

USING SENTIMENT AND EMOTION
ANALYSIS OF NEWS ARTICLES
TO ANALYZE THE EFFECTS OF
LEADER'S STATEMENTS ON
COVID-19 SPREAD

by

POOJITHA THOTA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science and
Engineering at The University of Texas at Arlington
May, 2021

Arlington, Texas

Supervising Committee:

Ramez A Elmasri, Supervising Professor
David Levine
Leonidas Fegaras

ABSTRACT

USING SENTIMENT AND EMOTION ANALYSIS OF NEWS ARTICLES TO ANALYZE THE EFFECTS OF LEADER'S STATEMENTS ON COVID-19 SPREAD

Poojitha Thota, Master's

The University of Texas at Arlington, 2021

Supervising Professor(s): Dr Ramez A Elmasri

Leaders generally include government officials, politicians, etc. Their statements can highly affect people's decisions in many ways. Currently, in the pandemic situation, many statements were being passed every hour and day, which showed an impact on the spread of corona virus cases at certain location. So, this paper proposes a supervised model to analyze the variations of COVID-19 data based upon the leader's statements passed at certain time and location. The proposed methodology consists of sentiment and emotion analysis for the leader's statements to determine the true intentions of the leader. The leader's statements are a collection of data obtained by scraping webpages of popular news channels like CNN. And the COVID-19 data has been extracted from the largest collection, which is accumulated by John Hopkins University, every day. Then, NLP text processing techniques were used to prepare the dataset and pre-process the text, through which we obtain a labelled dataset. From this, our methodology includes analyzing the obtained dataset.

Copyright by
Poojitha Thota
2021

ACKNOWLEDGEMENTS

I wish to express my gratitude to my thesis advisor, Dr Ramez A Elmasri, for his generous advice, inspiring guidance, and encouragement throughout my research for this work. I would also like to thank my peers Mary Elizabeth Koone and Mohammad A.Shaito, members of MAST-DB lab at UTA, for their guidance and support throughout this project.

TABLE OF CONTENTS

ABSTRACT	I
ACKNOWLEDGEMENTS.....	III
LIST OF FIGURES	V
LIST OF TABLES.....	VII
LIST OF ABBREVIATIONS.....	VIII
CHAPTER 1: INTRODUCTION.....	9
CHAPTER 2: RELATED WORK.....	12
CHAPTER 3: DATA COLLECTION	21
CHAPTER 4: DATASET PREPARATION.....	42
CHAPTER 5: SENTIMENT ANALYSIS	59
CHAPTER 6: EMOTION ANALYSIS	75
CHAPTER 7: RESULTS AND ANALYSIS	79
CHAPTER 8: CONCLUSION AND FUTURE WORK	83
APPENDIX	
1. APPENDIX 1.....	85
2. APPENDIX 2.....	87
REFERENCES	88

LIST OF FIGURES

Figure	Page
1. Figure 2.1 Sentiment Classification Techniques.....	15
2. Figure 2.2 Plutchik's Wheel of Emotions.....	20
3. Figure 3.1 Overview of Web Scraping Process.....	24
4. Figure 4.1 Output when Punkt tokenizer is applied to text at sentence level.....	44
5. Figure 4.2 Output when Punkt tokenizer is applied to text at word level.....	45
6. Figure 4.3 Output showing differences between Stemmer and Lemmatizer.....	46
7. Figure 4.4 Language processing tasks and corresponding NLTK modules with examples of functionality.....	49
8. Figure 4.5 Summarized process of finding named entities in text.....	53
9. Figure 4.6 Required libraries imported from NLTK.....	54
10. Figure 4.7: Instance of tokenized output of input text.....	55
11. Figure 4.8: Instance of POS Tagging.....	56
12. Figure 4.9: Instance of NE Chunker.....	57
13. Figure 4.10: Instance of Final output obtained from Named Entity Algorithm.....	57
14. Figure 5.1 Sample output after removing stop words from dataset.....	61
15. Figure 5.2 Sample input text before lemmatization.....	62
16. Figure 5.3 Sample output after lemmatization.....	62
17. Figure 5.4 Movie Reviews Dataset after Cleaning.....	63
18. Figure 5.5 Movie Reviews Dataset for training Classifiers.....	63
19. Figure 5.6 Accuracy with Linear SVC.....	64
20. Figure 5.7 Improved accuracy with Linear SVC with n-gram range (1,2).....	65

21. Figure 5.8 Improved accuracy with Linear SVC.....	65
22. Figure 5.9 Input Layers of Multi-layer perceptron model.....	67
23. Figure 5.10 Accuracy obtained from Multi-layer perceptron model.....	67
24. Figure 5.11 Pre-training and Fine-Tuning procedures for BERT.....	68
25. Figure 5.12 Output of Classifier after using BERT.....	69
26. Figure 5.13 Model Summary of BERT.....	71
27. Figure 5.14 Usage of function InputExample.....	71
28. Figure 5.15 Fine tuning of BERT Model.....	72
29. Figure 7.1 Figure showing number of confirmed cases of COVID-19 in the state of Georgia during the month of July.....	80
30. Figure 7.2 Figure showing different Emotions in Positive Statements given by Governor, State of Georgia.....	81
31. Figure 7.3 Figure showing different Emotions in Negative Statements given by Governor, State of Georgia.....	82

LIST OF TABLES

Table	Page
1. Table 2.1 Different types of Lexicons.....	17
2. Table 2.2 Basic Emotion classes identified by researchers.....	19
3. Table 3.1 Output showing top 5 news stories and headlines on August 13th, 2020.....	41
4. Table 4.1 Sample of dataset showing statements provided by leaders in state of Georgia.....	58
5. Table 5.1 Summary of Experiments conducted for sentiment classifier.....	69
6. Table 5.2 Sample of dataset showing statements provided by leaders in state of Georgia with Sentiment of Headline and Story.....	73
7. Table 6.1 Summary of Experiments conducted for Emotion classifier.....	77
8. Table 6.2 Sample of dataset showing statements provided by leaders in state of Georgia with Emotion of Headline and Story.....	78
9. Table A-1: NLTK Pos tags with their meanings.....	85

LIST OF ABBREVIATIONS

1. NLP: Natural Language Processing
2. ML: Machine Learning
3. HTML: Hyper Text Markup Language
4. DOM: Document Object Model
5. NER: Named Entity Recognition
6. POS Tagging: Parts Of Speech Tagging
7. NLTK: Natural Language ToolKit
8. BoW: Bag of Words
9. TF-IDF: Term Frequency- Inverse Document Frequency
10. GPE: Geo-Political Entities
11. MaxEnt: Maximum Entropy
12. ACE: Automatic Content Extraction
13. NEC: Named Entity Chunker
14. BERT: Bidirectional Encoder Representation from Transformers
15. ADAM: Derived from Adaptive Moment Estimation
16. ISEAR: International Survey on Emotion Antecedents and Reactions

CHAPTER 1

INTRODUCTION

1.1 Introduction and Problem Statement

In the current era, COVID-19 is the biggest pandemic which rapidly affected our day-to-day life in huge number of ways. Many leaders like state and federal government officials, during this pandemic, have played a major role in motivating people, issued several safety precautions and preventive measures to public and showed concern towards their assigned regions. Many safety measures like mask mandate, six feet social distancing, staying sanitized, etc. have been introduced to prevent the spread of the virus. But there were many areas where such measures have not been followed by the public, as there were no strict orders from the government, that lead to growth in the number of cases. Looking at such scenarios has given us an idea that there might be some relationship existing between the type of leader statement or order, its implementation, and the number of cases in that region.

But to find such relationships, the crucial component is a dataset that can consist of different leader's statements from news channels, the region they have been assigned for during the pandemic, the sentiment and emotion involved in their statements, and finally the COVID-19 data for the entire pandemic, focusing mainly on the states of USA. To create such huge dataset, one of the solutions opted in this project is to scrape the websites of popular news channels like CNN, which regularly stream live news and store the archives. As we focused on states of USA, the news obtained is filtered into categories like International, National, and states. Each statement once categorized, leader stating is also found through NLP text processing techniques. The international category of statements was not used for final analysis but was used in training dataset.

Then to judge how the leader speaks and to know the nature of statements, sentiment analysis is considered as the best option. Sentiment analysis in brief is a basic process of analyzing the subjective information involved in the sentence and classify it into three major categories of positive, negative, and neutral. To analyze the way, the leader speaks and to know how concerned they are towards their region, such type of classification is insufficient and cannot provide a complete understanding of their statements. This limitation has led us to have an idea of performing emotion analysis of the leader's statements, along with sentiment analysis. This gave us a scope to classify both sentiment and emotion involved in the statements to have a complete understanding.

To perform such classification, we have opted for supervised method of classification by building two different machine learning models with two different datasets. One of the machine learning models performs sentiment classification and the other performs the emotion classification. The predictions obtained by these two models were integrated, and then transformed the actual dataset into a labelled dataset. After obtaining the desired dataset, a new machine learning model is trained with it and used for analyzing the relationship between leader's statements and the spread of COVID-19 in the leader's region. Based on the output, we can conclude how concerned the leader was towards his/her state and type of emotions they had during the pandemic. Detailed explanation about each model and the methodology implemented will be discussed in later sections.

This type of analysis not only limits to COVID-19 pandemic, but it can also be used for other applications where leader's statements play a crucial role.

1.2 Structure of Thesis

This thesis is structured into 8 chapters. The problem statement and introduction about analysis is discussed in chapter 1. In chapter 2, literature review concerning Web scraping, Natural Language Processing methods, Sentiment and Emotion analysis, Deep learning methods for classification is provided. Chapter 3 illustrates the web scraping process to collect the dataset from popular news channel like CNN. Chapter 4 discusses about the NLP approaches to prepare and clean the dataset as per requirements of analysis. Chapters 5 and 6 describes the methodology used for sentiment and emotion analysis. Chapter 7 discusses about the results obtained through analyzing the datasets of Leader's statements and COVID-19. Conclusions are drawn and future work is stated in chapter 8.

CHAPTER 2

RELATED WORK

This thesis mainly covers three major research areas: (1) Web Scraping methods to collect and create dataset; (2) Natural Language Processing to clean and prepare the dataset for analysis; (3) Sentiment Analysis to acquire the polarity in leader's statements and (4) Emotion Analysis to extend towards predicting the true intentions of leaders by knowing the emotions, using deep learning methods.

This chapter is divided into three parts and each part provides the scientific background of the above-mentioned topics relevant to this thesis. The order of these parts follows the workflow of thesis. The subchapter 2.1 is about Web scraping processes, in which novel methods to web scrape and collecting the data results in a dataset that contains leader's statements to use further. The subchapter 2.2 is about natural language processing techniques, which are used in dataset preparation and cleaning. The subchapters 2.3 and 2.4 consists of works relevant to sentiment and Emotion analysis that can be performed on clean dataset to know the intentions of a leader through his/her statements.

2.1 Web Scraping

Web Scraping is important technique used for extracting unstructured data from the websites and transforming that data into structured. Web Scraping is also identified as web data extraction, web data scraping, web harvesting or screen scraping. Web scraping is a form of data mining [1]. The web scrapers were basically initiated with an aim to transform unstructured data to structured format, that can further be stored in databases. There are many methods to scrape information from the Web [2]. The most effective method is traditional copy-paste. But this might

not be feasible in real time scenarios where huge amount of data has to be extracted. One of the other methods is text grepping where regular expressions are used to find information that matches the pattern provided. More web scraping techniques include HTML Parsing, DOM Parsing, etc. which will be discussed in later sections providing detailed description. Finally, a Web scraping method consists of making scraper sites that are automatically generated from other Web pages by scraping their content [3].

2.2 Natural Language Processing

Natural Language Processing (NLP) is a tract of Artificial Intelligence and Linguistics, devoted to make computers understand the statements or words written in human languages [4]. Some of the researched tasks of NLP and which are relevant to this thesis are Named Entity Recognition, Parts Of Speech Tagging, Chunking, Stop-words removal, Lemmatization etc. *Named Entity Recognition* (NER) takes certain amount of text as input and determines which items in the text relates to proper names. *Parts Of Speech Tagging* (POS Tagging) similarly takes a sentence as input and determines the part of speech for each word present in it. Another major task in NLP is Chunking. *Chunking* allows segmenting sentences into words and each word with its own label or syntactic correlated keywords like Noun Phrase and Verb Phrase (NP, VP). Stop-words removal is another task in NLP where keywords provided must be removed before processing documents. But documents or statements can use different forms for a single word, which should be detected by the model. This can be done using the task, Lemmatization. *Lemmatization* usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma* [5]. Usage of such techniques have made the algorithms easier than before.

2.3 Sentiment Analysis

According to Liu [6], sentiment analysis is the field of study that analyzes people's opinions, sentiments, evaluations, appraisal, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes. The term can be used interchangeably with Opinion Mining. In general, the sentiment for text can be categorized as positive and negative polarity. Based on its extent to analyze, sentiment analysis is divided into levels which are document level (Sentiment is calculated for whole document), Sentence level (Sentiment is calculated for a sentence), and Feature/Entity level (extract features of target and determines the feature wise polarity).

The sentiment analysis in this project takes into consideration of two levels document and sentence. When leader statements are collected, the stories under their statements are also collected, which requires document level prediction of sentiment, and as the leader's statements contain a single sentence, it requires sentence level prediction of sentiment. Analyzing sentiment can be done in two methods: lexicon-based method and machine learning based method. These approaches are summarized as follows [7],

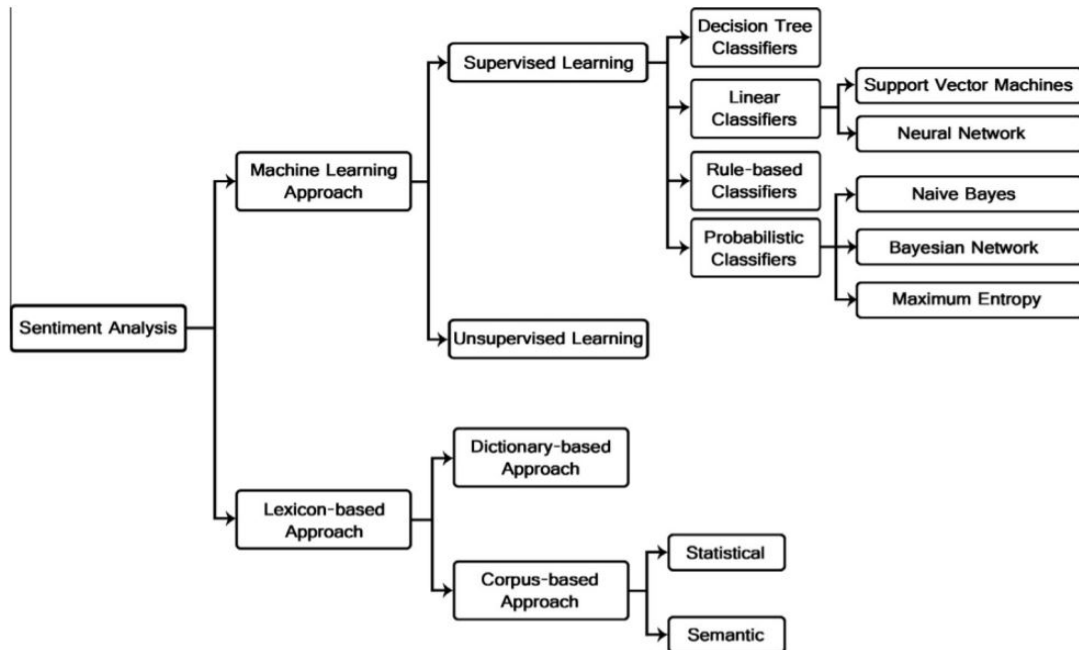


Figure 2.1 Sentiment Classification Techniques

Combining sentiment analysis with NLP techniques can provide us with many advantages. Such combinations have achieved great success in many fields like recommending, ranking of products, public opinion monitoring, emotion prediction [8]. Two main methods exist to perform sentiment analysis, which are given as follows.

Machine Learning Approach

Many methods were developed using supervised and semi-supervised learning to perform sentiment analysis.

1. Supervised Learning:

In supervised method of learning, classification models are trained using standard sentiment labelled documents. One of the earliest studies in supervised classification is performed on movie reviews. The dataset is trained on three machine learning models

which are Naïve Bayes, Maximum Entropy Classification and Support Vector Machine [9]. Based on this research, studies have been extended towards development of deep learning systems. Dong and Wei proposed adaptive recursive Neural Network (AdaRNN) for sentiment classification using Twitter data as an example in 2014 [10]. With an increase in deep learning techniques, a deep learning system named Coooolll has been developed to do sentiment analysis on Twitter data [11], with an accuracy of 87.61% to classify into categories of positive and negative.

2. Semi-supervised Learning:

Through semi-supervised learning, supervised methods and lexicon-based methods can be combined. In this process, a small amount of labelled data is used to train the classification model and large amount of unlabeled data is taken. Many researchers like Zhou and Chen have worked on it and proposed a method that uses both old and new unlabeled data. They have proposed a two-step semi supervised learning method called Fuzzy deep belief networks for sentiment classification. The model was tested on movie reviews and DVD reviews dataset and achieved an accuracy of 79.5% [12].

Lexicon-Based Approach

The lexicon-based approach is based on the concept of text sentiment polarity classification in word or sentence. Similar to machine learning approach, many have performed research on this method. As per an assumption, sentiment polarity of text is the sum of sentiment of each word or phrase [13]. The following table gives a glance about different types of lexicons that can exist while calculating the sentiment.

Table 2.1 Different types of Lexicons

Type of Lexicon	Description
Sentiment words	Positive or Negative sentiment words have a sentiment score of +1 or -1 to indicate the respective polarity.
Negation Words	Negation words are the words which reverse the polarity of sentiment.
Blind Negation Words	Blind negation words operate at a sentence level and points out the absence or presence of some sense that is not desired in a product feature.
Split Words	Split words are the words used for splitting sentences into clauses. The split words list consists of conjunctions and punctuation marks.

As each word in the sentence comes into different categories of lexicons, the sentiment can be calculated accordingly. The sentiment score of a sentence constitutes the sum of polarities of each word of the sentence [13].

With an aim to calculate sentiment labels and in a desire of getting highly accurate model, this project uses the reference of supervised method of learning among the machine learning approaches.

2.4 Emotion Analysis

Emotion recognition is an area of study that comes deep under sentiment analysis. Top levels of sentiment analysis can only restrict towards classifying the statement into positive, negative and neutral labels. But emotion proceeds to deeper level by knowing in which way the sentences or speeches were positive or negative. Human emotion can be obtained in the form of facial expressions, writings, speeches, and in actions and gestures. Study in the field of emotions has always fascinated researchers for long and is widespread into several fields.

The word “affect” is often used instead of “emotion” in the literature. This area of research is collectively known as Affective Computing, based on the work of Picard [14]. One of the most relevant works Alm et al. (2005) [15] have explored into the task of classifying the emotions in text, automatically. They have identified emotions in children’s fairy tales according to basic emotions identified by Ekman, 1992 [16]. The authors have also classified the sentences into emotional versus non-emotion, as well as into valences which are positive emotion, negative emotion, and no emotion. Using the concept of Ekman [16], all emotion classes are taken as *happy, sad, angry, disgusted, fearful, positively surprised, and negatively surprised*. Among these *happy and positively surprised* were classified into *positive emotion class*, and *sad, angry, disgusted, fearful, and negatively surprised* were classified into *negative emotion class*.

Later some researchers have come up with techniques for visualizing the acquired emotions in the text. For instance, the authors have mapped the affective content of a document to a color bar, such that change of colors in the bars represent the change of affect across the document [17]. They have constructed an affect-to-color correspondence scheme to encode 6 basic emotion categories provided by Ekman [16]. The table below shows basic emotion classes identified by different researchers from several years.

Table 2.2 Basic Emotion classes identified by researchers

Tomkins (1962)	Izard (1977)	Plutchik (1980)	Ortony (1988)	Ekman (1992)
Joy	Enjoyment	Joy	Joy	Happiness
Anguish	Sadness	Sorrow	Sadness	Sadness
Fear	Fear	Fear	Fear	Fear
Anger	Anger	Anger	Anger	Anger
Disgust	Disgust	Disgust	Disgust	Disgust
Surprise	Surprise	Surprise	Surprise	Surprise
Interest	Interest	Acceptance		
Shame	Shame	Anticipation		
	Shyness			
	Guilt			

Not only restricting to these classes of emotions, Plutchik has proposed a psycho evolutionary classification approach to classify these primary emotions anger, fear, sadness, disgust, surprise, anticipation, trust, and joy. He created a wheel of emotions to illustrate different emotions and their own categories. It shows how emotions were related to each other. This is also known as his circumplex model. Each emotion is also expressed at different intensities and can be mixed with one another to form different emotions. The Plutchik's wheel of emotions is shown as follows.

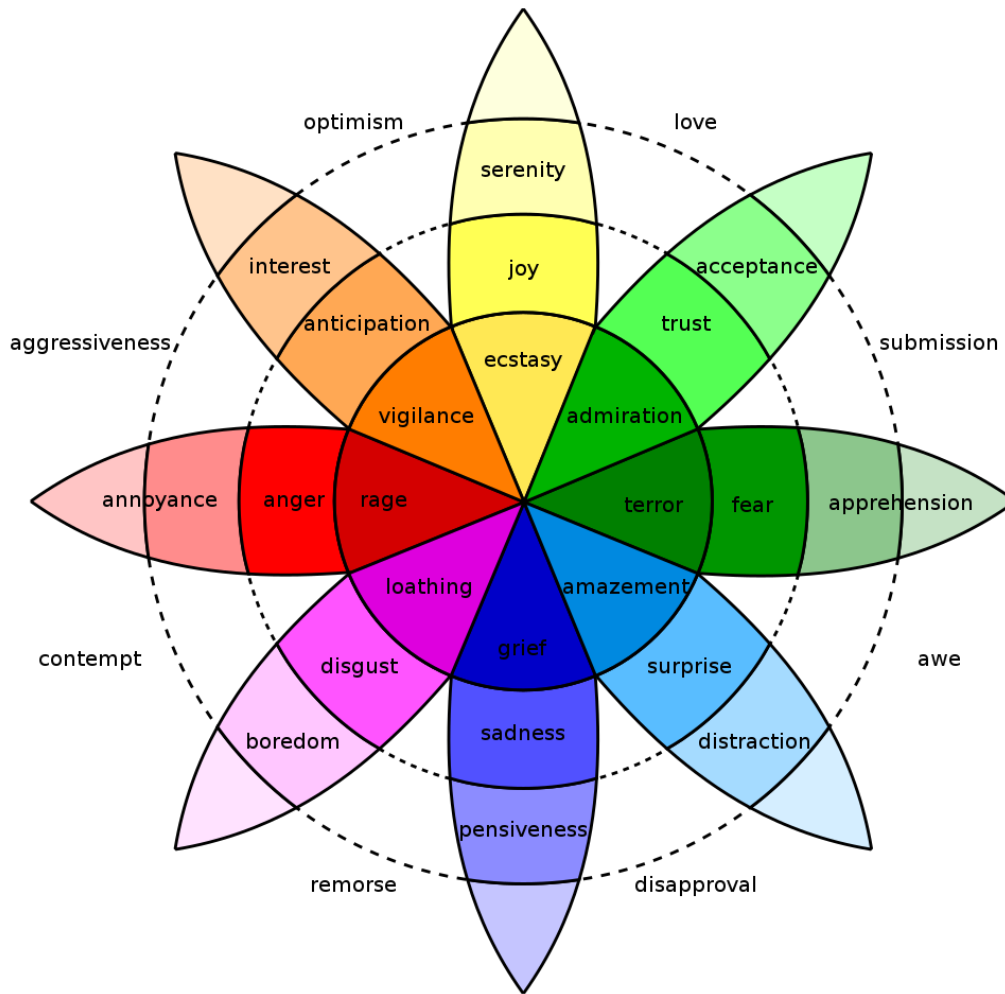


Figure 2.2 Plutchik's Wheel of Emotions

This thesis has utilized labels of emotions as per the situation of the pandemic during COVID-19. Statements which come under category “happy” consists of phrases like reduce in number of cases, measures taken, etc. Instead of the label “happy”, the label “Safety” has been replaced while classifying the emotions, as most of the leader’s statements were related to safety precautions. Detailed explanation about consideration of different emotions taken during thesis will be provided in chapters further.

CHAPTER 3

DATA COLLECTION

3.1 Data Collection

Every application has its own target and requirements to be satisfied. Different strategies should be applied to have a reliable output at the end. And such need leads to the idea of creating new datasets that can be used for multiple purposes later. Creating a large dataset is a tedious task if done manually. But methods like web scraping and web crawling can automate data collection and make it easier to create datasets for analysis. But initially, leader's statements must be stored, which we obtain from websites of several news channels. Being one of the most popular news channels in USA and Internationally too, we have chosen CNN as one of the major resources for creating the dataset. This channel not only streams the live news to people, but also stores the archives of several years. As the pandemic came into light in USA, from March 2020, we have focused on retrieval of leader's statements from that date. CNN maintains its own websites for each day containing different categories of news. As COVID-19 is the most popular update in recent times, a unique portal is being developed by them to have live updates of Coronavirus across the world.

For extracting such huge amounts of data, we chose web scraping as the best option. Through web scraping, we can scrape data across several websites using their HTML attributes. We have used programming language R to perform the process of scraping. There are several libraries for web scraping in other languages like Python, but we chose R due to several advantages which will be discussed in the next section. The whole process of web scraping, a glance about

choosing correct libraries and methodology implemented from them will be explained in detail in next section 3.2.

As we have seen in the previous sections about analyzing the relationship between leader's statements and spread of COVID-19, the dataset created using leader's statements is not sufficient. The COVID-19 data should be extracted to know the spread of number of cases across the region where the leaders have stated their orders. There are many sources that provide the spread of COVID-19 in spatial forms. But most of them extract their data from COVID-19 dataset created by CSE department of John Hopkins University [44]. They have a daily update of COVID-19 data involving many fields like County Name, Province_State, Confirmed cases and deaths, etc. They maintain and update data obtained from whole world, and in addition, they maintain separate folders for USA, which is being used in this project, for the final analysis.

3.2 Web Scraping

3.2.1 Overview of Web Scraping

Web scraping is a process that involves extraction of information from various websites. It is also known as web data extraction, web data scraping, web harvesting or screen scraping [18]. In general, websites contain unstructured data, where web scraping can be used to fetch and extract the information, transform it into an understandable format like csv and finally store it into an external database. This technique can be performed manually by applying traditional methods of copy-paste, but in most of the cases, it is automatically performed to achieve efficiency. The whole process of web scraping can be split into different phases and explained in brief, as follows.

Phase-1: Fetch data

Firstly, in this step, desired websites must be selected where the data is accessed from. Then fetching can be done using HTTP protocol, an Internet protocol used to send and receive requests from a web server. In this phase, libraries such as wget, curl and navigate can be used to send a HTTP GET request to the URL of a website (desired location), from which HTML document is sent back as a response [19].

Phase-2: Extracting Information

After fetching the desired HTML documents, the next step would be extracting our required information from the website. This can be performed using several techniques such as HTML Parsing, DOM Parsing, XPath, and Text pattern matching, which will be discussed in detail in the next section.

Phase-3: Data Transformation

Once the required information is extracted from the desired locations (URL), the data will be in unstructured format. It can then be transformed into structured format such as CSV, spreadsheet, or pdf, either for presentation or storage.

The phases described above in the process of web scraping can be summarized in Figure 3.1, as follows.

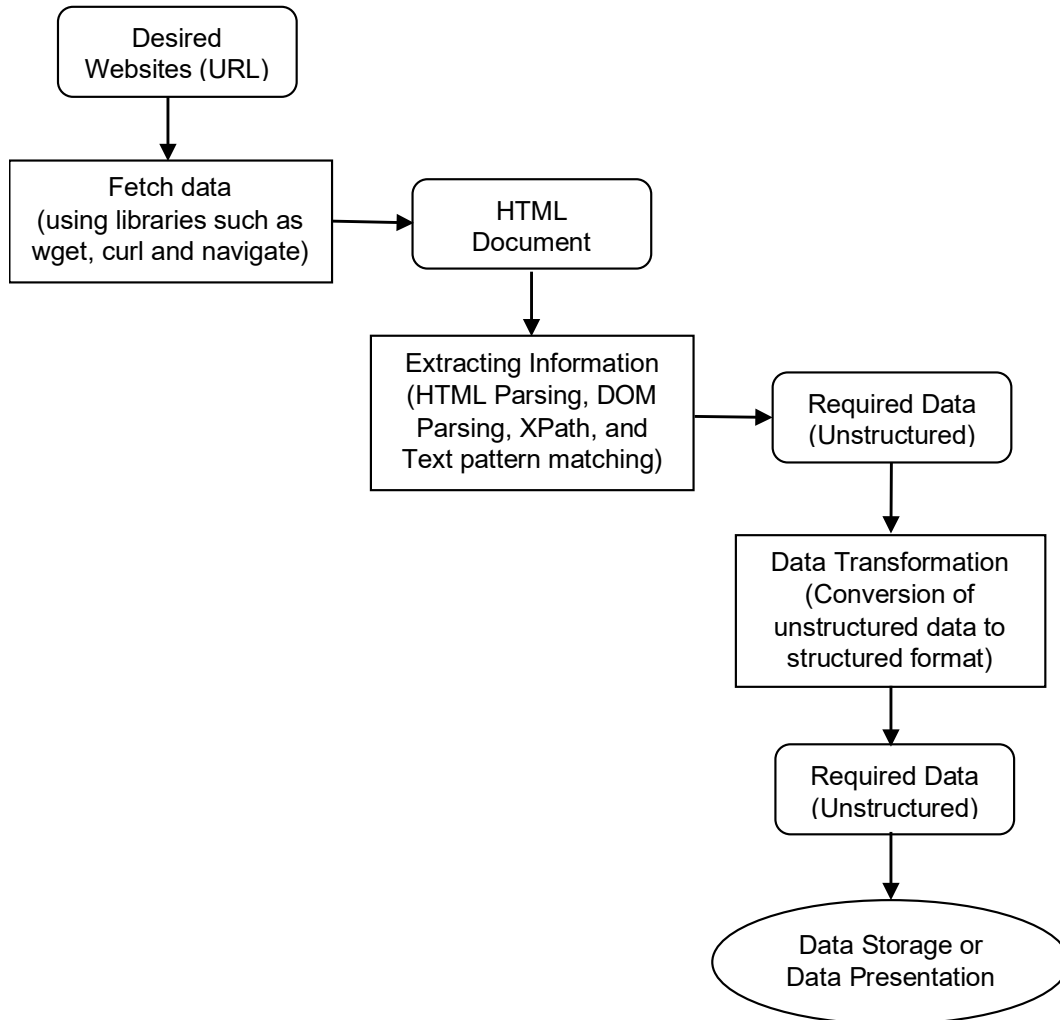


Figure 3.1 Overview of Web Scraping Process

3.2.2 Techniques for Web Scraping

To implement a web scraper, there are several approaches among which the best one can be selected based on a programmer's requirement. From the above, it can be observed that these techniques are used during the extraction phase, and they are mainly divided into two categories Manual web scraping and Automated web scraping. This section discusses about some of the major techniques in each category and gives a scope for selecting the suitable technique among them.

3.2.2.1 Manual Web Scraping

Traditional Copy Pasting

Traditional copy-pasting is a manual approach where data that has to be extracted from a website is manually copied as a group and pasted into a document. Then the required data would be collected from the group and arranged into a structured form. Occasionally, for less amount of data, this can be the best technique. But in the process of creation of large dataset, this technique could be tiresome and error prone as it involves lot of manual work [18].

3.2.2.2 Automated Web Scraping

HTML Parsing

In general, parsing is the process of analyzing a string of symbols either in natural language or in computer languages or data structures conforming to the rules of grammar. The result of parsing a document is usually a tree of nodes that represent the structure of it. In HTML Parsing, after the HTML document is fetched, a tree of nodes is formed during parsing, from which the relevant information like headings in the page, title of the page, paragraphs in the page can be extracted by detecting the html nodes. Some semi-structured data query languages, such as XQuery and the HTQL, can be used to parse HTML pages and to retrieve and transform page content [18].

DOM Parsing

The DOM is a common platform for managing document structures. It works on XML documents by creating an interface that would be helpful in providing access to structure and contents of these documents. Like HTML parsers, when an XML document is fetched and applied with a DOM parser, a tree structure containing all elements of the document is formed. With the

help of DOM, the contents and structure of the document can be examined and used for extraction [21].

XPath

XPath stands for XML Path Language [21]. It is a technology that takes path expressions into account, to access nodes in a document. It can be used on XML documents to access different elements in their structure and content. XPath can also be used to navigate HTML documents, as they share similar structure to XML. It allows us to write expressions which can directly access the HTML elements instead of examining the whole tree. In general, after DOM parsing, XPath can be used as web scraper to scrape the website's data. It is not a language but comes in a form of expressions, that requires its own syntax.

CSS Selectors

Another popular technique in web scraping for extracting data from HTML documents is the application of CSS selectors, where CSS (Cascading Style Sheets) is a language used for styling HTML documents. Mainly it describes the rendering of structured documents such as HTML and XML. Based on different CSS properties, different selectors such as type, attribute, ID, etc. are used to represent the structure and content of website. These can be used to match and extract HTML elements [21].

Text Pattern Matching

Text pattern matching is a matching technique where regular expressions are used to match HTML tags and extract data from HTML documents [22]. Regular expressions, in general, are sequence of characters that give rise to a search pattern. As HTML is virtually composed of many strings, regular expressions can come into action here, by matching different strings. But regular

expressions might not be the first choice in parsing HTML, as there is a chance of encountering mistakes such as missing tags.

Among popular techniques discussed above, based on the flexibility, and knowing about difficulties of their usage, it can be observed that HTML parsing is the most convenient technique. As there are numerous libraries and inbuilt tools available for performing this technique, it had been chosen for this project to start with scraping the news stories. The following section gives a detailed picture about different libraries containing HTML parser in python and R.

3.2.3 Libraries for Web Scraping

There are several libraries for web scraping. But being most popular languages in data science, python and R have bagged wide variety of built-in libraries that made the process easier for extracting relevant information from websites.

3.2.3.1 Libraries in Python

Requests

Requests is the most basic python library for web scraping. Through this, HTML requests can be made to a website server to retrieve data on the web page. This library is used in fetch phase of web scraping process, as discussed above in section-3.2.1, overview of web scraping. This python library gives high flexibility to users by providing various types of HTTP requests such as GET, POST, etc [26]. As this is basic library that can be used only for fetching web pages, it cannot be individually used to scrape data. It must be combined with other libraries to obtain a reliable output.

LXML

As we have seen above in requests, it had a limitation that it cannot be used as a parser. LXML, a high performance, fast, HTML and XML parsing python library [27] can be used along with it to start parsing and extracting data from webpages. This library is python in-built and uses HTML parsing technique, discussed above in automated web scraping, section 3.2.2.2. It is faster than most of the parsers and uses python API, making the functionality easier. But it does not work well with poorly designed HTML documents. This makes LXML less flexible for usage when compared to other libraries, which will be discussed later in this section.

Beautiful Soup

Beautiful soup is the most widely used python library for web scraping [21,27]. For parsing HTML and XML documents, it creates a parse tree, and this also uses HTML parsing technique discussed above in section 3.2.2.2. Beautiful soup has gained its popularity due to its ease of usage, but it is slower when compared to LXML, discussed above. One of the major advantages is that it is suitable to any type of website. And this can be used along with requests to perform fetch and extraction phases successfully. To speed up the process, it can be combined with LXML parser.

Selenium

The libraries mentioned above have a limitation that they cannot work with websites designed with JavaScript. This makes them difficult to work while scraping dynamic webpages. Working with dynamic webpages is one of the biggest challenges in area of web scraping which can be faced with effective usage of libraries. Detailed explanation about dynamic webpages and overcoming them will be discussed in later section of Challenges. But selenium is one of the python libraries that can overcome this difficulty.

Initially, it was made for automated testing of web applications, but later its ability to run JavaScript has extended its scope for using it as web scraping library. It consists of a web driver for rendering web pages, that can be used to perform clicks, highlights, scrolls, etc. automatically [27]. These features made it flexible for scraping dynamically populated web pages. But one of the major limitations of this library is that it loads and runs JavaScript for every page, which makes the processes slower and not suitable for large scale projects.

Scrapy

Scrapy is one of the most popular libraries in python. It is not just a library, instead it can be considered as an efficient web scraping framework, which is a complete package consisting of functionalities suitable for fetching and extraction phases, discussed above in section 3.2.1. It can be used to crawl websites as it provides spider bots and gives users a facility to create their own spider bot [28]. Scrapy can work in an asynchronous manner indicating that it can make multiple HTTP requests simultaneously, which speeds up the process and increases the efficiency. One of the limitations of scrapy is that it cannot handle JavaScript like selenium drivers. To be able to automate the process with scrapy, it must be combined again with other libraries like Splash, to extract data from dynamic websites.

As we can observe from the libraries mentioned above, each has its own advantages. But none of the libraries mentioned above can be used individually for complete process. At least two or three of them must be combined to obtain the complete functionality of a web scraper. And if the requirement is not satisfied, a greater number of libraries must be explored and used in python, to acquire a reliable output of extraction. This limitation gave us a scope for designing a scraper using the programming language R.

3.2.3.2 Libraries in R

There are numerous libraries in R, that were designed using statistical methods. Some of the major libraries in R have been studied for this project and proved to be sufficient for designing an efficient web scraper embedding all the functionalities.

Rvest

Rvest is popular package that is used for scraping data from html web pages [23]. It is inspired by libraries like beautiful soup in python. It is used along with the package magrittr in R, to have the piping functionality. Working with rvest is easier when compared to other libraries. This can be shown through an example.

Scraping data from “The Lego movie” from IMDB is used as an example here. While using rvest, first step would be downloading and parsing the file with the help of method `html()`.

```
library(rvest)
```

```
lego_movie <- html("http://www.imdb.com/title/tt1490017/")
```

Then pipelining can be done to extract data from css selectors that exist in the web page. Methods such as `html_node()` can be used to find all the nodes that match with the selector and then contents from the location can be extracted using method `html_text()`. The code snippet below shows the extraction of IMDB rating of the movie using these methods.

```
rating <- lego_movie %>%
```

```
html_nodes("strong span") %>%
```

```
html_text() %>%
```

```
as.numeric()
```

```
rating
```

```
#> [1] 7.7
```

From the code snippet, it can be observed that the rating is obtained as 7.7 for “The Lego movie” in IMDB website.

As this library contains fetching and extraction phases of a web scraping process, discussed above in section 3.2.1, most of the extraction can be completed using this. But the goal to create a huge dataset requires the process to be automated. And, as rvest does not provide this facility, it must be combined with a driver that helps in automating web scraping. R does contain such powerful library named as “RSelenium”, which will be elaborated in the following section.

RSelenium

RSelenium is a combination of R and selenium, where it establishes a connection to Selenium Server/ Remote Selenium Server from within R [24]. Selenium as discussed above in section 3.2.3.1, works with JavaScript, and focuses on automating web browsers. Integrating the Rvest library along with RSelenium gives a facility for the users to access a web driver in a web scraper. Usage of this library involves few steps that would be discussed in detail in later sections of methodology implemented.

Stringr

Stringr is another popular package present in R, which contain wide range of set of functions that can be used for data cleaning or for string manipulations [29]. As there is text cleaning process in the final stage of web scraping process, that is Data Transformation phase, this package can be of great use for dealing such operations.

From the packages discussed above, we can observe that packages in R have more features when compared to that present in python. Instead of using several libraries to achieve functionality, R provides a facility for users to complete the operations with minimal usage of packages and methods. These are some of the reasons with which we can decide the languages and packages to be used. Based on all the advantages mentioned above, R had been selected and the packages rvest and RSelenium were used to achieve web scraper for this project. Their detailed integration and working will be shown in the methodology in later sections.

3.2.4 Challenges while designing web scrapers

Websites are designed in different ways and the goal of creation would include many factors such as user interactive, way of presenting large datasets, including latest features for styling, etc. And as the number of features increase, the complexity of the website increases, and this in turn challenges the web scrapers in many ways. But our goal is to design a web scraper that is suitable to any type of website. This section describes about some of the major challenges faced while designing an efficient web scraper, and brief description of solutions that can be used to overcome these challenges. Detailed description of solutions using code snippets will be given in the later section consisting of implementation.

Infinite Scrolling Webpages

Infinite scrolling is a technique used in websites where users scroll without ever reaching the end of the page. In such scenarios, new content gets loaded as user scrolls down. Several websites like Facebook, Twitter, etc. have included infinite scrolling as their main feature. The inclusion of this feature totally depends on the nature of the website and the requirement on which

it is designed. For example, websites such as blogs, or others that give information about an organization are built towards a motive to provide information to people. Such websites would not require this feature. But the websites that intend to be browsed and explored by users tries to engage users with their information. This kind of websites embed this feature into the program, so that user invests time in the page. Our project mainly uses CNN website for scraping data. For news sites like this, the purpose is to display the latest and most relevant stories first, here infinite scrolling can ensure users to have a look at latest stories, instead of all headlines and stories at once. One of the solutions can be to scroll the page multiple times, but that involves re-loading each time, the page gets scrolled. But this can be resolved using RSelenium driver which will be clearly shown in later sections of implementation.

Data Repository

In general, data extraction at large scale can give rise to huge amount of information. If data repositories are not designed properly, filtering, and organizing this kind of data could become cumbersome and time-consuming task.

Dynamic Webpages

Websites are mainly classified into two different types: static and dynamic. Static webpages consist of fixed number of pages and they have a specific layout. The content is static and does not change in response to user actions and are intended to be purely informational. They can be created using HTML and CSS. Whereas, dynamic web pages are a combination of server-side and client-side scripting, and they rely mainly on JavaScript or PHP. Here users can access the website and the browsers change based on their actions. As each user can opt different options, the webpages will have more possible outputs making difficult for web scraping.

Frequent Structural Changes

Another challenge is that websites undergo structural changes quite often, to either add novel features or improve user experience. As web scrapers are designed based on the code elements present at the time of setup, any changes that made in between can prevent the web scraper from extracting information.

Data Quality Assurance

Sometimes, extracted data can have missing information and may not match with a pre-defined template. In such cases, data accuracy could be reduced. To resolve such difficulties, quality assurance test should run on the data and each field has to be validated. These tests can be done automatically using selenium drivers, but in some cases, manual testing could provide reliable results.

Denial of Service

Denial of Service is a common method where web scraping can be interrupted from accessing data from the website. It generally happens when a website detects a greater number of GET requests from the same IP address [20]. If users get a warning, the scraping has to be stopped completely. Disregarding such warnings can lead to a ban or restriction of IP address to break the web scraping process.

As we can see, the above-mentioned challenges are regularly faced during the process of web scraping. And having a target to scrape news sites like CNN, some of the important challenges like infinite scrolling webpages, data quality assurance, frequent structural changes have been faced in this project. But they have been resolved again with effective solutions while

programming the web scraper. The next section includes the methodology and way of implementation of the web scraper by overcoming the above- mentioned challenges.

3.2.5 Methodology and Implementation of Web Scraper

Websites are designed in different ways and the goal of creation would include many factors such as user interactive, way of presenting large datasets, including latest features for styling, etc. And as the number of features increase, the complexity of the website increases, and this in turn challenges the web scrapers in many ways. But our goal is to design a web scraper that is suitable to any type of website. This section describes about some of the major challenges faced while designing an efficient web scraper, and brief description of solutions that can be used to overcome these challenges. Detailed description of solutions using code snippets will be given in the later section consisting of implementation.

Selecting Packages

First step of the web scraping process includes selection of relevant packages. As we have discussed above in section 3.2.3 about libraries, we have chosen the programming language to be R and suitable packages to be “rvest”, “RSelenium”. And the package “stringr” can be used to manipulate and clean data. To use the packages into scraper program, they should be installed and downloaded before moving forward. This can be done with following commands,

```
install.packages("rvest")
```

```
install.packages("RSelenium")
```

```
install.packages("stringr")
```

To add them into our scraper program, we can use the following commands,

```
library(RSelenium)
```

```
library(rvest)
```

```
library(stringr)
```

After adding the packages, they are ready to be used and all the features, methods under them can be used.

Connecting to Selenium Driver

In R, RSelenium is used to establish a connection between Selenium Server and R. This allows us to automate the web scraping process. But firstly, to drive a browser on same machine that RSelenium is running on, we will need to have a Selenium server running on that machine. To run a selenium driver through R, the function “rsDriver” can be used, which manages the binaries required for running a Selenium server [24].

```
rD <- rsDriver(browser=c("firefox"))
```

The above command indicates that selenium driver and browser “Mozilla Firefox” has started. The value of the driver would be a list containing server and client. Now the client, which is indirectly the browser can be stored in a variable and used further.

```
driver <- rD$client
```

To connect and disconnect from the server, the methods open and close can be used, as shown below.

```
driver$open()
```

```
driver$close()
```

Navigating URL's

Recollecting the phases of web scraping process from section 3.2.1, the first phase involves finding a URL and fetching data from it. This paper aims to collect news stories and news headlines during COVID-19, i.e., starting from March 2020. As it focuses mainly on CNN news sites, the relevant URL's can be found, which contains international news related to COVID-19. The URL can be shown as follows,

<https://www.cnn.com/world/live-news/coronavirus-pandemic-intl-04-17-20/index.html>

From the above URL, we have observed that the modulating part is the date of the pandemic. Each day in period of pandemic has its own webpage, which made the process of web scraping easier. As we know the start date of pandemic, we can set end date and create a sequence of dates. This can be shown as follows,

```
dates=format(seq(as.Date("2020-03-25"), as.Date("2020-12-10"),  
by="days"), format="%m-%d-%y")
```

As we have a set of dates now, we can iterate the process to all webpages. We can navigate through the webpages using the method “navigate” by taking URL as argument. The commands can be shown as follows,

```
url=paste(c("https://www.cnn.com/world/live-news/coronavirus-pandemic-  
",dates[d],"-intl/index.html"), collapse="")  
  
driver$navigate(url)
```

The navigate function opens the webpages on our selected browser and waits for the next action input from the user or from the program.

Data Extraction from Webpages

As the relevant URL's have been collected, data can be extracted from these sources. Here, the HTML parsers come into play. The method that acts as parser is `getPageSource()`, which gives the complete set of html nodes present in the webpage. The command to request from client is shown as follows,

```
page_source<-driver$getPageSource()
```

Now from the obtained page source, we can acquire the desired content such as dates, headlines, and stories under a headline. This can be done using the function "read_html". The whole command is shown as follows,

```
headlines<-read_html(page_source[[1]])%>%  
html_nodes("h2") %>% html_text()  
stories<-read_html(page_source[[1]])%>%  
html_nodes("article") %>% html_text()  
headlines<-read_html(page_source[[1]])%>%  
html_nodes(".date") %>% html_text()
```

From the commands above, we can observe that headlines, stories and their corresponding dates have been obtained using the html nodes, h2, article and .date respectively.

This extraction can be iterated to all the pages using a for loop.

Handling Infinite Scrolling Webpages

The extraction of data can be done successfully. But the CNN news site that is being used in this scenario is an infinite scrolling webpage, as discussed above in section 3.2.4. This should be handled in order to get the complete information existing in the page.

As we know, for infinite scrolling pages, from section 3.2.4, more information is loaded when page is scrolled down. One way to handle it is by scrolling down with the help of method “executeScript()”, calculating the scroll height and finally updating it in a variable. Then the newly updated height can be compared to the initial height and start scrolling operation to that extent. This process can be stored in a function or it can be iterated using loops, and the complete loop has been shown as a code snippet,

```
last_height = 0

repeat {

driver$executeScript ("window.scrollTo(0,document.body.scrollHeight);")

Sys.sleep(10)

updated_height=driver$executeScript ("return
document.body.scrollHeight")

if(unlist(last_height) == unlist(updated_height)) {

break

} else {

last_height = updated_height

}
```

In the code snippet above, “sleep” function is used, which gives time for a webpage to be loaded completely in the new scroll area.

The above-mentioned method can be embedded into any project in common, where the scenario of infinite scrolling webpages exists.

Data Transformation

From the overview of web scraping, discussed above in section 2, the last phase of the process is “Data Transformation”, where the extracted data will be transformed into desired format like csv or spreadsheet.

In this project, a csv is desired. The method “write.csv()” can be used to complete the operation and store the data into a csv file.

Closing the Client

Finally, after completion of scrapping, we can close the driver using “close()” method, which closes the connection established between R, selenium, and web browser.

```
driver$close()
```

Result of Web Scraper

From the methodology discussed above, we can observe that it includes all phases of web scraping process. But the result desired in web scraping should be transformed and structured data. After transforming into csv file, the output of top 5 rows of news stories and headlines collected on August 13th, 2020, can be shown in Table 3.1, as follows.

Table 3.1 Output showing top 5 news stories and headlines on August 13th, 2020

corona_dates	Headlines	Stories
08-13-20	Through violence, the pandemic and the fight...	Chicago's West Englewood neighborhood...
08-13-20	Venezuela reports highest daily number of new...	Venezuela reported 1,281 new Covid-19 cases...
08-13-20	Japan reports 1,177 new coronavirus cases	Japan reported 1,177 new cases of Covid-19 and...
08-13-20	Trump coronavirus adviser claims “young...	Dr. Scott Atlas listens as President Trump speaks...
08-13-20	New Zealand reports 12 new coronavirus cases...	New Zealand reported 12 new coronavirus cases...

From the above table, we can observe that the news acquired by web scraping contains both International and National (USA) updates. As the thesis project focuses only on National headlines and leaders across 50 states of the USA, the headlines and stories obtained here should be filtered as per the region. And the leader’s stating exists in the headline but not separately. These concerns require us to add attributes to the dataset such as Region of statement, Leader stated, etc. As the dataset is quite large, manual filtering and addition of attributes could be tiresome, which lead us towards using NLTK toolkit to prepare the complete dataset. Brief introduction about NLP and detailed description on usage of its techniques to prepare dataset is given in the following chapter.

CHAPTER 4

DATASET PREPARATION

Natural language processing (NLP) comes under the broad spectrum of Artificial Intelligence (AI). This thesis uses concepts of NLP to complete the intermediate procedures required to create a complete dataset. According to the author Chowdhury [30], “*Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things*”.

There are several techniques in NLP which were combined into Natural Language ToolKit (NLTK) that can be applied into programs using Python. The later sections of this chapter consists of descriptions about several techniques relevant to this thesis, and their usage in the project at each level.

4.1 NLP Techniques for Text

NLP is a wide topic and contains huge number of techniques for each procedure when dealing with text. But there are few techniques that has to be followed before processing the text. These are listed as follows.

1. *Tokenization*

Tokenization is a process of dividing the set of strings into sub-components. This can be performed at both sentence and word level. Sentence tokenization when performed on a sentence divides the set of strings based on punctuation marks. Whereas tokenization at word level splits the whole sentence into list of words, that can be processed further as per requirements. Before proceeding to any operation, tokenization is the basic procedure that has

to be done for a document or simple sentence in order to get the words or sentences. The process of tokenization can be shown through an example.

As NLTK already consists of several libraries, it constitutes methods for tokenization too. Initially to use NLTK into an application, the library has to be imported using python. The command can be given as follows,

```
import nltk
```

Along with this NLTK has inbuilt tokenizer which is known as Punkt. This tokenizer divides sentences using an inbuilt unsupervised algorithm. Other tokenizers exist but Punkt is the most popularly used sentence tokenizer and has been used in this thesis. It can be downloaded once after we import the NLTK package. The command for the download is given as follows,

```
nltk.download('punkt')
```

After obtaining the required packages, the tokenization can be started using the methods “sent_tokenize” and “word_tokenize” respectively for tokenizing into sentences and words. This can be shown through an example as follows.

Let us take a piece of text on few statements given by a leader during the COVID-19 pandemic.

“The state of Texas is strong, our people, resilient. As we have seen in years past, when tested by fire, flood, or hurricane, Texans respond with resilience and calm resolve. And just as we overcame those challenges, we will overcome this one. When neighbors help neighbors, our resilience is redoubled. I have no doubt that Texans will continue to work together in that spirit over the coming days and weeks. And we are here to help.”

The code snippet to perform tokenization at sentence level can be given as follows,

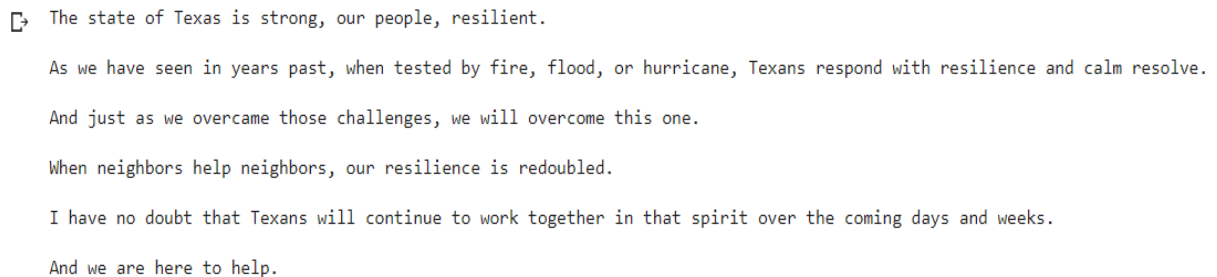
```
leader_statement = "The state of Texas is strong, our people, resilient. As we have seen in years past, when tested by fire, flood, or hurricane, Texans respond with resilience and calm resolve. And just as we overcame those challenges, we will overcome this one. When neighbors help neighbors, our resilience is redoubled. I have no doubt that Texans will continue to work together in that spirit over the coming days and weeks. And we are here to help."

tokenized_sentences = nltk.sent_tokenize(leader_statement)

for sentence in tokenized_sentences:

    print(sentence)

    print()
```



```
☐ The state of Texas is strong, our people, resilient.
As we have seen in years past, when tested by fire, flood, or hurricane, Texans respond with resilience and calm resolve.
And just as we overcame those challenges, we will overcome this one.
When neighbors help neighbors, our resilience is redoubled.
I have no doubt that Texans will continue to work together in that spirit over the coming days and weeks.
And we are here to help.
```

Figure 4.1 Output when Punkt tokenizer is applied to text at sentence level

The figure above shows the output obtained when the Punkt tokenizer is applied to the leader's statements at sentence level.

The tokenization can be performed at the level of words where the method "*word_tokenize*" can be applied. It can be applied to same group of statements and is shown below. The code snippet can be given as follows,

```
leader_statement = "The state of Texas is strong, our people, resilient. As we have seen in years past, when tested by fire, flood, or hurricane, Texans respond with resilience and calm resolve. And just as we overcame those challenges, we will overcome this one. When neighbors help neighbors, our resilience is redoubled. I have no doubt that Texans will continue to work together in that spirit over the coming days and weeks. And we are here to help."
```

```
for sentence in tokenized_sentences:  
    words = nltk.word_tokenize(leader_statement)  
    print(words)
```

```
['The', 'state', 'of', 'Texas', 'is', 'strong', ',', 'our', 'people', ',', 'resilient', '.']  
['As', 'we', 'have', 'seen', 'in', 'years', 'past', ',', 'when', 'tested', 'by', 'fire', ',',  
['And', 'just', 'as', 'we', 'overcame', 'those', 'challenges', ',', 'we', 'will', 'overcome',  
['When', 'neighbors', 'help', 'neighbors', ',', 'our', 'resilience', 'is', 'redoubled', '.']  
['I', 'have', 'no', 'doubt', 'that', 'Texans', 'will', 'continue', 'to', 'work', 'together',  
['And', 'we', 'are', 'here', 'to', 'help', '.']
```

Figure 4.2 Output when Punkt tokenizer is applied to text at word level

The figure above shows the output obtained when the Punkt tokenizer is applied to the leader's statements at level of words. Words from each sentence are formed as different lists. In this thesis, tokenizer at word level is used before trying different operations on text, that we will discuss in later sections of this chapter.

2. Stemming and Lemmatization

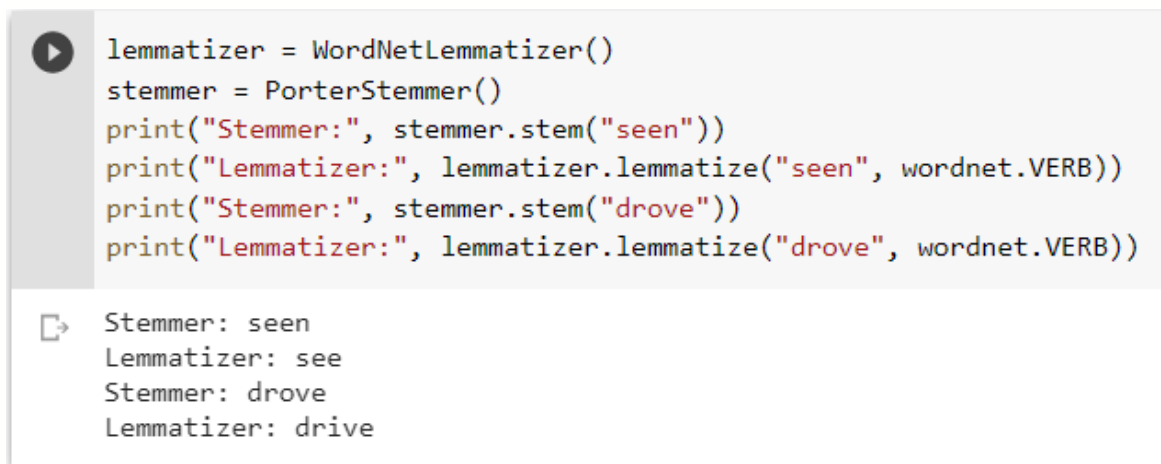
Different forms can exist for a word such as organize, organizes and organizing based on the tenses. Through stemming and lemmatization, such formations can be identified, and all the related words can be compressed to a base word. Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of

the time, and often includes the removal of derivational affixes. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma [5]. The difference between stemmer and lemmatizer can be shown through an instance.

Firstly, the stemmer and lemmatizer can be imported from NLTK Library. Along with these the wordnet library [30] can also be downloaded, which is a large word database of English nouns, Adjectives, Adverbs and Verbs. The commands can be given as follows,

```
nltk.download("wordnet")  
  
from nltk.stem import PorterStemmer, WordNetLemmatizer  
  
from nltk.corpus import wordnet
```

To analyze the difference, two words “seen” and “drove” have been taken as instances.



```
▶ lemmatizer = WordNetLemmatizer()  
  stemmer = PorterStemmer()  
  print("Stemmer:", stemmer.stem("seen"))  
  print("Lemmatizer:", lemmatizer.lemmatize("seen", wordnet.VERB))  
  print("Stemmer:", stemmer.stem("drove"))  
  print("Lemmatizer:", lemmatizer.lemmatize("drove", wordnet.VERB))  
  
↳ Stemmer: seen  
  Lemmatizer: see  
  Stemmer: drove  
  Lemmatizer: drive
```

Figure 4.3 Output showing differences between Stemmer and Lemmatizer

From the figure above, we can observe that the stemmer gives the same output as the input text, whereas lemmatizer gives an output of the base words by identifying the parts of speech.

To process the text and acquire uniformity among all the words in a sentence, a lemmatizer or stemmer has to be used after tokenization. This project uses lemmatizer after tokenization as it has more benefits when compared to stemmer, as discussed above.

3. Stop Word Removal

Stop words can be considered as the most frequently used words in grammar. Words like “a”, “an”, “the”, etc. are used for forming a sentence. But while pre-processing, these do not provide any importance or weight to the sentence. Instead, they can bring noise to the analysis. Therefore, they must be removed before processing the text. The list of stop words can change based on the requirements of an application. Fortunately, to reduce the manual work of listing out the stop words, NLTK constitutes a list of stop words that can be downloaded and used to remove from the sentences.

4. Bag of Words

To process text, machine learning algorithms cannot work with raw text like English words directly. The text should be pre-processed and converted into vectors of numbers. This is called feature extraction. Bag of words works as a feature extraction technique where each word will have a number listed to it, and the count of each word is stored in the vector, which is also known as Bag-of-Words (BoW) model. From NLTK using python, to create a BoW model, *CountVectorizer* class can be used which counts the occurrences of each word.

5. *N*-grams

The words in the text can also be grouped to form n-grams. Unigram is one word, bigram is a combination of two words, trigram is a sequence of three words, etc. With n-grams, instead of storing count of each word, frequencies of each word out of all words can be calculated as a score for each word. With this we can obtain the most frequent words across the text. But sometimes most frequently used words cannot give much information. To correct this, TF-IDF (Term Frequency- Inverse Document Frequency) can be used to evaluate how important the most frequent word is to the text.

TF-IDF increases proportionally to the frequency of a word appeared in a document, but it is inversely offset by the number of documents in the corpus that contain the word.

The formula for TF can be given as,

$$TF \text{ (Term Frequency for word)} = \frac{\text{Frequency of word in a text document}}{\text{Total number of words in a text document}}$$

The formula for IDF can be given as,

$$IDF \text{ (Inverse Document Frequency for word)} = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents with word in it}}\right)$$

To find the final TF-IDF score for a word, it can be calculated as,

$$TF\text{-}IDF \text{ score for word} = TF \text{ (for word)} * IDF \text{ (for word)}$$

In python, to find TF-IDF score for a word, we can use the *TF-IDF vectorizer* instead of *count vectorizer*.

As the basic NLP techniques for any type of text are being discussed in this section, the later sections of this chapter focuses on operations on text using libraries and methods present in NLTK Toolkit. This chapter also shows preparation of leader's statements dataset obtained in previous chapter by adding more attributes as per requirements.

4.2 NLTK Toolkit

NLTK Toolkit was originally created with aim to provide practical knowledge of NLP techniques. As python is the most popular language in the current generation, the NLTK Toolkit involved in python can be used in any type of text processing task present in an algorithm. The below figure lists out the text processing tasks, corresponding NLTK modules that can be used for the task, and the functionality of each module in brief [32].

Language processing task	NLTK modules	Functionality
Accessing corpora	corpus	standardized interfaces to corpora and lexicons
String processing	tokenize, stem	tokenizers, sentence tokenizers, stemmers
Collocation discovery	collocations	t-test, chi-squared, point-wise mutual information
Part-of-speech tagging	tag	n-gram, backoff, Brill, HMM, TnT
Machine learning	classify, cluster, tbl	decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	chunk	regular expression, n-gram, named-entity
Parsing	parse, ccg	chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	sem, inference	lambda calculus, first-order logic, model checking
Evaluation metrics	metrics	precision, recall, agreement coefficients
Probability and estimation	probability	frequency distributions, smoothed probability distributions
Applications	app, chat	graphical concordancer, parsers, WordNet browser, chatbots
Linguistic fieldwork	toolbox	manipulate data in SIL Toolbox format

Figure 4.4 Language processing tasks and corresponding NLTK modules with examples of functionality

Among the above listed Language processing tasks, this thesis uses tasks such as Accessing Corpora, String Processing, Part-of-speech tagging, Machine learning, Chunking, Parsing, which will be discussed in detail along with the steps to prepare the dataset for further operations.

4.3 Dataset Preparation using NLTK Toolkit

In general, to analyze the leader's statement, only the statement will not be sufficient. The other attributes to know are the statement spoken region, statement spoken date or time, the leader's name, and their role in federal or state government. There can be more attributes required if needed, but these are the basic ones among them which are necessary for any type of analysis dealing with leader's statements.

As we can observe from the output obtained after web scraping at the end of Chapter-3, the raw data does consist attributes statement spoken date, but not the other attributes such as statement spoken region, the leader's name and their role as discussed above. These attributes can be created and noted by finding the names and regions in the statements and stories obtained. But that could be quite tedious, if done manually. As NLTK Toolkit covers most of the NLP Text processing techniques as its methods, they can be used here to find such attributes.

This section discusses about different NLTK modules that can be used to prepare the require dataset and finally shows the dataset that can be used further for sentiment and emotion analysis.

4.3.1 Required modules from NLTK Toolkit

Before starting text processing, some of the libraries from NLTK should be downloaded, which can be used in the algorithm. The required modules are listed as follows.

1. Punkt

Punkt is a sentence tokenizer which splits the list of sentences into lists of words. It uses an unsupervised algorithm to tokenize the text. It can be downloaded after importing the

NLTK toolkit into the algorithm. The commands can be given as follows,

```
nltk.download('punkt')
```

2. Averaged Perceptron Tagger

With the help of Average Perceptron Tagger, the parts of speech tagging can be implemented. It uses average, structured perceptron algorithm. In general, this algorithm is split into two components, *structured prediction*, and *perceptron*. Structured Prediction in any algorithm relates to breaking down large structure into sub problems and then start prediction for each problem. And then the results will be used in later stages. This algorithm is used for POS tagging, where each word is tokenized and then each token is tagged with its own parts of speech.

The perceptron algorithm stores a linear vector of components in pairs of features and their weights. For scoring each observation, dot product is calculated between feature vectors and its corresponding perceptron. By combining both the algorithms, the final ones would be structured perceptron, where the score from each term is accumulated to get the score of output. This applies here in Averaged perceptron tagger and is used to acquire parts of speech for the words. This tagger can be downloaded in the same manner as that of 'punkt'. The command can be given as follows,

```
nltk.download('averaged_perceptron_tagger')
```

3. Maxent NE chunker

The next basic library to be downloaded is Maxent NE chunker. It contains two pre-trained English named entity chunkers trained on ACE corpus. The NLTK's NEC works by using a supervised machine learning algorithm known as a MaxEnt classifier [33]. A MaxEnt

Classifier gets its name from maximum entropy. In discrete probability distribution, maximum entropy is obtained when distribution is uniform.

The machine learning model used here takes data from ACE corpus which has been manually annotated for NE's. This annotator marks named entities or NE's for each tokenized word in the text. Different Chunk types or chunk labels can be obtained through this process. The chunk types can be listed as follows [32],

1. LOCATION
2. ORGANIZATION
3. PERSON
4. DURATION
5. DATE
6. CARDINAL
7. PERCENT
8. MONEY
9. MEASURE

These chunk types are decided based upon the words that match with the ACE Corpus. It can then be used into our algorithms as per the requirements. The process to download the chunker is same as the above two downloads. The command can be provided as follows,

```
nltk.download('maxent_ne_chunker')
```

The usage of these libraries into algorithm and finding the NE's to prepare the dataset, will be discussed in the following section.

The whole process discussed above in 3 parts can be summarized in the following figure.

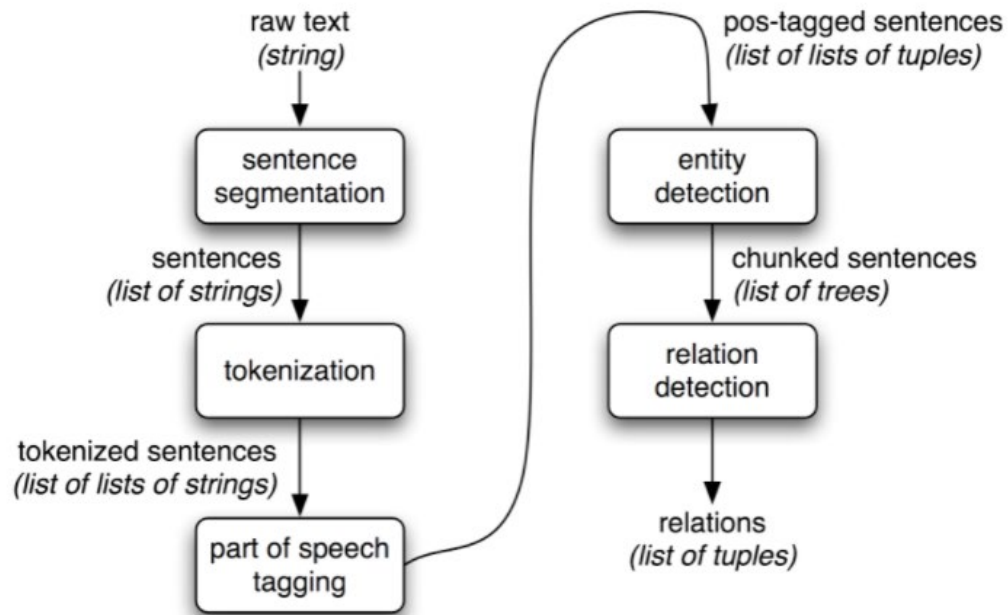


Figure 4.5 Summarized process of finding named entities in text

4. Words

As we deal with English words, the final download that should be performed is the 'words' library. The command to download it can be provided as follows.

```
nltk.download('words')
```

Now, as all the downloads can be performed using these commands, the algorithm can be started.

4.3.2 Algorithm to find Named Entities

As we have already discussed about the requirements of additional attributes in previous sections, they can be added into the raw dataset using the algorithm given in this section. The detailed description about each stage will be given along with instances across this section.

The detailed process in this algorithm is explained in the form of steps, which are provided as follows,

Step-1: Imports from NLTK

After downloading the required packages, which we have seen in previous section, the corresponding methods have to be imported from nltk. This can be provided as follows in the form of code snippets.

```
from nltk import ne_chunk, pos_tag, word_tokenize
from nltk.tree import Tree
```

Figure 4.6 Required libraries imported from NLTK

From the above figure, we can observe that the libraries `ne_chunk`, `pos_tag`, `word_tokenize` have been imported from NLTK package. From the previous sections, we know that the basic libraries to be downloaded have to be related to tokenizer, tagger, and chunker. Therefore, the relevant methods can be obtained by importing these libraries into the algorithm. The library “Tree” is also imported from NLTK because when the text is processed till the step of chunking, it splits the text in form of category called trees, which can be captured through this library. These operations will be seen in detailed in the following steps of this section.

Step-2: Applying Tokenizer, Tagger and Chunker

Firstly, text input is taken, and a tokenizer is applied. This can be shown in the form of code snippet as follows.

```
▶ text = ""Governor Andrew Cuomo is highly concerned about New York City""
  tokenized_text = word_tokenize(text)
  tokenized_text|

['Governor',
 'Andrew',
 'Cuomo',
 'is',
 'highly',
 'concerned',
 'about',
 'New',
 'York',
 'City']
```

Figure 4.7: Instance of tokenized output of input text

The figure above shows an instance of how a tokenizer works with an input text. For instance, one of the leader’s statement which is statement given by Governor Andrew Cuomo is taken for processing. As we can see, the words have been tokenized after using the method “word_tokenize”.

The process after tokenizing the text into words is POS Tagging, which is tagging each word with its own parts of speech. We can use the method “pos_tag” to tag each word, which is shown in the figure as follows.


```
▶ parts_of_speech_words = pos_tag(tokenized_text)
parts_of_speech_words

[('Governor', 'NNP'),
 ('Andrew', 'NNP'),
 ('Cuomo', 'NNP'),
 ('is', 'VBZ'),
 ('highly', 'RB'),
 ('concerned', 'VBN'),
 ('about', 'IN'),
 ('New', 'NNP'),
 ('York', 'NNP'),
 ('City', 'NNP')]
```

Figure 4.8: Instance of POS Tagging

There are quite many NLTK POS Tags which will be provided in the Appendix 1. From the figure, we can observe that the method applied to tokens has given an output of POS tag to each word in the list. Here NNP is proper noun, which is assigned to all the Names, roles, etc. VBZ represents verb, present tense, and VBZ represents verb, past tense. The tag IN represents and tags words like conjunctions or prepositions.

After getting the parts of speech tag for each word in the sentence, the next process is to start with chunking the words into tree structure. The method “ne_chunk” can be used to start chunking. It creates a list by assigning each word into a categorized tree. This can be shown in the figure as follows,

```
▶ chunked = ne_chunk(pos_tag(word_tokenize(text)))
  for i in chunked:
    print(i)

↳ ('Governor', 'NNP')
  PERSON Andrew/NNP Cuomo/NNP)
  ('is', 'VBZ')
  ('highly', 'RB')
  ('concerned', 'VBN')
  ('about', 'IN')
  (GPE New/NNP York/NNP City/NNP)
```

Figure 4.9: Instance of NE Chunker

From the figure above, we can observe that the name “Andrew Cuomo” of Governor has been detected and allocated into a chunk type or label called “PERSON”. And one of the other important detection by chunker is the words “New York City”. It has been identified into the category of GPE (Geo-Political Entity). But these are still in the form of trees which is not the required form. We can extract each chunk, its tree, and each leaf by using for loops. The complete algorithm is given in Appendix-2. The loop formations and the conditions have been stored in a function, as it has to work for huge amounts of data. The final output of the function is provided in the following figure.

```
txt = """Governor Andrew Cuomo is highly concerned about New York City"""
print(named_entity(txt, 'PERSON'))
print(named_entity(txt, 'GPE'))

['Andrew Cuomo']
['New York City']
```

Figure 4.10: Instance of Final output obtained from Named Entity Algorithm

From the above figure, we can observe that we have obtained the required outputs. The Leader’s name in the statement is “Andrew Cuomo”, which is being identified as a “PERSON”,

and the statement spoken city in the statement, or the statement is related to the city “New York”, which is being identified as “GPE”. In this way, we can get the required attributes like Leader’s name, their statement spoken region into the raw dataset, that is obtained at the end of chapter 3.

4.3.3 Applying Algorithm to the Dataset

Finally, a revised dataset can be obtained when the algorithm is applied to the previous raw dataset. The output of this operation can be shown, and a sample amount of entire dataset is arranged as follows in the table below.

Table 4.1 Sample of dataset showing statements provided by leaders in state of Georgia

Statemen t_Dates	Headlines	Stories	Headline Region	Type of Leader	Leader Name
07-01-20	Georgia has no plan to order...	Gov. Brian Kemp speaks at a press...	Georgia	Governor	Brian Kemp
07-08-20	Atlanta mayor says she is...	Atlanta Mayor Keisha Lance...	Georgia	Mayor- Atlanta	Keisha Lance...
07-23-20	Atlanta mayor says she's...	Atlanta Mayor Keisha Lance...	Georgia	Mayor- Atlanta	Keisha Lance...
07-31-20	Georgia governor extends public...	Gov. Brian Kemp signed two....	Georgia	Governor	Brian Kemp

The above table consists of sample amount of dataset related to state of Georgia, and the statements given by some of the leaders in that State. Now the dataset is completely prepared with the required additional attributes. And the usage of this data in analyzing the sentiment and emotion is provided in the following chapters.

CHAPTER 5

SENTIMENT ANALYSIS

5.1 Introduction

As discussed before, currently during this pandemic, there is a huge amount of change in the lifestyles of people. Many preventive measures have been implemented by leaders and their words have affected a lot during this situation. To analyze how the words of leaders in a certain region have affected people can be known through a measure called sentiment. By capturing sentiment, people's opinions about the scenarios can be judged. Each statement or any form of text can be classified into its own polarity based on the words it contains. In general, through sentiment analysis, each sentence can be classified into polarities positive or negative.

This thesis, as discussed in previous chapters, uses supervised learning methods to train the model and then labels the statements with the help of highly accurate and trained machine learning model.

This chapter discusses the work on polarity classification, where the leader's statements collected and prepared along with additional attributes from previous chapters will be classified as either positive or negative according to the situations of the pandemic. This chapter shows experiments performed to achieve a highly accurate model and gives detailed description about efficient model that is used for labelling the dataset. Firstly, this chapter discusses about the dataset that can be taken to train the required model and then enters towards selection of an efficient model and finally shows a labelled output with high amount of accuracy.

5.2 Dataset Description

The dataset used in this work to train the models is the famous Movie Review Dataset [35]. This dataset is used for binary classification which consists of 25000 highly polar movie reviews for training and 25000 for testing. This dataset is used in such a way that 20000 reviews were used to train the model and 5000 reviews were used for validation. Each review in the dataset has a 'LABEL_COLUMN' where the sentiment labels are present. Sentiment label 1 represents Positive reviews and sentiment label 0 represents Negative reviews. This dataset also contains 25000 unlabeled reviews. But they were removed as this is a supervised learning task. Finally, the 'DATA_COLUMN' and 'LABEL_COLUMN' were taken for training and testing. The following section gives detailed description about machine learning models that were used for experimenting and working of the final selected model to perform sentiment labeling.

5.3 Selection of Machine Learning Model

Basically, the problem in this thesis is classification. And as we know, a classification model aims to draw some conclusion from observed inputs. The conclusions are labels that can be applied to the dataset and used for analysis further.

There are numerous classification models that include decision tree, random forest, logistic regression, Naïve Bayes, gradient-boosted tree, support vector machine and multilayer perceptron. To select an efficient model, firstly the dataset is experimented on different models to find accuracy of labeling for the movie reviews dataset.

In order to start with experimentation, the movies review dataset is cleaned by using the text pre-processing techniques that have been discussed earlier in Chapter 4. The cleaning process and output after each step is discussed in brief as follows.

5.3.1 Dataset Cleaning

Both the dataset used for training labels and the leader's statements dataset has to be cleaned before processing the text in them. The process of cleaning is described in the form of steps given as follows.

Step-1: Remove Punctuations

Beginning with cleaning, punctuations are removed from dataset. Both the train and test datasets of movie reviews dataset are modified in the same way. This is repeated to the leader's statements dataset that is revised and obtained at the end of Chapter-4.

Step-2: Removal of Stop Words

After removal of punctuations, unwanted words which are also known as stop words have to be eliminated from all the datasets. The stop word removal procedure is shown in the below figure.

```
['went saw movie last night coaxed friends mine ill admit reluctant see
ed character jake fischer well kevin costner played ben randall profess
ntire theater sold overcome laughter first half movie moved tears secur
ll grown men well trying desperately let anyone see crying movie great
'actor turned director bill paxton follows promising debut gothic horr
n young american caddy rises humble background play bristish idol dubbe
erdog sports flicks dime dozen recently done grand effect miracle cinde
ning credits imagine disneyfied version animated opening credits hbos c
ur action moves us open things pick well paxton nice job shows knack ef
ntage action day two open propel plot add unexpected psychological dept
nt british harry vardon haunted images aristocrats black suits top hats
e also good job visually depicting goes players heads pressure golf pai
so given set designers costume department creating engaging period piec
ury know going end based true story also films genre follow template pa
ent behind camera ever front despite formulaic nature nice easy film rc
'recreational golfer knowledge sports history pleased disneys sensitiv
depicted well psychological battles harry vardon fought within childhoc
ents accepted equal english golf society likewise young ouimet goes cla
scoff attempts rise standing loved best however theme class manifested
```

Figure 5.1 Sample output after removing stop words from dataset

Step-3: Lemmatization

From the previous section 4.1, we get to know that Lemmatization is better than Stemming, as Lemmatization considers the parts of speech of words and then transforms the words into root words, which lacks in stemming. So, we have performed lemmatization here for both the datasets. And the figures below show the difference between non-lemmatized and lemmatized text.

```
['bromwell high cartoon comedy ran time programs school life teachers  
ighs satire much closer reality teachers scramble survive financially  
pettiness whole situation remind schools knew students saw episode stu  
ed high classic line inspector im sack one teachers student welcome br  
gh far fetched pity isnt',
```

Figure 5.2 Sample input text before lemmatization

```
['bromwell high cartoon comedy ran time program school life teacher  
satire much closer reality teacher scramble survive financially insi  
ss whole situation remind school knew student saw episode student re  
lassic line inspector im sack one teacher student welcome bromwell h  
hed pity isnt',  
'homelessness houselessness george carlin stated issue year never p
```

Figure 5.3 Sample output after lemmatization

After obtaining the lemmatized text, the datasets can be prepared for sentiment analysis, which is shown as follows.

Train Dataset after Preprocessing

review_numbers	Reviews	ratings	Category	Reviews after Cleaning
0	0	Bromwell High is a cartoon comedy. It ran at...	9 positive	bromwell high cartoon comedy ran time program ...
1	10000	Homelessness (or Houselessness as George Carli...	8 positive	homelessness houselessness george carlin state...
2	10001	Brilliant over-acting by Lesley Ann Warren. Be...	10 positive	brilliant acting lesley ann warren best dramat...
3	10002	This is easily the most underrated film inn th...	7 positive	easily underrated film inn brook cannon sure f...
4	10003	This is not the typical Mel Brooks film. It wa...	8 positive	typical mel brook film much le slapstick movie...
...
24995	9998	Towards the end of the movie, I felt it was to...	4 negative	towards end movie felt technical felt like cla...
24996	9999	This is the kind of movie that my enemies cont...	3 negative	kind movie enemy content watch time bloody tru...
24997	999	I saw 'Descent' last night at the Stockholm Fi...	3 negative	saw descent last night stockholm film festival...
24998	99	Some films that you pick up for a pound turn o...	1 negative	film pick pound turn rather good 23rd century ...
24999	9	This is one of the dumbest films, I've ever se...	1 negative	one dumbest film ive ever seen rip nearly ever...

25000 rows × 5 columns

Figure 5.4 Movie Reviews Dataset after Cleaning

From the figure above we can observe that the Category and Reviews after Cleaning attributes are only useful for training the classifiers. So, they have been separated from the dataset and stored in new files which can be used for further processing.

```

reviews_train
0      bromwell high cartoon comedy ran time program ...
1      homelessness houselessness george carlin state...
2      brilliant acting lesley ann warren best dramat...
3      easily underrated film inn brook cannon sure f...
4      typical mel brook film much le slapstick movie...
      ...
24995  towards end movie felt technical felt like cla...
24996  kind movie enemy content watch time bloody tru...
24997  saw descent last night stockholm film festival...
24998  film pick pound turn rather good 23rd century ...
24999  one dumbest film ive ever seen rip nearly ever...
Name: Reviews after Cleaning, Length: 25000, dtype: object

reviews_test
0      went saw movie last night coaxed friend mine i...
1      actor turned director bill paxton follows prom...
2      recreational golfer knowledge sport history pl...
3      saw film sneak preview delightful cinematograp...
4      bill paxton taken true story 1913 u golf open ...
      ...
24995  occasionally let kid watch garbage understand ...
24996  anymore pretty much reality tv show people mak...
24997  basic genre thriller intercut uncomfortable me...
24998  four thing intrigued film firstly star carly p...
24999  david bryces comment nearby exceptionally well...
Name: Reviews after Cleaning, Length: 25000, dtype: object

```

Figure 5.5 Movie Reviews Dataset for training Classifiers

5.3.2 Experiments with different classifiers

The dataset obtained after cleaning can now be used on to different classifiers for testing and training. This section focuses on different type of classifiers used for experiments and selecting the most efficient one among them. The description and working about the selected model will be discussed in the next section, along with labeling the dataset containing leader's statements.

Before using the datasets to train, it has to be vectorized. For vectorization, two vectorizers have been used for experimenting, Count Vectorizer and TF-IDF vectorizer, which in brief, have been discussed in Chapter-4. After vectorizing the movie reviews dataset, it has been sent to the classifiers.

EXPERIMENT-1:

Firstly, the data has been tested among classifiers, Logistic Regression, Support Vector Machine and Support Vector Machine with Linear Kernel. Among three classifiers, better results were obtained when dataset is trained and tested through Linear SVC with count vectorizer, when compared to others. The training accuracy is shown as follows,

```
Accuracy for C=0.01: 0.85008
Accuracy for C=0.05: 0.88032
Accuracy for C=0.25: 0.89536
Accuracy for C=0.5: 0.89568
Accuracy for C=1: 0.89312
Final Accuracy: 0.8558
```

Figure 5.6 Accuracy with Linear SVC

EXPERIMENT-2:

For improving the accuracy, n-gram range is introduced along with the count vectorizer used. During this experiment, several n-gram ranges were taken such as (1,2), (1,3), (1,4) which represents taking 2 words at once, 3 words at once, and 4 words at once respectively. Among these, better results were obtained when (1,2) n-gram range was taken. The accuracy has been improved to 88%, which is shown as follows.

```
Accuracy for C=0.01: 0.88592
Accuracy for C=0.05: 0.88608
Accuracy for C=0.25: 0.8848
Accuracy for C=0.5: 0.88496
Accuracy for C=1: 0.88528
Final Accuracy: 0.88712
```

Figure 5.7 Improved accuracy with Linear SVC with n-gram range (1,2)

EXPERIMENT-3:

In an attempt to increase accuracy further, a special list of stop words is added to the model, to remove if there is any noise and to increase the efficiency of count vectorizer. Through this there was an improvement in accuracy, but not to a level that is satisfiable. The accuracy after experiment 3 is shown as follows,

```
Accuracy for C=0.01: 0.88736
Accuracy for C=0.05: 0.88592
Accuracy for C=0.25: 0.8856
Accuracy for C=0.5: 0.8856
Accuracy for C=1: 0.88544
[1 1 1 ... 0 0 1]
Final Accuracy: 0.88716
```

Figure 5.8 Improved accuracy with Linear SVC

5.3.3 Experiments with deep learning networks

There are techniques like hyper parameter tuning, with which accuracy can be boosted. But using them for conventional classification models could increase the complexity and run time of the model when higher amount of data has to be labeled. This limitation has given a scope to explore the concepts of deep learning models.

After cleaning the dataset, before feeding the data to deep learning networks, the data should be transformed into floating-point tensors. To perform this conversion, word-embeddings can be used. To obtain word embeddings, the process can be started with learning random word vectors. But the package keras provides conversion into word embedding by an Embedding layer.

EXPERIMENT-1:

The first experiment is by developing a simple multi-layer perceptron model with a single hidden layer. As an input layer, embedding layer can be used by setting word size to 32 and vocabulary to 5000, and the input length to 500. From this, the output obtained from the first layer will be 32x500 sized matrix. Then this output will be flattened to one dimension. Then a dense hidden layer of rectifier activation function is used. After this, a dense layer with sigmoid activation function is used to predict 0 and 1.

In this experiment, the model uses ADAM optimizer for optimization. This model is plotted using the keras package, which is shown as follows.

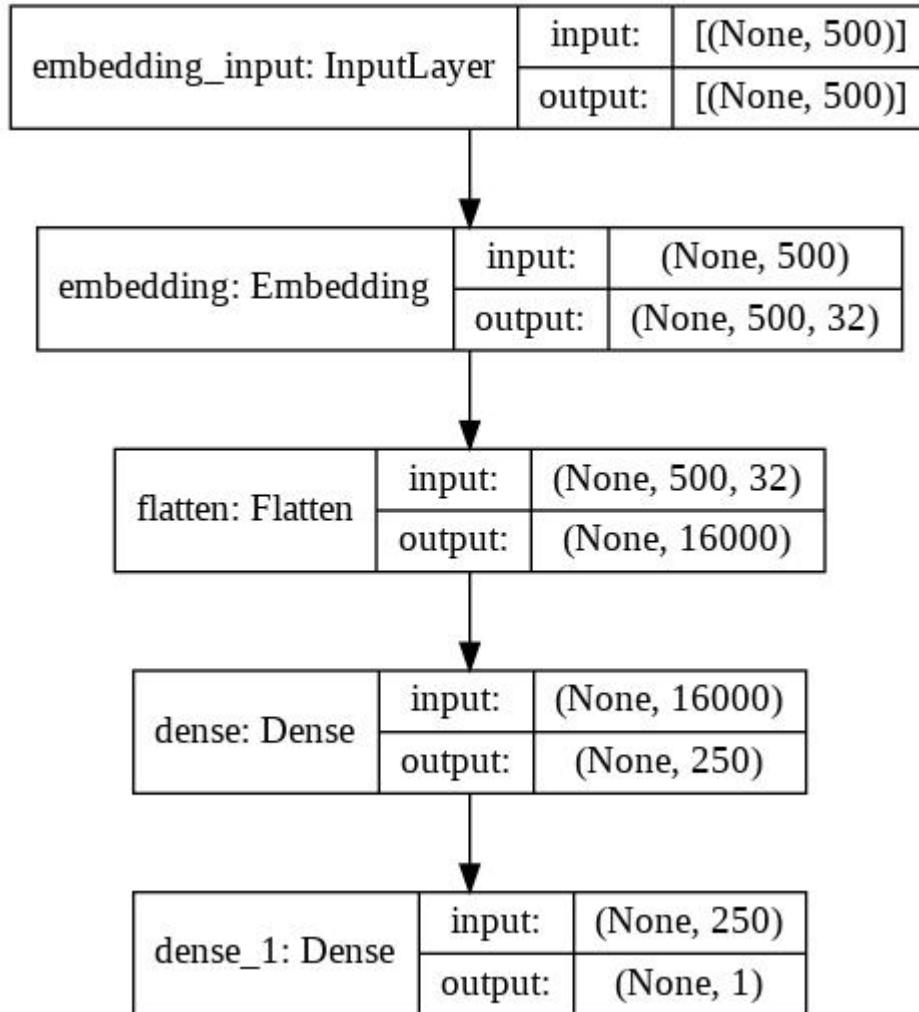


Figure 5.9 Input Layers of Multi-layer perceptron model

The accuracy obtained from the above model is 93.30% after 2 Epochs and is shown as follows.

```

Epoch 1/2
196/196 - 25s - loss: 0.4886 - accuracy: 0.7357 - val_loss: 0.3264 - val_accuracy: 0.8586
Epoch 2/2
196/196 - 24s - loss: 0.1778 - accuracy: 0.9330 - val_loss: 0.3244 - val_accuracy: 0.8657
Accuracy: 86.57%
  
```

Figure 5.10 Accuracy obtained from Multi-layer perceptron model

EXPERIMENT-2:

The second experiment is building a sentiment classifier with pre-trained NLP model that is BERT. BERT stands for Bidirectional Encoder Representation from Transformers [36]. There are two pre-trained variations of BERT. The base model is a 12-layer, 768-hidden, 12-heads, 110M parameter neural network architecture, whereas the large model is a 24-layer, 1024-hidden, 16-heads, 340M parameter neural network architecture. The architecture can be visualized as follows [36].

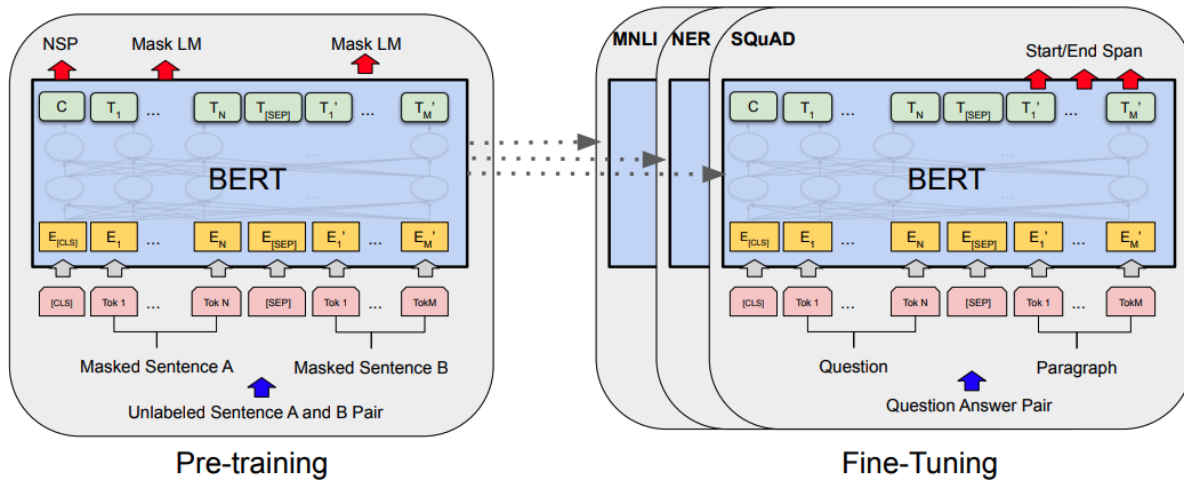


Figure 5.11 Pre-training and Fine-Tuning procedures for BERT

As BERT comes under the category of transformers, it can be imported from the Transformers library in Python. The dataset is tokenized and fed same as that of Experiment-1. This experiment uses ADAM as optimizer too. But the difference is dataset is fed in the form of input sequences, and fine tuning the model with BERT. This improved the accuracy to 96% in two epochs. The output can be shown as follows,

```

WARNING:tensorflow:from the parameter 'return_dict' cannot be set in graph mode and will always be set to 'True'.
1250/1250 [=====] - 1129s 860ms/step - loss: 0.3814 - accuracy: 0.8082 - val_loss: 0.3470 - val_accuracy: 0.8804
Epoch 2/2
1250/1250 [=====] - 1080s 864ms/step - loss: 0.0982 - accuracy: 0.9640 - val_loss: 0.4341 - val_accuracy: 0.8778
<tensorflow.python.keras.callbacks.History at 0x7f39a08c1d50>

```

Figure 5.12 Output of Classifier after using BERT

The training accuracies obtained from the experiments is summarized in table as follows.

Table 5.1 Summary of Experiments conducted for sentiment classifier

Models Experimented	Accuracy Obtained
Logistic Regression	83.4%
Linear SVC	85.6%
Linear SVC with count vectorizer	88.72%
Multi-layer perceptron model	93.30%
Classifier with BERT pre-trained model	96.4%

As the accuracy obtained from experiment-2 using deep learning networks is the highest among all, it had been selected as the best classifier to label the sentiments for the leader’s statements dataset. Detailed explanation about creating input sequences, fine tuning with BERT and making predictions or labeling is provided in next section for the selected classifier.

5.3.4 Working of BERT Tokenizer and BERT Classifier

5.3.4.1 Brief Introduction to BERT

BERT is constructed upon the concepts of pre-training contextual representations. Pre-trained representations can either be context-free or contextual, and contextual representations can further be unidirectional or bidirectional [43]. BERT represents bidirectional context-free functionality by considering its previous and next context to generate a word embedding. This feature makes it unique from other context-free models. To minimize the problems that occur during bidirectional predictions, BERT constitutes straightforward technique of masking out some of the words in the input and then condition each word bidirectionally to predict the masked words [43].

5.3.4.2 BERT Model

In general, BERT embeddings are trained for performing two major tasks. These are listed as follows.

1. Sequence Classification Task: In this task, the classifier returns the category of the input sentence.
2. Next sentence prediction Task: In this task, the model determines whether the next sentence follows the current sentence or not.

In order to perform these tasks, the sentences have to be tokenized. BERT constitutes a tokenizer which can be installed from “transformers” package in Python. Using this package, the model and tokenizer can be extracted. The commands can be shown as follows,

```
model = TFBertForSequenceClassification.from_pretrained("bert-base-uncased")
```

```
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
```

The model summary can be shown as follows.

```
▶ model.summary()
↳ Model: "tf_bert_for_sequence_classification"
```

Layer (type)	Output Shape	Param #
bert (TFBertMainLayer)	multiple	109482240
dropout_37 (Dropout)	multiple	0
classifier (Dense)	multiple	1538

```
=====  
Total params: 109,483,778  
Trainable params: 109,483,778  
Non-trainable params: 0  
=====
```

Figure 5.13 Model Summary of BERT

Creation of Input Sequences:

To feed the input into the BERT model, sequences have to be created from the required dataset. To perform this operation, the function “*InputExample*” can be used to create input sequences from train and test dataset. The usage of this function is shown as follows,

```
train_InputExamples = train.apply(lambda x: InputExample(guid=None, text_a = x[DATA_COLUMN], text_b = None, label = x[LABEL_COLUMN]), axis = 1)  
validation_InputExamples = test.apply(lambda x: InputExample(guid=None, text_a = x[DATA_COLUMN], text_b = None, label = x[LABEL_COLUMN]), axis = 1)
```

Figure 5.14 Usage of function InputExample

The obtained sequences can then be used for sequence classification. Now, the sequences can be tokenized using the BERT tokenizer mentioned above, to create an input format with tokenized objects. Detailed lines of code will be shown in Appendix-2.

Fine Tuning the BERT Model:

The model can be compiled using “Adam” as optimizer, “SparseCategoricalAccuracy” as accuracy metric and “CategoricalCrossentropy” as loss function, for achieving better results. Fine tuning the model with the help of these will give 96% training accuracy, as shown in the figure below.

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=3e-5, epsilon=1e-08, clipnorm=1.0),
              metrics=[tf.keras.metrics.SparseCategoricalAccuracy('accuracy')],
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True))

model.fit(train_data, epochs=2, validation_data=validation_data)
```

```
Epoch 1/2
WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when calling a model.They ha
WARNING:tensorflow:AutoGraph could not transform <bound method Socket.send of <zmq.sugar.socket.Socket object at 0x7f3a97e48d00>> and will r
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and atta
Cause: module, class, method, function, traceback, frame, or code object was expected, got cython_function_or_method
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING: AutoGraph could not transform <bound method Socket.send of <zmq.sugar.socket.Socket object at 0x7f3a97e48d00>> and will run it as-i
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and atta
Cause: module, class, method, function, traceback, frame, or code object was expected, got cython_function_or_method
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING:tensorflow:AutoGraph could not transform <function wrap at 0x7f3ab36f5d40> and will run it as-is.
Cause: while/else statement not yet supported
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING: AutoGraph could not transform <function wrap at 0x7f3ab36f5d40> and will run it as-is.
Cause: while/else statement not yet supported
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when calling a model.They ha
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
1250/Unknown - 1084s 824ms/step - loss: 0.3815 - accuracy: 0.8081WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_st
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
1250/1250 [=====] - 1129s 860ms/step - loss: 0.3814 - accuracy: 0.8082 - val_loss: 0.3470 - val_accuracy: 0.8804
Epoch 2/2
1250/1250 [=====] - 1080s 864ms/step - loss: 0.0982 - accuracy: 0.9640 - val_loss: 0.4341 - val_accuracy: 0.8778
<tensorflow.python.keras.callbacks.History at 0x7f39a08c1d50>
```

Figure 5.15 Fine tuning of BERT Model

Then the sentiments for tokenized sequences can be predicted and required labels can be generated which are shown in the next section.

5.3.5 Labeling Sentiments using BERT Classifier

After using the BERT Classifier to label the sentiments positive and negative, the dataset can be transformed as follows,

Table 5.2 Sample of dataset showing statements provided by leaders in state of Georgia with Sentiment of Headline and Story

Statement Dates	Headlines	Stories	Headline Region	Type of Leader	Leader Name	Sentiment Headline	Sentiment Story
07-01-20	Georgia has no plan to order...	Gov. Brian Kemp speaks...	Georgia	Governor	Brian Kemp	Negative	Negative
07-08-20	Atlanta mayor says she's si...	Atlanta Mayor Keisha Lance...	Georgia	Mayor-Atlanta	Keisha Lance...	Positive	Positive
07-23-20	Atlanta mayor says she's wo...	Atlanta Mayor Keisha Lance...	Georgia	Mayor-Atlanta	Keisha Lance...	Positive	Positive
07-31-20	Georgia governor exte...	Gov. Brian Kemp signed tw...	Georgia	Governor	Brian Kemp	Positive	Positive

From the above table, we can observe that the sentiments have been labeled for each statement using the BERT Classifier model. As we can see, the classes obtained from sentiment labels are positive and negative which does not provide detailed intentions present in leader's statements. There can be many factors related to how leader's think. For instance, some of them were highly concerned about their states, and have always given statements related to safety measures. And some of the other leaders were not concerned about their states and have always given such statements which show that no precautions or measures were not taken during

pandemic, which led to increase in number of COVID-19 cases. This incompleteness in the opinion or intention after calculating the sentiment led to finding emotions in the statements.

The following chapter provides description about emotion analysis, supervised classification to label emotions, experiments to select classifier model and finally an upgrade to the dataset of leader's statements by adding the emotion labels.

CHAPTER 6

EMOTION ANALYSIS

Emotion Recognition from text is related to the field of NLP, and it is closely related to sentiment analysis. As we have seen in the previous chapter, the sentiments were restricted to categories Positive and Negative. Emotions can also be split into two kinds which are positive and negative emotions. The positive emotion can consist of happiness, interest, concerned, joy, laughter, etc. Whereas negative emotions could be fear, angry, sadness, unconcerned and so on.

Emotions can be classified using supervised learning methods, similar to that to labeling sentiments, which was discussed in previous chapter. The same procedure can be followed to classify emotions, by taking an emotion rich dataset, cleaning and pre-processing it, train a classifier accordingly, and start labeling or predicting emotions for new datasets.

This chapter discusses about the datasets that are available for emotion analysis, experiments using datasets and finally labeling the leader's statements dataset that is revised in previous chapter.

6.1 Dataset Description

There are many datasets which contain the basic set of emotions such as happiness, anger, sadness, surprise, disgust, and fear. But when situations related to pandemic are considered, the emotion labels can change. It can be shown through instances.

Ex: "Leader says, masks are not mandatory. It is okay to not wear masks."

In the present scenario, wearing a mask should be mandatory as the number of cases were not controllable at a point of time and it can also prevent the increase in number of cases. Such

restrictions have reduced the number of cases in many other states. So, this statement looks *fearful* according to the present situation. It can also be considered as an *unconcerned* statement when we look from the aspect of leader. Therefore, the emotion in this statement can be classified as *unconcerned*. So, to train a machine learning model with such statements and emotions, normal datasets with the conventional set of emotions will not be sufficient, it has to be increased with the data that is related to COVID-19, in order to judge how concerned or unconcerned the leaders were towards their states.

The dataset is prepared by combining the datasets acquired from sources such as dailydialog [39], ISEAR (International Survey On Emotion Antecedents And Reactions) [40], emotion stimulus [38] and Emotional responses to COVID-19 [41]. After combining these datasets, there were several emotion labels, which can be listed as follows,

1. Happiness
2. Sadness
3. Concerned
4. Unconcerned
5. Safety
6. Anger
7. Disgust
8. worry
9. Neutral

As the dataset was intended to be used for such purposes, not much data cleaning and pre-processing was necessary except for the operations like removal of punctuations, tokenization, and lemmatization.

6.2 Experiments with different Classifiers

The dataset during experiments is split into test and train. The train and test consist of 9834 and 3959 emotional statements with the above listed emotion labels. Several experiments were performed on this dataset by choosing different classifiers, starting from Naïve Bayes to deep learning network with BERT pre-trained model. The F1-scores have been noted into the table as follows.

Table 6.1 Summary of Experiments conducted for Emotion classifier

Models Experimented	F1-Score Obtained
Naïve Bayes	56.34%
Logistic Regression	67.02%
Linear SVC	68.345%
LSTM with word2vec embedding layer	71.92%
CNN with word2vec embedding layer	73.8%
Classifier with BERT pre-trained model	81.2 %

As the accuracy obtained from experiment using BERT pre-trained model in deep learning networks is the highest among all, it had been selected as the best classifier to label the emotions for the leader's statements dataset.

6.3 Labeling Emotions using BERT Classifier

The emotion labels can be classified in the same way as sentiments have been classified earlier using BERT model in section 5.3.4. After using the BERT Classifier to label the emotions, the dataset is upgraded as follows,

**Table 6.2 Sample of dataset showing statements provided by leaders in state of Georgia
with Sentiments and Emotions of Headline and Story**

Statement Dates	Headlines	Stories	Headline Region	Type of Leader	Leader Name	Sentiment Headline	Emotion Headline	Sentiment Story	Emotion Story
07-01-20	Georgia has no plan to order...	Gov. Brian Kemp speaks at a...	Georgia	Governor	Brian Kemp	Negative	Unconc...	Negative	Sad, Unc...
07-08-20	Atlanta mayor says she's si...	Atlanta Mayor Keisha Lance...	Georgia	Mayor-Atlanta	Keisha Lance...	Positive	Sad, Unconc...	Positive	Angry, Unc...
07-23-20	Atlanta mayor says she's wo...	Atlanta Mayor Keisha Lance...	Georgia	Mayor-Atlanta	Keisha Lance...	Positive	Safety	Positive	Sad, Unc...
07-31-20	Georgia governor exte...	Gov. Brian Kemp signed two exe...	Georgia	Governor	Brian Kemp	Positive	Safety	Positive	Safety, Unc...

From the above table, we can observe that the required attributes which are emotion labels for both stories and headlines, and sentiment labels for both stories and headlines have been added. This dataset can be used to analyze the leader's statements during the pandemic. The analysis and results will be shown in the following chapter.

CHAPTER 7

RESULTS AND ANALYSIS

7.1 Analysis of Leader's statements

Once a dataset is ready, we can start analyzing several aspects like most unconcerned statements given by leaders, types of emotions in their positive and negative statements, graphical analysis of both news statements and COVID-19 data in time series [42]. As we have seen earlier, each emotion can be sub categorized from positive and negative sentiments, the analysis is based upon this. Different types of analysis performed on the statements of leader's is shown as follows.

Analysis-1:

From the leader's statements dataset that was ready in previous chapter, we can choose which leader's statements we want to analyze. As instances were already shown relating to state of Georgia, the analysis will be continued with the same state. In analysis 1, the leader chosen from the state of Georgia is governor, Brian Kemp. When most unconcerned and negative statements are analyzed, these statements have been printed, which are given as follows.

Most Unconcerned and negative statements of Governor-State of Georgia:

Statement Date: 07-01-2020

Headline: Georgia's governor said they could reopen. More than 50 restaurateurs said in a newspaper ad they're not ready.

Statement Date: 07-16-2020

Headline: Georgia has no plan to order people to wear masks, governor says

From the above statements, we can notice that the governor had taken measures to re-open the state in peak of pandemic such as month of July. And also, many restaurants were not ready with his orders to re-open. This can be verified by looking at the number of confirmed cases of COVID-19 in the state of Georgia, during the month of July. These can be extracted from the resources provided by CSE department of John Hopkins University [44], which provides several attributes like number of confirmed cases, number of deaths, locations, etc.

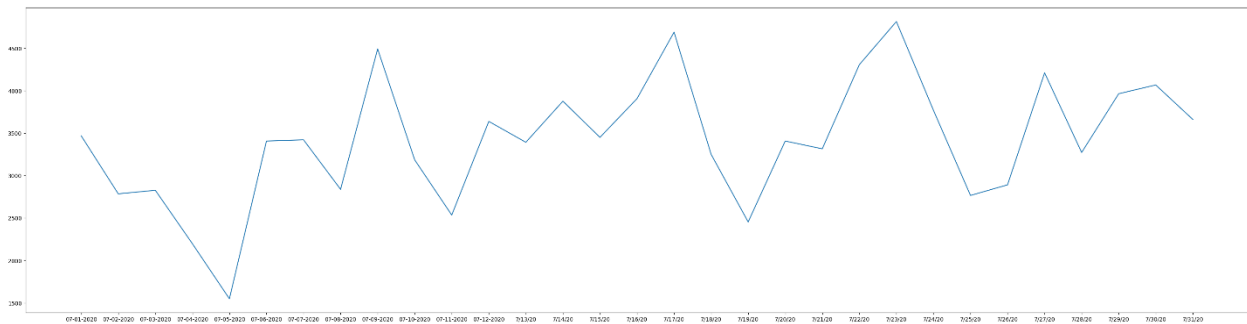


Figure 7.1 Figure showing number of confirmed cases of COVID-19 in the state of Georgia during the month of July

From the above figure, we can observe that the number of confirmed COVID-19 cases were nearly high in the mid of July. But at the same time, the governor had passed statements that include severe terms like re-opening of state. Such statements prove to be instances where leaders have not concerned about their ruling states.

Analysis-2:

In the second analysis, the leader’s statements were visualized based on emotions that come under each category of sentiment. When the same leader, governor of state of Georgia is chosen, there were more statements taken by him which come under category *safety* among positive

statements. This represents that the governor has taken safety and preventive measures but was also unconcerned at times. The output can be shown as follows.

```
Emotions in positive statements by Governor  
<seaborn.axisgrid.FacetGrid at 0x7f650b795550>
```

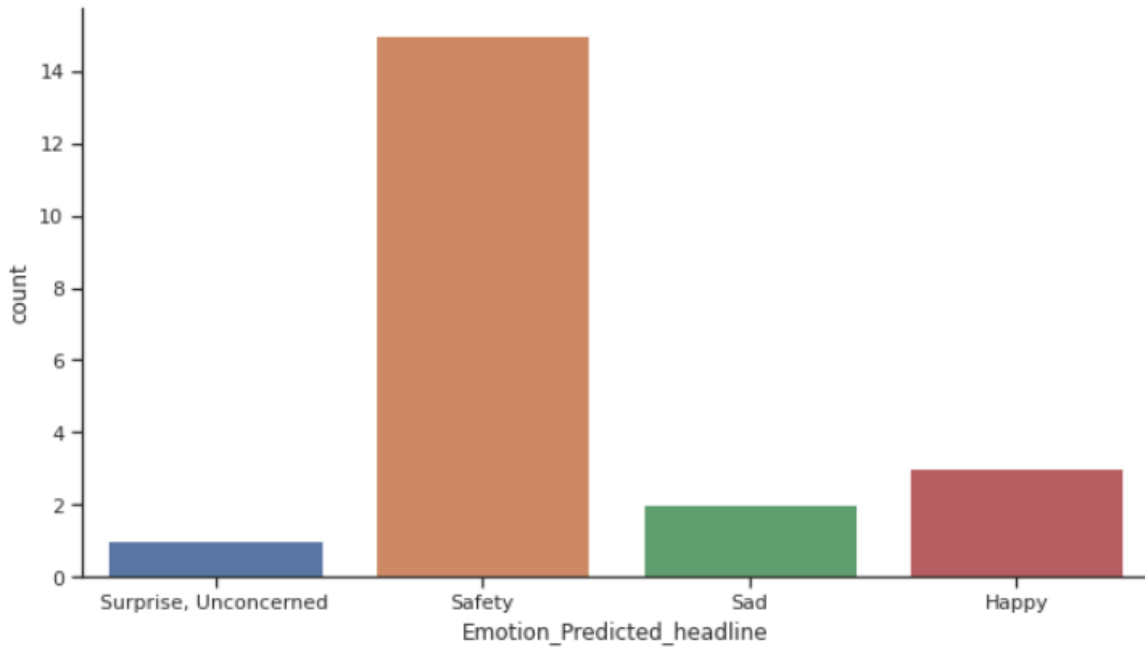


Figure 7.2 Figure showing different Emotions in Positive Statements given by Governor, State of Georgia

As we have discussed earlier, there are unconcerned statements provided by governor at times. This can be shown from the output as follows.

Emotions in negative statements by Governor
<seaborn.axisgrid.FacetGrid at 0x7f650b770050>

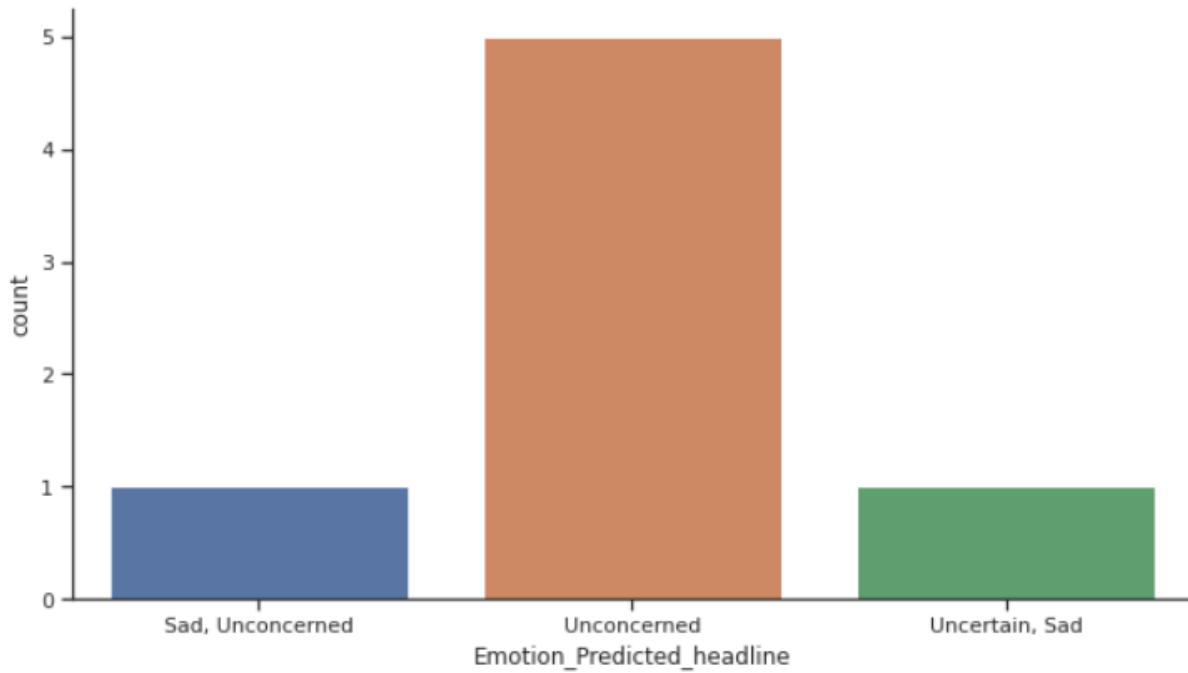


Figure 7.3 Figure showing different Emotions in Negative Statements given by Governor, State of Georgia

The above figures are provided as an example in analysis to see what type of emotions the leader had during the statements, split into main categories of sentiment as positive and negative. In this way, the acquired dataset can be used to analyze several aspects in such type of scenarios.

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 Conclusion

As we have seen from previous chapters that leader's statements not only provide orders, but they also affect people's opinion, which was evident during this pandemic. Not only during COVID-19, but a deeper analysis can also prove that it is the same case for almost every pandemic situation. Such research towards statements concludes that leaders should always be aware of the situation and take actions accordingly, instead of getting influenced by political factors, as it severely affects people's decisions, and it indirectly shows an impact on their daily life.

This thesis reports an analysis of different emotions and sentiments in leader's statements by considering the high-level objective of the work to be analyzing the effects of leader's statements during COVID-19 pandemic. This work shows several intermediate tasks that were performed with a goal to create a dataset for analysis.

Tasks such as Web Scraping have been discussed in chapter 3, where the process to web scrape the data from CNN news channel using efficient methodologies was shown. And in a desire to create more attributes for analysis, NLP techniques used were discussed and shown with several instances. Functions to create required attributes were built in this process. And as the primary objective of this work is to analyze the leader's statements, sentiments and emotions existing were found using highly accurate deep learning networks with the latest BERT- pre trained model. I believe the performance of this model could be increased and accuracy can be improved further if there were more annotated data available for training the emotion classifiers. As the work is summarized in this section, the next section includes the roadmap to future work.

8.2 Future Work

The work presented in this thesis can be pursued further in several directions. One of the major tasks to be addressed is, here the focus was mainly towards leader's statements restricting to the United States of America. This can be extended towards collecting the leader's statements based on the international news giving a scope for extending the analysis, worldwide.

Another direction for future work is to build unsupervised models for sentiment and emotion classification. For sentiment classification, supervised classification was sufficient for labeling sentiments. But emotion classification basically lacks in the amount of data as per the scenarios. So, building an unsupervised classifier according to the targets of particular application can open the doors to future research in this field of analysis.

NLP is another part of this research which has endless strings. Text analysis is always incomplete and numerous tasks can occur based on the problem present in the applications. Extending dataset to analyze international views can also give scope to creating more functions using NLP techniques. This could be yet another possible line of research.

APPENDIX 1

NLTK Pos Tags are used before determining Named Entities present in a sentence. There are numerous tags present in the library, among which some of the identified tags were described above. The complete list of tags can be shown in the following table.

Table A-1: NLTK Pos tags with their meanings

Pos Tags	Meaning
CC	Coordinating Conjunction
CD	cardinal digit
DT	determiner
EX	existential there
FW	foreign word
IN	preposition/subordinating conjunction
JJ	adjective (large)
JJR	adjective, comparative (larger)
JJS	adjective, superlative (largest)
LS	list market
MD	modal
NN	noun, singular
NNS	noun plural
NNP	proper noun, singular
NNPS	proper noun, plural
PDT	predeterminer

POS	possessive ending
PRP	personal pronoun
PRP\$	possessive pronoun
RB	adverb
RBR	adverb, comparative
RBS	adverb, superlative
RP	particle
TO	infinite marker (to)
UH	interjection
VB	Verb
VBG	Verb gerund
VBD	Verb past tense
VBN	Verb past participle
VBZ	Verb, present tense
WDT	Wh-determiner
WP	Wh-pronoun
WRB	Wh-adverb

APPENDIX 2

This appendix consists of algorithms used during each phase of the thesis.

- Code used for Web Scraping of headlines and stories from CNN News Channel:

- Language used: R



web_scraping.txt

- Algorithm used for extracting named entities from input text:

- Language used: Python



named_entities.txt

- Classification using BERT Tokenizer and model:

- Language used: Python



Classification_BERT.txt

REFERENCES

- [1] Saurkar, Anand V., Kedar G. Pathare, and Shweta A. Gode. "An overview on web scraping techniques and tools." *International Journal on Future Revolution in Computer Science & Communication Engineering* 4, no. 4 (2018): 363-367.
- [2] Mehlführer, Andreas. *Web scraping: a tool evaluation*. na, 2009.
- [3] Penman, Richard Baron, Timothy Baldwin, and David Martinez. "Web scraping made simple with sitescraper." *online*, [retrieved Nov. 30, 2015]. Retrieved from the internet (2009).
- [4] Khurana, Diksha, Aditya Koli, Kiran Khatter, and Sukhdev Singh. "Natural language processing: State of the art, current trends and challenges." *arXiv preprint arXiv:1708.05148* (2017).
- [5] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. "An Introduction to Information Retrieval" online, Retrieved from the internet (2009).
- [6] Liu, Bing. "Sentiment analysis and opinion mining." *Synthesis lectures on human language technologies* 5, no. 1 (2012): 1-167.
- [7] Medhat, Walaa, Ahmed Hassan, and Hoda Korashy. "Sentiment analysis algorithms and applications: A survey." *Ain Shams engineering journal* 5, no. 4 (2014): 1093-1113.
- [8] Montoyo, Andrés, Patricio Martínez-Barco, and Alexandra Balahur. "Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments." *Decision Support Systems* 53, no. 4 (2012): 675-679.
- [9] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment classification using machine learning techniques." *arXiv preprint cs/0205070* (2002).

- [10] Dong, Li, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. "Adaptive recursive neural network for target-dependent twitter sentiment classification." In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)*, pp. 49-54. 2014.
- [11] Tang, Duyu, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. "Cooooo!!!: A deep learning system for twitter sentiment classification." In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pp. 208-212. 2014.
- [12] Zhou, Shusen, Qingcai Chen, and Xiaolong Wang. "Fuzzy deep belief networks for semi-supervised sentiment classification." *Neurocomputing* 131 (2014): 312-322.
- [13] Palanisamy, Prabu, Vineet Yadav, and Harsha Elchuri. "Serendio: Simple and Practical lexicon based approach to Sentiment Analysis." In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pp. 543-548. 2013.
- [14] Picard, Rosalind W. *Affective computing*. MIT press, 2000.
- [15] Alm, Cecilia Ovesdotter, Dan Roth, and Richard Sproat. "Emotions from text: machine learning for text-based emotion prediction." In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pp. 579-586. 2005.
- [16] Ekman, Paul. "Basic emotions." *Handbook of cognition and emotion* 98, no. 45-60 (1999): 16.

- [17] Liu, Hugo, Henry Lieberman, and Ted Selker. "A model of textual affect sensing using real-world knowledge." In *Proceedings of the 8th international conference on Intelligent user interfaces*, pp. 125-132. 2003.
- [18] Sirisuriya, De S. "A comparative study on web scraping." (2015).
- [19] Glez-Peña, Daniel, Anália Lourenço, Hugo López-Fernández, Miguel Reboiro-Jato, and Florentino Fdez-Riverola. "Web scraping technologies in an API world." *Briefings in bioinformatics* 15, no. 5 (2014): 788-797.
- [20] Anand, V., G. Kedar, and A. Gode Shweta. "An Overview On Web Scraping Techniques and Tools." *International Journal on Future Revolution in Computer Science & Communication Engineering* 4 (2018).
- [21] Persson, Emil. "Evaluating tools and techniques for web scraping." (2019).
- [22] Salem, Hamza, and Manuel Mazzara. "Pattern Matching-based scraping of news websites." In *Journal of Physics: Conference Series*, vol. 1694, no. 1, p. 012011. IOP Publishing, 2020.
- [23] Hadley Wickham., RStudio. *Easily Harvest (Scrape) Web Pages*. (2015)
- [24] Harrison, John, and Maintainer John Harrison. "Package 'RSelenium'." (2020).
- [25] Thomas, David Mathew, and Sandeep Mathur. "Data analysis by web scraping using python." In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 450-454. IEEE, 2019.
- [26] Reitz, Kenneth. "Requests: HTTP for Humans—Requests 2.9. 1 documentation." *URL <http://docs.python-requests.org/en/v2.9>* (2015).

- [27] Mitchell, Ryan. *Web scraping with Python: Collecting more data from the modern web*. " O'Reilly Media, Inc.", 2018.
- [28] Scrapy developers. Scrapy Tutorial. Revised in 2020
- [29] Hadley Wickham, RStudio. *stringr, Simple, Consistent Wrappers for Common String Operations*. (2019).
- [30] Chowdhury, Gobinda G. "Natural language processing." *Annual review of information science and technology* 37, no. 1 (2003): 51-89.
- [31] Miller, George A. "WordNet: a lexical database for English." *Communications of the ACM* 38, no. 11 (1995): 39-41.
- [32] Bird, Steven, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.
- [33] Matt Johnson. Lifting the hood on NLTK's NE Chunker. (2016)
- [34] Mount, John. "The equivalence of logistic regression and maximum entropy models." *URL: <http://www.win-vector.com/dfiles/LogisticRegressionMaxEnt.pdf>* (2011).
- [35] Maas, Andrew, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. "Learning word vectors for sentiment analysis." In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142-150. 2011.
- [36] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

- [37] Calefato, Fabio, Filippo Lanubile, and Nicole Novielli. "Emotxt: a toolkit for emotion recognition from text." In *2017 seventh international conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pp. 79-80. IEEE, 2017.
- [38] Ghazi, Diman, Diana Inkpen, and Stan Szpakowicz. "Detecting emotion stimuli in emotion-bearing sentences." In *International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 152-165. Springer, Cham, 2015.
- [39] Li, Yanran, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. "Dailydialog: A manually labelled multi-turn dialogue dataset." *arXiv preprint arXiv:1710.03957* (2017).
- [40] Scherer, Klaus R., and Harald G. Wallbott. "Evidence for universality and cultural variation of differential emotion response patterning." *Journal of personality and social psychology* 66, no. 2 (1994): 310.
- [41] Kleinberg, Bennett. "Measuring emotional responses to COVID-19". World Pandemic Research Network. WPRN-459652, 11/06/2020
- [42] "Dong E, Du H, Gardner L. An interactive web-based dashboard to track COVID-19 in real time. *Lancet Inf Dis.* 20(5):533-534. doi: 10.1016/S1473-3099(20)30120-1"
- [43] Jacob Devlin, Ming-Wei Chang. Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing. 2018
- [44] Dong E, Du H, Gardner L. An interactive web-based dashboard to track COVID-19 in real time. *Lancet Inf Dis.* 20(5):533-534. doi: 10.1016/S1473-3099(20)30120-1"

[45] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed representations of words and phrases and their compositionality." arXiv preprint arXiv:1310.4546 (2013).

[46] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543. 2014.