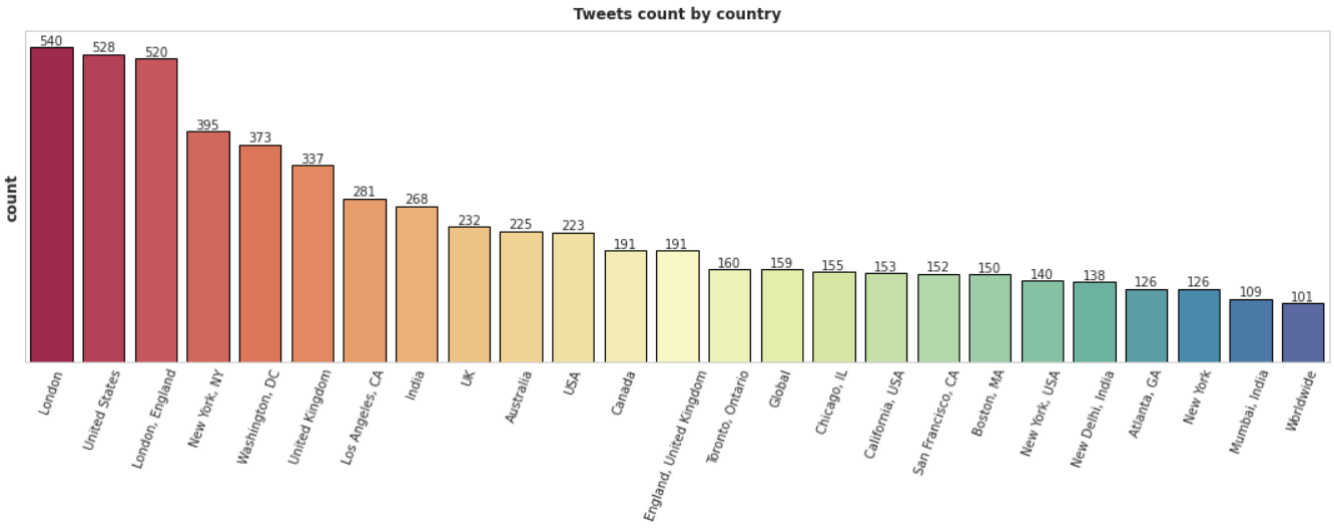


Figure 3.1: Tweet count by location

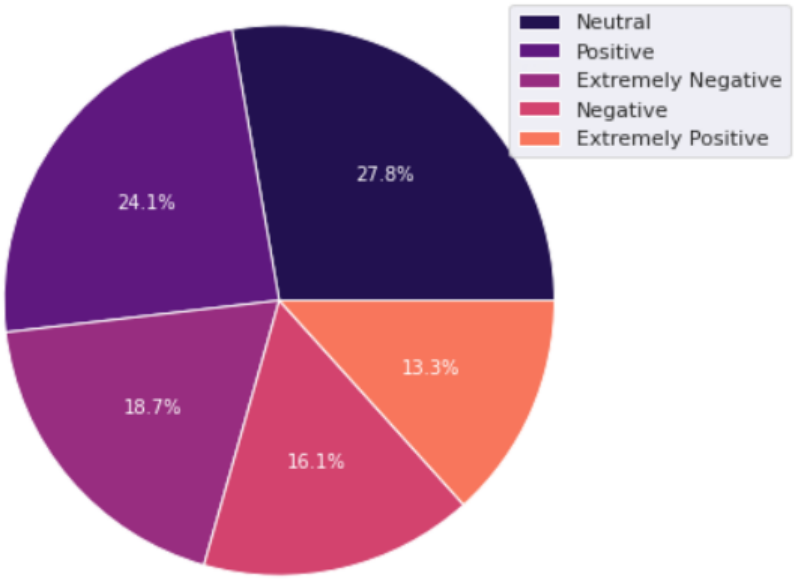


Figure 3.2: Distribution of five sentiment classes in the train dataset

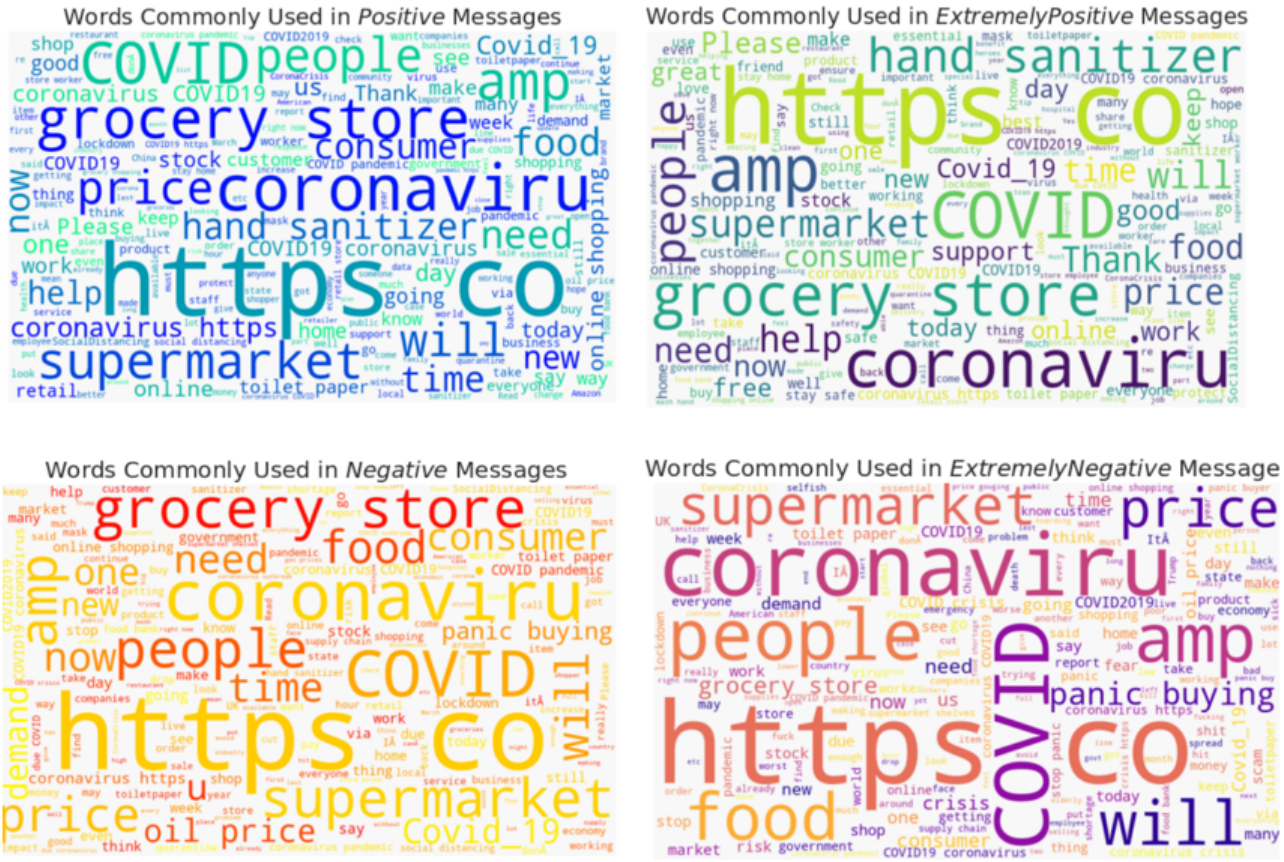


Figure 3.3: Word cloud for the sentiment class: positive, extremely positive, negative and extremely negative

Chapter 4

Evaluate the Model Performances

By preprocessing the dataset, we got our numeric representation of the OriginalTweet. After that, we implemented different models to get the proper result in sentiment analysis.

4.0.1 LSTM:

Recurrent Neural Network follows a few steps to perform its operation where output coming from the preceding step is taken as input in the next step. When the input gap is significant, RNNs made up of sigma cells or tanh cells are incapable of learning the pertinent information from the input data. Long short-term memory RNNs are a special case of RNNs. Since the same operations are performed in both the input and hidden layers, the same parameters must be utilized for each input in order to create an output. Multiplicative gate units control access to the cells and can be programmed to allow access as necessary. Tanh is used to hold on to the information in layers and this works as a recurring module. To obtain a sense of how the network functions, as well as the associated learning techniques, there is decent documentation available.

Juergen Schmidhuber's mainly came with the idea of LSTM. LSTM is constructed expressly to label the problem of long-term dependency. A recurrent neural network is built from a set of neural network modules that are designed to interact with one another on a regular basis[12]. It is their natural tendency to recall knowledge for extended periods of time. RNNs are incapable of linking the pertinent information when the distance between the important input data is wide. This is where LSTM becomes handy. In order to function, LSTMs rely on the cell state, which is represented as a horizontal line that pierces the image's peak. The cell is in the same condition as a conveyor belt. It goes all the way through the chain with just a few small linear exchanges. RNN's biggest drawback is vanishing gradient problem, but LSTM solves this problem. In order to stimulate the desired consequence from the error gradient using many gates change on each time step of the learning procedure, LSTMs address the issue by utilizing an original additive gradient structure that includes direct ties to the forget gate signals[12]. Data may easily and unalteredly travel through it. Protecting and regulating the cell state is the job of LSTM's three gates. Input, output, and a forget gate are the three main components of an LSTM. But some LSTMs are also built without a forget gate.

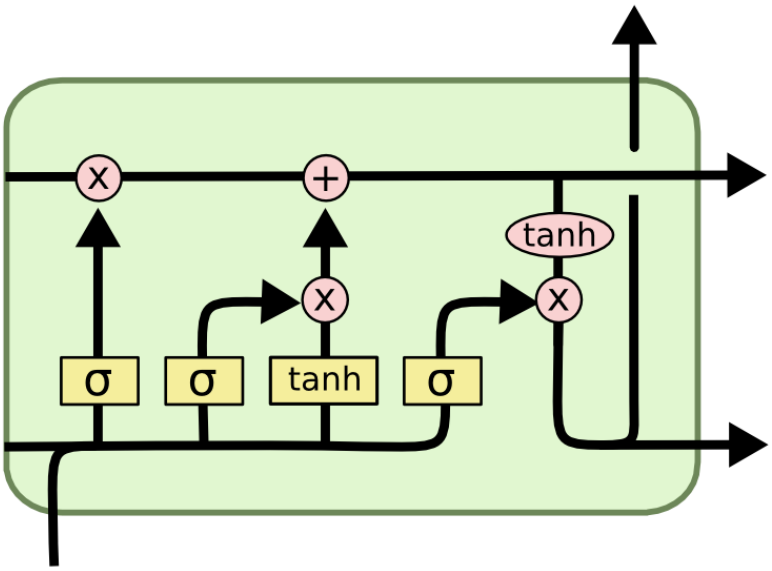


Figure 4.1: LSTM 1

To make tiny changes to data as it moves through cell states, LSTM employs simple addition or multiplication. This is how LSTM outperforms RNNs by selectively forgetting and remembering things.

First of all, the information has to be picked which we are going to get rid of. This is done using the forget gate that is situated in the sigmoid layer. The values of ht and xt generate a number in the middle of 0 and 1. Number 1 is identified as the information that should be completely stored and 0 is identified as the information should be completely forgotten.

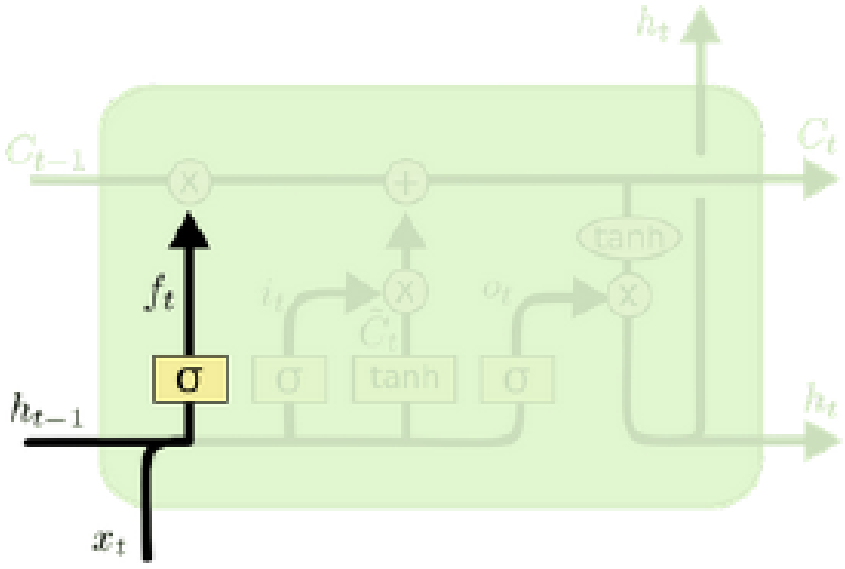


Figure 4.2: LSTM 2

$$f_t = \sigma\left(W_{f\cdot}\left[h_{t-1}, x_t\right] + b_f\right) \tag{4.1}$$

The second stage involves determining what supplementary data may be retained in the cell state as a result of the first stage's choice. There are 2 elements in this stage. A sigmoid layer firstly decides which values should be altered, also known as the "input layer gate". After that, a layer of tanh produces a vector of latest values, indicated as Ct. Using this, the system's status might be updated.

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right) \\ C &= \tanh\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_C\right) \end{aligned} \tag{4.2}$$

Moving from Ct-1 in the previous stage to the new cell state Ct is the next step. Now, considering the last decisions made, we need to put them to practice.

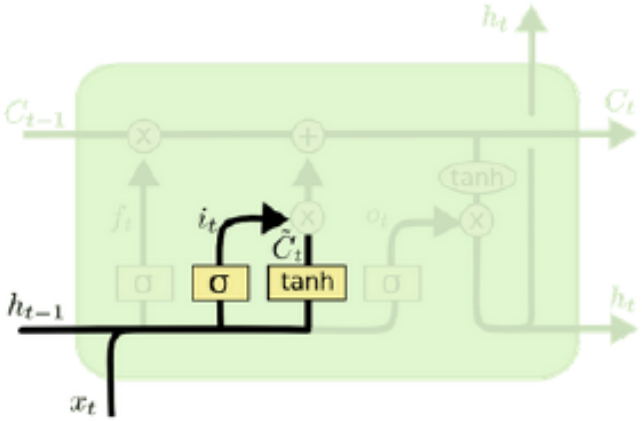


Figure 4.3: LSTM 3

Then we duplicate the past stage, by not remembering the items we previously admitted to ignore. Consequently, we transfer it as Ct. This is labeled as new values, that we chose to switch in every state value.

According to the model of this language, in this position the information about the old subject's gender is to be removed and alter it with updated information as we chose in the old stages.

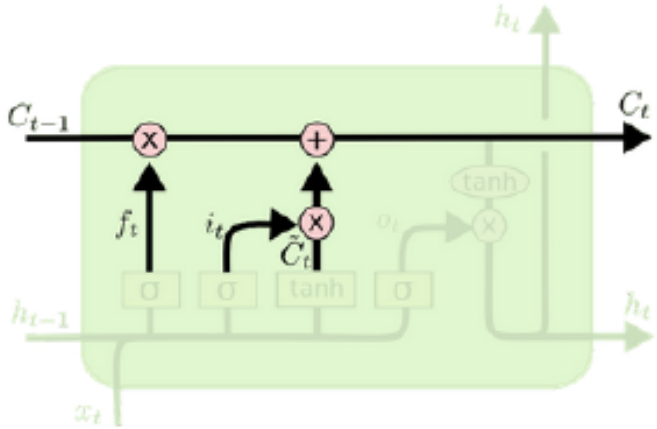


Figure 4.4: LSTM 4

$$C_t = f_t * C_{t-1} + i_t * C_t^r \tag{4.3}$$

Finally, the production must be chosen. The result will come on the basis of the current cell status, where it will be a purified version of the information. At first, we pass a sigmoid layer. This layer confirms that we solely get the output parts of the cell gate which is chosen for output.

According to the model, since it witnessed a thread, it may want to output data which is connected to verb in the circumstance that is what is going to happen afterwards.

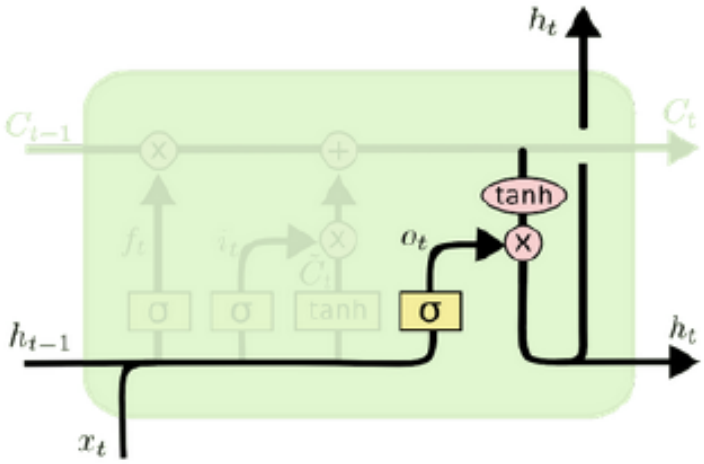


Figure 4.5: LSTM 5

$$\begin{aligned} o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \tag{4.4}$$

4.0.2 Naive Bayes classifier:

Naive Bayes classifiers are algorithms that use the Bayes theorem to categorize data. Classifiers based on the Naive Bayes algorithm place heavy (or naive) emphasis on the independence of data point properties. Some typical applications of naive Bayes classifiers are text analysis, spam filters, and medical diagnosis [13]. Since these classifiers are straightforward to design, they see an extensive application in machine learning. When it comes to constructing quick machine learning models that can quickly predict outcomes, of all the algorithms, the Naive Bayes Classifier is one of the easiest to use and most effective. To classify data, Naive Bayes calculates the likelihood in the given data point which refers to a specific class rather than the actual label. Unlike other Machine Learning algorithms, which can often handle both Regression and Classification tasks, this one is exclusively suitable for Classification jobs. Since its underlying assumptions are so unlikely to hold in the real world, the Naive Bayes algorithm is often mocked as simplistic. It multiplies the possibilities of its constituent parts using conditional probability. This indicates that, given the class variable, the algorithm presupposes the appearance or absence of a class feature that is entirely unrelated to any other quality. You may think of Naive Bayes as a subset of Machine Learning (ML), which might help you grasp how the two concepts are connected. The two types are known as Supervised learning and Unsupervised learning. Furthermore, these learnings are divided into further

several categories. Now, supervised learning is spitted into two parts which are Classification and Regression. Classification is where you'll find the naive Bayes method.

The Naive Bayes classifier has the following salient features:
1. The Naive Bayes technique is a method for solving classification issues that makes use of Bayes theory and a supervised learning strategy.

2. Also, this classifier is a significant factor in determining your creditworthiness.

3. It has a place in the categorization of health-related data.

4. Since the Naive Bayes Classifier is so open to new information, it can be utilized for instantaneous forecasts.

5. It is most often used to classify text when there is a large training dataset.

6. The Naive Bayes Classifier is a straightforward classification method that still possesses considerable power which may be used to rapidly construct accurate and predictive models for machine learning.

7. It makes predictions based on the object's probabilities because it is a probabilistic classifier.

8. Among the many applications of the Naive Bayes Algorithm are filtering spam, Sentiment analysis, and categorizing publications.

What is Bayes's Theorem?

The Bayes theorem, sometimes referred to as the Bayes Rule, is a mathematical formula that determines how likely a hypothesis is to be correct when given only some early evidence. The conditional probability is what makes this outcome certain. The following formula denotes the Bayes theorem in Fig-4.6:

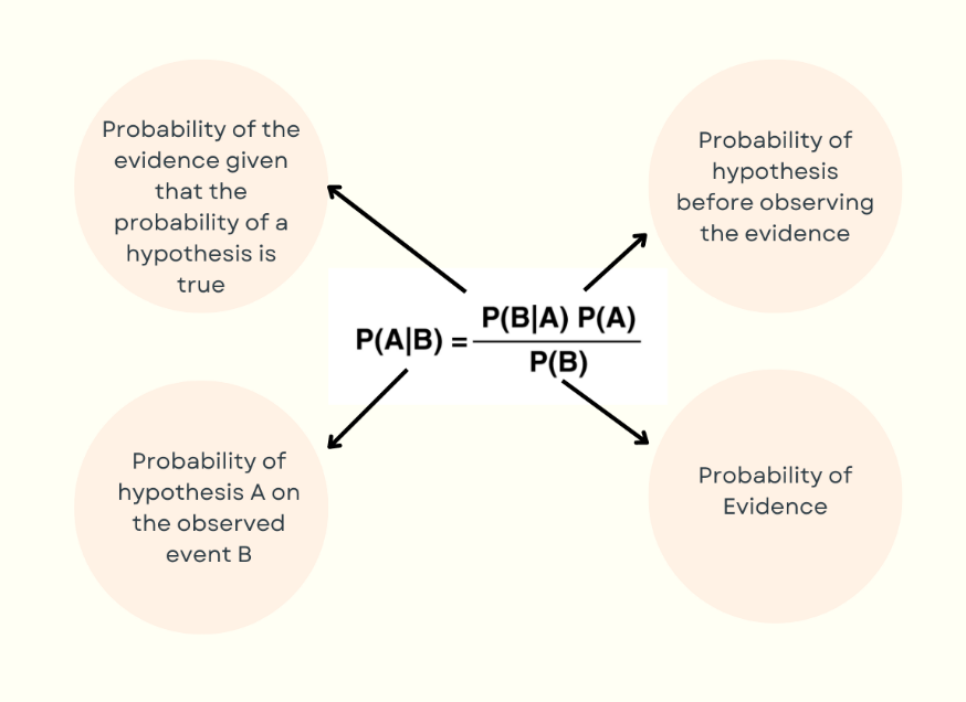


Figure 4.6: Naive Bayes 1

Variations on Naive Bayes Model

For improving studies, Naive Bayes Classifier is divided into three models. They are as follows:

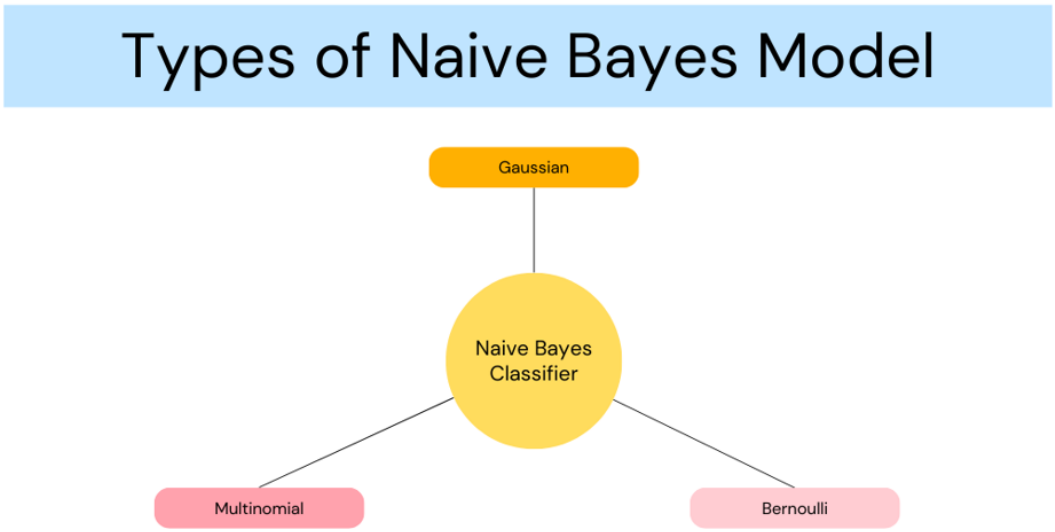


Figure 4.7: Types of Naive Bayes

Gaussian

Gaussian model assumes key attributes follow a normal distribution. In the event that the model parameters are continuous, then the values that are projected are presumed to be drawn from a Gaussian distribution

Multinomial

When dealing with data that follows a multinomial distribution, the Multinomial Naive Bayes classifier is the method of choice. Its primary application is in resolving issues about the classification of documents; this implies determining which category a specific document fits, such as education, politics, sports, etc. The classifier bases its predictions on the frequency with which words appear.

Bernoulli

The Bernoulli classifier performs its function in a manner that is analogous to that of a Multinomial classifier; however, the explanatory variables, in this case, are the independent Boolean variables. For example, if a particular term appears or does not appear in a given text. Additionally well-known for its use in document categorization problems is this model.

4.0.3 SVM:

Support Vector Machine is the full abbreviation for SVM[14]. Among the popular algorithms used in ML, SVM is one. This is a supervised learning algorithm used in regression as well as classification. SVM is used in a wide range of operations and detecting outliers such as handwriting identification, detection of intrusion, facial recognition, categorisation of emails, categorization of genes, websites etc. This algorithm can be used in both nonlinear as well as linear data. As we are doing text classification, we selected SVM because it can handle text features.

This approach is best suited for a limited number of data. The basic notion behind this strategy is that a hyper-plane should fit optimally between two input corpus groups. The vectors closest to a given hyperplane are called "support vectors," and deleting one of them will cause the other vectors' positions to shift. The target of SVM method is to find out the finest route or decisions with keeping in mind that it has restrictions of classification. The data set offered must be gone through classification in order to give it a simple form so that it can be worked through extended data points in the planned way with prolonged term. This creates the opportunity to just place the latest data point in the appropriate category in future. This algorithm designs the most effective border or line for classifying n-dimensional space[14].

For creating the hyperplane, it selects the most extreme points or vectors. The vectors or data points that are most near the hyper-plane. These cases are known as support vectors. The number of features determines the hyper-plane's size. When the input number is 2, then the hyper-plane is represented as a line. But when the input number is 3, it becomes a 2D plane. Imagining something with more than three features gets challenging.

So, we can see that SVM can be divided mainly into two categories. First one is Linear and the second one is Non-Linear. Linear SVM can be separated using a single line which means a Two categories can be used to categorize a dataset. On the other hand, Non-Linear SVM can not be separated using only one straight line which means dataset cannot be classified using a line. Now, let us see the most

appropriate hyper-plane generated from a dataset.

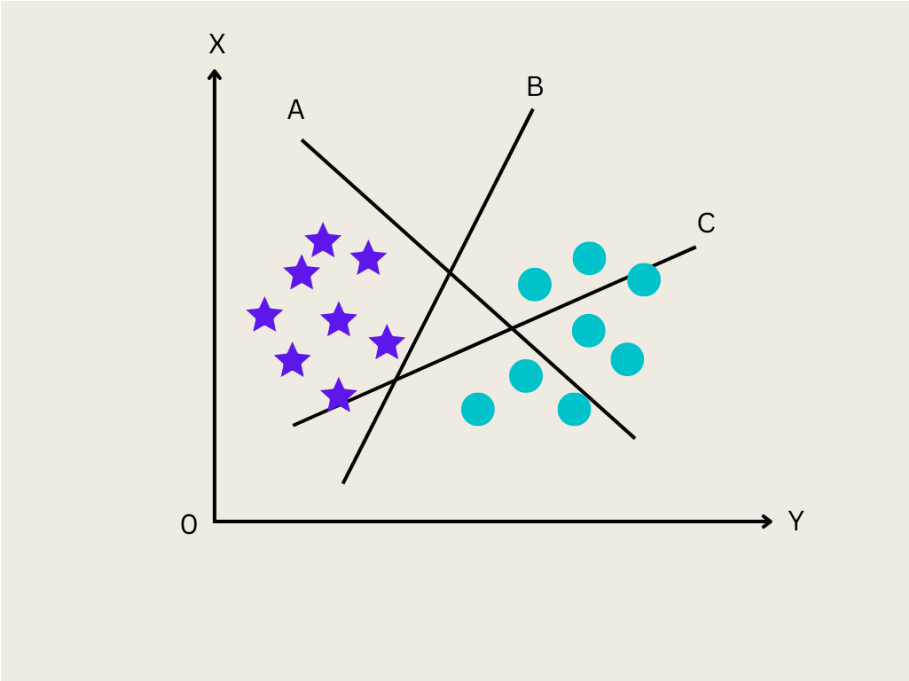


Figure 4.8: SVM 1

The graph shown above, can be classified using Linear SVM technique which will solve it in a progressing way. The finest way to select a hyperplane is when it shows the greatest distance or difference between the two groups using a line.

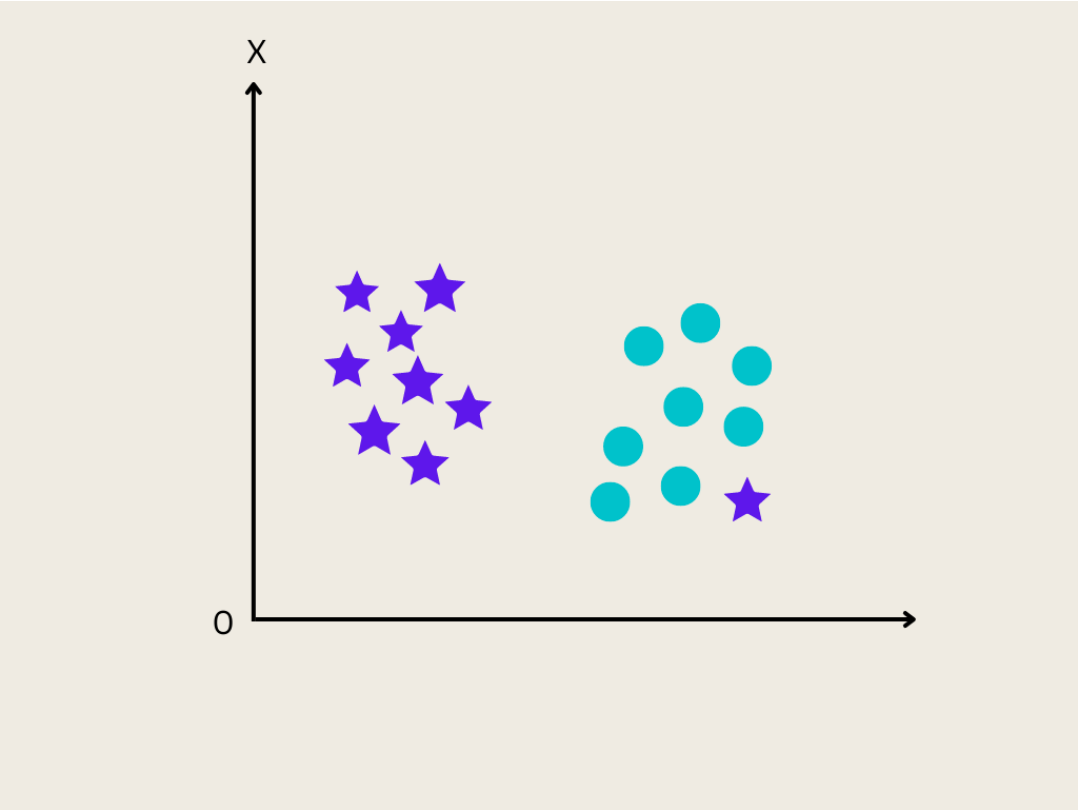


Figure 4.9: SVM 2

Furthermore, this picture states another type where it is not fully categorized. This category shows that when the line increases, the odd ones or exceptions are neglected. In this case, it determines the maximum margin like it has in prior data sets, and it also puts on a fine every time a point exceeds the limit.

When there is a very unorganized clustered dataset then the dataset needs to be handled using Non-linear SVM. For this picture in Fig-4.10,

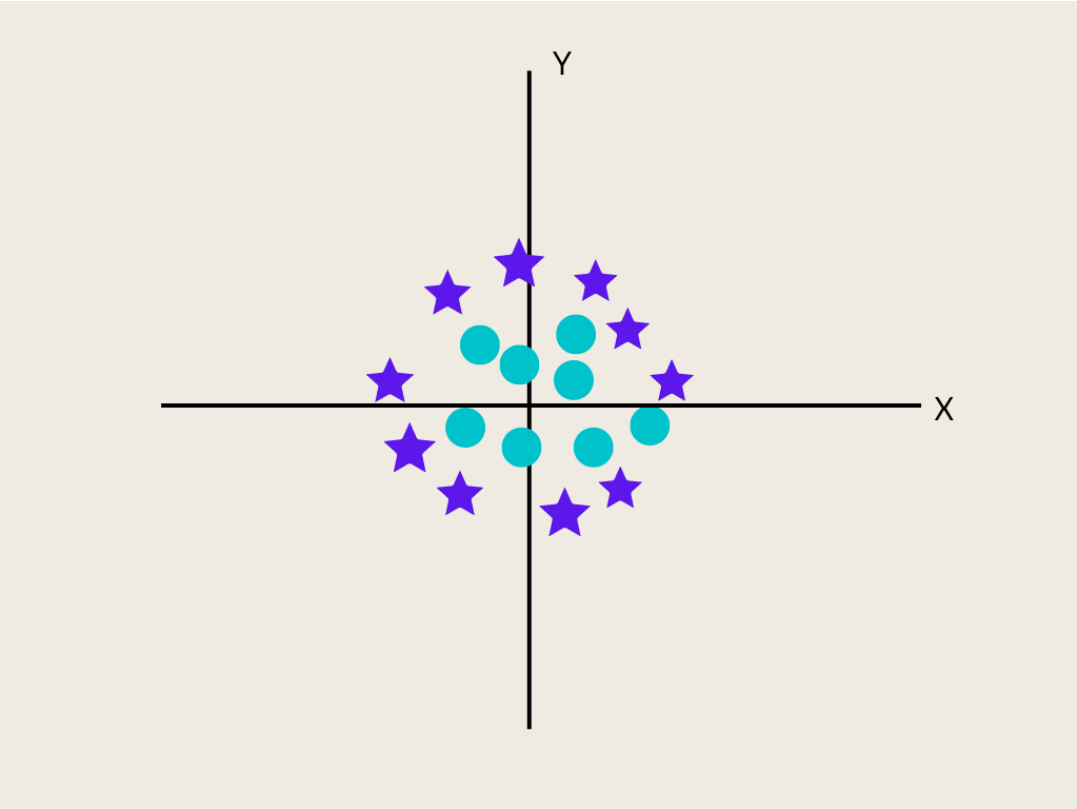


Figure 4.10: SVM 3

SVM will apply the formula of the circle to solve the problem. This is an exceptional technique to detect a hyperplane. This is known as SVM Kernel. The formula takes a low-dimensional provided space and expands it into a higher-dimensional space, making the issue separable. It does some really complicated data manipulations before figuring out how to segregate the data according to the names or outcomes specified.

4.0.4 RoBERTa:

RoBERTa is an advanced technique of BERT. The distinction between critical and minor jobs in BERT allows the former to achieve their objectives, perhaps at the latter's price. It addresses the issue of scheduling cycles by utilizing the artificial clock packet scheduling mechanism and takes advantage of the link between the actual and simulated time that the virtual clock keeps. The complete form of this is the robustly optimized BERT approach. Over the previously described BERT, RoBERTa offers a significant development. Additional gains are detected in efficiency across all subsequent tasks, demonstrating the value of data quantity and variety in pre-training[15].

Now, in order for us to comprehend RoBERTa, we need to get some background information on BERT.

The BERT model generates the contextual description of individual tokens. Additionally, it can construct the context of a whole sentence, as well as the context of many sentences.

Regarding language modeling, BERT pre-trains the model in an unsupervised way using a huge dataset, and transformers provide the foundation for most of BERT's architecture. The BERT system employs multilayer bidirectional transforming encoders to represent languages. In general, BERT does all of the functions, including NLI, classification, and question-answering, using the same model architecture with just a few minor modifications, such as adding an output layer for classification.

As explained previously, the design that RoBERTa uses is identical to that of BERT. In opposition to BERT, however, it just goes through simple pre-training with Masked Language Modeling during the pre-training phase.

In the world of Transformer architecture, RoBERTa is considered one of the recent model of this generation[16]. It solves the over fitting problem as It is more resistant to over fitting training data. Also, RoBERTa achieves better performance on several natural language understanding tasks[16]. It uses Byte-Pair Encoding (BPE) instead of WordPiece encoding like BERT, which helps it better handle out-of-vocabulary words. However, RoBERTa integrates ideas from BERT and OpenAI GPT, so it performs with the combined efficiency of these two models. One point differentiating the model from BERT is that RoBERTa can handle longer sequences than BERT. It uses a technique called Knowledge Distillation to improve performance on downstream tasks.

There are several fields where RoBERTa can be used for a progressive output. It plays a vital role in Sentiment Analysis. Not only Sentiment analysis but also Topic and text classification. The model is used for different purposes in different fields. However, RoBERTa is suitable for text-related approaches and contributes to translation. The algorithm works in a mechanism where it generates the salience of the text and then classifies them into categories. Here are some related fields where RoBERTa can be implemented for efficient output.

Following in the footsteps of BERT comes the uncomplicated and immensely popular RoBERTa, which serves as both a successor and an alternative. The careful and strategic optimization of BERT's training hyper-parameters is the critical contribution it has made to develop that algorithm. As a result of a combination of elementary and uncomplicated modifications, RoBERTa's overall performance has been improved. It now outperforms BERT on virtually all of the challenges that BERT was developed to address. RoBERTa now outperforms BERT on virtually all of the challenges that BERT was developed to address.

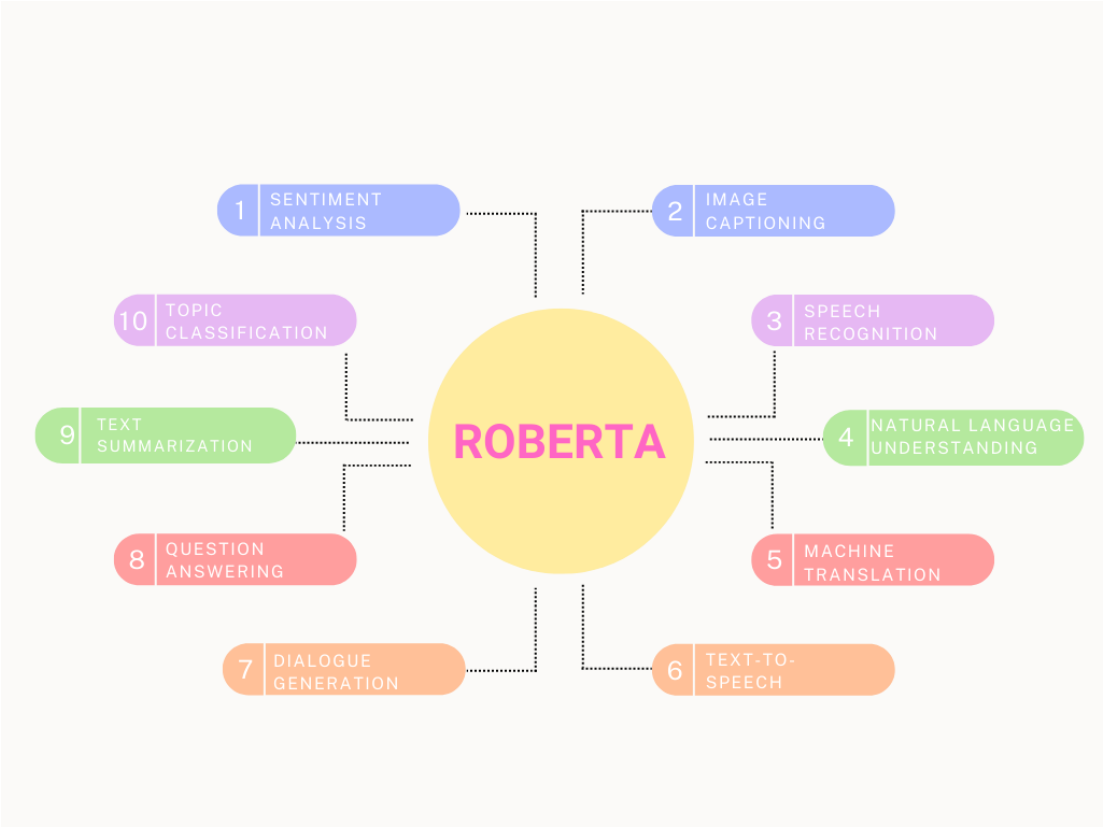


Figure 4.11: RoBERTa

Chapter 5

Result and Limitations

5.1 Precision, Recall, F1 score

Model results may be described using a variety of metrics, such as accuracy, recall, F1, and confusion matrix. In the end, but using various methods, the outcome compares the anticipated value and the actual labelled values. Before moving on to these result analysis procedures, it is important to understand the critical variables that these performance assessment techniques relies on.

True positives:
If the projected value and the values that were actually identified match, it is considered a true positive. This may be thought of as positively expected values. For instance, if the labelling of the value is "Positive" and the predicted value is also "Positive" then it can be called as true positive (TP).
True Negative:
Values that were correctly anticipated to be negative are referred to as true negatives. If both the estimated value and the labeled value are negative, it may be referred to as true negative (TN). For example, if the labelling of the value is "Not Positive" and the predicted value is also "Not Positive" then it can be called as true negative (TN).
False Positive:
If the outcome is a false positive, the expected value is yes but the class is really negative (FP). For example, if the actual class says "Positive" but when a value is expected to be "Negative," As such, it is classified as a false positive.
False Negative:
When there is a discrepancy between the expected and recognized classes, usually a negative one. For instance, we will consider it a false negative (FN) if the projected value is "Positive," but the actual label is "Negative".
Accuracy:
This method of calculating performance is the simplest. By contrasting the prediction with the actual facts, this is accomplished. It demonstrates how accurately the model foresees the categories.
Accuracy = TP+TN / TP+FP+TN+FN

Recall:
Recall reveals the percentage of correctly predicted events when all are positive. The

formula is shown below-
Recall = T P / T P + F N

Precision:
When all forecasts are positive, precision reveals how much of a prediction is actually accurate. The formula is shown below-
Precision = T P / T P + F P

F1 Score:
F1 Score provides us with a performance statistic for classifiers based on Precision and Recall. Thus, it takes into account both false positives and false negatives. If there is a significant dissimilarity in the class distributions, the F1 Score will be more useful than accuracy. Accuracy works best when the costs of false positives and false negatives are equal.
F1 Score = 2 ∗ (Recall*Precision) / Recall + Precision

5.2 Result

In the dataset tweets were divided into six sentiment classes. Next, we reduced the original six groups down to only three basic types. Negative, neutral, and positive describe them. For instance, we used two different vectorization techniques to vectorize or tokenize the comments (data) and one hot encoding to convert the data. Then SVM, Multinomial Naive Bayes, Roberta, LSTM is used for sentiment analysis. We evaluated the performance of our classification system by computing its accuracy, precision, recall, and F1 score. Below, you'll see a tally of the classifiers' discoveries broken down by tokenization technique and class type.

5.2.1 TF-IDF tokenized Multi-Class Dataset

Classifiers	Accuracy	Precision	Recall	F1 score
LSTM	0.83	0.83	0.82	0.82
Naive Bayes Classifier	0.67	0.64	0.62	0.63
RoBERTa	0.89	0.87	0.85	0.86
SVM	0.79	0.78	0.76	0.76

Table 5.1: TF-IDF tokenized Multi-Class Dataset

LSTM and RoBERTa both achieve above 80% in accuracy, precision, recall, and F1 score from TF-IDF.. Among all of the classifiers, RoBERTa gives the best result in case of all the four parameters with the highest accuracy of 0.89. The second highest is the LSTM classifier with 0.83 accuracy. Naive Bayes and SVM have accuracy of 0.67 and 0.79 respectively. RoBERTa clearly has the best results as it takes less time while preprocessing and while training data it takes larger numbers of data. In case of precision, the highest is again RoBERTa and the lowest is Naive Bayes with 0.87 and 0.64 respectively. The third and fourth appropriate result or we simply know as precision is of 0.78 and 0.64 which is generated by SVM and Naive Bayes.

Positive predictive value can also be called as precision and the lowest value stands for precision is 64%. The third parameter which is recall is also known as performance indicators based on various things. Here in recall, LSTM and RoBERTa have almost same scores with only difference of 0.03. RoBERTa as usual has the most recall value of 85%. LSTM has just 0.03 less recall than RoBERTa which is 0.82. The least recall value is 0.62 carried by Naive Bayes. Regarding the recall values, third comes SVM with 0.76 recall value which 0.06 less than the second highest value. Then finally comes the F1 score and for SVM the value remains same as the recall value with again 76%. F1 value increased a little bit in RoBERTa than of recall value. The change is 0.01 and now RoBERTa has F1 score of 0.86. LSTM and RoBERTa both have F1 score above 80%. They only differ with the value of 0.04. LSTM has score of 0.86. Furthermore, Naive Bayes has the least F1 score consisting of 0.63. With TF-IDF, no classifier got less than 60% whereas none also got more than 90% result. This comparison clearly shows that RoBERTa is the most appropriate and Naive Bayes is the least appropriate model for TF-IDF.

5.2.2 Bag of Words tokenized Multi-Class Dataset

Classifiers	Accuracy	Precision	Recall	F1 score
LSTM	0.82	0.82	0.81	0.81
Naive Bayes Classifier	0.69	0.66	0.65	0.66
RoBERTa	0.90	0.87	0.86	0.87
SVM	0.78	0.76	0.76	0.76

Table 5.2: Bag of Words tokenized Multi-Class Dataset

The Bag of Words method is used to tokenize the data in a multiclass dataset. The Roberta classifier has the highest accuracy of 0.90, as can be shown. That's an excellent conclusion for a multiclass dataset. The second-place finisher, LSTM, had a score of 0.82. The accuracy of SVM and Naive Bayes Classifier, in contrast, was 0.78 and 0.69, respectively. We can see from the above table, Naive Bayes Classifier has the lowest accuracy, 0.69. But surprisingly none of the classifier showed less than 50% accuracy. It follows that Bag of Words tokenization of both positive and negative tweets may lead us to the conclusion that Roberta is the most effective classifier for a multiclass dataset.

Multi-class categorization of good, negative, and neutral tweets was performed with the aid of Bag of Words, and the results are displayed below. To do this, we employed four distinct classes of algorithm. We'll break down each model's results using its Precision, Recall, and F1 Score metrics now.

In the first place, Roberta has a better Recall value, which indicates that this classifier is more accurate at predicting positive results. This classifier is able to make a better informed determination as to whether the provided data represents a positive tweet or not, or if it is neutral. In addition, we get comparable outcomes when we use LSTM and SVM. Roberta ranks first, with a Recall of 0.86, among them. After that comes LSTM (0.81) and SVM (0.76). The Recall value of the alternative

classifier is poor. Naive Bayes estimates a recall of 0.65 which is lowest among all these classifiers.

Similarly, Roberta's 0.87 Precision rating is the highest of any classifier. Accurately anticipating positive values is a hallmark of classification algorithms, and a higher accuracy number indicates this. Precision is defined as the proportion of "positive" samples or cases that were correctly classified. To a lesser extent than RNN, LSTM provides the second-highest precision value. In terms of precision, LSTM is quite close to the gold standard, Roberta, with a value of 0.82. SVM follows LSTM, which has a value of 0.76 at the moment. The naive bayes method, as expected, returns the lowest result, which is 0.66 in this case. This indicates that the naive bayes algorithm is the least accurate at making predictions for positive values.

In terms of the F1 Score, Roberta has the highest rating (0.87). Followed by, LSTM (0.81), SVM (0.76) whose values are pretty close to each other. Lastly, naive bayes has shown the least value, stands at 0.66. The first three classifiers in this set have values between 0.87 and 0.76. As a group, these three are outperforming the other. By harmonically averaging accuracy and recall, the F1-score provides an all-encompassing measure of a classifier's efficacy. We can see that Roberta achieves the highest possible F1 score by calculating the harmonic mean of the accuracy and recall of a classifier, thereby unifying them into a single measure. However, when it comes to displaying the F1 score, naive bayes has proven to be the least effective.

In conclusion, Roberta showed great performance and outperformed all the other classifiers.

5.2.3 One Hot Encoded tokenized Multi-Class Dataset

Classifiers	Accuracy	Precision	Recall	F1 score
LSTM	0.84	0.83	0.83	0.82
Naive Bayes Classifier	0.70	0.68	0.65	0.66
RoBERTa	0.90	0.86	0.86	0.86
SVM	0.78	0.77	0.76	0.76

Table 5.3: One Hot Encoded tokenized Multi-Class Dataset

The data in our multiclass dataset is tokenized using the One Hot Encoding approach. We can see that RoBERTa has the highest accuracy among all of the classifiers. RoBERTa has a high accuracy of 0.90. In our case, we have used the same dataset and in terms of a multiclass dataset, this is a very great outcome. We can observe that LSTM came second in terms of accuracy. It holds an accuracy of 0.84 which makes the comparison more competitive. Naive Bayes and SVM on the other hand hold an accuracy below 0.80. SVM has an accuracy of 0.78 and Naive Bayes has an accuracy of 0.70. This means Naive Bayes has the lowest accuracy among all the classifiers. If tokenization is applied via One Hot Encoding then we may conclude that RoBERTa is an advanced classifier that performs better than

LSTM, Naive Bayes, and SVM. Not to mention LSTM also consists of good accuracy but RoBERTa is undoubtedly the best classifier for a multiclass dataset.

There are four distinct categorization algorithms that we employed. Now we are going to talk about the performance of each model using the Precision, Recall, and F1 Score.

Firstly, Multinomial Naive Bayes has the lowest Recall value, which means that this classifier does not predict values less correctly. So, this classifier cannot predict the sentiment more perfectly. In this case, SVM Performs at a neutral level but LSTM (0.83) and RoBERTa(0.86) predict emotion more perfectly. RoBERTa has the highest recall value which is 0.86. It can be assumed that RoBERTa detects sentiment in a faster process than other classifiers mentioned here.

On the other hand, a higher precision value means that these classification algorithms are predicting sentiments correctly. Here, RoBERTa holds the highest precision value which is 0.86. The second highest value is LSTM which is 0.83. Naive Bayes and SVM give us a precision value of 0.68 and 0.77.

The F1 score is widely recognized as a top metric for evaluation in the area of machine learning. RoBERTa has an F1 score of 0.86 which is the highest F1 score and Naive Bayes has a F1 score of 0.68 which is the lowest F1 score. Also, SVM and LSTM hold F1 scores of 0.76 and 0.83 which is a good average level.

After all these evaluations, we can see that RoBERTa is scoring the maximum in every category. This states that RoBERTa is the best classifier for the One Hot Encoding method in the multiclass dataset and the second best is LSTM.

5.3 Accuracy Report:

	TF-IDF	Bag Of Words	One Hot Encoding
LSTM	0.83	0.82	0.84
Naive Bayes Classifier	0.67	0.69	0.70
RoBERTa	0.89	0.90	0.90
SVM	0.79	0.78	0.78

Table 5.4: Score Comparison

We have used “Corona Virus Tweets NLP- Text Classification” as our dataset and we implemented this data set to compare the performance of each methodology. Here by methodology, we are denoting the combination of pre-processing scheme and classifier. We neutralized our dataset using TF-IDF, BOW and One Hot Encoding and then we tried to implement the output to different classifiers to see what mechanism performs efficiently. For generating better performance, we focused on the methods which are used in several platforms and this makes the comparison more competitive.

These classifiers are algorithms itself which get trained using a dataset. In our research, we Implemented Bag of words, TF-IDF and One HOT encoding on four classifiers. They are LSTM, SVM, Naive Bayes, RoBERTa. We can observe that One Hot Encoding giver better performance in terms of better pre-processing than TF-IDF and Bag of Words. In the case of SVM, TF-IDF gives 0.001 percent better performance because the Recall value of TF-IDF is higher than the recall value of One Hot Encoding. The majority of report says that One Hot Encoding stands for better output. Also, Bag of Words and TF-IDF give close output in different classifiers but on average these two give almost the same performance. However, we analyze that, roberta performs best with Bag Of Words and One Hot encoding. After observing a bit more we found that, One Hot Encoding merges with the classifiers very smoothly as this methodology gives better performance in every case except SVM. However, LSTM gives a higher percentage while compiling with One Hot Encoding. Furthermore, RoBERTa holds the highest recall percentage in every scenario with the classifiers when One Hot Encoding is used for tokenization.

We can count LSTM as the second best classifier as it gives a balanced performance with every other preprocessing, and holds a better F1 score and recall value than Naive Bayes and SVM. On average, Bag Of Words gives 0.002 percent better output in terms of tokenization than TF-IDF. But One Hot Encoding gives 0.01 percent better output than both TF-IDF and Bag of Words. Eventually, we can say that by using One Hot Encoding tokenization and RoBERTa as a classifier we can generate a better output and a better version. This is the novelty of our work as we found out that RoBERTa gives 0.03 percent advantage in different cases. We can conclude with the fact that our methodology can generate better sentiment analysis reports if we use the proposed combination. We also analyzed why RoBERTa is best among these models. By diving into this matter, we found that RoBERTa can implement larger datasets for usage in its analysis. The RoBERTa transformer model is more complex since it includes 24 layers of transformers instead of the 12 levels which creates faster working mechanism. RoBERTa can more accurately show the contextual representation of the words contained inside a phrase. That means it can predict sentiments more accurately if the right pre-processing is used. Not to mention, RoBERTa has a maximum token processing capacity of 512 and this is one of the reasons why it outperforms LSTM, SVM and Naive Bayes.

5.4 Comparative analysis

In our paper, we tried to implement several methods and different preprocessing techniques to reach a better result. After analyzing the result, RoBERTa overall performs the best and achieved F1 and accuracy scores around 90%. If we look for similar studies, we can see Untari N. Wisesty [9] came up with an accuracy of 80% and F1-result of 78 using SVM. They also implemented LSTM which have F1-score of 0.67 and Bert with F1-score of 0.74. In another study [17], Rustam, Furqan used the BiLSTM and CNN-LSTM and achieved the accuracy score of 0.579 and 0.61 reflectively. They also implemented TF-IDF and BOW model and get score of 0.89 and 0.87. In another paper [18], they performed several deep learning algorithms on covid-19 tweets for sentiment analysis. After using linear SVM, MKH-SVM and

neural net model, they achieved a F1-score of 0.77 and precision of 0.86 with MKH-SVM model. In another study [19], an accuracy of 84.5% was achieved by using LSTM model and they used VADER to label the dataset of coronavirus tweets. Besides these, same dataset which we used was also used in this paper [9] and the achieved a F1-score of 0.75 and precision of 0.87 using Multinomial Naive Bayes. From Table 5.5, we can see our RoBERTa model out performs other mentioned models previously implemented on similar studies.

Paper Name	Dataset Class	Samples	Algorithm	Performance
[9]	3	41157	BERT	0.83 (F1-score)
[18]	5	500	SVM	0.77 (F1-score)
[19]	3	165116	LSTM	0.84 (accuracy score)
[9]	3	41157	MNB	0.87 (accuracy score)
Our work	3	41157	RoBERTa	0.90 (accuracy score)

Table 5.5: Performance comparison with previous works

5.5 Classification Report

	precision	recall	f1-score	support
Negative	0.79	0.81	0.80	4266
Neutral	0.75	0.60	0.67	2058
Positive	0.81	0.85	0.83	4915
accuracy			0.79	11239
macro avg	0.78	0.76	0.76	11239
weighted avg	0.79	0.79	0.79	11239

Table 5.6: SVM with TF-IDF

	precision	recall	f1-score	support
Negative	0.79	0.77	0.78	4242
Neutral	0.69	0.67	0.68	2058
Positive	0.80	0.83	0.82	4939
accuracy			0.78	11239
macro avg	0.76	0.76	0.76	11239
weighted avg	0.78	0.78	0.78	11239

Table 5.7: SVM with BOW

	precision	recall	f1-score	support
Negative	0.80	0.79	0.80	1629
Neutral	0.70	0.65	0.67	614
Positive	0.80	0.82	0.81	1544
accuracy			0.78	3787
macro avg	0.77	0.76	0.76	3787
weighted avg	0.78	0.78	0.78	3787

Table 5.8: SVM with One Hot Encoding

	precision	recall	f1-score	support
Negative	0.81	0.85	0.83	1633
Neutral	0.80	0.74	0.77	619
Positive	0.86	0.83	0.84	1546
accuracy			0.78	3798
macro avg	0.82	0.81	0.81	3798
weighted avg	0.83	0.82	0.82	3798

Table 5.9: LSTM with BOW

	precision	recall	f1-score	support
Negative	0.85	0.81	0.83	1633
Neutral	0.81	0.76	0.78	619
Positive	0.82	0.88	0.85	1546
accuracy			0.83	3798
macro avg	0.83	0.82	0.82	3798
weighted avg	0.83	0.83	0.83	3798

Table 5.10: LSTM with TF-IDF

	precision	recall	f1-score	support
Negative	0.71	0.77	0.74	1629
Neutral	0.61	0.46	0.52	614
Positive	0.73	0.73	0.73	1544
accuracy			0.70	3787
macro avg	0.68	0.65	0.66	3787
weighted avg	0.70	0.70	0.70	3787

Table 5.11: Naive Bayes with One Hot Encoding

	precision	recall	f1-score	support
Negative	0.70	0.77	0.73	1629
Neutral	0.55	0.51	0.53	614
Positive	0.74	0.69	0.71	1544
accuracy			0.69	3787
macro avg	0.66	0.65	0.66	3787
weighted avg	0.69	0.69	0.69	3787

Table 5.12: Naive Bayes with BOW

	precision	recall	f1-score	support
Negative	0.69	0.74	0.71	1629
Neutral	0.53	0.44	0.48	614
Positive	0.70	0.69	0.69	1544
accuracy			0.67	3787
macro avg	0.64	0.62	0.63	3787
weighted avg	0.67	0.67	0.67	3787

Table 5.13: Naive Bayes with TF-IDF

	precision	recall	f1-score	support
Negative	0.86	0.94	0.90	1629
Neutral	0.78	0.81	0.79	614
Positive	0.94	0.83	0.88	1544
micro avg	0.88	0.88	0.88	3787
macro avg	0.86	0.86	0.86	3787
weighted avg	0.88	0.88	0.88	3787
samples avg	0.88	0.88	0.88	3787

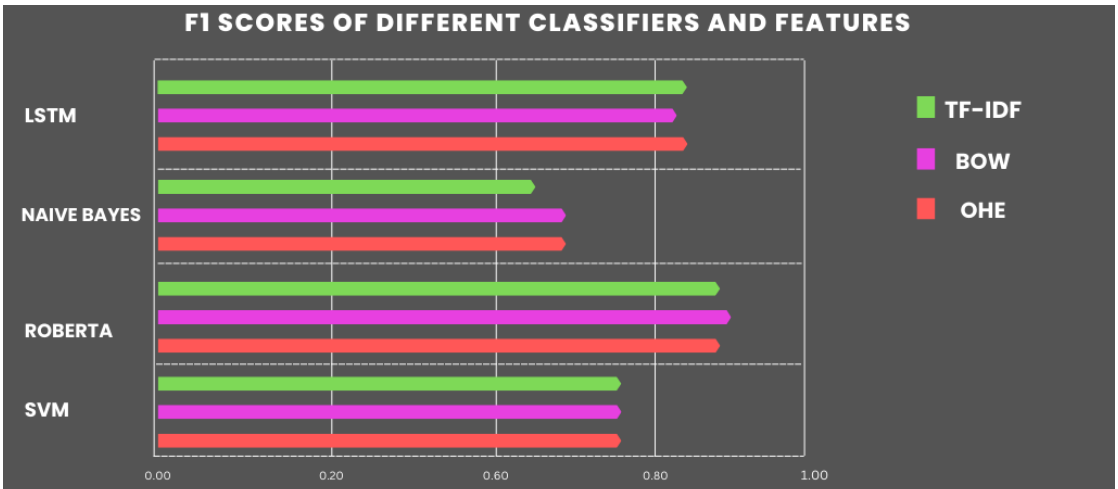
Table 5.14: RoBERTa with One Hot Encoding

	precision	recall	f1-score	support
Negative	0.87	0.93	0.90	1629
Neutral	0.82	0.78	0.80	614
Positive	0.92	0.88	0.90	1544
micro avg	0.88	0.88	0.88	3787
macro avg	0.87	0.86	0.87	3787
weighted avg	0.88	0.88	0.88	3787
samples avg	0.88	0.88	0.88	3787

Table 5.15: RoBERTa with BOW

	precision	recall	f1-score	support
Negative	0.89	0.89	0.89	1629
Neutral	0.84	0.76	0.80	614
Positive	0.87	0.91	0.89	1544
micro avg	0.88	0.88	0.88	3787
macro avg	0.87	0.85	0.86	3787
weighted avg	0.88	0.88	0.88	3787
samples avg	0.88	0.88	0.88	3787

Table 5.16: RoBERTa with TF-IDF



	TF-IDF	Bag of Words	One Hot Encoding
LSTM	0.82	0.81	0.82
NAIVE BAYES	0.63	0.66	0.66
ROBERTA	0.86	0.87	0.86
SVM	0.76	0.76	0.76

Figure 5.1: F1 scores of different classifiers and features

This chart shows the comparison between the F1-Score between four classifiers by inputting three different techniques and the overall score shows that RoBERTa gives better output in processing. Naive Bayes shows poor results and SVM shows the average. LSTM on the other hand gives a standard result but RoBERTa exceeds all three classifiers.

5.6 Confusion Matrix

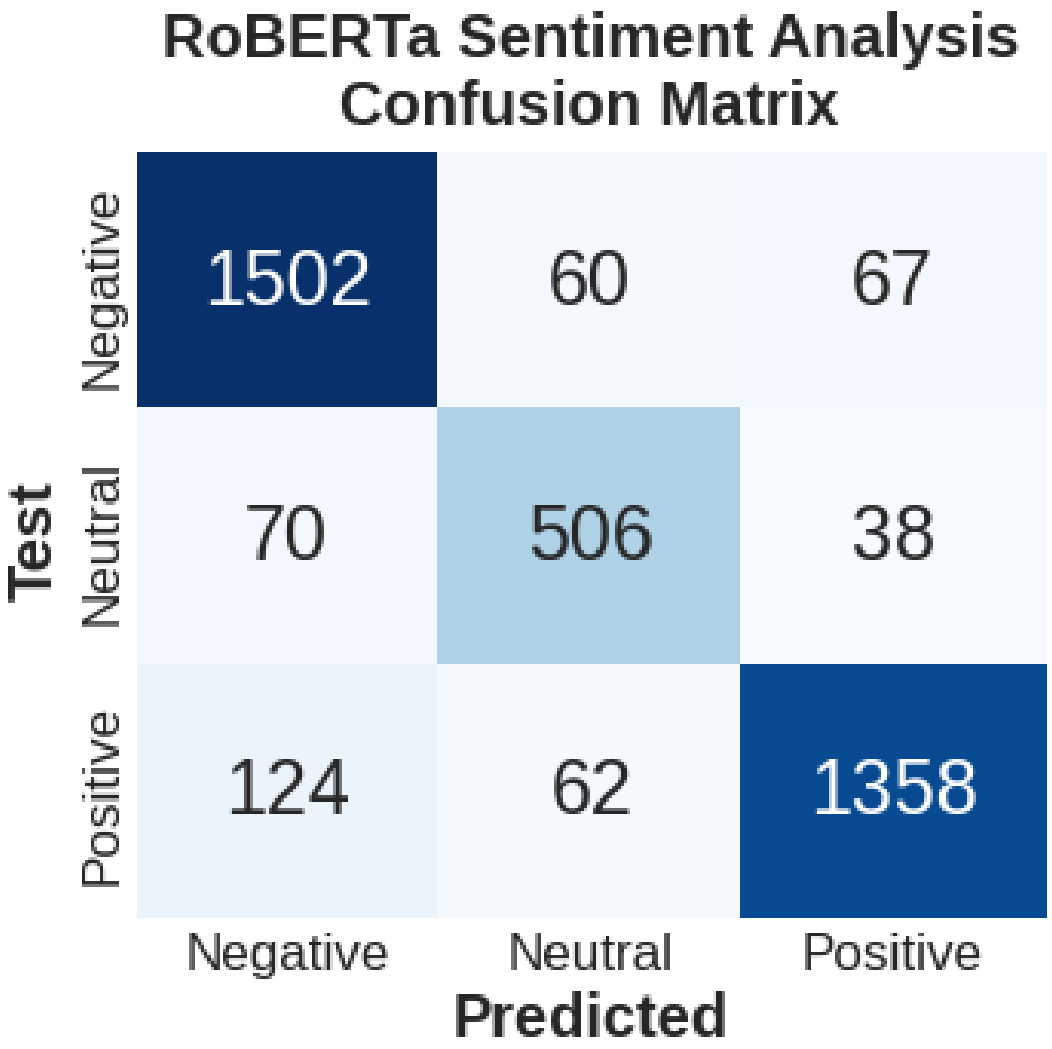


Figure 5.2: Confusion Matrix of RoBERTa

Here is the generated confusion matrix for roberta which provided the better result in classification report compared to other models.

Chapter 6

Conclusion

In our paper, we tried to implement several methods and different preprocessing techniques to reach a better result in sentiment classification of coronavirus tweets. In our dataset tweets were classified in six sentiment classes. We worked with three simplified sentiment class for better result. Here, four models; SVM, Naive Bayes Classifier, LSTM and RoBERTa were implemented. Every model was implemented on preprocessed data which were generated using One Hot Encoding, TF-IDF and Bag of Words. Among them RoBERTa outperforms other models by achieving an accuracy score and F1 score around 0.90. RoBERTa is considered one of the recent model of this generation of Transformer architecture [16]. RoBERTa can implement larger datasets for usage in its analysis compared to BERT[15].

According to the findings of the study that was carried out in this study, we are confident to reach accuracy and F1-score range of 0.92-0.94 just by cleaning the dataset more thoroughly and using detailed preprocessing method. Furthermore, we are also working on implementing other leading language models in future. Next we are planning on using ALBERT and XLNet to get better classification result. Moreover, we will also train our model using six sentiment classes from our dataset to get more precise classification result.

Bibliography

[1] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams engineering journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[2] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[3] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM neural network for text classification," *CoRR*, vol. abs/1511.08630, 2015. arXiv: 1511.08630. [Online]. Available: http://arxiv.org/abs/1511.08630.

[4] I. M. Alsmadi and K. H. Gan, "Short text classification using feature enrichment from credible texts," *International Journal of Web Engineering and Technology*, vol. 15, no. 1, pp. 59–80, 2020.

[5] K. Y. Kamath and J. Caverlee, "Expert-driven topical classification of short message streams," in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, IEEE, 2011, pp. 388–393.

[6] Z. N. Aziza and D. Y. Kristiyanto, "Prediction of the level of public trust in government policies in the 1st quarter of the covid 19 pandemic using sentiment analysis," in *E3S Web of Conferences*, EDP Sciences, vol. 317, 2021, p. 05 013.

[7] J. A. Jijon-Vorbeck and I. Segura-Bedmar, "Exploring the impact of covid-19 on social life by deep learning," *Information*, vol. 12, no. 11, p. 459, 2021.

[8] D. Laines and C. López Castañeda, "Sentiment analysis of covid-19 tweets," Nov. 2021.

[9] U. N. Wisesty, R. Rismala, W. Munggana, and A. Purwarianti, "Comparative study of covid-19 tweets sentiment classification methods," in *2021 9th International Conference on Information and Communication Technology (ICoICT)*, IEEE, 2021, pp. 588–593.

[10] S. Soni, S. S. Chouhan, and S. S. Rathore, "Textconvonet: A convolutional neural network based architecture for text classification," *arXiv preprint arXiv:2203.05173*, 2022.

[11] A. Miglani, *Coronavirus tweets nlp-text classification*, 2020.

[12] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[13] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, 2001, pp. 41–46.

[14] S. Huang, N. Cai, P. P. Pacheco, S. Narrandes, Y. Wang, and W. Xu, "Applications of support vector machine (svm) learning in cancer genomics," *Cancer genomics & proteomics*, vol. 15, no. 1, pp. 41–51, 2018.

[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[16] L. You, F. Han, J. Peng, H. Jin, and C. Claramunt, "Ask-roberta: A pretraining model for aspect-based sentiment classification via sentiment knowledge mining," *Knowledge-Based Systems*, vol. 253, p. 109 511, 2022.

[17] F. Rustam, M. Khalid, W. Aslam, V. Rupapara, A. Mehmood, and G. S. Choi, "A performance comparison of supervised machine learning models for covid-19 tweets sentiment analysis," *Plos one*, vol. 16, no. 2, e0245909, 2021.

[18] H. Kaur, S. U. Ahsaan, B. Alankar, and V. Chang, "A proposed sentiment analysis deep learning algorithm for analyzing covid-19 tweets," *Information Systems Frontiers*, vol. 23, no. 6, pp. 1417–1429, 2021.

[19] M. Mansoor, K. Gurumurthy, V. Prasad, *et al.*, "Global sentiment analysis of covid-19 tweets over time," *arXiv preprint arXiv:2010.14234*, 2020.