Sentiment Analysis of Lockdown in India During COVID-19: A Case Study on Twitter

A PROJECT REPORT

Submitted by

T.AJAY KUMAR	**(Reg. No. 202006005)**
P.RAJA ATHIBAN	**(Reg. No. 202006036)**
P.K.VASANTH RAMM	**(Reg. No. 202006254)**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKAS

(An Autonomous Institution affiliated to Anna University, Chennai)

March 20

BONAFIDE CERTIFICATE

Certified that this project report titled **Sentiment Analysis of Lockdown in India During COVID-19: A Case Study on Twitter** is the bonafide work of **T.Ajay Kumar (Regno: 202006005), P.Raja Athiban (Regno: 202006036), P.K.Vasanth Ramm (Regno: 202906254)** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Mrs.M.BlESSA BINOLIN PEPSI, M.E.**,** (Ph.D).,	Dr.T.REVATHI, M.E., Ph.D.,
Internal guide	**Head of the Department**
Assistant Professor,	**Senior Professor,**
Department of Information Technology,	**Department of Information Technology,**
Mepco Schlenk Engineering College,	**Mepco Schlenk Engineering College,**
Sivakasi.	**Sivakasi.**

Submitted for Viva-Voce Examination held at **MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI (AUTONOMOUS)** on **…………………………**

Internal Examiner **External Examiner**

ABSTRACT

ABSTRACT

With the rapid increase in the use of the Internet, sentiment analysis has become one of the most popular fields of natural language processing (NLP). Using sentiment analysis, the implied emotion in the text can be mined effectively for different occasions. People are using social media to receive and communicate different types of information on a massive scale during COVID-19 outburst. Mining such content to evaluate people's sentiments can play a critical role in making decisions to keep the situation under control. The objective of this study is to mine the sentiments of Indian citizens regarding the nationwide lockdown enforced by the Indian government to reduce the rate of spreading of Coronavirus. In this work, the sentiment analysis of tweets posted by Indian citizens has been performed using NLP and machine learning classifiers. We have a total of 13000 tweets having the keywords "Covid-19" are extracted. Data have been extracted from Twitter using Tweepy API, annotated using TextBlob and VADER lexicons, and preprocessed using the natural language tool kit provided by the Python. The experiment achieved the highest accuracy of 93.5% with LinearSVC classifier and unigrams. This study concludes that the majority of Indian citizens are supporting the decision of the lockdown implemented by the Indian government during corona outburst.

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

Apart from our efforts, the success of our project depends largely on the encouragement of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of our project. We would like to express our immense pleasure to thank our college management for giving required amenities regarding our project.

We would like to convey our sincere thanks to our respected Principal, **Dr.S.Arivazhagan, M.E.,Ph.D.,** Mepco Schlenk Engineering College, for providing us with facilities to complete our project.

We extend our profound gratitude and heartfelt thanks to **Dr.T.Revathi, M.E.,Ph.D.,** Senior Professor and Head of the Department of Information Technology for providing us constant encouragement.

We are bound to thank our project coordinator **Mrs. M.Blessa Binolin Pepsi, M.E.,(Ph.D).,** Assistant Professor of Information Technology. We sincerely thank our project guide **Mrs. M.Blessa Binolin Pepsi, M.E.,(Ph.D).,** Assistant Professor, Department of Information Technology, for her inspiring guidance and valuable suggestions to complete our project successfully.

The guidance and support received from all the staff members and lab technicians of our department who contributed to our project, was vital for the success of the project. We are grateful for their constant support and help.

We would like to thank our parents and friends for their help and support in our project.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES

LIST OF SYMBOLS

LIST OF SYMBOLS

NOTATION	MEANING
X	Dataset
w_i	Weights
P	Precision
F	F1-Score
L	Labeling
R	Recall

LIST OF ABBREVATION

LIST OF ABBREVATION

S.NO	ACRONYMS	ABBREVIATIONS
1	VADER	Valence Aware Dictionary sEntiment Reasoner
2	FP	False Positive
3	TP	True Positive
4	NLP	Natural Language Processing
5	FN	False Negative
6	FP	False Positive

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 SENTIMENT ANALYSIS

Sentiment analysis, also referred to as opinion mining, is an approach to natural language processing (NLP) that identifies the emotional tone behind a body of text. This is a popular way for organizations to determine and categorize opinions about a product, service, or idea. Sentiment analysis focuses on the polarity of a text (positive, negative, neutral) but it also goes beyond polarity to detect specific feelings and emotions (angry, happy, sad, etc) and even intentions. Many emotion detection systems use lexicons (i.e. lists of words and the emotions they convey) or complex machine learning algorithms.

1.2 NLP

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

1.2.1 TEXTBLOB

TextBlob is a Lexicon-based sentiment analyzer It has some predefined rules, where it has some scores that help to calculate a sentence's polarity. That's why the Lexicon-based sentiment analyzers are also called “Rule-based sentiment analyzers”. TextBlob is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

1.2.2 VADER

VADER (Valence Aware Dictionary and sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. Vader is optimized for social media data and can yield good results when used with data from twitter, facebook, etc. VADER uses a combination of sentiment lexicon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive or negative. VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is

1.3 STEPS OF NLP

The Steps of NLP includes:

•Tokenization

•Lemmatization

•Stemming

•POS Tagging

•NER

•Chunking

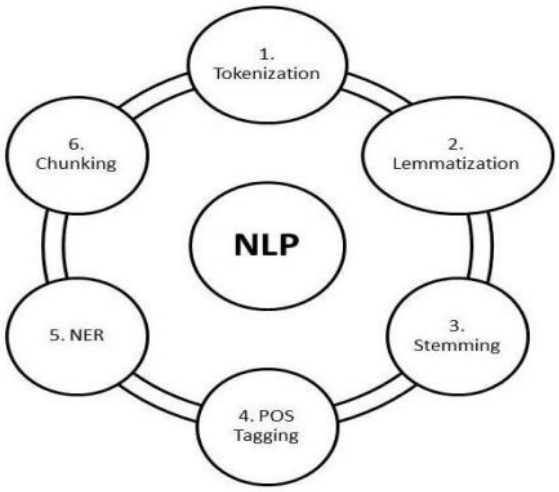


Figure 1.3.1 Steps of NLP

1.4 TYPES OF NLP

There are many different natural language processing algorithms, but two main types are commonly used is

•Rule-based system:

This system uses carefully designed linguistic rules. This approach was used early on in the development of natural language processing, and is still used.

•Machine learning-based system:

Machine learning algorithms use statistical methods. They learn to perform tasks based on training data they are fed, and adjust their methods as more data is processed. Using a combination of machine learning, deep learning and neural networks, natural language processing algorithms hone their own rules through repeated processing and learning.

1.5 HOW DOES NLP WORK?

NLP enables computers to understand natural language as humans do. Whether the language is spoken or written, natural language processing uses artificial intelligence to take real-world input, process it, and make sense of it in a way a computer can understand. Just as humans have different sensors such as ears to hear and eyes to see computers have programs to read and microphones to collect audio. And just as humans have a brain to process that input, computers have a program to process their respective inputs. At some point in processing, the input is converted to code that the computer can understand.

LITERATURE STUDY

CHAPTER 2
LITERATURE STUDY

2.1 OVERVIEW:

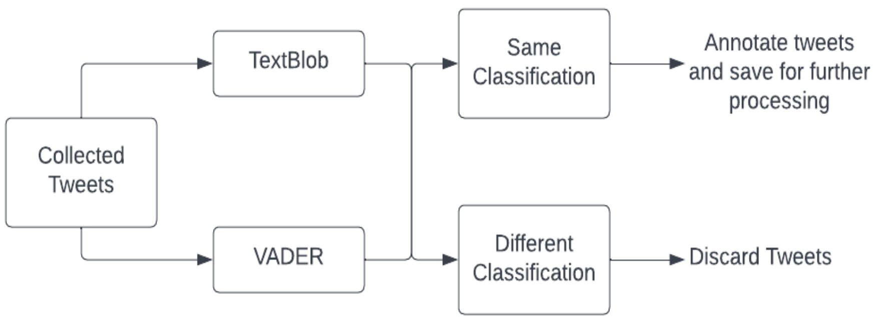
To analyze the sentiment of Indian citizens whether the data is positive , negative or neutral ,Which helps the government to reduce the spread of the disease. To Handle Sentiment Analysis using NLP and Machine Learning Algorithms.

2.2 DATA EXTRACTION:

Tweepy is an open-sourced, easy-to-use Python library for accessing the Twitter API. It gives you an interface to access the API from your Python application.In this project, we used the tweepy to extract the tweets from twitter for the keyword “Covid-19”.

2.3 DATA LABELING:

After tweets collection, we have used the following approach shown in below to label the tweets as positive, neutral, and negative. We have generated each tweet’s polarity using the TextBlob library and VADER (Valence Aware Dictionary for sEntiment Reasoning) tool of the Python. Next, we have taken the intersection of TextBlob and VADER results to consolidate the polarities. After this step, we are left with 7284 tweets having 3545 with positive polarity, 2097 with neutral polarity, and 1642 with negative polarity.



Data Annotation Process

2.4 DATA PREPROCESSING:

The data we have collected may hold some unsought and sentiment fewer words like links, Twitter-specific words such as hashtags (starts with #) and tags (starts with @), single letter words, numbers, etc. These types of words can play the role of noise in our classifier training and testing. To amend classifier efficiency, it is necessary to remove noise from the labeled data set before feeding the classifier. Our pre-processing module separates noise from the labeled data set. The steps of pre-processing are shown below. In this step, we implemented a module to remove the above-specified impurities, converted the data set into a data frame, and then executed removal of string punctuations, tokenization, and removal of English stop words, stemming, and lemmatization.

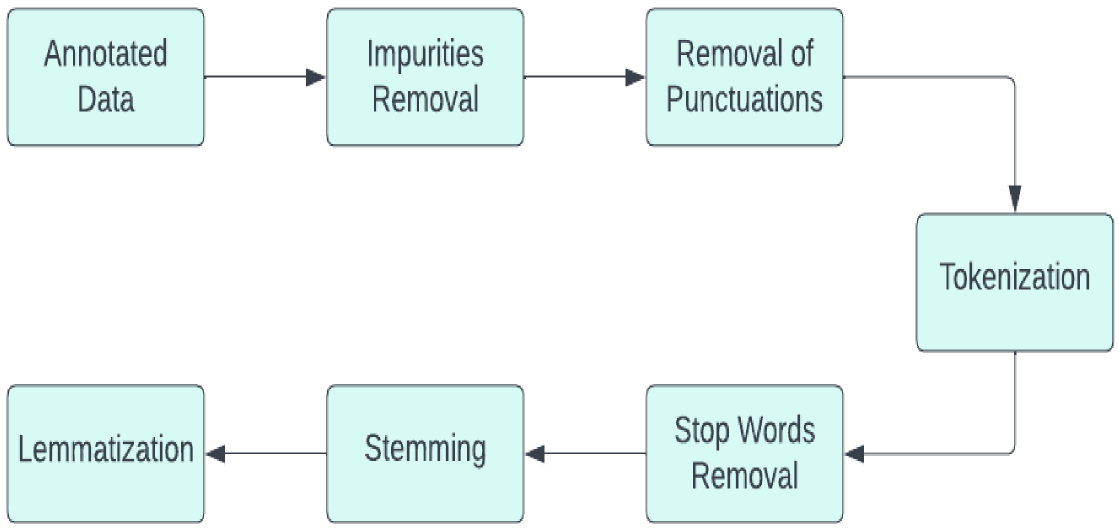
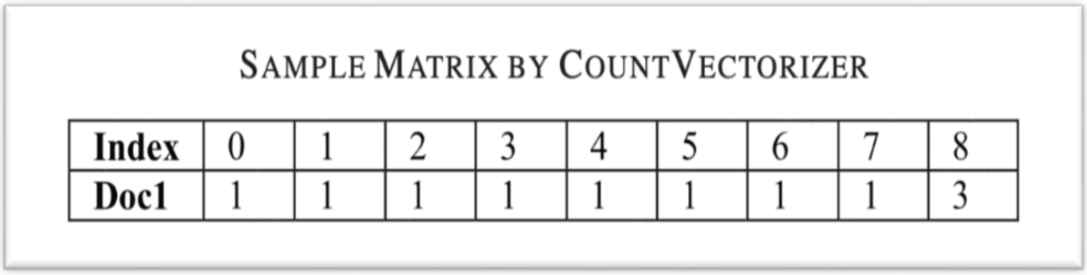


Figure 2.4.1 Data Preprocessing

2.5 DATA VECTORIZATION:

The machine learning classifiers cannot take the input written in any language except numbers. Thus, before using the text data for predictive modeling, it is required to convert it into features. We have used the CountVectorizer feature extractor to calculate word frequencies. CountVectorizer counts the frequency of each word present in the document and creates a sparse matrix, as shown below. For example, Doc1:

"She was young the way an actual young person is young." CountVectorizer will convert this text into the following sparse matrix with an index of the words in alphabetical order as follows: {"she": 4, "was": 6, "young": 8, "the": 5, "way": 7, "an": 1, "actual": 0, "person": 3, "is": 2}. This matrix is not sparse because we are converting the only single document. In the case of multiple documents, it is frequent that a word present in one document can be missing from some other documents, and hence the corresponding cells are filled up with zero, and the resultant matrix will become sparse.


SAMPLE MATRIX BY COUNTVECTORIZER

Index	0	1	2	3	4	5	6	7	8
Doc1	1	1	1	1	1	1	1	1	3

Figure 2.5.1 CountVectorizer

2.6 TRAINING AND TESTING THE CLASSIFIERS:

After feature extraction of the preprocessed data set, we have passed the data to machine learning classifiers. We have used eight classifiers (Multinomial NaiveBayes, Bernoulli NaiveBayes, LogisticRegression, LinearSVC, AdaBoostClassifier, RidgeClassifier, PassiveAggressiveClassifier, and Perceptron) for this purpose. We have used 80% data for training and 20% data for testing the classifiers. We have extracted the performance of the classifiers mentioned above using 1-g, 2-g, and 3-g.

SYSTEM STUDY

CHAPTER 3
SYSTEM STUDY

3.1 SCOPE:

The scope of Sentiment Analysis of Lockdown in India During COVID-19: A Case Study on Twitter is to analyze the setiments of all the people and classify the tweets according to the sentiments using NLP.

3.2 PRODUCT FUNCTION:

- We have used Tweepy Open source package for extracting data for the keyword "Covid-19" nearly 13,000 tweets have been scrapped using tweepy package.
- We have labelled the data using TextBlob and VADER which is used to classify the tweets as Positive, Negative or Neutral tweets.
- Once, We have labelled the tweets using TextBlob and VADER we plotted graph and counted how much Positve, Negative or Netural tweets are present.
- After finishing the data labelling, the data we have collected may hold some unsought and sentiment fewer words like links, Twitter-specific words such as hashtags (starts with #) and tags (starts with @), single letter words, numbers, etc. These types of words can play the role of noise in our classifier training and testing. To amend classifier efficiency, it is necessary to remove noise from the labeled data set before feeding the classifier
- Our pre-processing module separates noise from the labeled data set.
- The steps of pre-processing are shown above. In this step, we implemented a module to remove the above-specified impurities, converted the data set into a data frame, and then executed removal of string punctuations, tokenization, and removal of English stop words, stemming, and lemmatization
- We have used the CountVectorizer feature extractor to calculate word frequencies.
- CountVectorizer counts the frequency of each word present in the document and creates a sparse matrix.
- The matrix is not sparse because we are converting the only single document.
- In the case of multiple documents, it is frequent that a word present in one document can be missing from some other documents, and hence the corresponding cells are filled up with zero, and the resultant matrix will become sparse.

- After feature extraction of the preprocessed data set, we have passed the data to machine learning classifiers
- We have used eight classifiers (Multinomial NaiveBayes,Bernoulli NaiveBayes, LogisticRegression,LinearSVC,AdaBoostClassifier,RidgeClassifier, PassiveAggressiveClassifier, and Perceptron) for this purpose.
- We have used 80% data for training and 20% data for testing the classifiers.

3.3 SYSTEM REQUIREMENTS

The software and hardware requirements of the system are as follows:

3.3.1 HARDWARE INTERFACES

- Intel® Core^{TM} i5-8265U 1.6GHz
- 8 GB RAM

3.3.2 SOFTWARE INTERFACES

- Platform – Anaconda
- IDE – Spyder
- Technologies used – Python
- API– Tweepy
- Google Colab

3.3.2.1 ANACONDA

Anaconda platform is used for machine learning and any other large-scale data processing. It is an open source distribution which is capable of working with R and Python programming languages and free of cost. It consists of more than 1500 packages and virtual environment manager. The virtual environmental manager is named as Anaconda Navigator and it comprises all the libraries to be installed within. It holds certain default navigators like Spyder, JupyterLab, Jupyter Notebook, Orange, Rstudio etc.

3.3.2.2 SPYDER

To implement the proposed system the IDE used is Spyder environment. It is an open source cross-platform integrated development environment. It is the combination of advanced features such as debugging, editing, and analysis of huge data. This tool helps in interactive execution, data exploration and visualization of data.

3.3.2.3 PYTHON

Python is an interpreter, high-level data structures, general-purpose programming language. It can be used for creating web applications on server side. Python is also suitable as an extension language for customized applications.

3.3.2.4 TWEEPY

Tweepy is an open source Python package that gives you a very convenient way to access the Twitter API with Python. Tweepy includes a set of classes and methods that represent Twitter's models and API endpoints, and it transparently handles various implementation details, such as: Data encoding and decoding. This Twitter API gives developers access to almost all of Twitter's functionalities like likes, retweets, tweets, etc. Tweepy, a python package, helps us in achieving all this. Tweepy is a python package that smoothly and transparently accesses Twitter's endpoints made available for the developers.

3.3.2.4 GOOGLE COLAB

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs.

SYSTEM DESIGN

CHAPTER 4
SYSTEM DESIGN

4.1 OVERVIEW

This section presents the overview of the whole system. The Section 4.2 shows the system Section 4.2 defines the main three modules used Section 4.3.1 defines how clusters are formed with the unbounded data streams Section 4.3.2 defines the merge operation with the previously formed rough clusters Section 4.3.3 describes how the clusters are categorized and stored offline.

4.2 OVERALL ARCHITECTURE:

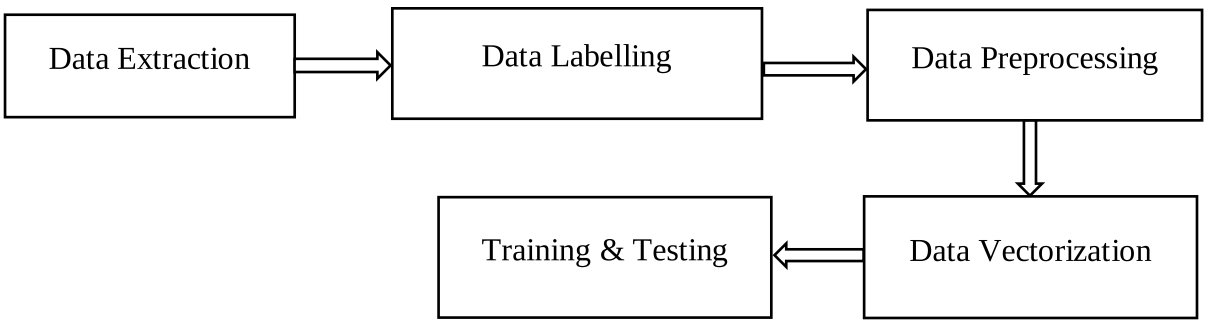


Figure 4.2.1 System architecture

4.3 MODULES

- Multinomial NaiveBayes,
- Bernoulli NaiveBayes
- LogisticRegression
- LinearSVC

- AdaBoostClassifier
- RidgeClassifier
- PassiveAggressiveClassifier
- Perceptron

4.3.1 MULTINOMIAL NAIVEBAYES:

MultinomialNB implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors for each class , where the number of features (in text classification, the size of the vocabulary) and is the probability of feature appearing in a sample belonging to class.

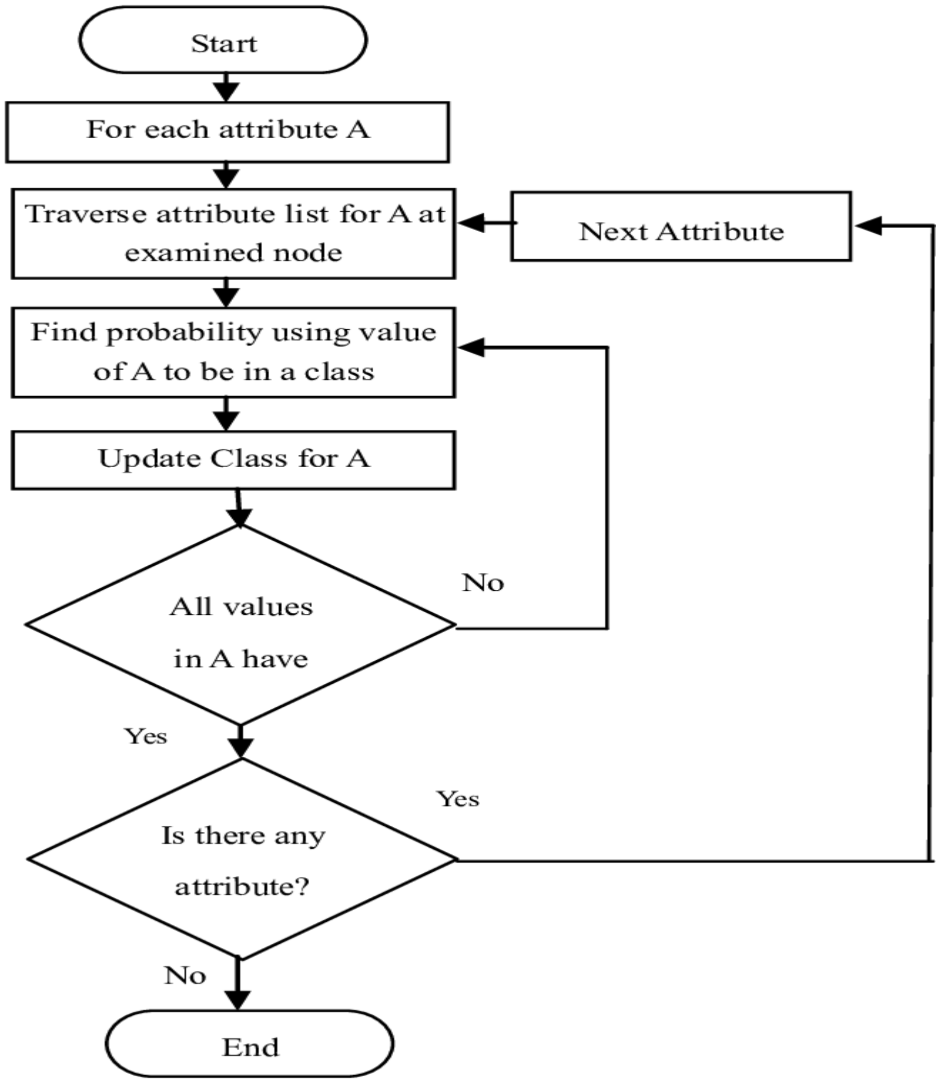


Figure 4.3.1.1 Work Flow of Multinomial NaiveBayes

$$P(A|B) = P(A) * P(B|A)/P(B)$$

Where we are calculating the probability of class A when predictor B is already provided.

P(B) = prior probability of B

P(A) = prior probability of class A

P(B|A) = occurrence of predictor B given class A probability

4.3.2 : BERNOULLI NAIVEBAYES:

BernoulliNB implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors; if handed any other kind of data, a BernoulliNB instance may binarize its input (depending on the binarize parameter).

The decision rule for Bernoulli naive Bayes is based on

$$P(x_i \mid y) = P(x_i = 1 \mid y) x_i + (1 - P(x_i = 1 \mid y))(1 - x_i)$$

which differs from multinomial NB's rule in that it explicitly penalizes the non-occurrence of a feature that is an indicator for class , where the multinomial variant would simply ignore a non-occurring feature.

In the case of text classification, word occurrence vectors (rather than word count vectors) may be used to train and use this classifier. BernoulliNB might perform better on some datasets, especially those with shorter documents. It is advisable to evaluate both models, if time permits.

4.3.3 LOGISTICREGRESSION:

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

•We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

•In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

• But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

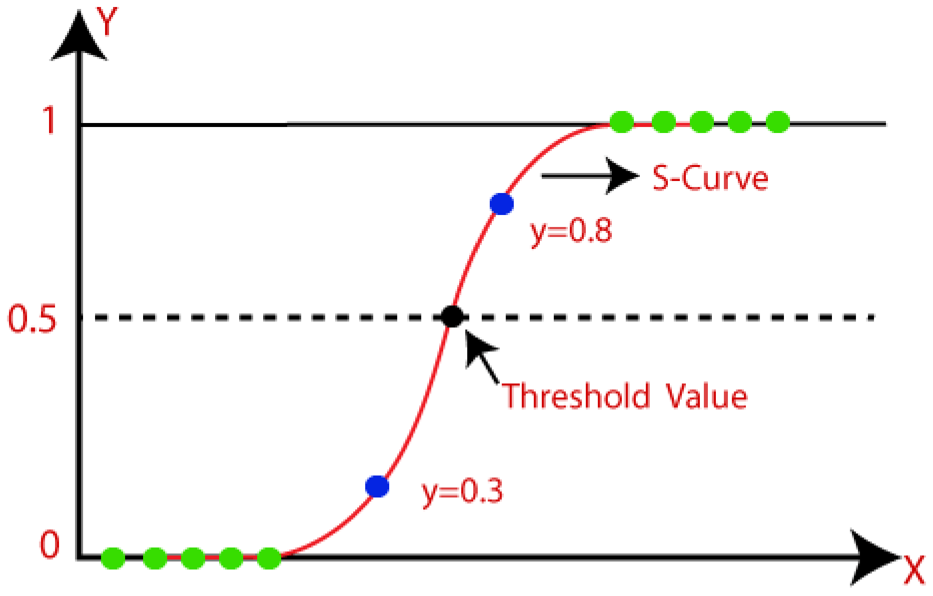


Figure 4.3.3.1 Sigmoid Function in Logistic Regression

Formula for Sigmoid Function:

$$p = \frac{1}{1 + e^{-y}}$$

4.3.4: LINEARSVC:

Linear Support Vector Machine (Linear SVC) is an algorithm that attempts to find a hyperplane to maximize the distance between classified samples.

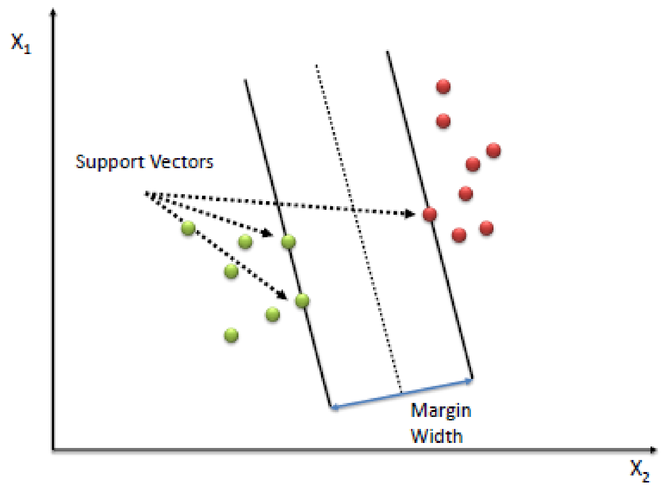
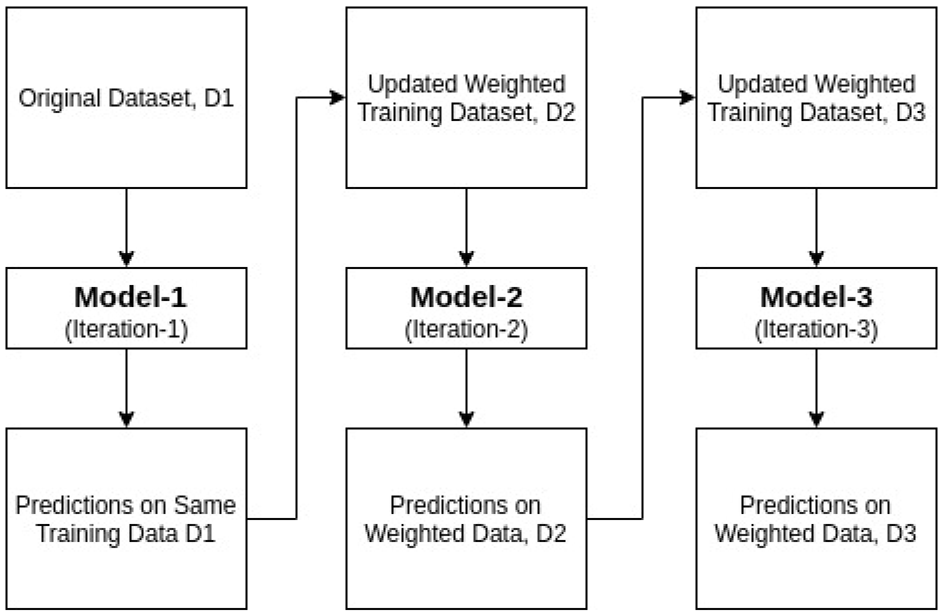


Figure 4.3.4.1 Support Vectors and Margin Width

4.3.5: ADABOOSTCLASSIFIER:

Ada-boost classifier combines weak classifier algorithm to form strong classifier. A single algorithm may classify the objects poorly. But if we combine multiple classifiers with selection of training set at every iteration and assigning right amount of weight in final voting, we can have good accuracy score for overall classifier.



Original Dataset, D1
Updated Weighted Training Dataset, D2
Updated Weighted Training Dataset, D3
Model-1 (Iteration-1)
Model-2 (Iteration-2)
Model-3 (Iteration-3)
Predictions on Same Training Data D1
Predictions on Weighted Data, D2
Predictions on Weighted Data, D3

Figure 4.3.5.1 Work Flow of AdaBoost Classifier

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Where,

h_t(x) is the output of weak classifier t for input x

alpha_t is weight assigned to classifier.

alpha_t is calculated as follows:

alpha_t = 0.5 * ln((1 — E)/E) : weight of classifier is straight forward, it is based on the error rate. Initially, all the input training example has equal weightage.

4.3.6: RIDGECLASSIFIER:

A Ridge regressor is basically a regularized version of a Linear Regressor. i.e to the original cost function of linear regressor we add a regularized term that forces the learning algorithm to fit the data and helps to keep the weights lower as possible.

The regularized term has the parameter 'alpha' which controls the regularization of the model i.e helps in reducing the variance of the estimates.

Cost Function for Ridge Regressor is,

$$J(\Theta) = \frac{1}{m}(X\Theta - Y)^2 + \alpha\frac{1}{2}(\Theta)^2$$

The first term is our basic linear regression's cost function and the second term is our new regularized weights term which uses the L2 norm to fit the data. If the 'alpha' is zero the model is the same as linear regression and the larger 'alpha' value specifies a stronger regularization.

4.3.7: PASSIVEAGRESSIVECLASSIFIER:

The passive aggressive classifier algorithm falls under the category of online learning algorithms, can handle large datasets, and updates its model based on each new instance it encounters.

Let's suppose to have a dataset:

$$\begin{cases} X = \{\bar{x}_0, \bar{x}_1, \dots, \bar{x}_t, \dots\} \; where \; \bar{x}_i \in \mathbb{R}^n \\ Y = \{y_0, y_1, \dots, y_t, \dots\} \; where \; y_i \in \{-1, +1\} \end{cases}$$

Given a weight vector w , the prediction is simply obtained as

$$\tilde{y}_t = sign(\bar{w}^T \cdot \bar{x}_t)$$

All these algorithms are based on the Hinge loss function (the same used by SVM):

$$L(\bar{\theta}) = max\big(0, 1 - y \cdot f(\bar{x}_t; \bar{\theta})\big)$$

The value of L is bounded between 0 (meaning perfect match) and K depending on f(x(t),θ) with K>0 (completely wrong prediction). A Passive-Aggressive algorithm works generically with this update rule:

$$\begin{cases} \bar{w}_{t+1} = argmin_{\bar{w}} \frac{1}{2} \|\bar{w} - \bar{w}_t\|^2 + C\xi^2 \\ L(\bar{w}; \bar{x}_t, y_t) \leq \xi \end{cases}$$

To understand this rule, let's assume the slack variable ξ=0 (and L constrained to be 0). If a sample x(t) is presented, the classifier uses the current weight vector to determine the sign. If the sign is correct, the loss function is 0 and the argmin is w(t). This means that the algorithm is passive when a correct classification occurs. Let's now assume that a misclassification occurred:

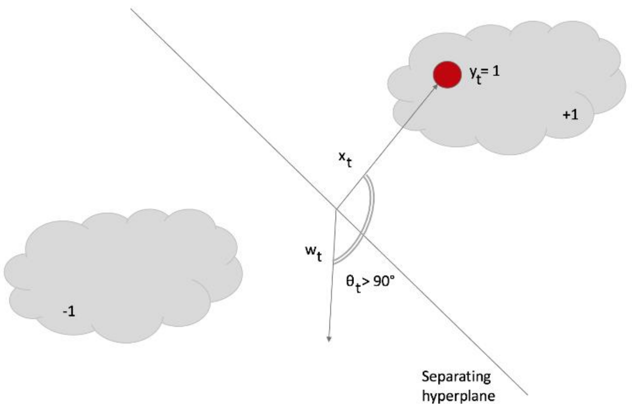


Figure 4.3.7.1 Separating Hyperplane in PassiveAggressive Classifier

4.3.8: PERCEPTRON:

Perceptron is a machine learning algorithm which mimics how a neuron in the brain works. It is also called as single layer neural network consisting of a single neuron. The output of this neural network is decided based on the outcome of just one activation function associated with the single neuron.

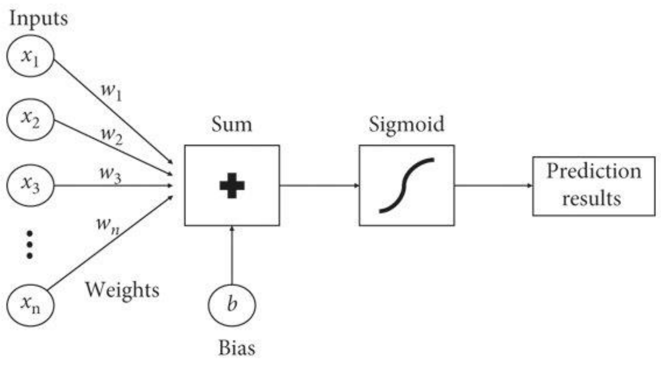


Figure 4.3.8.1 Work Flow of Perceptron

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \underset{\text{learning rate}}{\eta}(\underset{\text{target value}}{t} - \underset{\text{perceptron output}}{o})\underset{\text{input value}}{x_i}$$

IMPLEMENTATION METHODOLOGY

CHAPTER 5

IMPLEMENTATION METHODOLOGY

5.1 OVERVIEW:

The implementation methodology describes the main functional requirements, which needed for doing the project.

5.2 ESSENTIAL LIBRARIES:

The library used in this project are pandas, tweepy, textblob, nltk, countvectorizer, matplotlib, snowballstemmer.

5.2.1 PANDAS:

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

5.2.2 TWEEPY:

Tweepy is an open source Python package that gives you a very convenient way to access the Twitter API with Python. Tweepy includes a set of classes and methods that represent Twitter's models and API endpoints, and it transparently handles various implementation details, such as: Data encoding and decoding.

5.2.3 TEXTBLOB:

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

5.2.4 NLTK:

NLTK stands for Natural Language ToolKit, which is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc…

5.2.5. COUNTVECTORIZER:

CountVectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.

5.2.6 MATPLOTLIB:

Matplotlib is one of the most common packages used for data visualization in python. It is a cross-platform library for creating 2-Dimensional plots from data in arrays. Matplotlib is written in Python. Matplotlib with NumPy can be used as the open source.

5.2.6.1 matplotlib.pyplot:

The matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In this project matplotlib.pyplot is used to generate graph such as Bar graph, Cartesian graphs which represents the different Parameters Vs performance of the algorithm.

5.2.4 SNOWBALLSTEMMER:

It is a stemming algorithm which is also known as the Porter2 stemming algorithm as it is a better version of the Porter Stemmer since some issues of it were fixed in this stemmer.

5.3 FUNCTIONS USED FOR IMPLEMENTATION:

The user defined function used for the implementation of the project are

5.3.1 Data Extract:

In this method, the data is extracted using tweepy which gives request to the twitter API and the twitter will send a response as tweets. We have extracted the tweets for the keyword Covid-19 and the collected 13,000 tweets.

5.3.2 Clean Tweet:

The tweet we collected may have noisy data for removing that we used the method clean tweet. This method will remove the links, @mentions, RT, punctuations, etc. to have a structed tweets will meaningful information.

5.3.3 Subjectivity:

The Subjectivity methods is used inside the textblob. Subjectivity is the output that lies within [0,1] and refers to personal opinions and judgments.

5.3.4 Polarity:

In this method, textblob uses a polarity to classify the tweets and the polarity range between -1 to +1. The polarity classify the tweets as positive, negative and netural tweets.

5.3.5 Tokenization:

Word tokenization is the done for splitting a large sample of text into words. This is a requirement in natural language processing tasks where each word needs to be captured and subjected to further analysis like classifying and counting them for a particular sentiment etc.

5.3.6 Stemming:

Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language.

5.3.7 Lemmatization:

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meanings to one word.

PERFORMANCE METRICS

CHAPTER 6
PERFORMANCE METRICS

6.1 OVERVIEW:

Our algorithm is evaluated across three metrics: 1) confusion matrix 2) F1-Score 3) Precision 4) Recall. The performance metrics of the collected tweets is compared with eight important supervised machine learning algorithms they are Multinomial NaiveBayes, Bernoulli NaiveBayes, LogisiticRegression, LinearSVC, AdaBoostClassifier, RidgeClassifier, PassiveAggressiveClassifier and Perceptron.

6.2 CONFUSION MATRIX:

It is the easiest way to measure the performance of a classification problem where the output can be of two or more type of classes. A confusion matrix is nothing but a table with two dimensions viz. "Actual" and "Predicted" and furthermore, both the dimensions have "True Positives (TP)", "True Negatives (TN)", "False Positives (FP)", "False Negatives (FN)" as shown below –

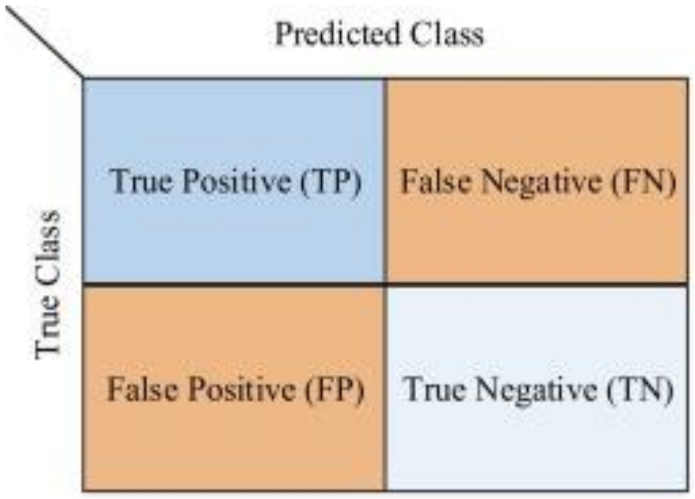


Figure 6.2.1 Confusion Matrix

Explanation of the terms associated with confusion matrix are as follows −

•True Positives (TP) − It is the case when both actual class & predicted class of data point is 1.
•True Negatives (TN) − It is the case when both actual class & predicted class of data point is 0.

•False Positives (FP) − It is the case when actual class of data point is 0 & predicted class of data point is 1.
•False Negatives (FN) − It is the case when actual class of data point is 1 & predicted class of data point is 0.

6.3 F1-SCORE:

This score will give us the harmonic mean of precision and recall. Mathematically, F1 score is the weighted average of the precision and recall. The best value of F1 would be 1 and worst would be 0. We can calculate F1 score with the help of following formula –

F1 = 2 * ((Precision * Recall) / Precision + Recall)

6.4 PRECISION:

Precision, used in document retrievals, may be defined as the number of correct documents returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula –

Precision=TP/TP+FP

`Where,

TP - True Positive

FP- False Positive

6.5 RECALL:

Recall may be defined as the number of positives returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

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

Where,

TP - True Positive

FN - False Negative

RESULTS AND DISCUSSION

CHAPTER 7

RESULTS AND DISCUSSION

7.1 OVERVIEW

This chapter explains the result of our project and the screenshots for each step are included and explained

7.2 DATASETS:

The datasets used in our projects are:

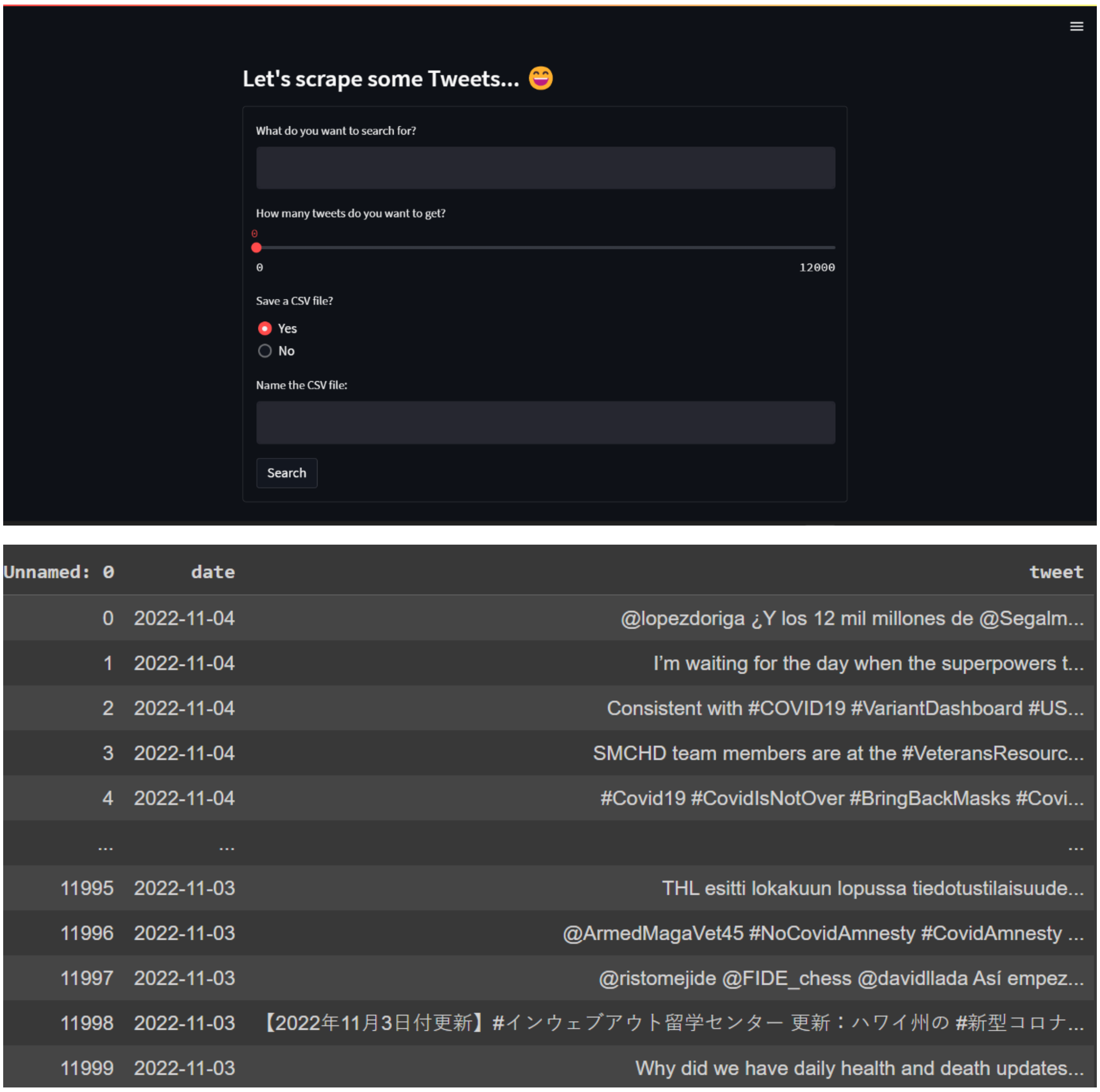
7.2.1 STATIONARY DATASET:

A Stationary dataset is one whose statistical properties such as the mean, variance and autocorrection are all constant over time.

DATASETS	CLASSES	FEATURES	EXAMPLES
Covid19	2	4	13,000
Seoul	2	13	5787
Firing	2	17	24,000

Table 7.2.1 Stationary Dataset

7.3 SCREENSHOTS



Let's scrape some Tweets... 😄

What do you want to search for?

How many tweets do you want to get?

0

0 — 12000

Save a CSV file?

[✓ Yes] No

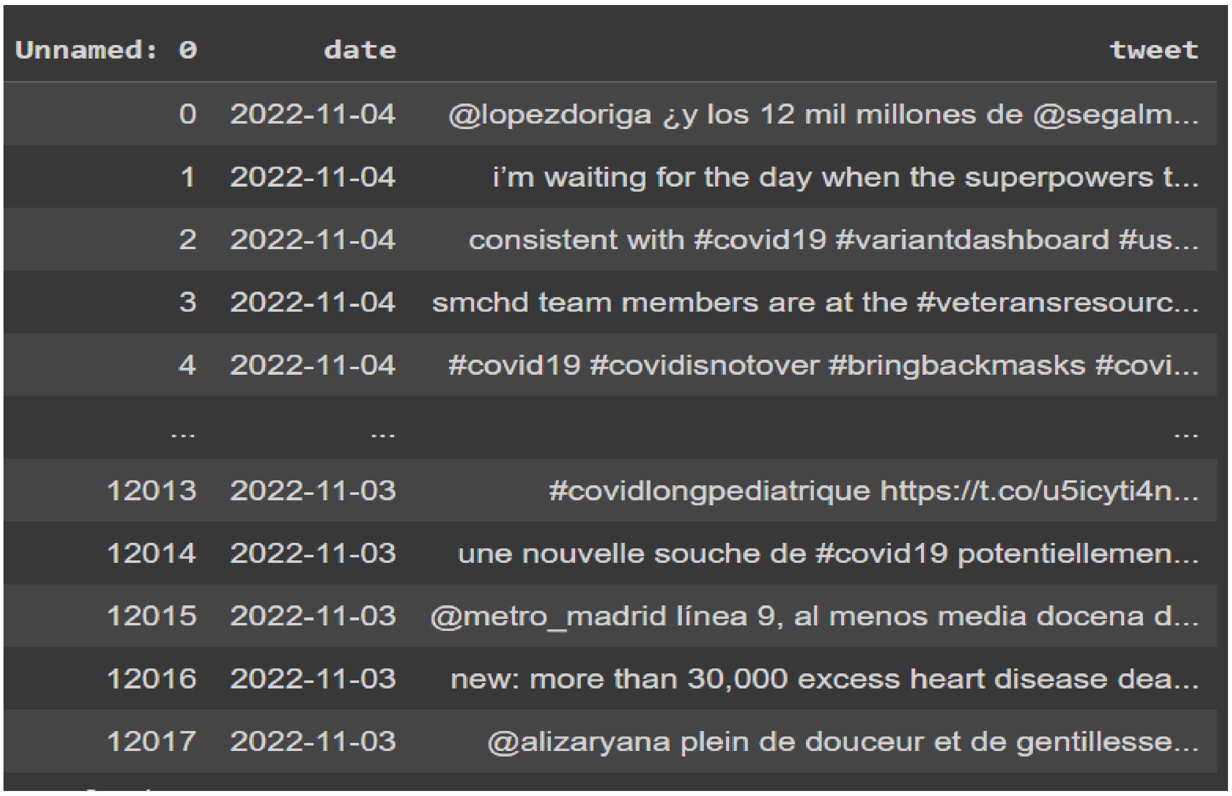
Name the CSV file:

Search

Unnamed: 0	date	tweet
0	2022-11-04	@lopezdoriga ¿Y los 12 mil millones de @Segalm...
1	2022-11-04	I'm waiting for the day when the superpowers t...
2	2022-11-04	Consistent with #COVID19 #VariantDashboard #US...
3	2022-11-04	SMCHD team members are at the #VeteransResourc...
4	2022-11-04	#Covid19 #CovidIsNotOver #BringBackMasks #Covi...
...	...	...
11995	2022-11-03	THL esitti lokakuun lopussa tiedotustilaisuude...
11996	2022-11-03	@ArmedMagaVet45 #NoCovidAmnesty #CovidAmnesty ...
11997	2022-11-03	@ristomejide @FIDE_chess @davidllada Así empez...
11998	2022-11-03	【2022年11月3日付更新】#インウェブアウト留学センター 更新：ハワイ州の #新型コロナ...
11999	2022-11-03	Why did we have daily health and death updates...

Figure 7.3.1 Data Extraction using Twint

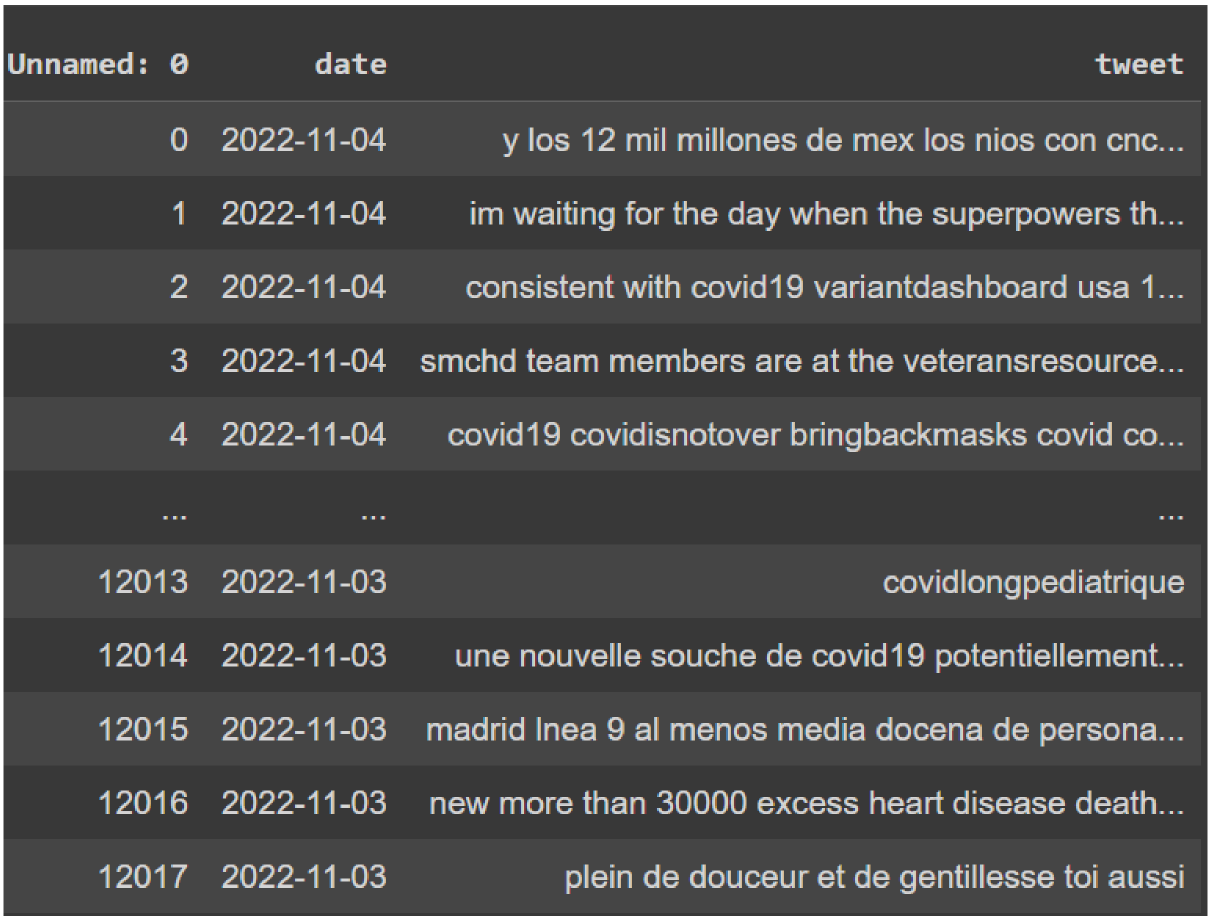
Figure 7.3.1 shows the data extracted using Twint. In which 12,000 tweets has been scrapped from the twitter.



```
Unnamed: 0        date                                               tweet
         0  2022-11-04   @lopezdoriga ¿y los 12 mil millones de @segalm...
         1  2022-11-04       i’m waiting for the day when the superpowers t...
         2  2022-11-04        consistent with #covid19 #variantdashboard #us...
         3  2022-11-04  smchd team members are at the #veteransresourc...
         4  2022-11-04    #covid19 #covidisnotover #bringbackmasks #covi...
       ...         ...                                                 ...
     12013  2022-11-03            #covidlongpediatrique https://t.co/u5icyti4n...
     12014  2022-11-03      une nouvelle souche de #covid19 potentiellemen...
     12015  2022-11-03  @metro_madrid línea 9, al menos media docena d...
     12016  2022-11-03     new: more than 30,000 excess heart disease dea...
     12017  2022-11-03        @alizaryana plein de douceur et de gentillesse...
```

Figure 7.3.2 Converting the collected tweets into lower text

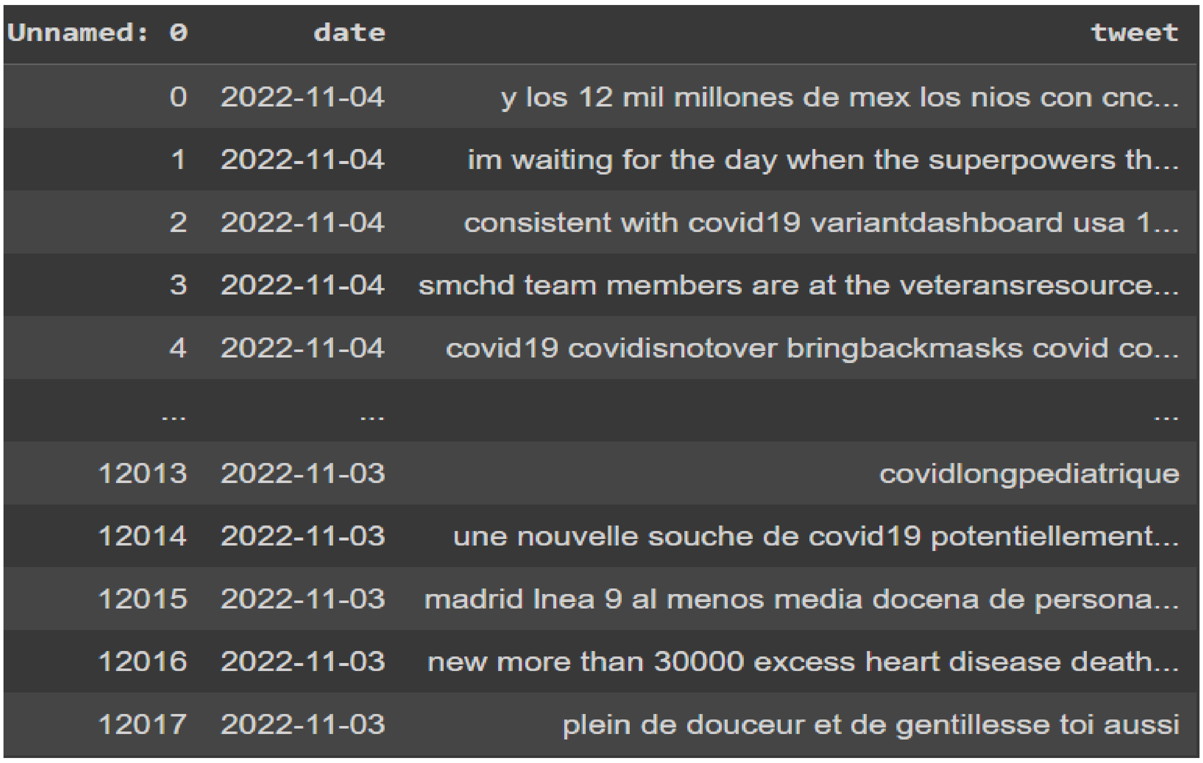
Figure 7.3.2 shows the scrapped tweets is converted into lower text.



Unnamed: 0	date	tweet
0	2022-11-04	y los 12 mil millones de mex los nios con cnc...
1	2022-11-04	im waiting for the day when the superpowers th...
2	2022-11-04	consistent with covid19 variantdashboard usa 1...
3	2022-11-04	smchd team members are at the veteransresource...
4	2022-11-04	covid19 covidisnotover bringbackmasks covid co...
...	...	...
12013	2022-11-03	covidlongpediatrique
12014	2022-11-03	une nouvelle souche de covid19 potentiellement...
12015	2022-11-03	madrid lnea 9 al menos media docena de persona...
12016	2022-11-03	new more than 30000 excess heart disease death...
12017	2022-11-03	plein de douceur et de gentillesse toi aussi

Figure 7.3.3 Removing the mentions

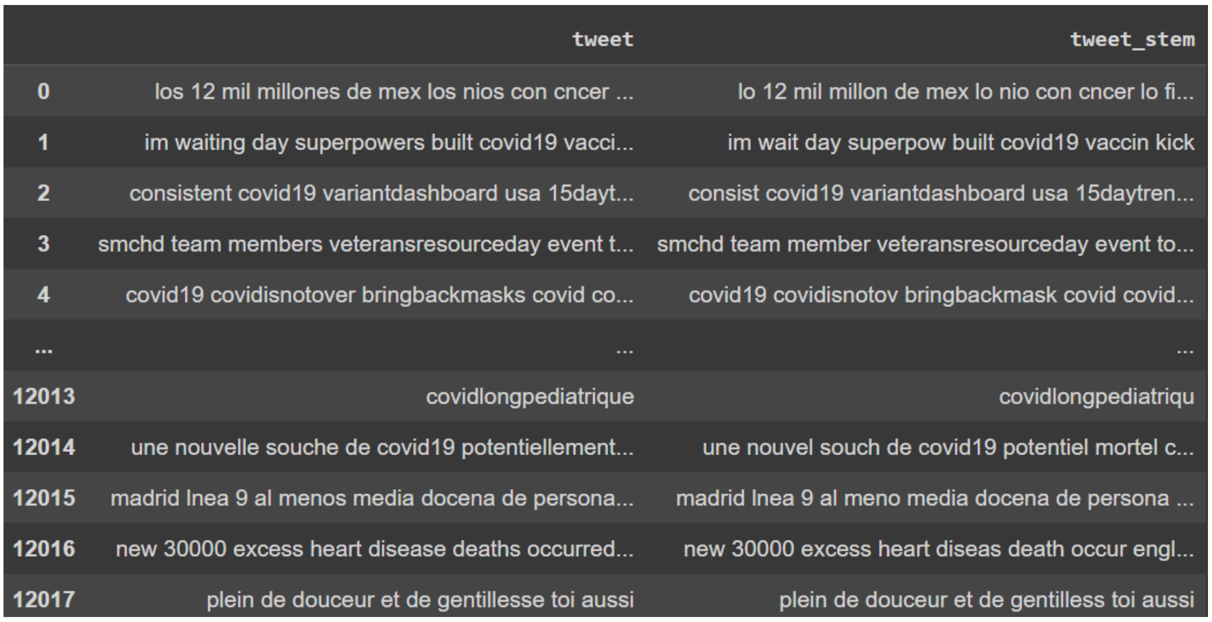
Figure 7.3.3 shows the removal of mentions in the tweets and removing multiple spaces



Unnamed: 0	date	tweet
0	2022-11-04	y los 12 mil millones de mex los nios con cnc...
1	2022-11-04	im waiting for the day when the superpowers th...
2	2022-11-04	consistent with covid19 variantdashboard usa 1...
3	2022-11-04	smchd team members are at the veteransresource...
4	2022-11-04	covid19 covidisnotover bringbackmasks covid co...
...	...	...
12013	2022-11-03	covidlongpediatrique
12014	2022-11-03	une nouvelle souche de covid19 potentiellement...
12015	2022-11-03	madrid lnea 9 al menos media docena de persona...
12016	2022-11-03	new more than 30000 excess heart disease death...
12017	2022-11-03	plein de douceur et de gentillesse toi aussi

Figure 7.3.4 Removing Punctuations

Figure 7.3.4 shows the removal of punctuations in the tweets.



	tweet	tweet_stem
0	los 12 mil millones de mex los nios con cncer ...	lo 12 mil millon de mex lo nio con cncer lo fi...
1	im waiting day superpowers built covid19 vacci...	im wait day superpow built covid19 vaccin kick
2	consistent covid19 variantdashboard usa 15dayt...	consist covid19 variantdashboard usa 15daytren...
3	smchd team members veteransresourceday event t...	smchd team member veteransresourceday event to...
4	covid19 covidisnotover bringbackmasks covid co...	covid19 covidisnotov bringbackmask covid covid...
...	...	...
12013	covidlongpediatrique	covidlongpediatriqu
12014	une nouvelle souche de covid19 potentiellement...	une nouvel souch de covid19 potentiel mortel c...
12015	madrid lnea 9 al menos media docena de persona...	madrid lnea 9 al meno media docena de persona ...
12016	new 30000 excess heart disease deaths occurred...	new 30000 excess heart diseas death occur engl...
12017	plein de douceur et de gentillesse toi aussi	plein de douceur et de gentilless toi aussi

Figure 7.3.5 Tweets after Stemming

Figure 7.3.5 shows the comparsion of normal tweets and tweets after stemming in done.

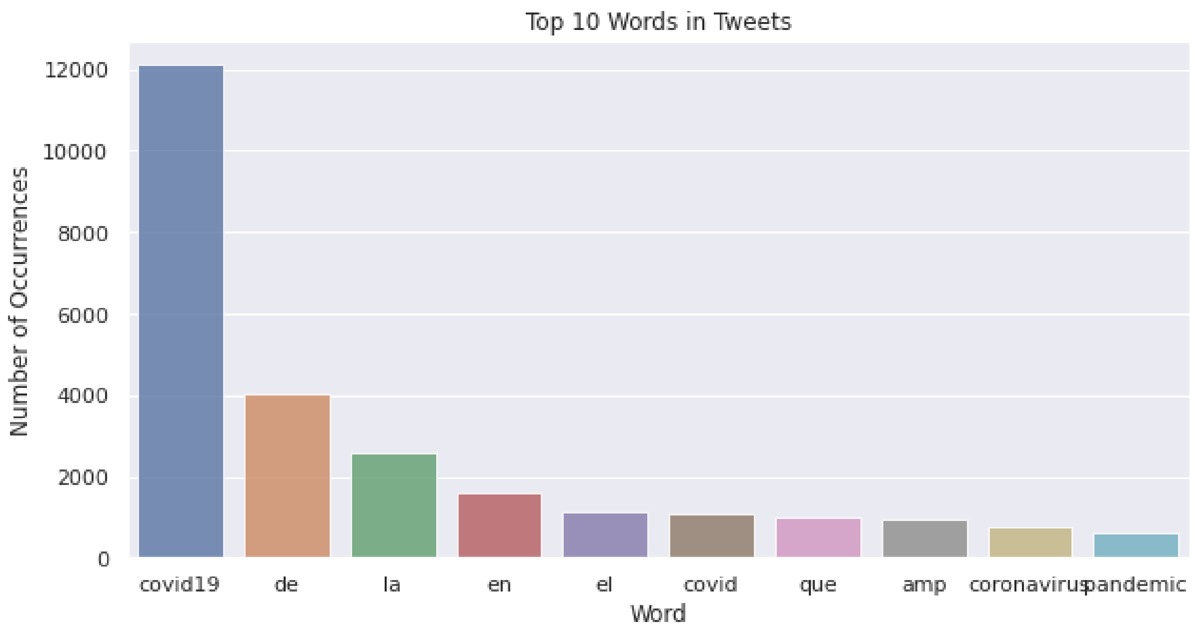


Figure 7.3.6 Unigram Analysis

Figure 7.3.6 shows the graph for top 10 words in the extracted tweets and how much times the word occurred in the overall tweets.

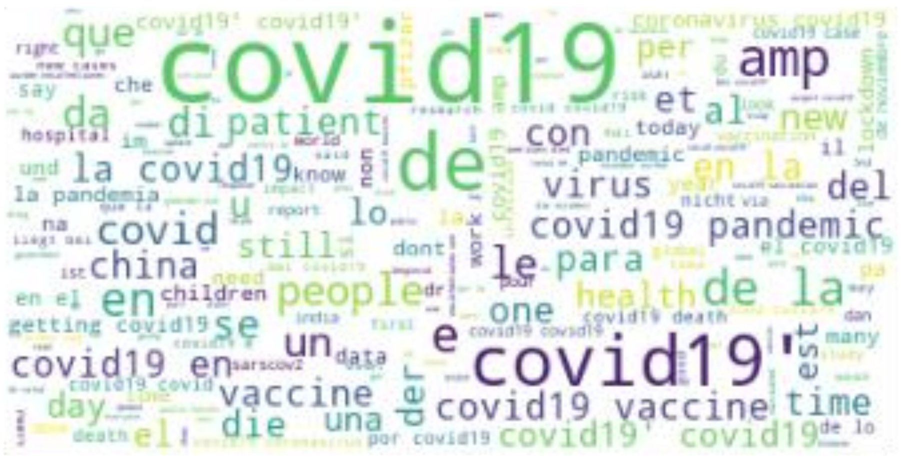


Figure 7.3.7 WordCloud

Figure 7.3.7 shows the wordcloud for what are the keywords repeadetly used by the users

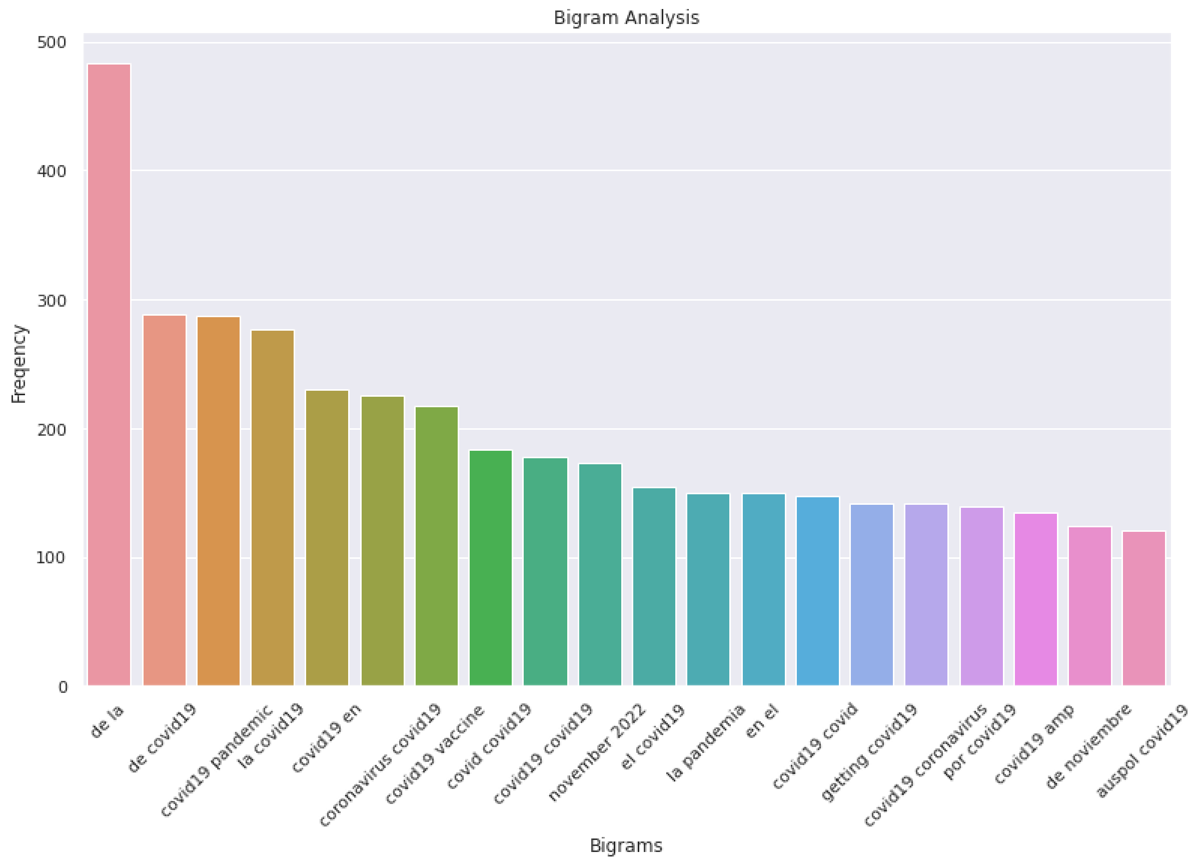


Figure 7.3.8 Bigram Analysis

Figure 7.3.8 shows the graph for bigram analysis in which top 20 words are visualized and the occurrences of the words

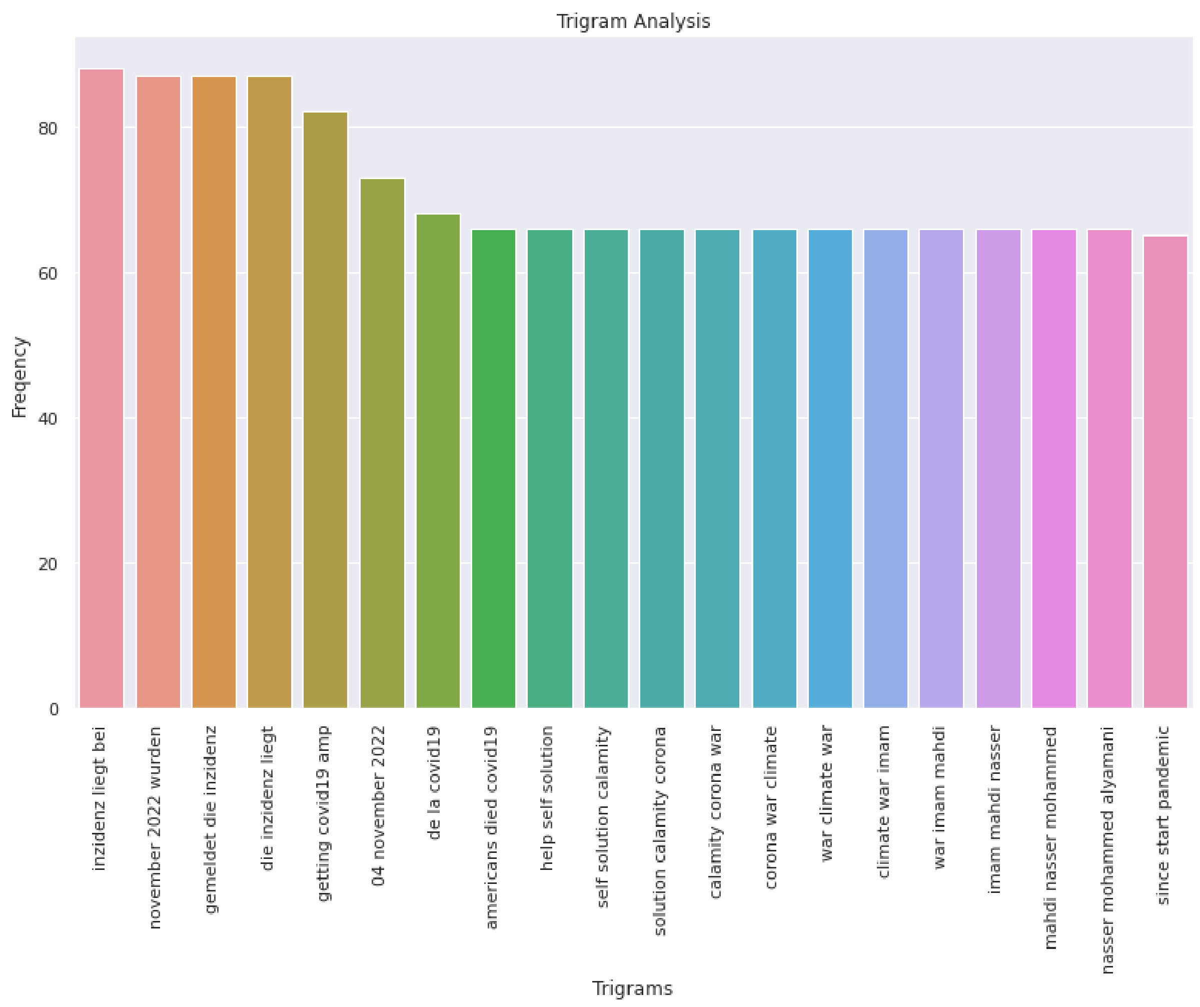


Figure 7.3.9 Trigram Analysis

Figure 7.3.9 shows the graph for trigram analysis in which top 20 words are visualized and the occurrences of the words

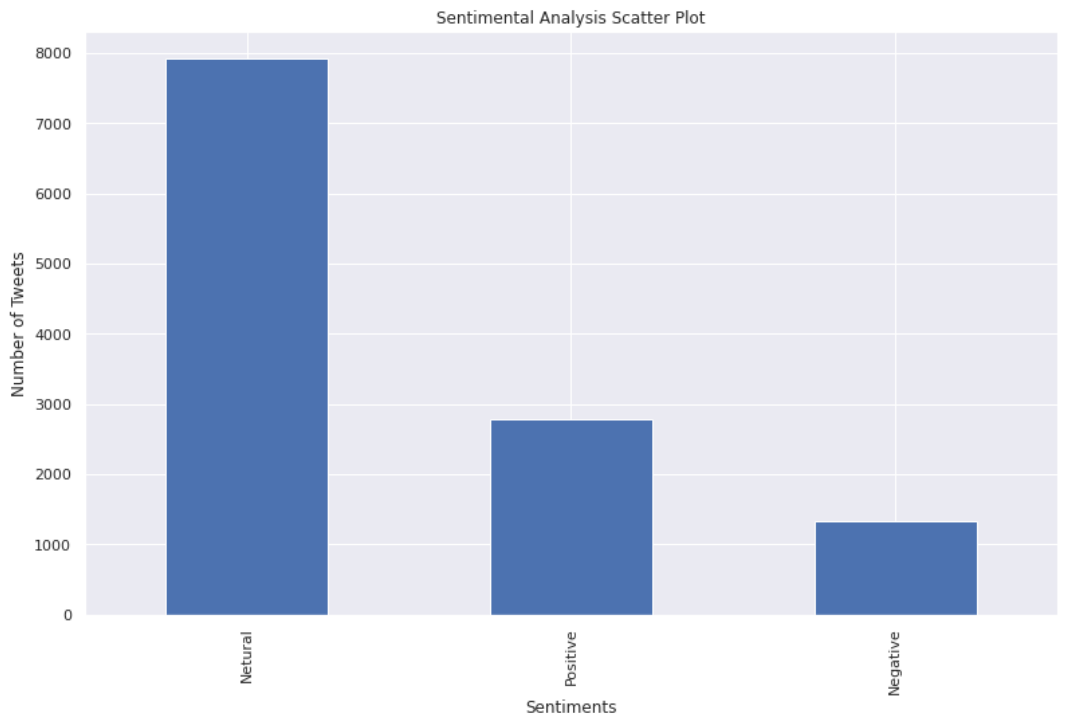


Figure 7.3.10 Sentiment Analysis Scatter Plot

Figure 7.3.10 shows the sentiment analysis scatter plot for Sentiments VS Number of Tweets and classify the tweets into positive, negative and neutral

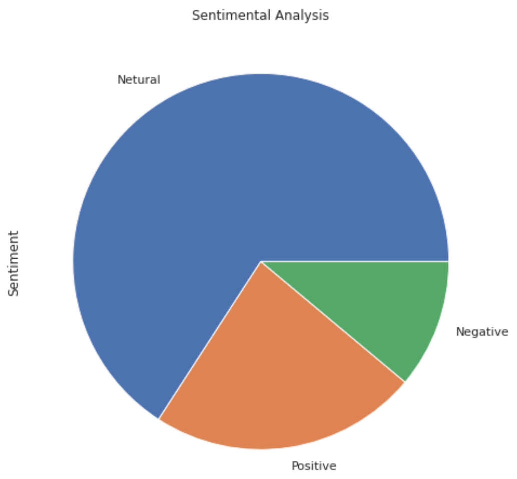


Figure 7.3.11 Sentiment Analysis Pie Chart

Figure 7.3.11 shows the sentiment analysis pie chart for Sentiments VS Number of Tweets and classify the tweets into positive, negative and neutral

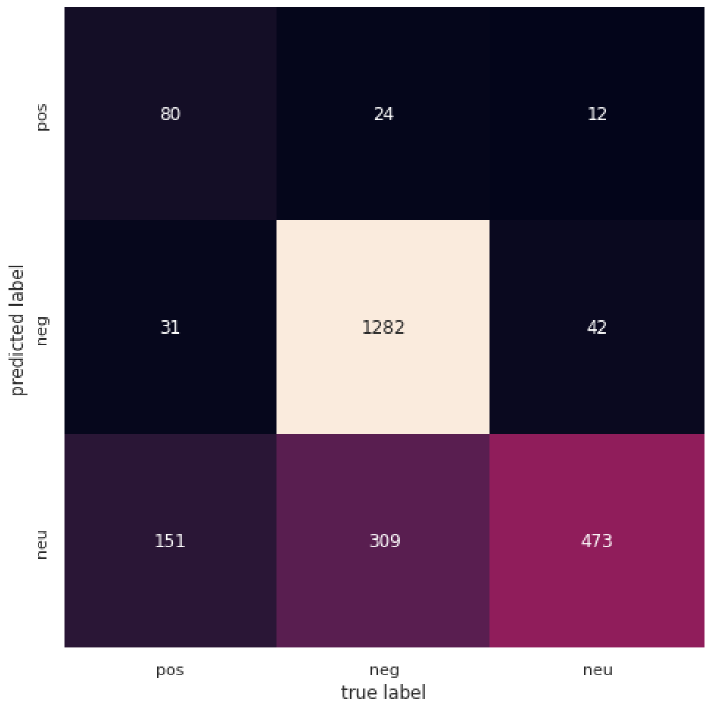


Figure 7.3.12 Confusion Matrix for True Label VS Predicted Label for Multinomial NaiveBayes

Figure 7.3.12 the graph shows the confusion matrix for Multinomial NaiveBayes Algorithm. Comparing with True Label VS Predicted Label

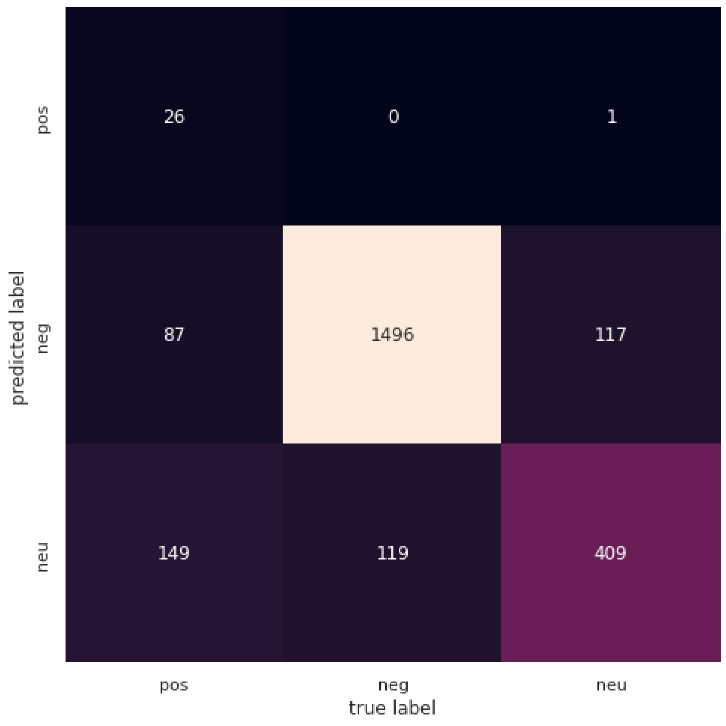


Figure 7.3.13 Confusion Matrix for True Label VS Predicted Label for BernoulliNaiveBayes

Figure 7.3.13 the graph shows the confusion matrix for Bernoulli NaiveBayes Algorithm. Comparing with True Label VS Predicted Label

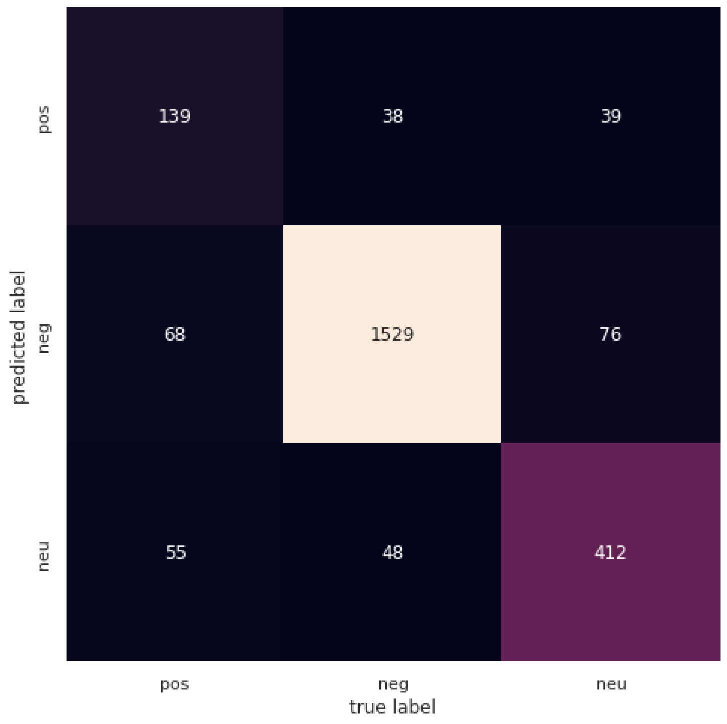


Figure 7.3.14 Confusion Matrix for True Label VS Predicted Label for Logistic Regression

Figure 7.3.14 the graph shows the confusion matrix for Logistic Regression Algorithm. Comparing with True Label VS Predicted Label

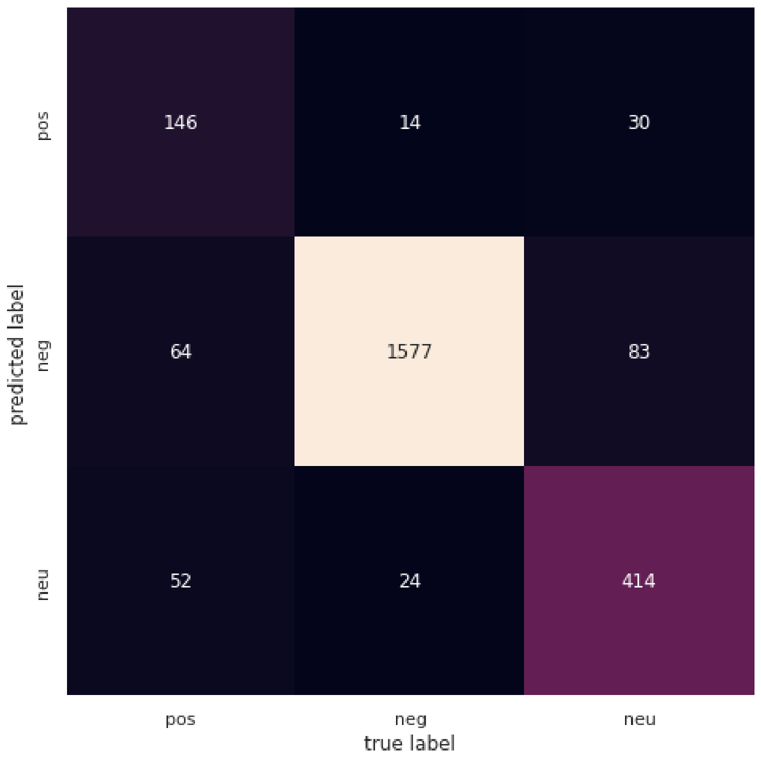


Figure 7.3.15 Confusion Matrix for True Label VS Predicted Label for Linear SVC

Figure 7.3.15 the graph shows the confusion matrix for Linear SVC Algorithm. Comparing with True Label VS Predicted Label

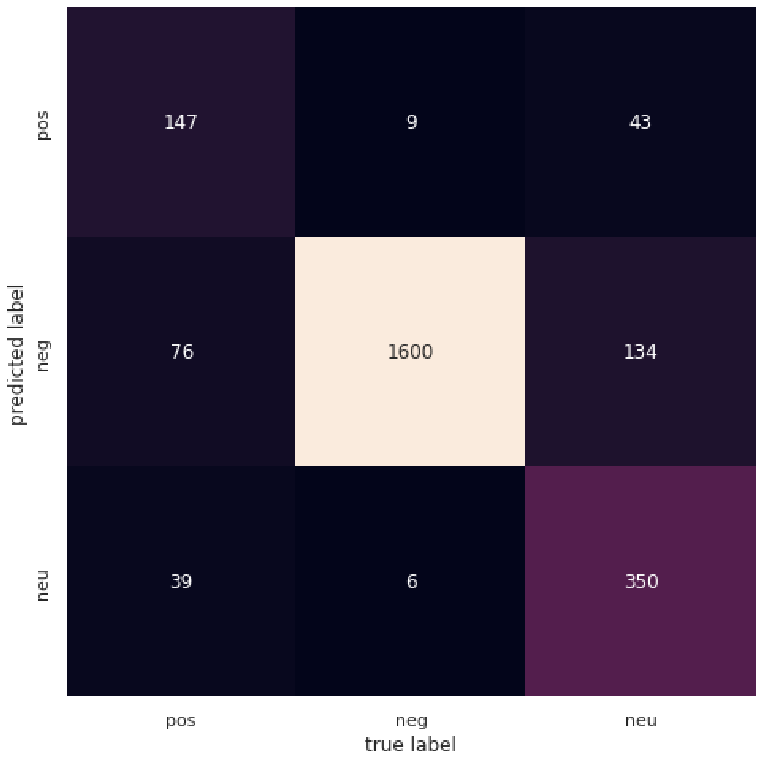


Figure 7.3.16 Confusion Matrix for True Label VS Predicted Label for AdaBoost Classifier

Figure 7.3.16 the graph shows the confusion matrix for AdaBoost Classifier Algorithm. Comparing with True Label VS Predicted Label

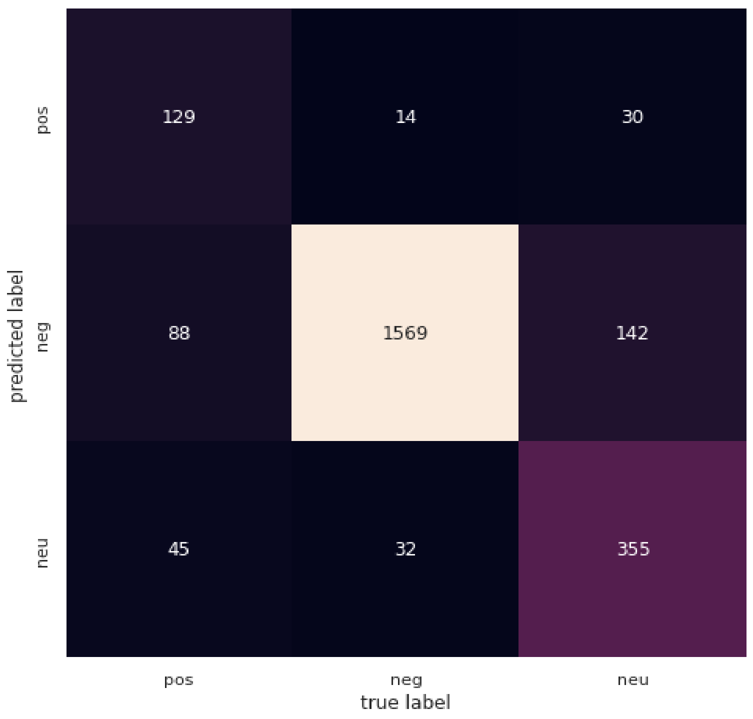


Figure 7.3.17 Confusion Matrix for True Label VS Predicted Label for RidgeClassifier

Figure 7.3.17 the graph shows the confusion matrix for Ridge Classifier Algorithm. Comparing with True Label VS Predicted Label

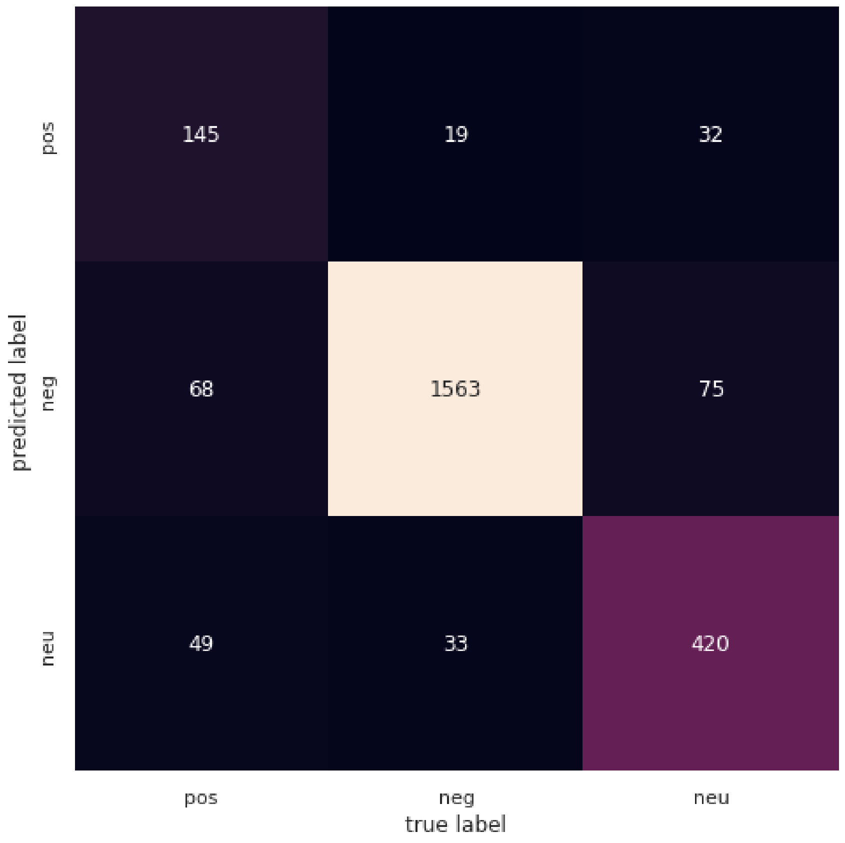


Figure 7.3.18 Confusion Matrix for True Label VS Predicted Label for PassiveAggressiveClassifier

Figure 7.3.18 the graph shows the confusion matrix for PassiveAggressive Classifier Algorithm. Comparing with True Label VS Predicted Label

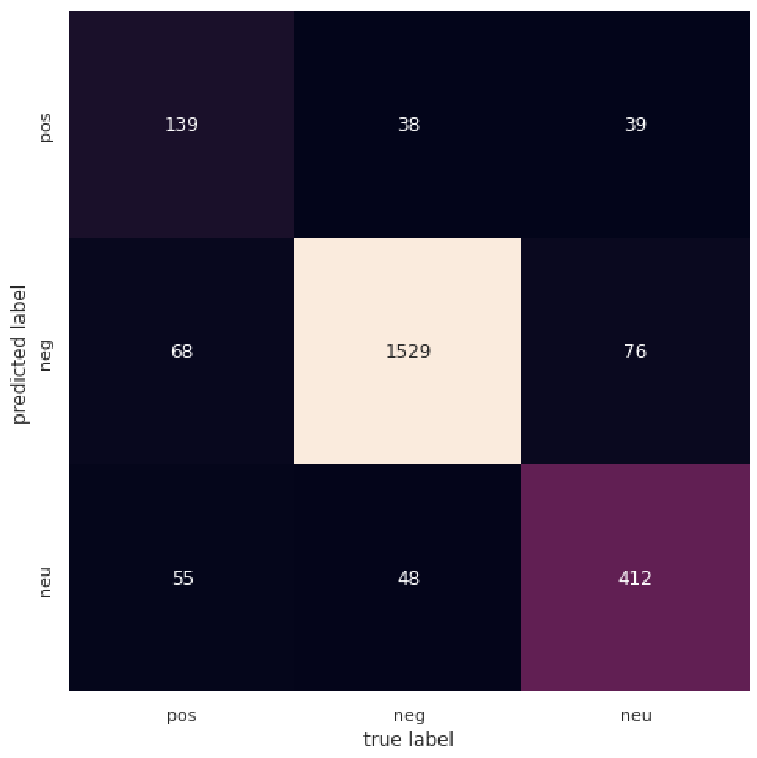


Figure 7.3.19 Confusion Matrix for True Label VS Predicted Label for Perceptron

Figure 7.3.19 the graph shows the confusion matrix for Perceptron Algorithm. Comparing with True Label VS Predicted Label

DATASET	PURITY	F-MEASURE
Covid19	0.88	0.86
Seoul	0.88	0.85
Firing	0.98	0.96

Table 7.3.1 Performance of stationary dataset

Table 7.3.1 shows the purity and F-measure value. The performance of stationary dataset like covid19, seoul, firing is displayed.

S.NO	**ALGORITHM**	**ACCURACY**
1.	MULTINOMIAL NAIVEBAYES	0.7633
2.	BERNOULLI NAIVEBAYES	0.8032
3.	LOGISTIC REGRESSION	0.8032
4.	LINEAR SVC	0.8889
5.	ADABOOST CLASSIFIER	0.8722
6.	RIDGE CLASSIFIER	0.8539
7.	PASSIVEAGGREESIVE CLASSIFIER	0.8851
8.	PERCEPTRON	0.8652

Table 7.3.2 Performance of all the supervised machine learning algorithms with the accuracy

CONCLUSION AND FUTURE WORK

CHAPTER 8
CONCLUSION

8.1 CONCLUSION

Social media is witnessing a massive increase in the number of users per day. People prefer to share their honest opinions on social media instead of sharing with someone in person. Using the posts from Twitter, we examined the common public's aggregate reaction toward the implementation of lockdown by the Indian government during the spread of COVID-19. Motivated by the mixed reactions coming after the announcement of lockdown in India, we collected tweets during phase 2 of lockdown in India. We have applied collected data to eight supervised machine learning techniques with different grams of text after annotation and preprocessing. We have observed the best performance with the LinearSVC classifier and unigram. The combination gives us an accuracy of 93.%, which is best in all the combinations which we have executed on our data set. We have consolidated the performance by calculating precision, recall, F1-Score, and tenfold cross-validation for all the combinations, and we got the best results with LinearSVC and unigram. So, we executed the sentiment analysis of tweets by the public during lockdown using this combination and found that almost half of the population (48.69%) is talking positive about the lockdown, 29.81% are neutral, and 21.5% of the people are feeling negative due to some reason

8.2 FUTURE WORK

In Future Work, we have planned to extract the trending tweets in the twitter and analyze the sentiments using NLP and we will improve the accuracy more using supervised machine learning algorithms.

APPENDIX

CHAPTER 9

APPENDIX

9.1 CODING:

```
#importing the required packages
import csv,re
from textblob import TextBlob
import tweepy
import re
import pandas as pd
import matplotlib.pyplot as plt

#Intializing the API Keys
consumer_key= "ennASdV55gqrbmVLxTon3furf"
consumer_secret= "v8GqFgbj9TLffgCLjE5r4fIiV2sIcKb68U7li8e1uVJb4R55Vu"
access_token= "1562124630651191296-lZBqOeK0R4DwIvmAIqKj10V7yFApJY"
access_token_secret= "sazY33olqA98GkHodxb8ucb74yd9BBw7wcrhvTZo58UeV"
auth= tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token= (access_token, access_token_secret)
api= tweepy.API(auth, wait_on_rate_limit=True)

#
def func():
    tweets = list(tweepy.Cursor(
        api.search, "{0}".format("COVID-19"), lang="en",tweet_mode='extended',
count=13000).items(13000))
    Tweet_outpt = []
    for i in tweets:
        text = i.full_text
        place = ""
        if i.place:
            place = i.place.full_name
        json_output = {
            "tweet": text,
        }
        Tweet_outpt.append(json_output)
    return Tweet_outpt
final_arr=func()
df = pd.DataFrame(final_arr)
df

#Cleaning the text
def cleanTwt(twt):
  twt = re.sub('COVID-19','COVID-19',twt)
  twt = re.sub('COVID19','COVID19',twt)
  twt = re.sub('#[A-Za-z0-9]+', '', twt)
  twt = re.sub('\\n', '', twt)
  twt = re.sub('https?:\/\/\S+', '', twt)
  twt = re.sub('@[^\s]+','',twt)
  twt = re.sub('RT','',twt)
  twt = re.sub('!','',twt)
  twt = ''.join([i if ord(i) < 128 else '' for i in twt])

  return twt

#cleaned tweets
df['cleaned tweets'] = df['tweet'].apply(cleanTwt)

df.head(100)
```

#Text Blob (Data Labeling)

```
def getSubjectivity(twt):
  return TextBlob(twt).sentiment.subjectivity

def getPolarity(twt):
  return TextBlob(twt).sentiment.polarity

#creating two columns subjectivity and polarity

df['Subjectivity'] = df['cleaned tweets'].apply(getSubjectivity)
df['Polarity'] = df['cleaned tweets'].apply(getPolarity)

df.head(100)
```

#Creating a function for sentiment text

```
def getSentiment(score):
  if score < 0:
    return 0

  elif score == 0:
    return 2
  elif score > 0:
    return 1

df['Sentiment'] = df['Polarity'].apply(getSentiment)

df.head(10000)
```

#VADER (Data Labelling)

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download("vader_lexicon")
sentiments = SentimentIntensityAnalyzer()
tweets["Positive"] = [sentiments.polarity_scores(i)["pos"] for i in tweets["tweet"]]
tweets["Negative"] = [sentiments.polarity_scores(i)["neg"] for i in tweets["tweet"]]
tweets["Neutral"] = [sentiments.polarity_scores(i)["neu"] for i in tweets["tweet"]]
tweets['Compound'] = [sentiments.polarity_scores(i)["compound"] for i in tweets["tweet"]]
tweets.head()
```

```
score = tweets["Compound"].values
sentiment = []
for i in score:
    if i > 0 :
        sentiment.append('Positive')
    elif i < 0 :
        sentiment.append('Negative')
    else:
        sentiment.append('Neutral')
tweets["Sentiment"] = sentiment
tweets.head(100)


print(tweets["Sentiment"].value_counts())


#libraries for preprocessing using NLTK
import nltk
from nltk import word_tokenize, FreqDist
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download
nltk.download('wordnet')
nltk.download('stopwords')
from nltk.tokenize import TweetTokenizer

nltk.download('punkt')
nltk.download('wordnet')
from nltk import sent_tokenize, word_tokenize


#Splitting Data
df=pd.DataFrame(tweets['cleaned tweets'])
split_data = df["cleaned tweets"].str.split(" ")
split_data

##Tokenization

data=split_data
s_new = []
def tokenization_s(data):
    for sent in (data[:][2]):
        s_token = sent_tokenize(sent)
        if s_token != '':
            s_new.append(s_token)
    return s_new

print(tokenization_s(data))

df1=pd.DataFrame(tweets['cleaned tweets'])
df1["Sentiment"]=tweets["Sentiment"]
```

```
df1['cleaned tweets'].dropna(inplace=True)
df1['cleaned tweets'] = df1['cleaned tweets'].astype(str)
df1.head(11000)


def tokenize(column):
    tokens = nltk.word_tokenize(column)
    return [w for w in tokens if w.isalpha()]
df1['tokenization'] = df1.apply(lambda x: tokenize(x['cleaned tweets']), axis=1)
df1[['tokenization']].head(13000)
```

#Stemming

```
from nltk.stem.snowball import SnowballStemmer
stemmer = SnowballStemmer("english")
df1['stemming'] = df1['tokenization'].apply(lambda x: [stemmer.stem(y) for y in x])
df1
```

#Lemmatization

```
import nltk
nltk.download('omw-1.4')
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()
def lemmatize_text(text):
    return [lemmatizer.lemmatize(w) for w in w_tokenizer.tokenize(text)]
df1['text_lemmatized'] = df1["cleaned tweets"].apply(lemmatize_text)

df1.head()
```

#Stop words

```
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize


filtered_word_list =tweets['cleaned tweets'] #make a copy of the word_list
for tweet in tweets: # iterate over word_list
  if tweet in stopwords.words('english'):
    filtered_word_list.remove(word)

filtered_word_list
```

#CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
count_vectorize = CountVectorizer()
vectorized = count_vectorize.fit_transform(df1["cleaned tweets"])

print(vectorized[0,:])

print(count_vectorize.get_feature_names_out())
print(count_vectorize.get_feature_names_out()[10000], vectorized[0,10000])
```

##MultinomialNB

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df1,df1["Sentiment"],test_size = 0.2, random_state
=26105111)

from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
vectorized = count_vectorize.fit_transform(X_train["cleaned tweets"])
clf.fit(vectorized, y_train)

vectorized = count_vectorize.transform(X_test["cleaned tweets"])
res=clf.predict(vectorized)

from sklearn import metrics
metrics.accuracy_score(res, y_test)

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
mat = confusion_matrix(y_test, res)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
        xticklabels=["pos","neg","neu"], yticklabels=["pos","neg","neu"])
plt.xlabel('true label')
plt.ylabel('predicted label')
```

##BernoulliNB

```
from sklearn.naive_bayes import BernoulliNB
clf = BernoulliNB()
vectorized = count_vectorize.fit_transform(X_train["cleaned tweets"])
clf.fit(vectorized, y_train)

vectorized = count_vectorize.transform(X_test["cleaned tweets"])
res=clf.predict(vectorized)

res

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
mat = confusion_matrix(y_test, res)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
        xticklabels=["pos","neg","neu"], yticklabels=["pos","neg","neu"])
plt.xlabel('true label')
plt.ylabel('predicted label')

from sklearn import metrics
metrics.accuracy_score(res, y_test)
```

## LogisticRegression

```
from sklearn.linear_model import LogisticRegression
logisticRegr = LogisticRegression()
vectorized = count_vectorize.fit_transform(X_train["cleaned tweets"])
logisticRegr.fit(vectorized, y_train)

vectorized = count_vectorize.transform(X_test["cleaned tweets"])
res=logisticRegr.predict(vectorized)

res

metrics.accuracy_score(res, y_test)

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
mat = confusion_matrix(y_test, res)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=["pos","neg","neu"], yticklabels=["pos","neg","neu"])
plt.xlabel('true label')
plt.ylabel('predicted label')
```

##LinearSVC

```
from sklearn.svm import LinearSVC
clf=LinearSVC()
vectorized = count_vectorize.fit_transform(X_train["cleaned tweets"])
clf.fit(vectorized, y_train)


vectorized = count_vectorize.transform(X_test["cleaned tweets"])
res=clf.predict(vectorized)

metrics.accuracy_score(res, y_test)

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
mat = confusion_matrix(y_test, res)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=["pos","neg","neu"], yticklabels=["pos","neg","neu"])
plt.xlabel('true label')
plt.ylabel('predicted label')
```

##AdaBoostClassifier

```
from sklearn.ensemble import AdaBoostClassifier
clf = AdaBoostClassifier(n_estimators=100, random_state=0)
vectorized = count_vectorize.fit_transform(X_train["cleaned tweets"])
clf.fit(vectorized, y_train)


vectorized = count_vectorize.transform(X_test["cleaned tweets"])
res=clf.predict(vectorized)
```

```
metrics.accuracy_score(res, y_test)

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
mat = confusion_matrix(y_test, res)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
        xticklabels=["pos","neg","neu"], yticklabels=["pos","neg","neu"])
plt.xlabel('true label')
plt.ylabel('predicted label')
```

##RidgeClassifier

```
from sklearn.linear_model import RidgeClassifier
clf = RidgeClassifier()
vectorized = count_vectorize.fit_transform(X_train["cleaned tweets"])
clf.fit(vectorized, y_train)


vectorized = count_vectorize.transform(X_test["cleaned tweets"])
res=clf.predict(vectorized)

metrics.accuracy_score(res, y_test)

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
mat = confusion_matrix(y_test, res)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
        xticklabels=["pos","neg","neu"], yticklabels=["pos","neg","neu"])
plt.xlabel('true label')
plt.ylabel('predicted label')
```

##PassiveAggressiveClassifier

```
from sklearn.linear_model import PassiveAggressiveClassifier
clf = PassiveAggressiveClassifier(random_state=0)
vectorized = count_vectorize.fit_transform(X_train["cleaned tweets"])
clf.fit(vectorized, y_train)


vectorized = count_vectorize.transform(X_test["cleaned tweets"])
res=clf.predict(vectorized)

metrics.accuracy_score(res, y_test)

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
mat = confusion_matrix(y_test, res)
```

```
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
        xticklabels=["pos","neg","neu"], yticklabels=["pos","neg","neu"])
plt.xlabel('true label')
plt.ylabel('predicted label')
```

##Perceptron

```
from sklearn.linear_model import Perceptron
clf = Perceptron(random_state=0)
vectorized = count_vectorize.fit_transform(X_train["cleaned tweets"])
clf.fit(vectorized, y_train)


vectorized = count_vectorize.transform(X_test["cleaned tweets"])
res=clf.predict(vectorized)

metrics.accuracy_score(res, y_test)

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
mat = confusion_matrix(y_test, res)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
        xticklabels=["pos","neg","neu"], yticklabels=["pos","neg","neu"])
plt.xlabel('true label')
plt.ylabel('predicted label')
```

REFERENCES

CHAPTER 10
REFERENCES

10.1 REFERENCES:

[1] J. T. Wu, K. Leung, and G. M. Leung, “Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China: A modeling study,” Obstetrical Gynecolo. Surv., vol. 75, no. 7, pp. 399–400, Jul. 2020.

[2] R. J. Medford, S. N. Saleh, A. Sumarsono, T. M. Perl, and C. U. Lehmann, “An ‘infodemic’: Leveraging high-volume Twitter data to understand public sentiment for the COVID-19 outbreak,” medRxiv, Jan. 2020, doi: 10.1101/2020.04.03.20052936.

[3] S. Li, Y. Wang, J. Xue, N. Zhao, and T. Zhu, “The impact of COVID-19 epidemic declaration on psychological consequences: A study on active Weibo users,” Int. J. Environ. Res. Public Health, vol. 17, no. 6, p. 2032, Mar. 2020.

[4] WHO Statement Regarding Cluster of Pneumonia Cases, WHO, Wuhan, China, 2020.

[5] R. Pandey et al., “A machine learning application for raising WASH awareness in the times of COVID-19 pandemic,” 2020, arXiv:2003.07074. [Online]. Available: http://arxiv.org/abs/2003.07074

[6] A. S. M. Kayes, M. S. Islam, P. A. Watters, A. Ng, and H. Kayesh, “Automated measurement of attitudes towards social distancing using social media: A COVID-19 case study,” Tech. Rep., Oct. 2020.

[7] C. K. Pastor, “Sentiment analysis on synchronous online delivery of instruction due to extreme community quarantine in the Philippines caused by Covid-19 pandemic,” Asian J. Multidisciplinary Stud., vol. 3, no. 1, pp. 1–6, Mar. 2020.

[8] A. D. Dubey, “Decoding the Twitter sentiments towards the leadership in the times of COVID-19: A case of USA and india,” SSRN Electron. J., Apr. 2009, doi: 10.2139/ssrn.3588623.

[9] L. Chen, H. Lyu, T. Yang, Y. Wang, and J. Luo, “In the eyes of the beholder: Analyzing social media use of neutral and controversial terms for COVID-19,” 2020, arXiv:2004.10225. [Online]. Available: http://arxiv.org/abs/2004.10225

[10] G. Barkur, Vibha, and G. B. Kamath, “Sentiment analysis of nationwide lockdown due to COVID 19 outbreak: Evidence from India,” Asian J. Psychiatry, vol. 51, Jun. 2020, Art. no. 102089.

[11] M. Alhajji, K. A. Al, M. Aljubran, and M. Alkhalifah, "Sentiment analysis of tweets in Saudi Arabia regarding governmental preventive measures to contain COVID-19," Dept. Social Behav. Sci., College Public Health, Temple Univ., Philadelphia, PA, USA, Tech. Rep., doi: 10.20944/preprints202004.0031.v1.
[12] J. Samuel, A. GG, M. Rahman, E. Esawi, and Y. Samuel, "Covid19 public sentiment insights and machine learning for tweets classification. Nawaz and Rahman, Md. Mokhlesur and Esawi, Ek and Samuel, Yana," Information, vol. 11, no. 6, pp. 1–22, Apr. 2020, doi: 10.3390/info11060314.
[13] R. Liu, Y. Shi, C. Jia, and M. Jia, "A survey of sentiment analysis based on transfer learning," IEEE Access, vol. 7, pp. 85401–85412, 2019.
[14] N. Kaka et al., "Digital India: Technology to transform a connected nation," McKinsey Global Inst., India, Tech. Rep., Mar. 2019. [Online]. Available: https://www.mckinsey.com/~/media/McKinsey/Business%20
Functions/McKinsey%20Digital/Our%20Insights/Digital%20India%20
Technology%20to%20transform%20a%20connected%20nation/MGIDigital-India-Report-April-2019.pdf
[15] A. Abd-Alrazaq, D. Alhuwail, M. Househ, M. Hamdi, and Z. Shah, "Top concerns of tweeters during the COVID-19 pandemic: Infoveillance study," J. Med. Internet Res., vol. 22, no. 4, Apr. 2020, Art. no. e19016.
[16] P. Burnap et al., "Tweeting the terror: Modelling the social media reaction to the woolwich terrorist attack," Social Netw. Anal. Mining, vol. 4, no. 1, p. 206, Dec. 2014.
[17] B. R. Naiknaware and S. S. Kawathekar, "Prediction of 2019 Indian election using sentiment analysis," in Proc. 2nd Int. Conf., Aug. 2018, pp. 660–665.
[18] D. D. Wu, L. Zheng, and D. L. Olson, "A decision support approach for online stock forum sentiment analysis," IEEE Trans. Syst., Man, Cybern. Syst., vol. 44, no. 8, pp. 1077–1087, Aug. 2014.
[19] J. Ding, H. Sun, X. Wang, and X. Liu, "Entity-level sentiment analysis of issue comments," in Proc. 3rd Int. Workshop Emotion Awareness Softw. Eng., Jun. 2018, pp. 7–13.
[20] M. Pota, M. Esposito, M. A. Palomino, and G. L. Masala, "A subwordbased deep learning approach for sentiment analysis of political tweets," in Proc. 32nd Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA), May 2018, pp. 651–656.