



UNIVERSITY OF THESSALY  
SCHOOL OF ENGINEERING  
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

## **Mining opinion on COVID-19 based on Twitter Data**

Diploma Thesis

**Athanasios Papanikolaou**

**Supervisor:** Michael Vassilakopoulos

Volos 2021





UNIVERSITY OF THESSALY  
SCHOOL OF ENGINEERING  
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

## **Mining opinion on COVID-19 based on Twitter Data**

### **Diploma Thesis**

**Athanasios Papanikolaou**

**Supervisor:** Michael Vassilakopoulos

Volos 2021





ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**Εξόρυξη γνώμης για την COVID-19  
με βάση δεδομένα του Twitter**

Διπλωματική Εργασία

**Αθανάσιος Παπανικολάου**

**Επιβλέπων:** Μιχαήλ Βασιλακόπουλος

Βόλος 2021



Approved by the Examination Committee:

Supervisor **Michael Vassilakopoulos**

Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **George Thanos**

Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Member **Athanasios Fevgas**

Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly





# Acknowledgements

I would like to thank my parents, because without them I would never be where I am today, as well as my supervisors Associate Professor Michael Vassilakopoulos and Emeritus Professor Elias N. Houstis for the support and guidance they provided me from the selection of my subject to the problems and questions that appeared during the whole process.



## **DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS**

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Athanasios Papanikolaou



# Abstract

In this thesis we explore, through the use of Jupyter Notebook from Anaconda and the programming language Python, the sentiment analysis of tweets made in the English language about the COVID-19 vaccine as well as detect any possible spread of misinformation through those tweets. The goal is to try to understand people's feelings when talking about the new vaccines online and specifically on Twitter and if they spread any misinformation through their posts. The tweets used in this project were downloaded through the API provided by Twitter in a live stream throughout different days and hours in August 2021. This was accomplished through the connection with the MySQL Workbench, which was used for the storage of said tweets, as well as through the connection with the Twitter App. There were two distinct ways used to extract the sentiment of each tweet. The first one was through the use of the TextBlob algorithm and the second was through machine learning and the use of classification models created by various algorithms. Lastly, the same classification models as before were used to predict the truthfulness of said tweets. The results were always split into the worldwide tweets and the tweets made exclusively from the USA, which was the leading country in most tweets created compared to all the rest. All results from each procedure were then compared to each other in order to show any potential differences in the outcomes and the usefulness of each algorithm and process.



# Περίληψη

Σε αυτή τη διατριβή διερευνούμε, μέσω της χρήσης του Jupyter Notebook από την Anaconda και της γλώσσας προγραμματισμού Python, την ανάλυση συναισθημάτων των tweets που έγιναν στην αγγλική γλώσσα σχετικά με το εμβόλιο COVID-19, καθώς και ανιχνεύουμε τυχόν διάδοση παραπληροφόρησης μέσω αυτών των tweets. Σκοπός είναι να προσπαθήσουμε να κατανοήσουμε τα συναισθήματα των ανθρώπων όταν μιλάμε για τα νέα εμβόλια στο διαδίκτυο και συγκεκριμένα στο Twitter και εάν διαδίδουν οποιαδήποτε παραπληροφόρηση μέσω των αναρτήσεών τους. Τα tweets που χρησιμοποιήθηκαν σε αυτό το έργο λήφθηκαν μέσω του API που παρέχεται από το Twitter σε ζωντανή ροή κατά τη διάρκεια διαφόρων ημερών και ωρών τον Αύγουστο του 2021. Αυτό επιτεύχθηκε μέσω της σύνδεσης με το MySQL Workbench, το οποίο χρησιμοποιήθηκε για την αποθήκευση των εν λόγω tweets, καθώς και μέσω της σύνδεσης με την εφαρμογή Twitter. Χρησιμοποιήθηκαν δύο διαφορετικοί τρόποι για να εξαχθεί το συναίσθημα κάθε tweet. Ο πρώτος ήταν μέσω της χρήσης του αλγορίθμου TextBlob και ο δεύτερος ήταν μέσω της μηχανικής μάθησης και της χρήσης μοντέλων κατηγοριοποίησης που δημιουργήθηκαν με διάφορους αλγόριθμους. Τέλος, οι ίδιοι με πριν αλγόριθμοι κατηγοριοποίησης χρησιμοποιήθηκαν για να προβλέψουν την ειλικρίνεια των tweets. Τα αποτελέσματα χωρίζονταν πάντα στα παγκόσμια tweets και στα tweets που αναρτώνταν αποκλειστικά από τις ΗΠΑ, που ήταν η χώρα των περισσότερων tweets που δημιουργήθηκαν σε σύγκριση με όλες τις υπόλοιπες. Όλα τα αποτελέσματα από κάθε διαδικασία συγκρίθηκαν στη συνέχεια μεταξύ τους προκειμένου να δείξουν τυχόν διαφορές στα αποτελέσματα και τη χρησιμότητα κάθε αλγορίθμου και διαδικασίας.





# Table of contents

<b>Acknowledgements</b>	<b>ix</b>
<b>Abstract</b>	<b>xiii</b>
<b>Περίληψη</b>	<b>xv</b>
<b>Table of contents</b>	<b>xvii</b>
<b>List of figures</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 The subject of this thesis . . . . .	2
1.2 Structure of thesis . . . . .	3
<b>2 Related Work</b>	<b>5</b>
2.1 Sentiment Analysis . . . . .	5
2.1.1 Lexicon Based Approach . . . . .	6
2.1.2 Classification Algorithms Based Approach . . . . .	6
2.2 Fake News Detection . . . . .	7
<b>3 Libraries, Connection and Collection</b>	<b>9</b>
3.1 Libraries . . . . .	9
3.2 Connection . . . . .	9
3.3 Tweet Collection . . . . .	10
<b>4 Sentiment Analysis</b>	<b>11</b>
4.1 Calling the notebooks . . . . .	11
4.2 TextBlob . . . . .	11

4.3	Sentiment Analysis . . . . .	13
4.3.1	World Data . . . . .	13
4.3.1.1	Time Series . . . . .	14
4.3.1.2	Natural Language Processing - NLTK . . . . .	15
4.3.2	USA Data . . . . .	17
4.3.3	Comparison . . . . .	21
<b>5</b>	<b>Sentiment Prediction</b>	<b>23</b>
5.1	Importing Datasets . . . . .	23
5.1.1	Pre-processing the Datasets . . . . .	25
5.1.2	TF-IDF . . . . .	27
5.2	Classification Algorithms . . . . .	28
5.2.1	Logistic Regression . . . . .	30
5.2.2	Naive-Bayes Classifier . . . . .	31
5.2.3	Decision Tree Classifier . . . . .	31
5.2.4	Random Forest Classifier . . . . .	31
5.2.5	Support Vector Machine Classifier . . . . .	32
5.2.6	K-Nearest Neighbours Classifier . . . . .	32
5.2.7	Stochastic Gradient Descent Classifier . . . . .	32
5.2.8	Gradient Boosting Classifier . . . . .	33
5.2.9	XGBoost Classifier . . . . .	33
5.2.10	Passive Aggressive Classifier . . . . .	33
5.2.11	LSTM Classifier . . . . .	34
5.2.11.1	Tokenizer . . . . .	34
5.2.11.2	LSTM Model . . . . .	35
5.2.12	Accuracy Results . . . . .	37
5.3	Sentiment Prediction . . . . .	38
5.3.1	World Data . . . . .	39
5.3.2	USA Data . . . . .	40
5.3.3	Comparison . . . . .	41
5.4	Extra Notes . . . . .	42

<b>6</b>	<b>Fake News Detection</b>	<b>45</b>
6.1	Importing Datasets . . . . .	45
6.2	Pre-processing and TF-IDF . . . . .	47
6.3	Classification Algorithms . . . . .	47
6.3.1	LSTM Model . . . . .	47
6.3.2	Accuracy Results . . . . .	48
6.4	Fake News Prediction . . . . .	49
6.4.1	World Data . . . . .	50
6.4.2	USA Data . . . . .	51
6.4.3	Comparison . . . . .	52
<b>7</b>	<b>Results, Conclusions and Future Extensions</b>	<b>57</b>
7.1	Results and Conclusions . . . . .	57
7.1.1	Chapter 4 . . . . .	57
7.1.2	Chapter 5 . . . . .	58
7.1.3	Chapter 6 . . . . .	59
7.2	Future Extensions . . . . .	59
7.2.1	Further Improvement . . . . .	59
7.2.1.1	Sentiment Analysis . . . . .	60
7.2.1.2	Fake News Detection . . . . .	60
7.2.2	Complete Evolution . . . . .	61
7.2.2.1	Sentiment Analysis . . . . .	61
7.2.2.2	Fake News Detection . . . . .	61
	<b>Bibliography</b>	<b>63</b>
	<b>Appendix</b>	
	<b>GitHub</b>	<b>65</b>
1	Contents . . . . .	66
2	Final Note . . . . .	69



# List of figures

4.1	Sample of 50000 Tweets . . . . .	13
4.2	Time Series Graph - World . . . . .	14
4.3	Polarity Pie - World . . . . .	15
4.4	WordCloud - World . . . . .	16
4.5	Word Frequency Bar Plot - World . . . . .	17
4.6	Sample of 14227 Tweets - USA . . . . .	18
4.7	Time Series Graph - USA . . . . .	18
4.8	Polarity Pie - USA . . . . .	19
4.9	WordCloud - USA . . . . .	19
4.10	Word Frequency Bar Plot - USA . . . . .	20
4.11	USA Tweet Map . . . . .	20
4.12	All Plots Together . . . . .	21
5.1	New Dataset - Polarity . . . . .	25
5.2	Polarity Pie - Datasets . . . . .	26
5.3	Polarity Pie - Datasets . . . . .	26
5.4	Confusion Matrix 2 x 2 . . . . .	29
5.5	Confusion Matrix 3 x 3 . . . . .	30
5.6	LSTM Model . . . . .	35
5.7	LSTM Loss . . . . .	36
5.8	LSTM Accuracy . . . . .	36
5.9	Algorithm Accuracy . . . . .	37
5.10	Time Series Graph - World 2 . . . . .	39
5.11	Polarity Pie - World 2 . . . . .	39
5.12	Time Series Graph - USA 2 . . . . .	40

5.13	Polarity Pie - USA 2	40
5.14	All Plots Together 2	41
6.1	New Dataset - News	46
6.2	Truth Pie	46
6.3	LSTM Loss 2	47
6.4	LSTM Accuracy 2	48
6.5	Algorithm Accuracy 2	48
6.6	Time Series Graph - World 3	50
6.7	Polarity Pie - World 3	50
6.8	Time Series Graph - USA 3	51
6.9	Polarity Pie - USA 3	51
6.10	All Plots Together 3	54
.1	All Files	65

# Chapter 1

## Introduction

COVID-19 and the lockdown it created are arguably the worst things that happened in the recent years of history on a global scale. From the deaths occurred and the economic crisis most countries had to face, to the mental problems created by the isolation, everyone got affected in some way. Today though we have a fighting chance against the virus with the help of the plentiful vaccines. After months of lock-downs, strict rules, 210 million cases of infection of which 4.41 million were deaths and plenty of cases that did not result in death but left deep scars and lasting problems, we have the chance to live properly and without fear once again.

Albeit the vaccine is a blessing, there are a lot of people being afraid of it or even completely denying it. Twitter, being considerably the best place to publish an opinion, is stormed by multitude tweets concerning the vaccines either promoting or denying it. In this crucial time period the people are not united as one. Everyone is baffled by this turn of events and both sides are asking why the rest do not agree with them as well as ask why there is a spread of so much misinformation from their point of view. In reality this "battlefield" is not beneficial for anyone.

However, this internet "war" is not totally pointless. Through everyone's tweets crucial data can be collected and analyzed in a multitude of ways showing great results. Notably, the most famous one is the classic sentiment analysis, closely followed by the upcoming fake news detection. For the world of data science these two topics are highly valuable and have already been used in countless text classification topics as well as for this "war" about the new vaccines, exactly what we will study in this thesis.

## 1.1 The subject of this thesis

In this project, which is run entirely on Python in Jupyter Notebook from Anaconda in Python, we will tackle exactly the same area of interest but first we have to collect some tweets. The tweets used in this project are all collected through the API of Twitter throughout August 2021 from all around the world in the English language and all containing the mention of COVID-19 vaccines. All these tweets are first stored in MySQL Workbench and later saved in csv files for later use but with the option to also further collect new tweets and store them in new databases or existing ones. After collecting the needed information from the tweets such as the user's location and id, the time of creation of the tweet and of course its text, we proceed to the "cleaning" of unnecessary characters and finally to the data analysis of them.

The first topic as mentioned above will be the sentiment analysis of everyone's tweets with two distinct ways. The first and albeit easiest way is through the use of the TextBlob algorithm, which targets each word from the tweet and appoints them a neutral, positive or negative value so to give the complete sentence the final neutral, positive or negative emotion. This is done by comparing the words of the text with the words stored in the dictionary of the algorithm. The second way of extracting the tweet's sentiment is done using machine learning. Many training models of classification algorithms, such as the Random Forest Classifier for example, are trained using datasets full of texts that were valued as neutral, positive or negative by human hands. The model with the best accuracy is then used to predict the sentiment of our collected tweets once again.

The second topic is the detection of misinformation by using the same classification algorithms as in the previous process and using different datasets containing texts valued as fake or true, valued again by human hands. The model with the best accuracy again proceeds to predict the truthfulness of our tweets and show all the possible fake news.

Through these methods we collect the information about how people feel in their discussions about the vaccines in the internet as well as see who spreads false information and how these fake information affect the overall emotions.



## 1.2 Structure of thesis

The following chapters will touch on the mentioned topics one by one and explain all the steps taken as well as what the results mean.

Starting by chapter 2, we will explain the current scientific situation on our topics and do a literature review.

Chapter 3 will show the processes of connecting both with MySQL Workbench and the Twitter App and the process of collecting the tweets will be thoroughly explained as well as the importing of all necessary libraries.

Chapters 4, 5 and 6 on the other hand will explain the sentiment analysis using TextBlob, the sentiment prediction through classification algorithms and the prediction of the truthfulness through the the same classification algorithms respectively. Each procedure produces a number of results which are divided in two categories, the worldwide tweets and the tweets created solely from USA citizens. The results are then shown through many graphs and even compared to each other to understand the effect of every algorithm and process.



# Chapter 2

## Related Work

As described in the introduction, we will specifically focus on two main topics, the sentiment analysis of tweets talking about the COVID-19 vaccines with two distinct ways, as well as investigate if these tweets contain any misinformation.

### 2.1 Sentiment Analysis

Sentiment analysis (also known as opinion mining or emotion AI) is the use of natural language processing, text analysis, computational linguistics, and bio-metrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

The main task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as enjoyment, anger, disgust, sadness, fear, and surprise.

In this project we will be tasked with the more basic polarity based approach and categorize our tweets as negative, positive or neutral. We will use both a lexicon based approach and a classification algorithms one, which will both be explained in greater details in the next two chapters.

### 2.1.1 Lexicon Based Approach

The lexicon based approach will use the TextBlob algorithm to extract the polarity of our tweets, which are all downloaded through Tweepy, are no retweets and always mention the vaccines. A similar approach is also taken by [1] and [2] which both use various lexicon based algorithms, such as OpinionFinder Lexicon, VADER and many more.

Having taken the polarity of the tweets' texts, we use the dataset, which also contains more information from the tweets such as the user's id, to show some important pieces of information through graphs just like [3] and [4], e.g. a time series graph that shows the amount of negative, positive and neutral tweets through the many hours they were created and downloaded. These graphs and information will be used again to compare it with the results of the second approach that uses classification algorithms instead of TextBlob.

### 2.1.2 Classification Algorithms Based Approach

On the other hand, the second approach does not use any lexicon but instead trains ten classification algorithms and one LSTM model to predict the polarity of said tweets, as [5] did in their own research using the Naive-Bayes algorithm, albeit for tweets made from some US presidents during the election procedure period.

The classification algorithms vary and use different ways to tackle the same problems, from linear separation to using trees but the LSTM model is the only one that is truly different compared to the others both in its setup but also how it operates and trains. Generally, neural networks have become more and more famous and are used more often in classification problems nowadays due to their incredible accuracy and ability to process data, especially unstructured data like text. Many researches, including [6] have used neural networks for sentiment analysis, just like us, but instead of using recurrent neural networks such as the LSTM, they have also used convolutional neural networks. Both ideas can be used for the same task but there are many differences between them resulting into different outputs.

The input data used to train these models will be various texts, specifically tweets, that have been labeled with the same three basic polarity values. In contrast, [7] does not use just the text of the tweets but also their hashtags as they think that they convey important information as well for the polarity of the tweets. On top of that, [7] does not use just classification algorithms (SVM) but also graph based algorithms.

The texts though are not used just like that, they are transformed into numerical arrays

through the TF-IDF vectorizer, just like with the lexicon based approach for some specific graphs.

In our LSTM model however, the texts are transformed once again through a different procedure, exactly as in [8]' work. The data is again transformed into vectors but in a different way only for the different than the rest LSTM model.

After everything is trained, we can use any model to predict and get the polarity of our own tweets again, plot again some graphs and as already described, compare it with the lexicon based approach's results.

## 2.2 Fake News Detection

With the advancement of technology, digital news is more widely exposed to users globally and contributes to the increment of spreading hoaxes and disinformation online. Fake news can be found through popular platforms such as social media and the Internet. There have been multiple solutions and efforts in the detection of fake news where it even works with artificial intelligence tools. However, fake news intends to convince the reader to believe false information which deems these articles difficult to perceive. The rate of producing digital news is large and quick, running daily at every second, thus it is challenging for machine learning to effectively detect fake news.

In this project we will try to find out if our mentioned tweets contain false information. As we did with the previous process, we will use classification algorithms and an LSTM model, train them with some datasets and use them to predict the truthfulness of our tweets. The datasets are again texts that have been labelled as fake or true by humans.

Even though plain texts with a true or fake label can work in our occasion, this is not the only data we can feed into the algorithms. [9] and [10] have shown through their work that we can extract far more information from these articles/news and use them in order to extract better results and accuracy. These extra features of knowledge can be anything from the location the news was created from to more complex ones such as psycholinguistic cues, subjectivity, bias, language structures, where the news is intended to be published at, such as only in a social media app, and many more. All this extra info can create a bigger picture of what is potentially fake and why we value it automatically as fake or not.

Once again the texts are of course transformed with TF-IDF vectorizer for all the algo-

rithms and with another way for the LSTM.

However these vectorizers are not the only way to transform our text data. As shown by [11] with their MVAE: Multimodal Variational Autoencoder and [12] with their BERT encoder or even with other examples such as the combination of PyCaret and the Universal Sentence Encoder, these different ways of transforming our data have been proven very effective and useful. They work in different ways and understand better the significance of words for detecting fake news.

In the end, when the models are trained once more, we can use any of them to get the final verdict for each tweet, delete all fake valued tweets and see if the polarity taken from the previous approach changed percentagely.

Despite all this procedure though there can be other ways to tackle this problem and one of them is as [13] suggested. Instead of using the classic algorithms, as in the previous procedure, they created and used their own unique algorithm which is graph based consisting of three phases, the "Label Seeding using Bi-cliques", the "Label Spreading within Bi-cliques" and lastly the "Full Dataset Labelling".

# Chapter 3

## Libraries, Connection and Collection

### 3.1 Libraries

The notebook "Libraries" contains all the libraries needed in the following notebooks. These libraries are imported in each notebook through the use of the "% run" command for the purpose of saving space in each call in each notebook, thus it will not be mentioned again when describing and explaining the rest of the notebooks because it is always called at the start of the notebook.

### 3.2 Connection

The first step of the project is the connection with the appropriate tools, MySQL Workbench and the Twitter App, inside the "Connecting" notebook.

The first connection is with the Twitter App where all credentials are already in place and the connection begins automatically.

The user is then asked at first to either connect with MySQL Workbench or use an existing csv file with pre-downloaded tweets. In case the user wants to connect with the Workbench, a series of questions begins. The first questions are about his credentials and if they are correct and the Workbench is properly running in the background, he will be connected.

The next questions revolve around how he wants to access the Workbench. There are choices to access existing databases or create new ones, to access existing tables or create new ones and of course if he wants to collect new tweets and add them to an existing database and table or to a new set or even access the pre-downloaded tweets from an existing database

and table.

In the case of collecting new tweets, the collecting procedure will begin and he is free to end it at his own discretion, proceeding then on the next step. On the other hand he can also choose a set of an existing database and table that contain tweets and do not wish to further collect more tweets. Lastly if he chooses to not use the Workbench at all but rather upload a csv file containing the pre-downloaded tweets, he is asked to choose one csv file from the directory and after that he proceeds again to the next step, which is the sentiment analysis described in chapter 4 and chapter 5 with two different and distinct procedures.

In the event of bad inputs in any of the previously mentioned questions the "chatbot" does not terminate but rather ask again until an acceptable answer is given.

### 3.3 Tweet Collection

In the event that the user connects to the Workbench and decides to collect new tweets, adding them in existing databases and tables or new ones, the collection process is triggered from the "Collecting" dataset.

Through the use of Tweepy we are able to live stream tweets with the parameters of our choosing. In this project we have selected to stream tweets written only in the English language, that contain the keyword "vaccine" and that are not retweets of other tweets.

With every tweet downloaded only some features are saved, more specifically the user's id and location, the time the tweet was created as well as the tweet's text. Using the algorithm TextBlob the sentiment and subsequently the polarity of the tweet is also extracted and are all together imported in our selected table and database that we selected in the previous step.

The connecting and collecting procedures are called from the "Sentiment Analysis" notebook, which will be explained in chapter 4, again with the help of the "% run" command.



# Chapter 4

## Sentiment Analysis

### 4.1 Calling the notebooks

Before of starting the main analysis procedure the notebooks "Connecting" and "Collecting" are called. This is where all the previous questions are actually being asked and answered. After answering everything the user either uploads an existing csv or is connected to a database and a table full of tweets. In each case a dataframe object is being created that includes all the tweets and all the features that were extracted before.

One last question is asked and it is about if the user wishes to use a smaller amount of tweets instead of all the data collected. This is asked because the procedures that are about to follow are sometimes gpu and ram consuming slowing down the computer or even stopping the whole process in certain cases. A good and easily manageable number is around 50000 tweets. It gives accurate results representing the whole dataset and the computer runs smoothly with this amount of data finishing all processes in about 10 hours in my case.

However, before moving on we have to understand what TextBlob is exactly and how it works. It is very important to understand these concepts as they affect dramatically the way the tweets are labeled as neutral, positive or negative.

### 4.2 TextBlob

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

project we use only its sentiment analysis part. The sentiment property returns a named tuple of the form `Sentiment(polarity, subjectivity)`. The polarity score is a float within the range  $[-1.0, 1.0]$ . The subjectivity is a float within the range  $[0.0, 1.0]$  where 0.0 is very objective and 1.0 is very subjective. In our case we only need the polarity as it is considered the emotion of the text.

In order though for this polarity number to be extracted, `TextBlob` compares each word from our tweets with its internal dictionary and values it with some numbers about neutrality, positivity or negativity. In the end of the tweet, a final result is being given after taking account the individual polarity of each word read. Then all tweets with polarities within the range  $(-0.5, 0.5)$  are considered neutral and are valued as 0 and all that are outside of that range, meaning all tweets in  $[-1.0, -0.5)$  and  $(0.5, 1.0]$  are considered negative and positive and are valued as -1 and 1 respectively.

This procedure however hides two major threats in plain sight. The first and maybe most easily observed is that a lot of words and subsequently whole tweets are valued wrongfully and this happens because the comparison between our words and the dictionary's words is not perfect. A simple sentence such as "I am dying from exhaustion." can be perceived easily by humans as something negative and a figure of speech in fact. However, the `TextBlob` algorithm values it exactly as (0.0) in terms of polarity, otherwise known as a true neutral, which is totally contradictory with a human's "sentiment analysis". This is the reason why a lot of words and tweets can be characterized as neutral but in reality be positive and negative. As far as we are concerned about positive and negative tweets, most of them are actually positive and negative, the only problem is with neutral tweets.

The second and probably the most misunderstood problem with this algorithm is what exactly we measure the sentiment of. A lot of readers could think that we measure the people's sentiment towards or against the vaccine but in reality it is not like that. The algorithm can only try to understand the sentiment of the tweet and not the sentiment of the tweet towards or against a topic, for example a tweet such as "I hate people that do not get vaccinated." can be easily analyzed by a human as positive towards the vaccine, because of the double negative, meaning that the user is negative about people being negative towards the vaccine, so he is in the end positive about it. In reality though the algorithm just understands a negative emotion devoid of any context or topic related. My project is not about understanding people's opinion towards the vaccine but actually measuring the sentiments in this internet "war" regarding

this topic, the sentiments being sent and received back and forth in Twitter and understanding if it is true that the users are "fighting" each other or actually agreeing in some kind of unison, albeit negative or positive towards our topic.

## 4.3 Sentiment Analysis

Having understood exactly how we do the connections, the collection of tweets and how TextBlob values its sentiments we can move on with the main procedure.

### 4.3.1 World Data

In our case, instead of connecting again with MySQL and downloading tweets, we will use a csv file called "mytweets.csv" which contains 500000 tweets all downloaded throughout August 2021 in different days and hours from all over the world and a sample of 50000 tweets is taken from it to be worked on, as seen bellow.

	id_str		text	created_at	polarity	user_location
314879	1425496491330842624	@Shawn_on_Games @nypost	Do you not realize the...	2021-08-11 16:37:02	0	Orange Park, FL
314880	1425496492685598725		Get ready!!! FAUCI warns FULL VACCINE MANDAT...	2021-08-11 16:37:02	0	Michigan, USA
314881	1425496493667098625		it's one thing to get the vaccine voluntarily,...	2021-08-11 16:37:03	0	Toronto, Ontario
314882	1425496493830643716	@ScruffyVandal @Justasnowmexic1 @JiQed @Newswe...		2021-08-11 16:37:03	1	NaN
314883	1425496494065528835		I can't believe we continue to fall into media...	2021-08-11 16:37:03	0	NaN
...	...	...	...	...	...	...
364874	1425862484397854725		If you decide to get vaccinated against COVID ...	2021-08-12 16:51:22	0	Lutz, FL
364875	1425862486830501888	@DavidRF34 @archerbandrodie @realTuckFrumper H...		2021-08-12 16:51:22	1	Born in the USA
364876	1425862488277520384		Of course the media is silent. WellIn their de...	2021-08-12 16:51:23	0	NaN
364877	1425862488285921289	@kayleighmcenany	Florida radio host hospitaliz...	2021-08-12 16:51:23	0	NaN
364878	1425862489405902851		We are getting closer to actual science. We k...	2021-08-12 16:51:23	0	KDIX

50000 rows × 5 columns

Figure 4.1: Sample of 50000 Tweets

Table 4.1 is a dataset with a collection of 50000 tweets taken randomly from the 500000 but all 50000 instances are adjacent to each other and not completely random. The sample is needed to be of contiguous tweets because of their time of creation. In the event that totally random tweets are selected, the graphs that will be discussed below would not function properly and misrepresent our information, that is why the tweets selected in this instance are between the numbers 314879 and 363878 and not randomly selected one by one.

We use a random sample of the dataset because we want to create the illusion of the live stream of data. If a user left the procedure of collecting tweets to run for some hours he could get a good amount of new just created tweets and execute the rest of the program and in some hours get all the results. Instead of collecting new tweets every time we just want to execute our program, we use the csv file of pre-downloaded tweets and select some of them randomly as if we just downloaded them. This randomness will be later shown that it affects greatly the results of some procedures, as if we used new tweets every time and got new results every time.

#### 4.3.1.1 Time Series

The first and most important task of all, the sentiment analysis, is to show the emotions of the collected tweets throughout the time they were created. The first step is to convert the time of creation in "datetime" values as well as rename the columns for easier explanation purposes. We then convert the entire dataset into groups of two seconds and count the number of sentiment for each kind of polarity, -1 for the negative, 0 for the neutral and 1 for the positive, in each time-interval group. We apply unstack-stack technology to make sure all categories in each group are displayed even when some of categories may not have any value, meaning that no tweets had this sentiment at that time frame.

In simpler terms, we divide all the time there is from the dataset into smaller two second groups grouped by the three sentiments. What we achieve is the display of all three emotions in a graph with the x axis being the time and the y axis being the number of emotions the tweets were valued with at each time instance.

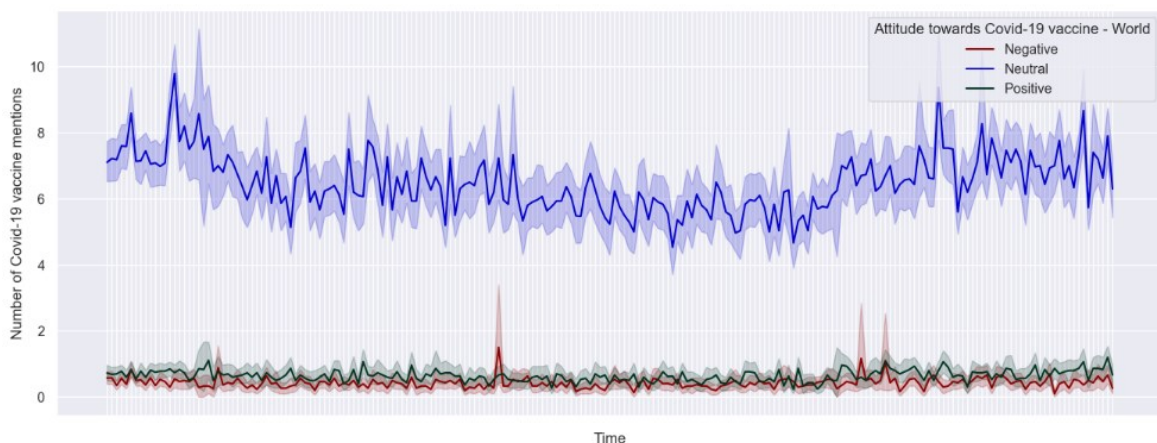


Figure 4.2: Time Series Graph - World

It is clearly obvious from Figure 4.2 that the TextBlob algorithm has deemed almost all tweets to be neutral, fewer to be positive and almost none of them negative. To be ever more accurate, here are shown exactly the numbers of each emotion in our 50000 tweets sample.

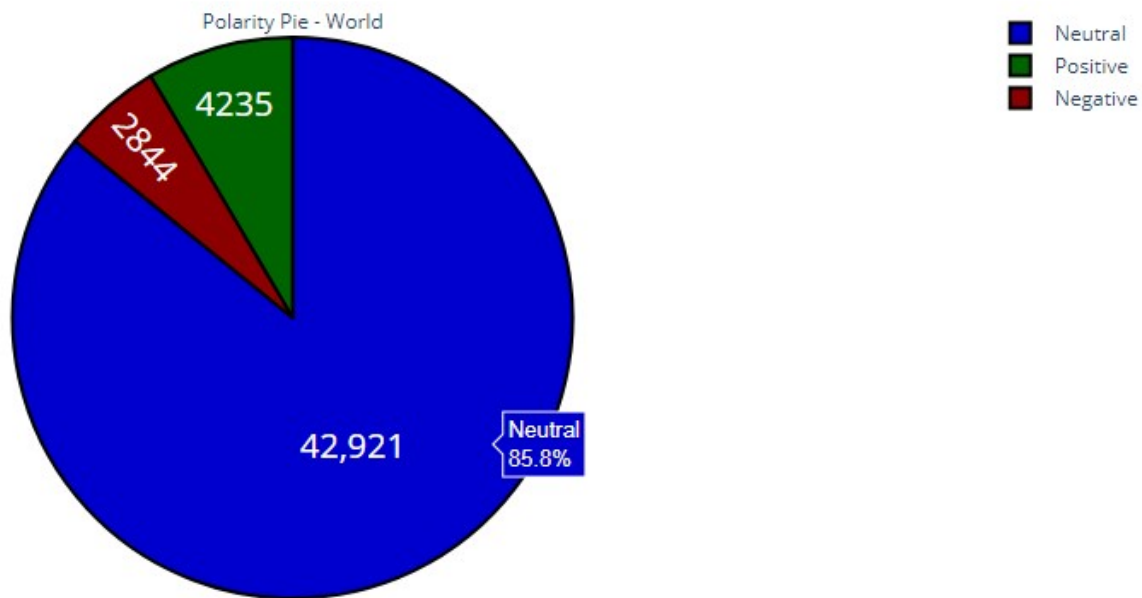


Figure 4.3: Polarity Pie - World

As Figure 4.3 shows, 85.8% of all tweets are considered neutral, 8.47% positive and only 5.69% negative. These percentages were the same in multiple runs of the program and represent all the 500000 tweets dataset. A multitude of samples were used, from different sizes, such as 50000 tweets and from different parts of the original 500000 collection. This however is very different from reality. As [2] states in their own research, the neutral category accounted for the 41% of the tweets, followed by the positive category accounting for 34% and negative category accounting for 25% and by using our basic logic it should be unrealistic to have this kind of one side dominance from neutral tweets. This is however the reason we will explore the same concept from a different angle and perspective and analyze it in chapter 5.

#### 4.3.1.2 Natural Language Processing - NLTK

Continuing our experiment we are going to use the famous NLTK in order to track words with Natural Language Processing (NLP).

NLTK is a leading platform for building Python programs to work with human language

data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

With the use of the NLTK tool set and the RE library we manage to "clean" the tweets by removing unnecessary characters and lowering all letters. In that way we are able to track the words used more often and present them in two different ways.

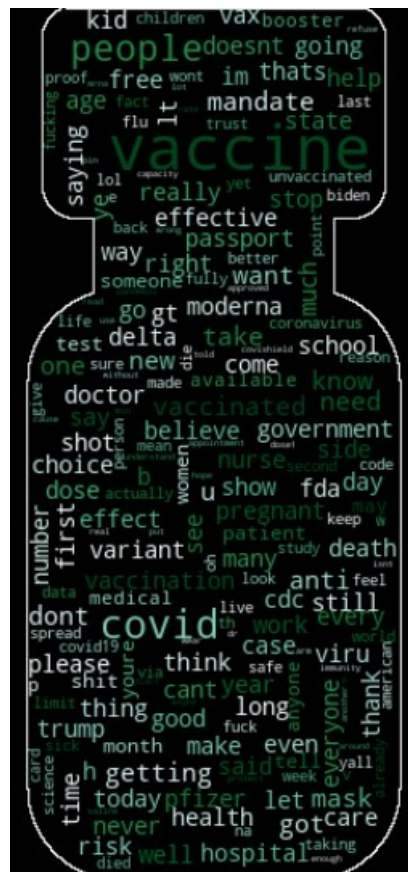


Figure 4.4: WordCloud - World

Figure 4.4 is known as a wordcloud. Inside the vaccine vial shape are shown the most frequently used words in the tweet sample, with the bigger clearer words representing a bigger frequency compared to the smaller blurrier ones.

In a less chaotic and more organized manner we can see specifically the top ten most used words and exactly how many times they were referenced in all the tweets.

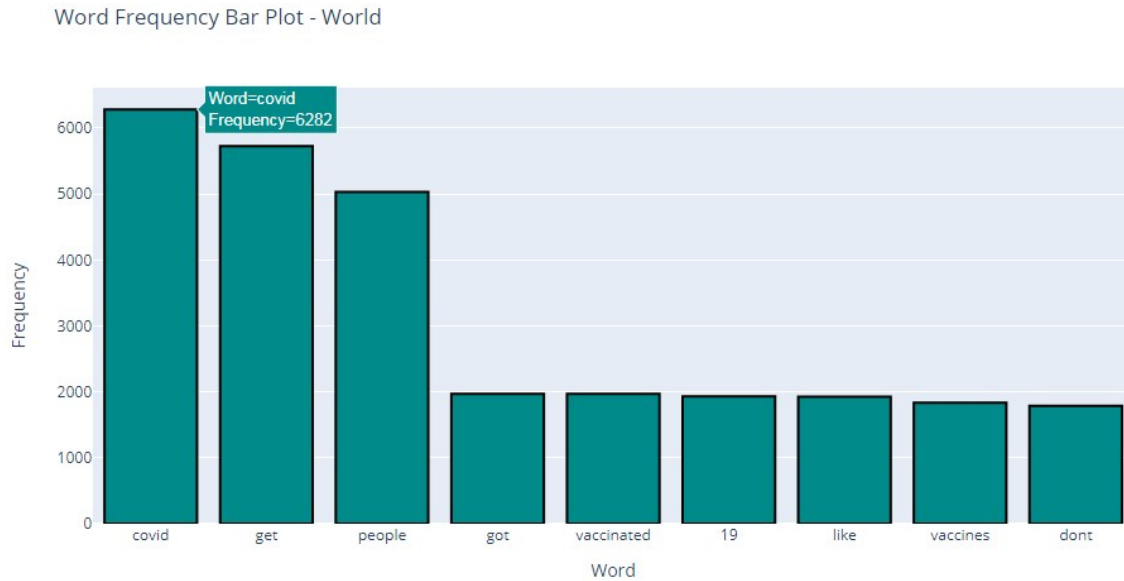


Figure 4.5: Word Frequency Bar Plot - World

As we can see from Figure 4.5, the most frequent words are actually what we would expect to see such as covid, vaccinated etc.

### 4.3.2 USA Data

As mentioned before, the 50000 tweets from the sample and of course the 500000 tweets from the csv file are collected from all around the world in the English language. A considerable portion of them however are created from USA citizens and more specifically around 30% and more of the total 500000 tweets.

The way to achieve this separation between USA made tweets and from anywhere else is by using the feature "user location" from our dataset. This location column contains the locations Twitter had for the user of each tweets. This location sometimes is an empty cell, contains something random the user put himself such as "my house" or it contains an actual location which can be a city, a country or even their combination e.g. "Paris, France".

Because of this we create a dictionary containing all the united states such as "NY, New York" and by going through each tweet we try to match at least one of the two, the full state name or its abbreviation. The rest of the tweets are dropped entirely. In this way we create a new dataframe object which contains only USA made tweets that can be traced back to a specific state.



	id_str		text	created_at	polarity	user_location
314879	1425496491330842624	@Shawn_on_Games @nypost	Do you not realize the...	2021-08-11 16:37:02	0	Orange Park, FL
314880	1425496492685598725		Get ready!!! FAUCI warns FULL VACCINE MANDAT...	2021-08-11 16:37:02	0	Michigan, USA
314885	1425496496825266182		: The investigation into Andrew Cuomo revealed...	2021-08-11 16:37:03	0	NYC
314888	1425496498964361216	@imillhiser @mcelhearn	Yes. The government has...	2021-08-11 16:37:04	0	El Cerrito, CA
314893	1425496502152040449		MOTB\nEndTimes	2021-08-11 16:37:05	0	New Mexico USA
...	...	...	...	...	...	...
364857	1425862473022939147	@asbell_tony @cachenca	No, you said he flip fl...	2021-08-12 16:51:19	0	Edmond, OK
364861	1425862474386087944		But mask wearing, hand washing, and physical d...	2021-08-12 16:51:19	0	EARTH
364863	1425862476491546632		If we could hurry up and make vaccines availab...	2021-08-12 16:51:20	0	Washington, DC
364870	1425862482971799566		I actually wouldnt have been able to be around...	2021-08-12 16:51:21	0	Washington, DC
364874	1425862484397854725		If you decide to get vaccinated against COVID ...	2021-08-12 16:51:22	0	Lutz, FL

14227 rows × 5 columns

Figure 4.6: Sample of 14227 Tweets - USA

The new dataset shown in Table 4.6 contains only 14227 tweets out of the starting 50000 of our sample, which makes it exactly 28.45% of the original, more that a quarter and as mentioned above these tweets are the ones we can surely identify as US made. Potentially there are even more tweets that are made from the US but have a non usable location input and could not be matched to any of the states from our dictionary, making the overall percentage rise even higher quite possibly even to one third of the dataset.

After this matching procedure we once again produce the same figures and graphs in an attempt to see if the US is different in any way with the rest of the world.

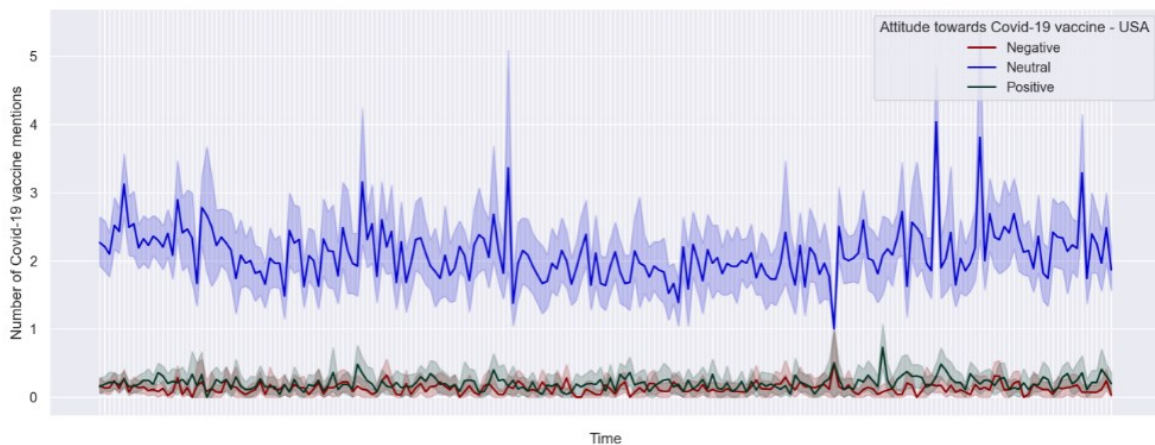


Figure 4.7: Time Series Graph - USA



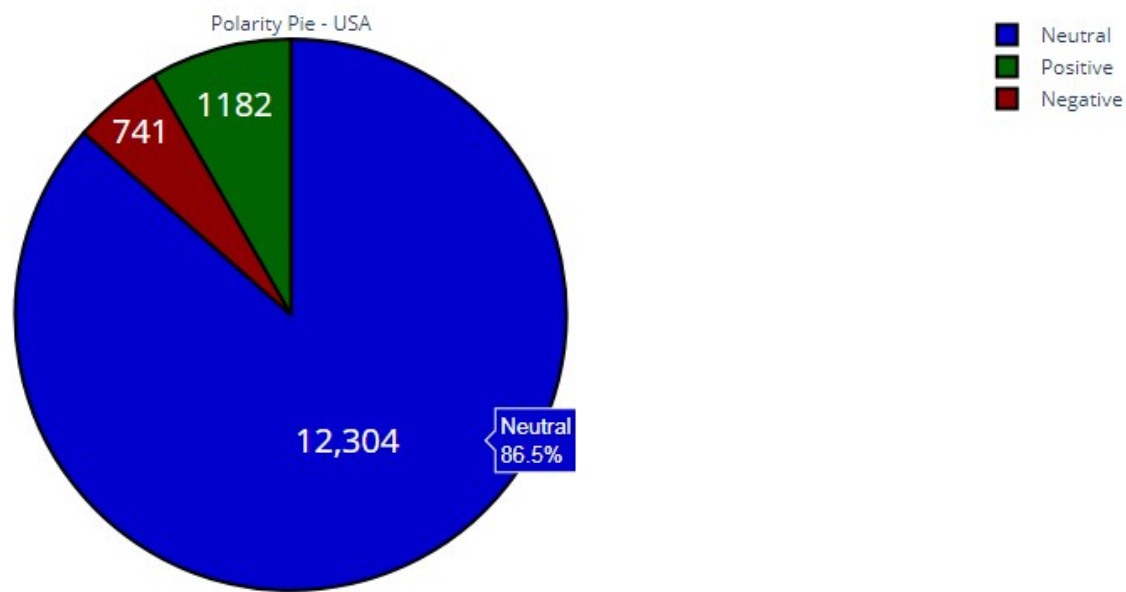


Figure 4.8: Polarity Pie - USA

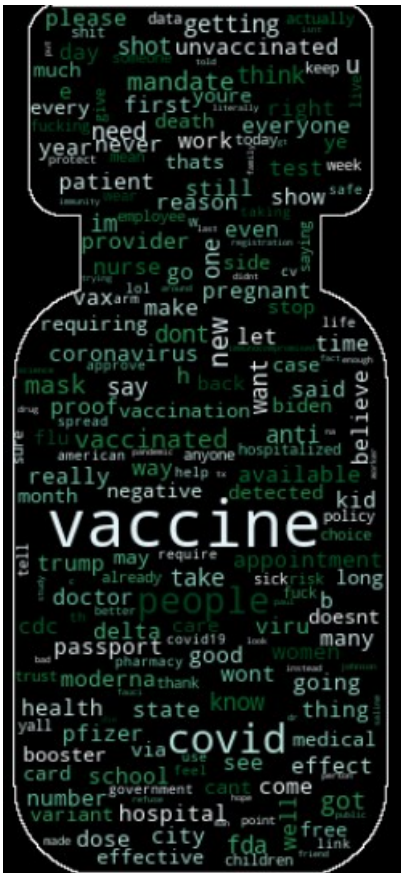


Figure 4.9: WordCloud - USA

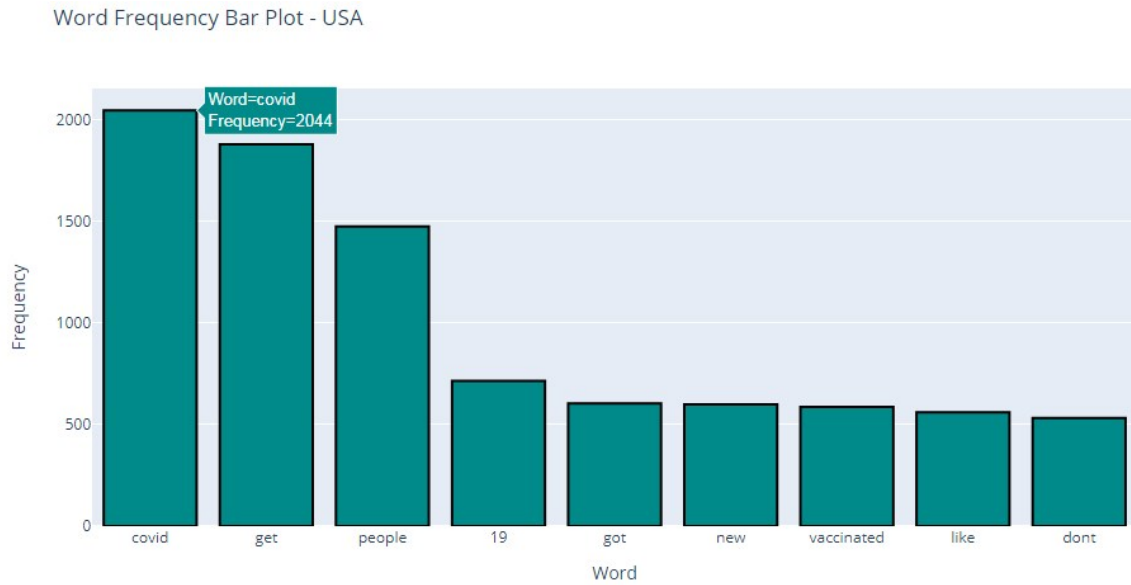


Figure 4.10: Word Frequency Bar Plot - USA

On top of that we also create an interactive USA map showing how many tweets were created per state.

USA Tweet Map

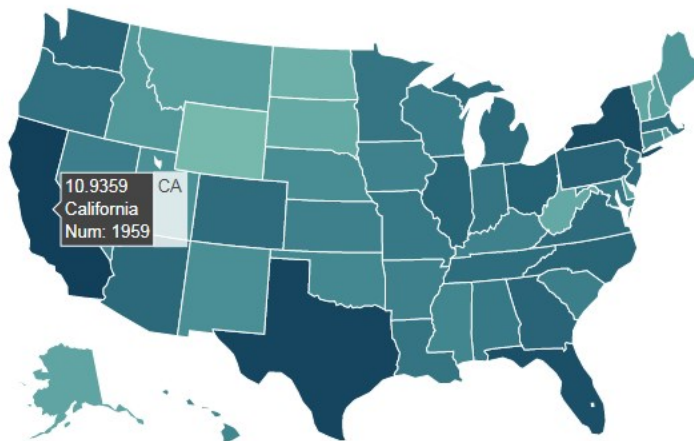


Figure 4.11: USA Tweet Map

Figure 4.11 is a map of USA colored in a way that the deeper the color the more tweets were created in that state and vice versa. As imagined the states with the most tweet creating contribution are the states of California, Texas, Florida and New York.

### 4.3.3 Comparison

In order to achieve maximum efficiency when we compare each graph to each other, we put them all together side by side in one massive plot.

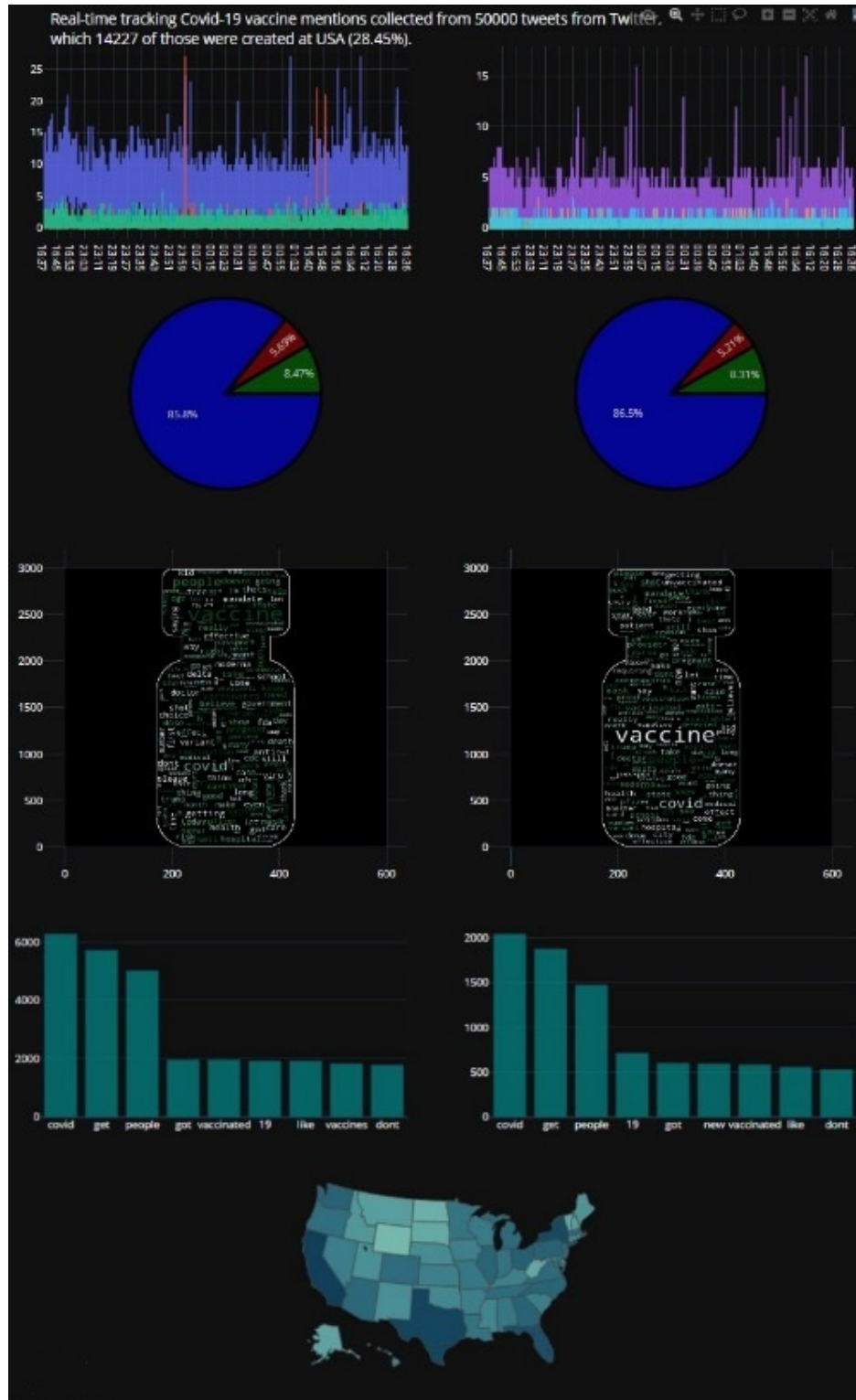


Figure 4.12: All Plots Together

In Figure 4.12 everything on the left is data acquired from all the tweets whereas on the right are only from the US. The differences are in fact negligible. All the graphs are almost identical to each other even to the finest details. This means that the US is not only a very active Twitter user, at least in the English language, but also almost identical to the rest of the world, at least in this topic.

Even when the US tweets were removed from the starting dataset, the percentages and numbers were once again very similar to each other. It is confirmed, at least in this experiment, that the US and the rest of the world are thinking and posting in very similar ways, at least emotionally.

# Chapter 5

## Sentiment Prediction

In the "Sentiment Analysis" notebook we experimented with our tweets having taken the polarity of them from TextBlob. However we quickly understood that the numbers must be wrong at least partially. In this chapter we will explore the "Sentiment Prediction" notebook which will show us another more sophisticated way of extracting the sentiment of the tweets and finally compare the two methods with each other.

The first step is to open the "store". Throughout the previous notebook we saved in the "store" some very important python objects with the use of the "%store" command. This command allowed us to store our sample of tweets, the time series data and pie data for both the world and USA counterparts as well as the states dictionary. This storing process was done in order to use these stored information in the next notebook and plot the necessary graphs to draw the comparisons between the two methods.

The only thing to do in the other notebook is to "open the store" with the command "%store -r", which allows us to use freely everything that was previously stored. The first that needs to be immediately used is a copy of the sample of tweets we were using in the previous notebook. This copy will be filled with the new polarity values and then be compared with the original sample which has the polarity values from TextBlob.

### 5.1 Importing Datasets

As described before we will use many classification algorithms in order to get the polarity of our tweets and to do that we need to train the models of these algorithms and for that we need datasets with any kind of text that has been evaluated as neutral, positive or negative

from humans. We need the training models to understand which words and what sentences give the three emotions and try to use that training to predict the sentiment from new texts, meaning our tweets.

Generally the more data we feed into the models the better the training will be in the end. However there are two problems. The first one is the overall sentiments of the combined texts from all the datasets. The models should be balanced in all three emotions, otherwise there is a high chance that one emotion can dominate in the prediction phase due to the overexpose of it during the training phase and that is something we do not want. That is the reason why we try to generally have some kind of balance with all three of them. The second one is the quantity of the data. Unsurprisingly once again the more data we use the harder it is for the computer to process it. In the best scenario we could use hundreds of thousands of instances to create the perfect model but this is unfortunately not possible with most everyday computers and that is why we will be limited with how many texts we will use.

The datasets we are gonna import and use for the training models are four.

1. The first one is called [Twitter US Airline Sentiment](#) and is about tweets addressed to US airlines commenting about the passenger's experience.
2. The second one is [Coachella 2015 Twitter](#) and is about tweets commenting about the user's experience at the Coachella festival.
3. The third one is [Coronavirus tweets NLP](#) and is about tweets commenting about COVID-19.
4. Last but not least the fourth one is [The Social Dilemma Tweets](#) and is about tweets commenting on the Social Dilemma documentary.

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	16-03-2020	@MeNyrble @Phil_Gahan @Chrisitv https://t.co/i...	Neutral
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	Positive
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV...	Extremely Negative
...	...	...	...	...	...	...
41152	44951	89903	Wellington City, New Zealand	14-04-2020	Airline pilots offering to stock supermarket s...	Neutral
41153	44952	89904	NaN	14-04-2020	Response to complaint not provided citing COVI...	Extremely Negative
41154	44953	89905	NaN	14-04-2020	You know it's getting tough when @KameronWilds...	Positive
41155	44954	89906	NaN	14-04-2020	Is it wrong that the smell of hand sanitizer i...	Neutral
41156	44955	89907	i love you so much    he/him	14-04-2020	@TartiiCat Well new/used Rift S are going for ...	Negative

41157 rows × 6 columns

Figure 5.1: New Dataset - Polarity

Table 5.1 is one example of how these datasets are.

### 5.1.1 Pre-processing the Datasets

The datasets at this point need some pre-processing. We go through every dataframe and delete any possible retweets and tweak the sentiment columns to contain only neutral, negative or positive values because some datasets may contain sentiment values such as "Extremely Negative", which of course cannot be used in that state.

After that we collect all the texts and all the sentiments into two separate lists to be used later in the training models. We do the same "cleaning" for our tweets and put them their texts in their own list. Let's first see the amount of instances per emotion.

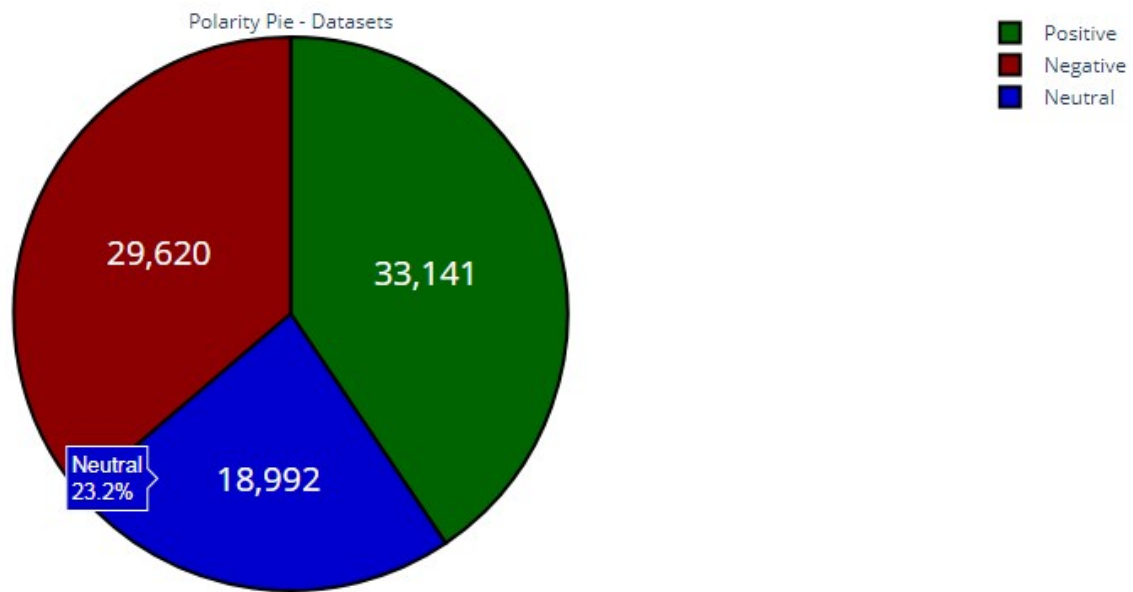


Figure 5.2: Polarity Pie - Datasets

The pie of Figure 5.2 shows a noticeable imbalance in the emotions. As mentioned above, this sort of imbalance could affect our results in a negative way. Too many or too less of one or more emotions could result in a potential dominance of one or two emotions. For that reason we will delete some instances of some of the datasets and bring balance to all three emotions.

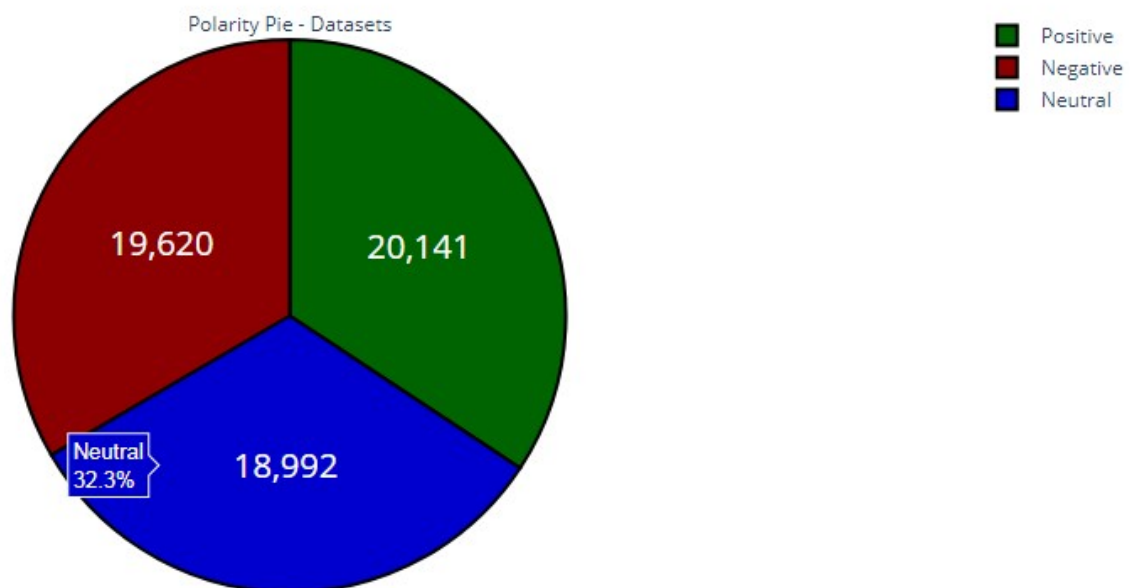


Figure 5.3: Polarity Pie - Datasets



After some deleting we can finally see in Figure 5.3 a better balance. The only thing left to do is to put again in new lists the now reduced tweets.

### 5.1.2 TF-IDF

However we cannot use our texts in that form. In order for the training models to use them we have to transform the data into numerical arrays and that will be accomplished by the TF-IDF transformer.

TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. It has many uses, most importantly in automated text analysis, and is very useful for scoring words in machine learning algorithms for Natural Language Processing (NLP).

The vectorizer that we have to use to transform our texts into arrays with floats representing the value of the words has some specific parameters that affect dramatically the performance of the training models.

1. The first one is called "max features" and is essentially the maximum of how many words the tool will use. In a list of so many tweets of different topics and opinions there can be words that are very rare and have minimal value affecting the overall tweet emotion, thus these words will have a smaller value in the transformed array. With that parameter we take only the words with the highest value in case we exceed the number of max features. In our case we set it at 2000. Due to ram and gpu limitations higher numbers would at some point extremely slow down the process or even terminate it. In a more advanced computer this number could be set even higher but this does not mean better results every time. After some experiments with fewer data, a max features of around 3000 could produce better results but after a point the use of more rare non significant words would actually drop the accuracy of the training models instead of boosting it.
2. The second one is called "min df" which regulates which words will not be used at all in the vectorizer. It can be set either by a positive integer or a float in the range [0, 1]. When an integer is put, the tool will not use all the words that were not found in documents equal or greater than the number set. In our situation it is set at ten, meaning

that if a word is not found in at least ten tweets it considered so rare it has no value. If the number was a float in that range, e.g. 0.2, then all words that are used in less than the 20% of all documents will not be used.

3. The third one is "max df" which is exactly the opposite of "min df". It is set at 0.8 for us, which means that if a word is used more than 80% in all documents will not be used. Overusing a word can actually decrease its value as when it is rarely used, because it becomes so common that does not actually affect the emotion so much.

## 5.2 Classification Algorithms

After the transformation of the tweets' text we split the texts and the polarity objects into the training and testing parts. The training data will be used to train the models and the testing one will test them to check their accuracy. We will use the classic 80% training and 20% testing and set the parameter random state at 0 in order to always split the data in the same way. This will let us compare results between different experiments. If the input data is always the same and the train-test split procedure always splits the data in the same way then we can be sure that any possible differences between experiments in their outputs will be due to the classification algorithms themselves or the different sample of tweets we get each time we try the whole process from the start.

The following classification algorithms are all used in the same way. The training model is created, we use it to predict the testing data and we proceed by showing the confusion matrix, the classification report and most importantly their accuracy score. Their accuracy is stored in a list which will be later used to find the most accurate algorithm and use that one to predict the sentiment of our own tweets.

A quick explanation about the metrics of such algorithms.

1. The confusion matrix is used to know the performance of a machine learning classification. It is represented in a matrix form and gives a comparison between actual and predicted values. Its matrix is a  $N \times N$  matrix, where  $N$  is the number of classes or outputs, in our case it is a  $3 \times 3$  matrix because of the 3 sentiments.

Before we explain the more complex  $3 \times 3$  matrix though let's understand the classic  $2 \times 2$ .

		Ground truth		
		+	-	
Predicted	+	True positive (TP)	False positive (FP)	Precision = $TP / (TP + FP)$
	-	False negative (FN)	True negative (TN)	
		Recall = $TP / (TP + FN)$		Accuracy = $(TP + TN) / (TP + FP + TN + FN)$

Figure 5.4: Confusion Matrix 2 x 2

Figure 5.4 is a prime example of what a 2 x 2 [confusion matrix](#) is. The columns are the actual values and the rows are the predicted ones, at least in this example. Each term can be explained as following.

- I True positive is when both predicted and actual values are positive.
- II True negative is when both values are negative.
- III False positive is when we predict it is positive but actually is negative.
- IV False negative is when we predict it is negative but actually is positive.

The three most important metrics that are calculated from the confusion matrix are the following.

- I Precision tells us how many of the correctly predicted cases actually turned out to be positive. This would determine whether our model is reliable or not.
- II Recall tells us how many of the actual positive cases we were able to predict correctly with our model.
- III Accuracy tells us the ratio of all correct predictions to all predictions that were made and it is what we use to determine the best algorithm to use for our predictions later.

After understanding the basic 2 x 2 confusion matrix we can continue with our unique 3 x 3 one.

[	[	2759	649	542	]
	[	507	2926	404	]
	[	542	608	2814	]

Figure 5.5: Confusion Matrix 3 x 3

The confusion matrix of Figure 5.5 is just an example from the Random Forest Classifier that we will discuss later on. In our confusion matrices the columns (negative, neutral and positive) are the predicted values and the rows are the actual values. The main problem is how to understand what is what. Which one is the true positive? Which one is the true negative? Let's analyze all four terms by examining the negative emotion, which is the first row and column here.

- I True positive is when the actual value and predicted value are the same, in our case the negative emotion, meaning it is the [1,1] cell.
  - II True negative is the sum of values of all columns and rows except the values of that class that we are calculating the values for, meaning it is the sum of [2,2], [2,3], [3,2] and [3,3] cells.
  - III False positive is the sum of values of corresponding columns except the true positive value, meaning it is the sum of [2,1] and [3,1] cells.
  - IV False negative is the sum of values of corresponding rows except the true positive value, meaning it is the sum of [1,2] and [1,3] cells.
2. The classification report displays our model's precision, recall, F1 score and support. It provides a better understanding of the overall performance of our trained model.
  3. The accuracy score, as described in the confusion matrix part, is the fraction of predictions our model got right to the sum of all predictions.

We are now ready to explore all the classification models we will be using.

### 5.2.1 Logistic Regression

Logistic regression is a machine learning algorithm for classification despite the "regression" in its name. In this algorithm, the probabilities describing the possible outcomes of a

single trial are modelled using a logistic function.

Advantages: Logistic regression is designed for this purpose (classification), and is most useful for understanding the influence of several independent variables on a single outcome variable.

Disadvantages: Works only when the predicted variable is binary, assumes all predictors are independent of each other and assumes data is free of missing values.

### 5.2.2 Naive-Bayes Classifier

Naive-Bayes algorithm based on Bayes' theorem with the assumption of independence between every pair of features. Naive-Bayes classifiers work well in many real-world situations such as document classification and spam filtering.

Advantages: This algorithm requires a small amount of training data to estimate the necessary parameters. Naive-Bayes classifiers are extremely fast compared to more sophisticated methods.

Disadvantages: Naive-Bayes is known to be a bad estimator.

### 5.2.3 Decision Tree Classifier

Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data.

Advantages: Decision Tree is simple to understand and visualise, requires little data preparation and can handle both numerical and categorical data.

Disadvantages: Decision Tree can create complex trees that do not generalise well and can be unstable because small variations in the data might result in a completely different tree being generated.

### 5.2.4 Random Forest Classifier

Random Forest Classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

Advantages: It reduces over-fitting and it is more accurate than decision trees in most cases.

Disadvantages: It has slow real time prediction and is difficult to implement because of its complexity.

### 5.2.5 Support Vector Machine Classifier

Support Vector Machine is a representation of the training data as points in space separated into categories by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

Advantages: It is effective in high dimensional spaces and uses a subset of training points in the decision function, which is very memory efficient.

Disadvantages: The algorithm does not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

### 5.2.6 K-Nearest Neighbours Classifier

Neighbours based classification is a type of lazy learning as it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the  $k$  nearest neighbours of each point.

Advantages: This algorithm is simple to implement, robust to noisy training data and effective if the training data is large.

Disadvantages: It needs to determine the value of  $K$  and the computation cost is high as it needs to compute the distance of each instance to all the training samples.

### 5.2.7 Stochastic Gradient Descent Classifier

Stochastic Gradient Descent is a simple and very efficient approach to fit linear models. It is particularly useful when the number of samples is very large. It supports different loss functions and penalties for classification.

Advantages: It is very efficient and easy to implement.

Disadvantages: Requires a number of hyper-parameters and it is sensitive to feature scaling.

### 5.2.8 Gradient Boosting Classifier

Gradient Boosting is a machine learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms Random Forest. It builds the model in a stage-wise fashion like other boosting methods do and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Advantages: Often provides predictive accuracy that cannot be trumped.

Disadvantages: It will continue improving to minimize all errors. This can overemphasize outliers and cause over-fitting.

### 5.2.9 XGBoost Classifier

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. It provides a parallel tree boosting that solves many data science problems in a fast and accurate way.

Advantages: It is an efficient and easy to use algorithm which delivers high performance and accuracy as compared to other algorithms.

Disadvantages: It can over-fit the data especially if the trees are too deep with noisy data.

### 5.2.10 Passive Aggressive Classifier

Passive Aggressive algorithms are generally used for large-scale learning. It is one of the few ‘online-learning algorithms’. In online machine learning algorithms, the input data comes in sequential order and the machine learning model is updated step-by-step, as opposed to batch learning, where the entire training dataset is used at once. We can simply say that an online-learning algorithm will get a training example, update the classifier, and then throw away the example.

How Passive Aggressive algorithms work:

1. Passive: If the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model.

2. Aggressive: If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.

Advantages: It is very useful in situations where there is a huge amount of data and it is computationally infeasible to train the entire dataset because of the sheer size of the data.

Disadvantages: The clear disadvantage is that we don't have the "big picture" of our data, so the end result will probably be affected by the order of presentation of the samples. We may not be able to achieve top accuracy thanks to that.

### 5.2.11 LSTM Classifier

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed-forward neural networks, LSTM has feedback connections. It can process not only single data points such as images, but also entire sequences of data such as speech or video.

Advantages: One of the key advantages of using LSTM networks lies in the fact that they address the vanishing gradient problem that makes network training difficult for a long sequence of words or integers. Gradients are used for updating Recurrent Neural Networks (RNN) parameters and for a long sequence of words or integers. These gradients become smaller and smaller to the extent that, effectively, no network training can take place. LSTM networks help to overcome this problem and make it possible to capture long-term dependencies between keywords or integers in sequences that are separated by a large distance.

Disadvantages: LSTMs are prone to over-fitting and it is difficult to apply the dropout algorithm to curb this issue. Dropout is a regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network.

#### 5.2.11.1 Tokenizer

Instead of using the TF-IDF tool like we did for the rest of classification techniques, we will use the "Tokenizer" from Keras, which is something very similar but works in a different way needed for the more complex and different than the rest LSTM. After creating the Tokenizer and transforming both our tweets and the tweets from the new datasets once again, we are ready to move on to the actual LSTM model.



### 5.2.11.2 LSTM Model

Model: "sequential\_3"

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 250, 100)	5000000
spatial_dropout1d_3 (Spatial	(None, 250, 100)	0
lstm_3 (LSTM)	(None, 100)	80400
dense_3 (Dense)	(None, 3)	303
Total params: 5,080,703		
Trainable params: 5,080,703		
Non-trainable params: 0		

Figure 5.6: LSTM Model

Figure 5.6 is the summary of our LSTM model and the following is each layer in details.

1. The first layer is the embedded layer that uses 100 length vectors to represent each word.
2. "SpatialDropout1D" performs variational dropout in NLP models.
3. The next layer is the LSTM layer with 100 memory units.
4. The output layer must create three output values for our three sentiments.

The activation function is softmax for multi-class classification and because it is a multi-class classification problem, "categorical\_crossentropy" is used as the loss function.

Here are the results of the training.

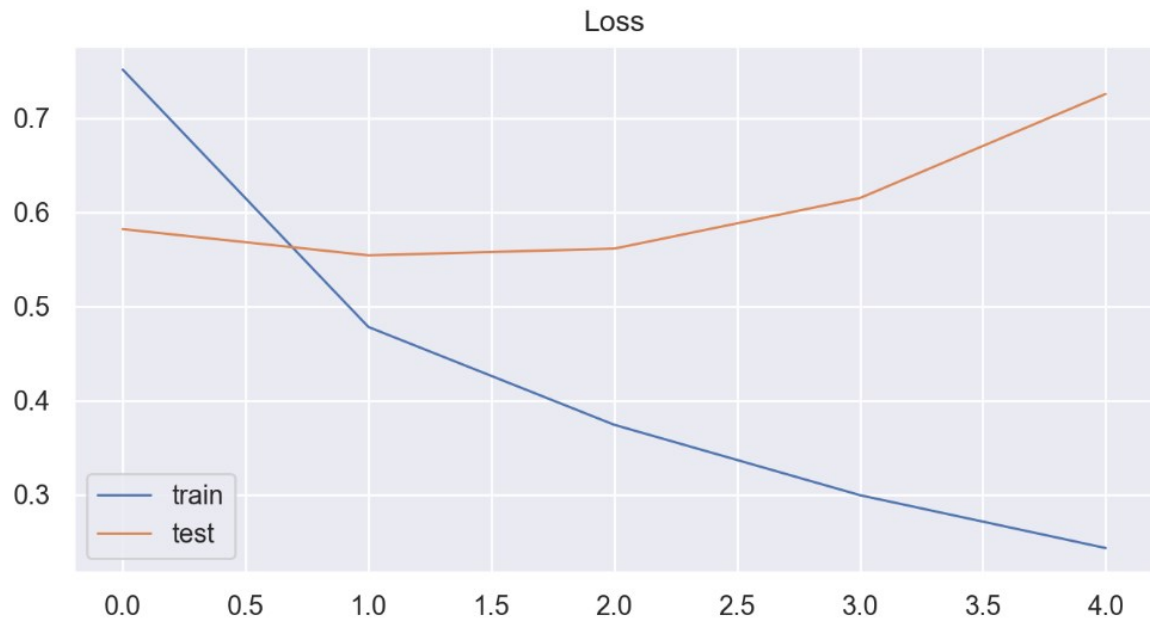


Figure 5.7: LSTM Loss

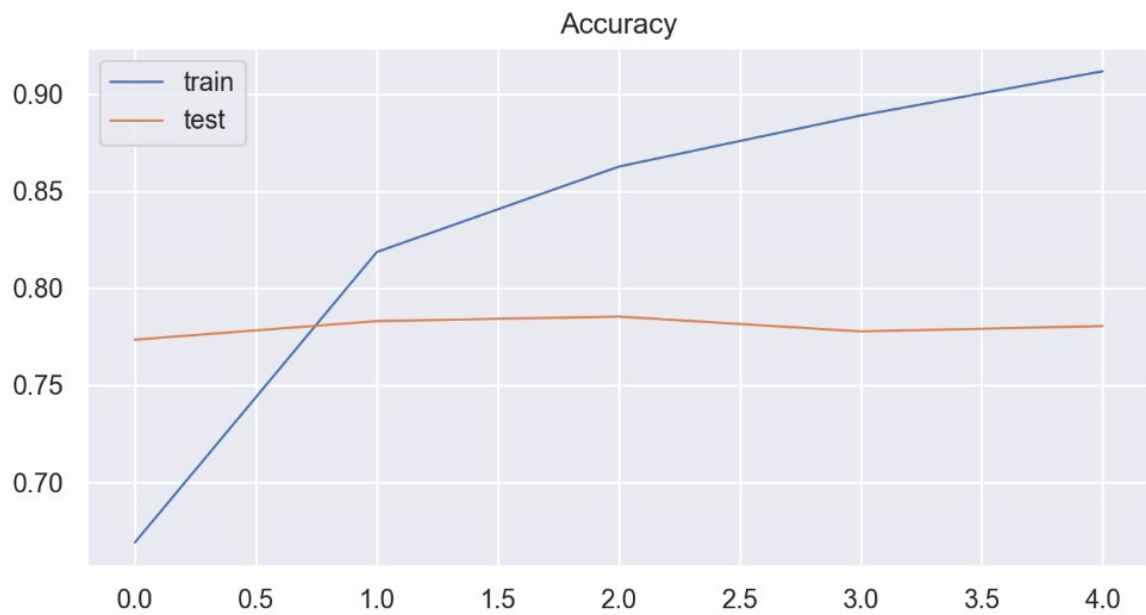


Figure 5.8: LSTM Accuracy

As we see from Figure 5.7 and Figure 5.8 the training model works extremely well with the training data but not so much with the testing. Our loss drops and our accuracy rises but the validation loss rises and the validation accuracy drops. This means that our model over-fits and probably focuses a lot on noise data. LSTM models as previously discussed are notorious for over-fitting and there are a lot of ways to combat this problem but they do

not always result in anything better just because they were used, some examples are adding dropout layers and adding weight regularization.

### 5.2.12 Accuracy Results

After running every training model and getting the accuracy scores we can see their results.

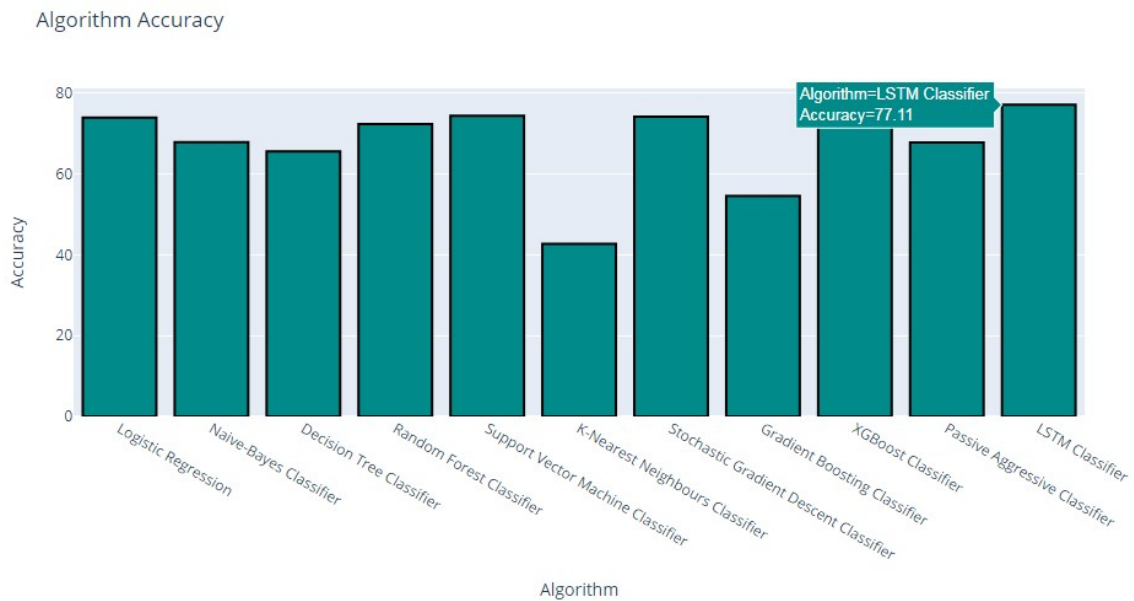


Figure 5.9: Algorithm Accuracy

Looking at Figure 5.9 we can see that some models unfortunately failed, notably the K-NN had and always has in multiple experiments the worst accuracy. Non-lazy classifiers follow a training process in which they try to optimize accuracy or error. K-NN, being lazy, has not a training process, and in consequence, it does not try to optimize any effectiveness measure. We can play around with different values of K, and intuitively, the bigger the K the higher the recall and the lower the precision, and the opposite. Gradient Boosting, albeit a bit better, had also bad results but sometimes is shown to improve a bit more.

On the other hand most of the other algorithms reached or almost reached the 70% threshold. Among them, SVM, Logistic Regression and Stochastic Gradient Descent always perform really well never dropping lower than the threshold. This cannot be said for Random Forest and XGBoost though which sometimes do not perform so well as they did in this experiment. They always perform well just not so well all the time. Both of them use decision trees which depending on the run and the input data can a lot of times under-perform, some-

thing that also applies to the Decision Tree algorithm which also performs well but not so well like SVM for example. Lastly Naive-Bayes and PAC once again reached a very good accuracy in a consistent way after many experiments.

An interesting fact is that our best and most consistent algorithms in multiple tweet samples, SVM with a linear kernel, Logistic Regression and Stochastic Gradient Descent as mentioned above, all use linear models into their classification compared to all the other algorithms, which is very interesting as it shows that our data is more linearly separable than at first glance.

However, even if a lot of our classification algorithms worked pretty well, the LSTM model surpassed all of them with an astounding 77.11% accuracy even when we clearly see an over-fitting problem, meaning it could reach even higher heights.

### 5.3 Sentiment Prediction

After seeing all the results and every algorithm's accuracy the user is asked to freely choose the method he wants to use to predict our own tweets. After his choice, the polarity will be predicted once again. We convert the categorical data (neutral, negative and positive) into the respective numerical ones (0, -1, 1) and insert them into the polarity column of the copy of the original dataset resulting into an identical dataset with the only difference being the values of the polarity columns.

In this experiment we will be using the LSTM approach as it reached the best accuracy and overall is a more interesting method being a neural network of course.

### 5.3.1 World Data

Once again we will plot the time series graph as well as the polarity pie of the world data.

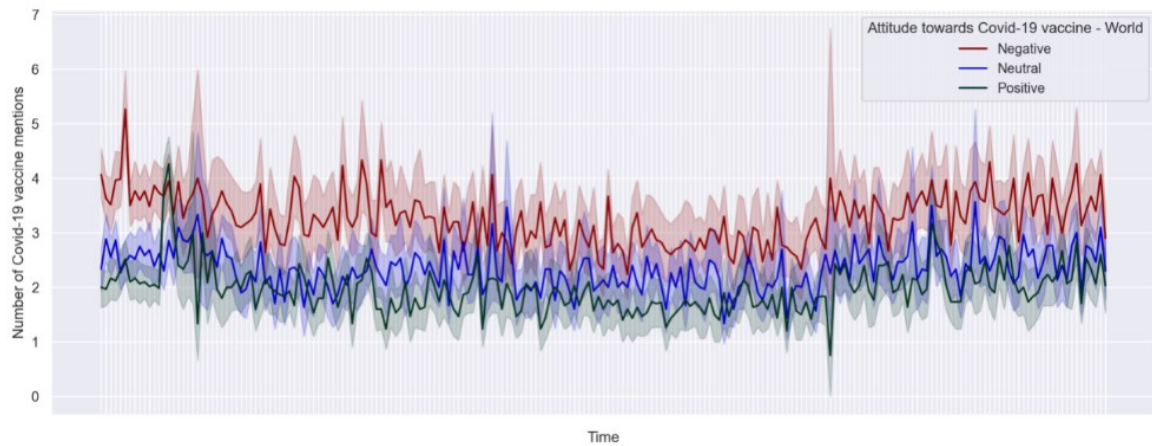


Figure 5.10: Time Series Graph - World 2

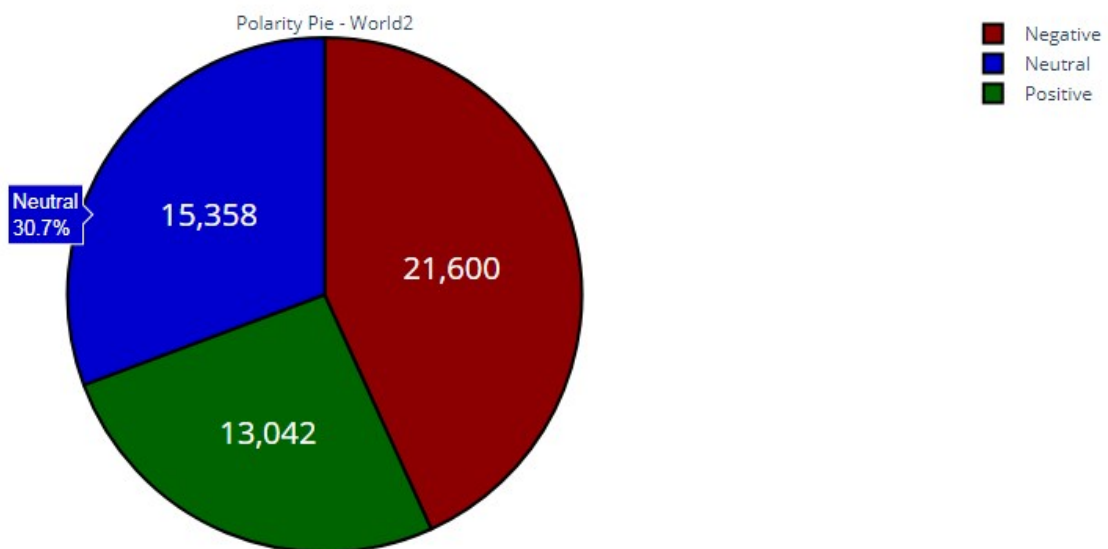


Figure 5.11: Polarity Pie - World 2

### 5.3.2 USA Data

We extract again the US made tweets and plot the same graphs as before.

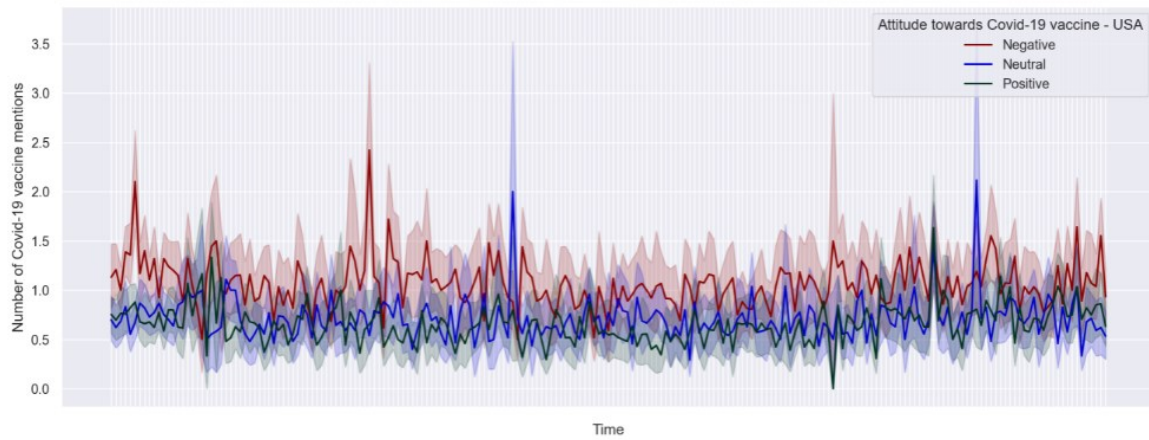


Figure 5.12: Time Series Graph - USA 2

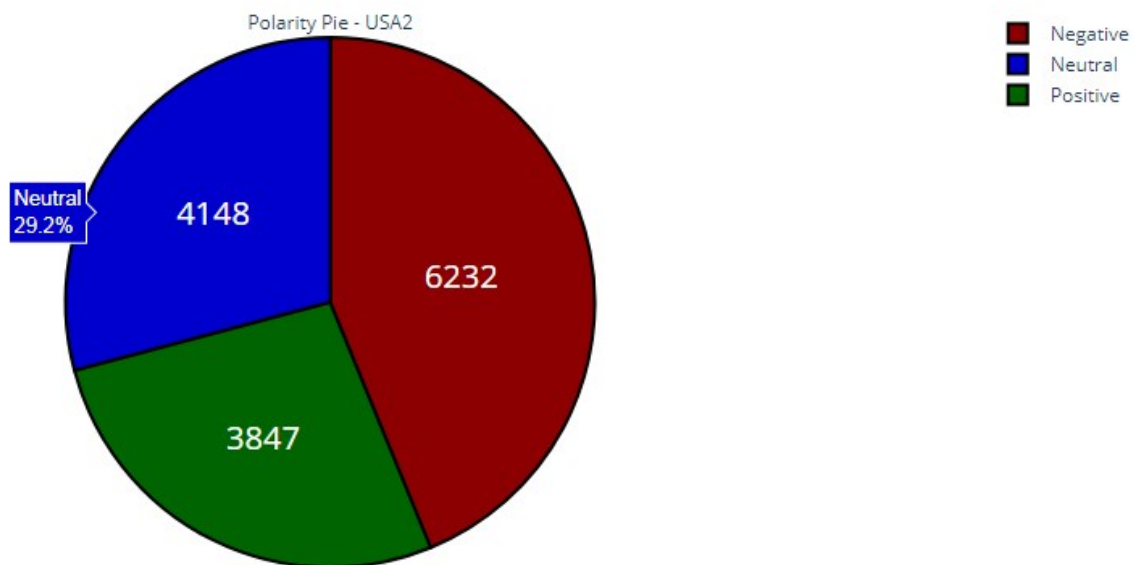


Figure 5.13: Polarity Pie - USA 2

### 5.3.3 Comparison

There seems that there are some differences but in order to be sure about it let's put everything together one more time and compare them.

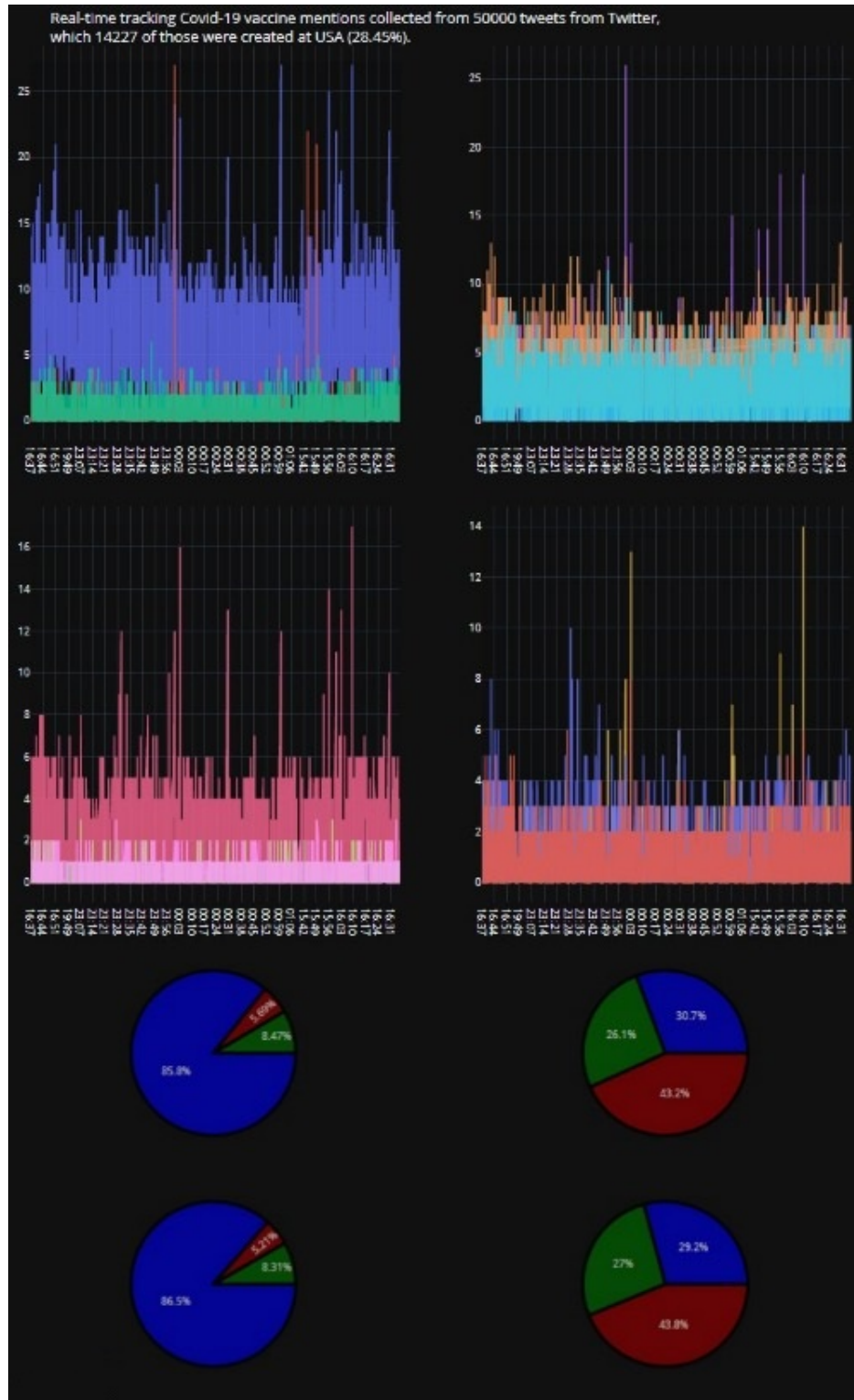


Figure 5.14: All Plots Together 2



In Figure 5.14 the two columns represent the two different ways we used to extract the polarity of our tweets, with the left being the one with TextBlob and the right the one with the classification algorithms. Every row is the comparison between the same graph, meaning that the first row is between the time series graphs of world data, the second the time series graphs of USA data, the third the polarity pies of world data and finally the fourth one the polarity pies of USA data. In that way we can compare both horizontally and vertically.

As expected the differences between world and USA data are meaningless in both procedures. However, there are noticeable differences between the two procedures themselves both for world and USA data. The overall number of sentiments from our tweets has shifted considerably. With the TextBlob algorithm the neutral emotion was dominant at around 85% and the rest 15% were shared to the positive and negative emotions in favor of the positive one. Nonetheless, with the classification procedure almost 45% of the neutral sentiment was converted, or better called reclassified, into positive and negative ones giving the advantage to the negative sentiment this time at a total of 30.7% neutral, 43.2% negative and 26.1% positive.

It seems as the second procedure came closer but not exactly to the findings of [2]. The differences between our results and [2]'s can be explained in many ways. The randomness of the downloaded tweets, the difference of the time of their creation and the training of the algorithms themselves are a few. Be that as it may, the classification models seem that are more able to value the polarity of the tweets in a more balanced and logical way compared to TextBlob.

## 5.4 Extra Notes

It was previously mentioned that in order for the training models to work correctly we would need a balanced set of texts and sentiments without one or two emotions dominating the dataset.

In previous experiments though, with datasets consisting mostly of negative emotions of 70% and more the end result was completely different. The neutral emotion of the TextBlob procedure was completely converted into negative of more than 80%. This happened because the training models got accustomed to more negative texts and perceived most of our tweets as negative as well. In theory the classification algorithms were working correctly as they



achieved great accuracy scores but in reality they were not tuned well to actually evaluate new texts. This shows the importance of correct and carefully selected data inputs even when the numbers show great results in the training/testing phase.

However, even when we balance the sentiments of our input dataset and avoid mistakes such as the the previously mentioned experiment, the output can vary between runs. As previously explained, even when we put the same input in the training algorithms by using the same texts and sentiments from our datasets and by splitting the data in the same train and test parts there can still be different outputs each time.

This is due to the inner workings of the classification algorithms themselves. The training procedure is not a simple task and in no way is it deterministic every time. One example is all algorithms that use decision trees such as Random Forest which is in fact stochastic and it is not the only one being stochastic. This does not mean that it is random but it is not deterministic either. Having the same texts and sentiments both from the training data, the testing data and our own tweets that we need to predict, does not necessarily mean that we will get the same result every time in its accuracy or in its predictions. This can be very tricky and often misleading in many ways and we must always take careful steps when dealing with these situations. One lapse of judgment or an "unlucky" experiment can lead to conclusions that do not represent the truth or at least all of it but a fraction of it.

No algorithm and model can be defined as the best just by their accuracy score and the experiment with the mostly negative input data was such an example. The algorithms performed superficially well with high accuracy scores but ended up predicting almost all the tweets to be negative, something that cannot be realistic.

For this reason we use a lot of algorithms and take into account both their accuracy in their prediction of the testing data and their final prediction of our own data. We can never be sure if everything that is predicted is 100% correct, either because of the stochastic nature of some algorithms or because they never actually reach the perfect accuracy score. False predictions are bound to always happen and this is especially true when we are dealing with unstructured text data in classification problems. All we can do is look at the results and try to understand which one comes closer to the "truth", something that can be extremely difficult and needs multiple experiments using the same data and much more gpu power and ram in order to execute more complicated, time and power consuming program runs as well as even better models especially for something as complicated as LSTM.



# Chapter 6

## Fake News Detection

In this chapter we will change the topic for a bit and focus on detecting fake news. The "Fake News Detection" notebook will use once again the same classification models but this time they will be trained to predict if our tweets are spreading misinformation or not.

Before of anything though we will again "open the store" in order to use all the objects that were saved in the previous notebook the "Sentiment Prediction" just as we did with the "Sentiment Analysis" one with the first one being a second copy of the original sample of tweets, not to be confused with the first copy of the previous notebook. The stored objects are the data that was created and could not transfer to the new notebook and it will be used for comparison purposes with the data that will be produced with the new procedures, more specifically the time series graph and the polarity pie for both world and USA data variants.

### 6.1 Importing Datasets

As in chapter 5, in order to train our classification models we will need new datasets containing various texts that have been labeled, by humans as per usual, as true or fake news.

The datasets we are gonna import and use for the training models are two.

1. The [ISOT Fake News](#) dataset is a compilation of several thousands fake news and truthful articles, obtained from different legitimate news sites and sites flagged as unreliable by Politifact.com.
2. [Fake News](#) is a compilation of multiple news articles of various authors similar with the ISOT Fake News dataset.

	id		title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...		1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...		0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...		1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Aistr...		1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print 'nAn Iranian woman has been sentenced to...		1
...	...	...	...	...	...	...
20795	20795	Rapper T.I.: Trump a 'Poster Child For White S...	Jerome Hudson	Rapper T. I. unloaded on black celebrities who...		0
20796	20796	N.F.L. Playoffs: Schedule, Matchups and Odds - ...	Benjamin Hoffman	When the Green Bay Packers lost to the Washing...		0
20797	20797	Macy's Is Said to Receive Takeover Approach by...	Michael J. de la Merced and Rachel Abrams	The Macy's of today grew from the union of sev...		0
20798	20798	NATO, Russia To Hold Parallel Exercises In Bal...	Alex Ansary	NATO, Russia To Hold Parallel Exercises In Bal...		1
20799	20799	What Keeps the F-35 Alive	David Swanson	David Swanson is an author, activist, journa...		1

20800 rows x 5 columns

Figure 6.1: New Dataset - News

Table 6.1 is one example of how these datasets are.

Before of taking any more steps though, we have to check one more thing, the balance of the data. As in our previous experiments with the amount of sentiments in our input data we also have to check here if the amount of fake and true news is balanced.

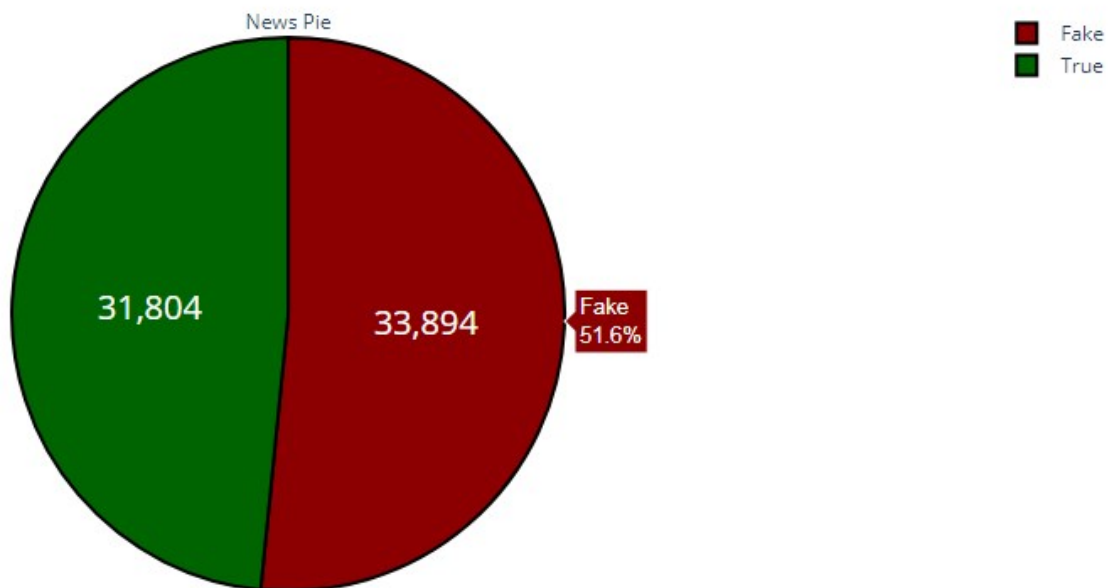


Figure 6.2: Truth Pie

Thankfully the amount of fake and true news in Figure 6.2 is balanced almost at 50-50. With this out of the way we can continue with the important stuff.

## 6.2 Pre-processing and TF-IDF

Once again we will clean our tweets and the texts from the imported news from any unnecessary characters, put them in separate lists and finally transform them with the TF-IDF vectorizer. The input data for the training models will be split again into the training and testing models and the same procedure of training will take place again.

## 6.3 Classification Algorithms

The models we will be using are exactly the same with the ones from chapter 5. The same applies to the LSTM model and the different tokenization procedure of its input data. This is why we will fast-forward a bit without mentioning each algorithm again except for the results of the LSTM model.

### 6.3.1 LSTM Model

Here are the results of the training.

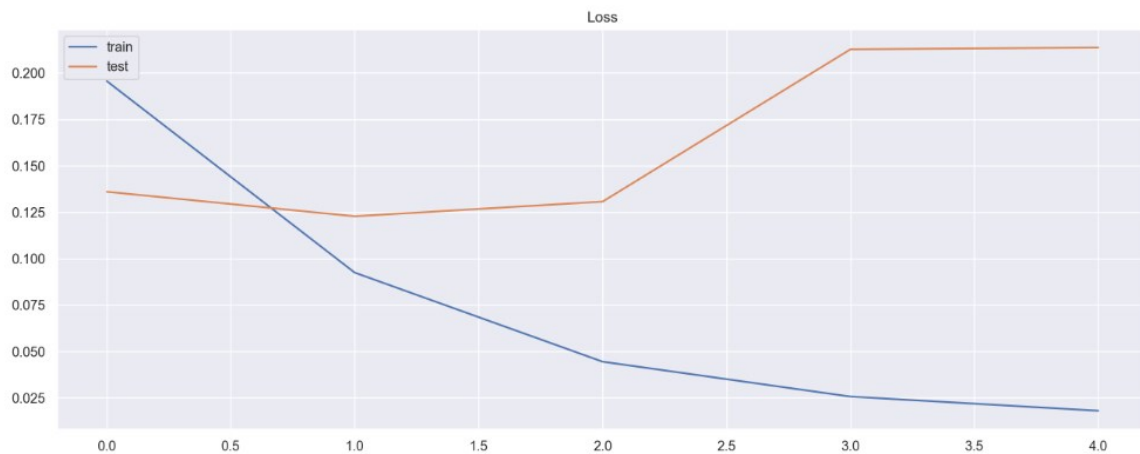


Figure 6.3: LSTM Loss 2

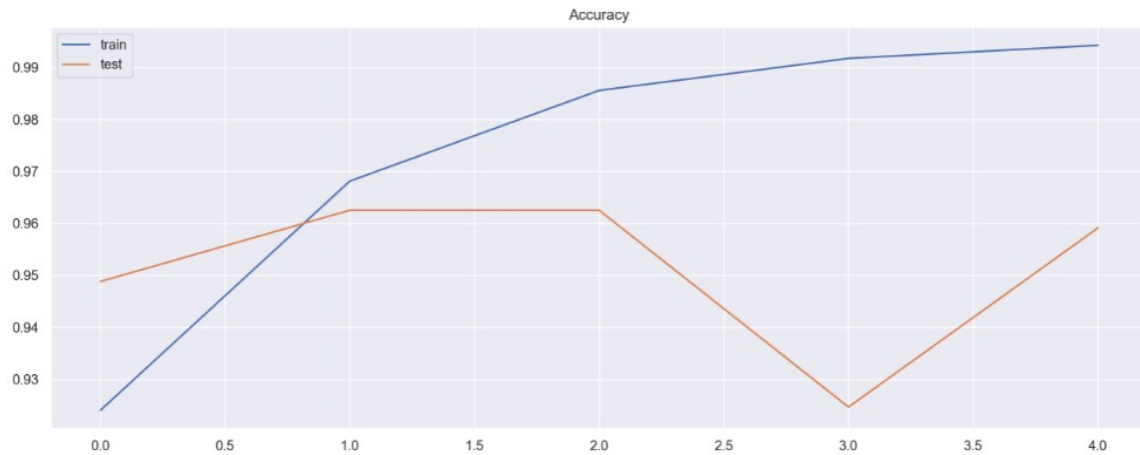


Figure 6.4: LSTM Accuracy 2

As we see from Figure 6.3 and Figure 6.4 the training model works with an amazing 96.26% accuracy but once again over-fits a lot and this time it is even worse than the last one. Even when we changed some parameters for the different input data still this was the best result.

### 6.3.2 Accuracy Results

After running every training model and getting the accuracy scores we can see their results.

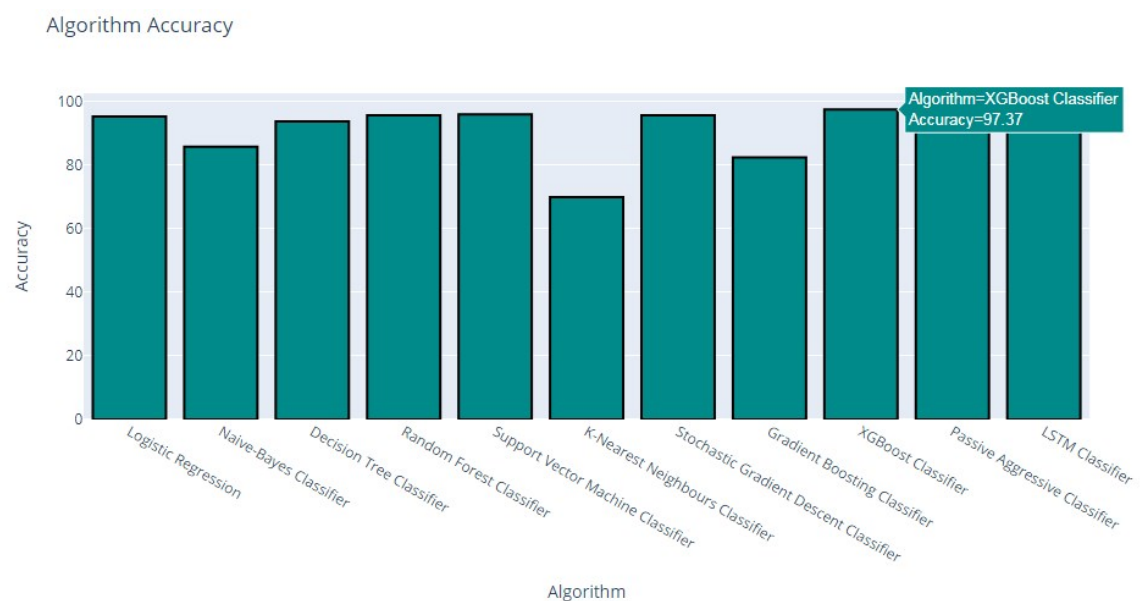


Figure 6.5: Algorithm Accuracy 2

Looking at Figure 6.5 almost all algorithms performed superficially well, even a little bit unrealistically well with XGBoost reaching 97.37% accuracy. For some reason the algorithms could predict really well the testing dataset which was not by any means small or unbalanced between fake and true news. Of course two classes are always easier to predict compared to three like in the previous chapter but for some reason, possibly because of the input datasets, the accuracy scores were phenomenal, something that may turn out to be actually bad for us. A dataset that can be very easily understood by the training models can have data instances so similar to each other that the models do not actually train for something as they get almost the same input over and over again, which of course is just a hypothesis but a realistic one.

## 6.4 Fake News Prediction

After seeing all the results and every algorithm's accuracy the user is asked once again to freely choose the method he wants to use on our own tweets. After his choice, the label of our tweets will be predicted, meaning if they are spreading misinformation or not. After that we will delete all tweets that were deemed as fake news and we will proceed to compare the new dataset with fewer tweets to the dataset of chapter 5 in order to see if these deleted tweets will actually affect the overall polarity or not.

In this experiment we will be using the Passive Aggressive approach. It did not reach the best accuracy out of every algorithm in this run but it is consistently good and considered one of the best for fake news detection.

### 6.4.1 World Data

Once again we will plot the time series graph as well as the polarity pie of the world data.

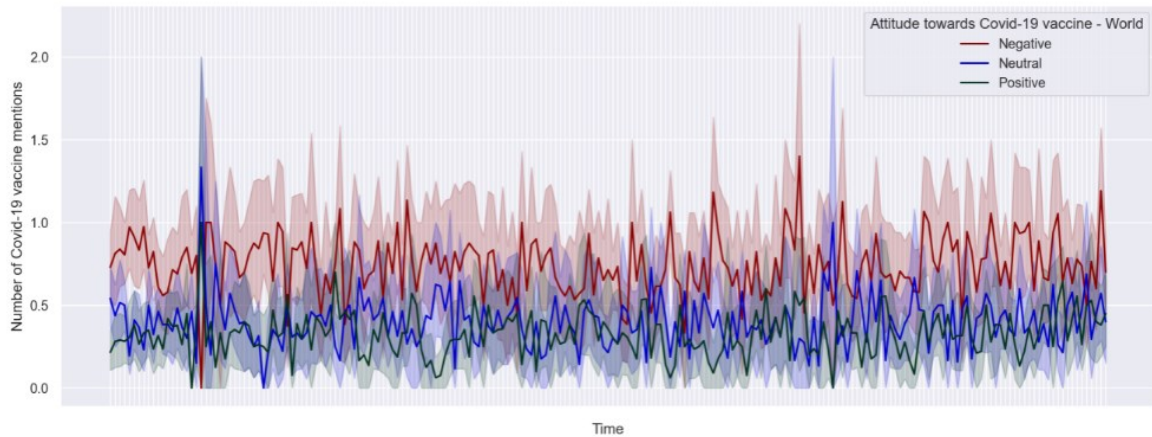


Figure 6.6: Time Series Graph - World 3

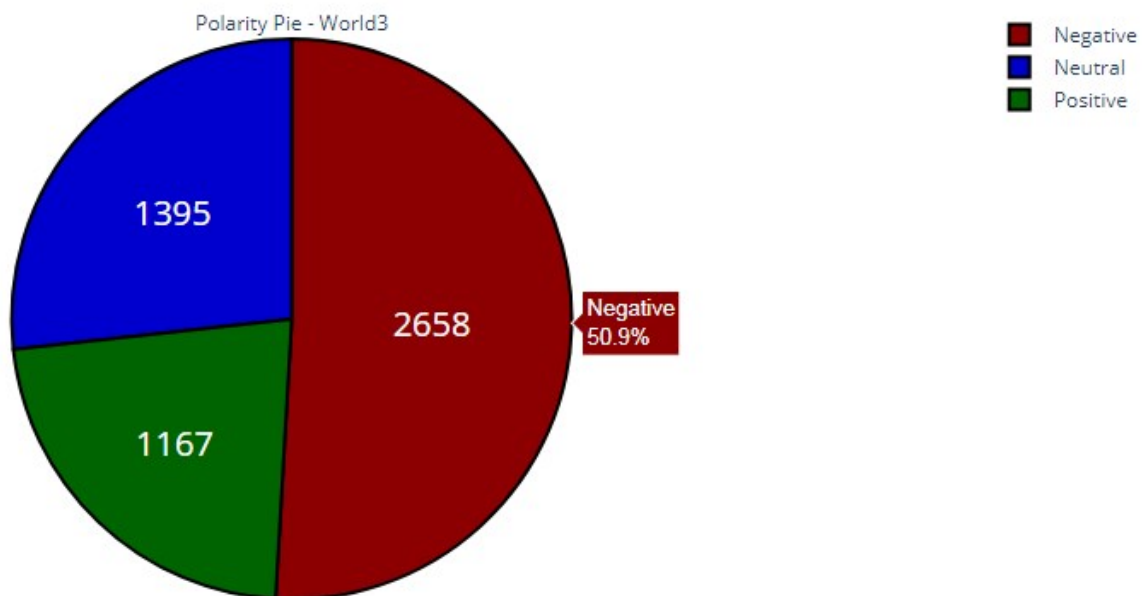


Figure 6.7: Polarity Pie - World 3



### 6.4.2 USA Data

We extract again the US made tweets and plot the same graphs as before.

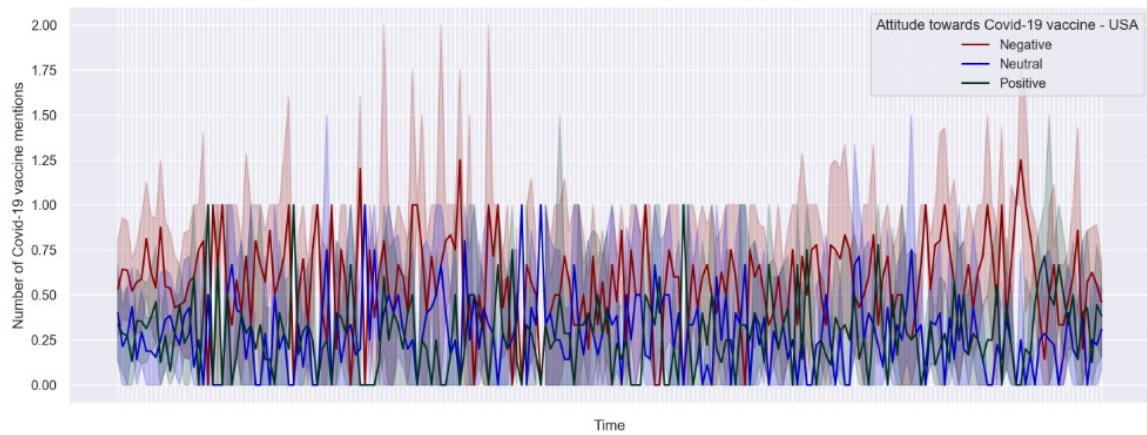


Figure 6.8: Time Series Graph - USA 3

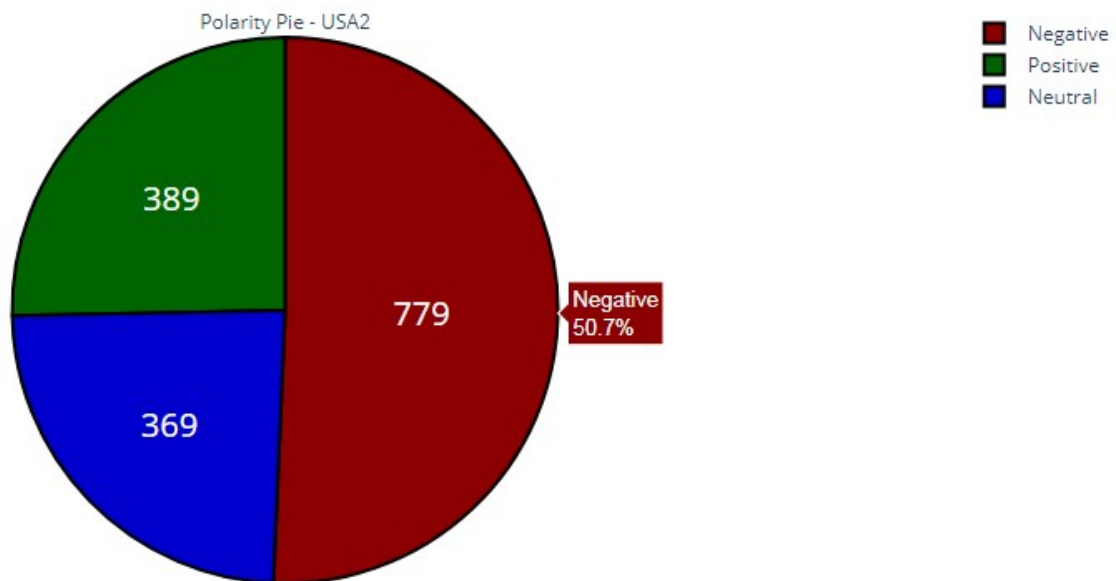


Figure 6.9: Polarity Pie - USA 3

### 6.4.3 Comparison

As we can all probably see the 50000 tweets were swiftly reduced to only 5220, which is in fact very weird. Of course depending on the algorithm and the specific run this number could change but on average it seems that the tweets are destined to be reduced to such low numbers.

We can all understand that this is highly unrealistic. Even if Twitter is full of misinformation these numbers are not acceptable. There is no way that only the 10% or even less of our tweets are considered not to be misinformation. However there are some possible explanations about this behaviour.

1. All the tweets are talking about the vaccine and the most common topic right now about the vaccines is if they are safe to use and if it is okay to be mandatory for every citizen instead of being each person's choice to get vaccinated or not. Unfortunately, by talking about this situation, either by promoting misinformation or just mentioning that this information is actually fake, the models will probably trigger and value it as fake news. It is almost the same with TextBlob. There is the chance that the algorithm cannot recognize easily between someone promoting misinformation and someone talking about it and possibly denying it. It perceives it both as spreading misinformation.
2. The tweets we used are not similar to the news we had in our input dataset. The news from the imported datasets were from articles and they were paragraph long texts about various topics. Our tweets are short sentences with a limitation of characters made by everyday people who most of the time type with abbreviations, some sort of slang and generally do not make the most structured sentences. This could be another reason why the model values these sort of sentences, together with the first reason, as misinformation.
3. Lastly, as mentioned above, the datasets used could be actually problematic. The algorithms really understood the input data so well that there is the suspicion that these instances are so close to each other that the models can train for them but not actually train well for different type of texts, exactly like the experiment done in chapter 5 where almost all the input data was negatively reviewed and almost all our tweets were also negatively reviewed. Great accuracy scores but unrealistic predictions.

These are the reasons I personally thought to explain this bizarre situation. We could all have guessed that a lot of the tweets would be spreading misinformation, it is the most deadly "virus" of our century after all, but 90% of all tweets seems like a stretch. It is highly possible that one or even multiple of the reasons I mentioned and maybe others I do not know made the training model to decide like that. I personally believe its due to the first reason, of not being able to fully understand when someone really promotes fake information for the vaccines compared to someone talking about it and disapproving it, as well as the not so diverse input data.

Nonetheless, we have to show everything and how it worked even if the results were not as pleasing as we would hope so.

So for the last time, we will show all graphs together comparing the results from chapter 5 and the results from this chapter as we did with chapters 4 and 5 previously.

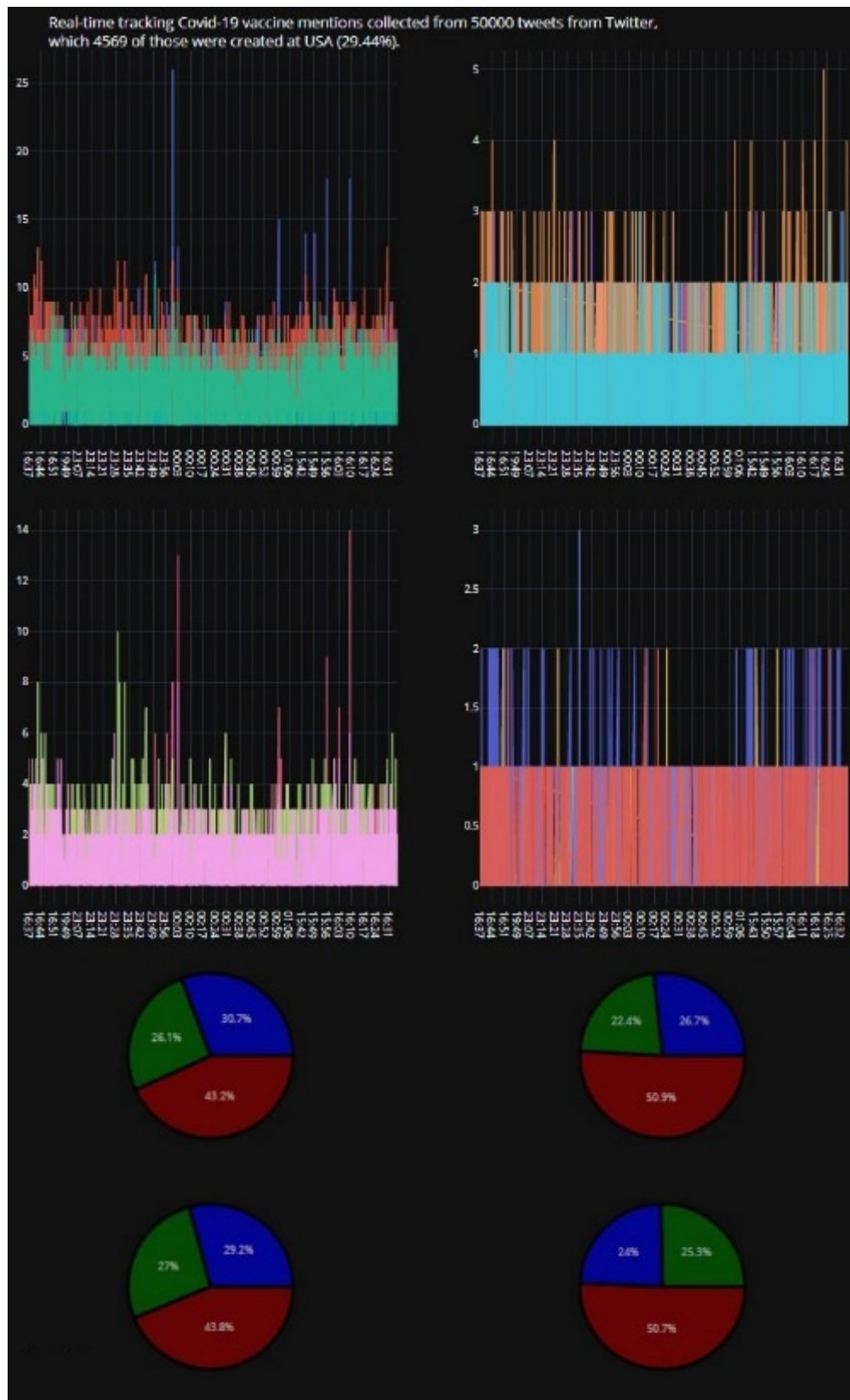


Figure 6.10: All Plots Together 3

In Figure 6.10 we see that generally there are small differences popularity wise between the two columns. Even after removing so many tweets, the overall polarity remained almost the same albeit of a slight change in the negative emotion, which means that the tweets considered to be fake were actually a balanced mix of all three emotions, not changing the final result by a lot. Also, once again the differences between the world and USA data are again negligible.

This proves that fake news in any kind of topic is not just hateful comments towards someone or an idea but more like a well hidden virus. People figuratively get infected with the fake information for a number of reasons. The most prevalent is due to the overexposure of this misinformation. With the power social media and generally the internet have in this day and age, people are subject to look constantly at fake news, which as many experts have stated, spread many times faster than the truth. A lot of them subsequently fall in the trap and perceive them as the actual truth and with a strong sense of justice in the recent years noone wants to fall victim to some kind of conspiracy by a higher power, in many cases their government. So, they perceive these fake news, which most of the times are in contrast to what that higher power says, as the truth and they fight against the higher power with a sense of justice and goodwill and ultimately spread these fake news to even more people in an attempt to pass the truth to them, exactly like a normal virus.

This misinformation is not anymore just an attack to someone. It is masked under the good heart of the people that do not want to get tricked with lies but end up unfortunately getting tricked. This is the reason why the tweets deleted in our case were not all just negative and neutral but a balanced combination of all three.



# Chapter 7

## Results, Conclusions and Future Extensions

This is the last chapter of this project. We will be discussing about our overall results, what conclusions can be drawn from these results as well as talk about future extensions both on this programs but also on the general idea.

### 7.1 Results and Conclusions

Generally text classification and anything about working with unstructured data like texts is bound to be a bumpy ride. We are given unlimited tools in our hands, different ideas and angles to approach some tasks but still a lot of them fail, under-perform or can be just predictions without really knowing how well we have really predicted the things we wanted to predict apart from when we handle them manually and inspect any inconsistencies. Chapters 4, 5 and 6 touched a lot on these ideas and created many results.

#### 7.1.1 Chapter 4

Chapter 4 showed us that simpler algorithms like TextBlob can give an idea of the sentiment of our input texts but ultimately cannot discern what is truly neutral with what is more positive or negative sided. On top of that, we saw that as far as we are talking about the English speaking users tweeting in English the USA goes along with the rest of the world and is actually the number one most tweeting country, at least about the vaccine. Even though the whole chapter was a bit uneventful, with mostly neutral tweets and the USA / World com-

parison being non existent, we still got to look at how we can treat text data and display it in the ways that we want to, from the time series graphs to the NLTK word frequency bars.

### 7.1.2 Chapter 5

Chapter 5 on the other hand completely turned the table around on the very same topic. Instead of using an algorithm that just compares words and guesses the overall sentiment, we used classifiers to train with tweets that were emotionally values by humans. In that way we tried to create models that will take in all the data at once, instead of word by word, and try to understand why this tweet is actually negative or why another is neutral and so on.

First of all we had the opportunity to use the two different ways of transforming our text data into arrays made of floats in order to be used in the training models, an essential step when dealing with these classification problems.

The results of course of all these classifications varied a lot. There were algorithms that performed better and others that unfortunately failed. Sometimes this happened due to the nature of the algorithm. As we saw some of the best algorithms based on their accuracy were linear models such as the Logistic Regression indicating that probably our data is more linearly separable, which some algorithms struggle with. Other times it was due to the specific input data and training model that was created. Not all algorithms performed the same way in each run and that is why their accuracy scores were also not the same. As it was explained, a lot of classifiers, even when the input data is exactly the same, do not create the same model as in the previous experiment resulting also in different predictions.

Additionally, even when a lot of models had the same or very similar accuracy results like Stochastic Gradient Descent and Logistic Regression, which are both linear models, their predictions were not the same and sometimes they were not even similar. Each algorithm has worked in its own way and even though they correctly predicted the same amount of sentiments in the input testing data, the final predictions of our own tweets do not have to be the same or even close to each other. The truth lies somewhere in the middle of all these predictions.

In the end though we saw the tremendously different results in the tweets' polarity compared to chapter 4. Instead of a neutral "monopoly", the results were always closer to an actual balance. Of course every iteration and different algorithm and different sample of the 500000 tweets resulted in different results, but every time these results were always much



more closer [2]’s findings or a true balance of one third for each emotion.

This is a prime example of how a procedure, that maybe seems to work at first, could always be improved and evolved further and further each time resulting in more complicated algorithms and more exciting and most of all truthful results.

### 7.1.3 Chapter 6

Unfortunately chapter 6 did not perform so well like chapter 5, at least in the results that were produced. The same processes were used as in chapter 5, the same algorithms, the same importing of input datasets and the same data transformations. However, the final result was the deletion of around 90% of all our tweets.

This number is of course not entirely correct as explained in chapter 6. There are some ideas why this happened but there is no concrete evidence. Despite that, even if we could not see the ”best” result of fewer tweets being deleted and shifting in some way the sentiment balance of chapter 5, we saw in practice how algorithms that score so well during their training/testing phase can in the end produce totally unexpected results, which for a human handling the same tweets by hand, could seem quite unrealistic.

No matter the ”failure”, the procedures are still there and as shown in chapter 5 they are working fine. The problem for these ”bizarre” results could lie in the input dataset itself, in the whole situation of different input dataset and different texts to actually predict, in the inability to understand who promotes and who references but denies false information or something else.

## 7.2 Future Extensions

Future extensions could go in two ways. One is the further improvement of what we already used and the second one is the evolution of this more basic approach to something much bigger.

### 7.2.1 Further Improvement

My project has shown us how we can work around text data and achieve some form of results in our goal to analyze the sentiment of our tweets and possibly detect any misinfor-

mation. However, while staying on this basis, there can be a lot of improvement in a number of ways.

#### 7.2.1.1 Sentiment Analysis

Both in chapters 5 and 6 we were tasked to find and explain the polarity of our tweets. There is always room for improvement even if we use a method like with TextBlob or a method as with the classifiers, such as with the case of [7] who also used other information from the tweets like their hashtags. One other simple example is a better LSTM model and the same applies for all methods. Even if we got some promising results, every algorithm used, especially the LSTM model which in our case had an over-fitting problem as well, can be adjusted in a better way, use different parameters or process the input data in a better and more efficient way by finding the most useful words and avoiding any potential noise.

On that notion, as explained before, a better computing system is also a way to get better results. More ram and more gpu power could possibly run more complicated algorithms and one example is the TF-IDF transformer where we selected a low number of "max features" due to the computer's limitations. In a case where there are no such limitations there is a possibility that more useful words would also be transformed and used in the training models giving a set of different maybe better results.

#### 7.2.1.2 Fake News Detection

Compared to the sentiment analysis, there seems to be even more room for improvement. The algorithms themselves are not incorrect and no possible different parameters can make such a huge change to produce more "normal" results than what we got but still something can always be done to change this outcome. One idea is instead of using the TF-IDF transformer to transform our text in numerical arrays, to use another encoder. One example is [11] which uses a different way, the Multimodal Variational Autoencoder (MVAE) to be specific in combination with the usage of different models compared to ours. Another really famous and upcoming encoder is BERT, which is frequently used for fake news detection purposes, as used in [12]. These different transformers can play a big role in how the data is used in the training models and produce completely different results, maybe even surpass the problems that I mentioned, such as the possible incapability of understanding who spreads misinformation and who just references it while rejecting it.

Apart from that, a different dataset could turn the situation up-side down. If we had a dataset of tweets, similar to ours, that was handled by humans for being true or fake news, instead of huge articles, there is a chance that we could see more prominent results, as it was with chapter 5 and its datasets being full of regular tweets like ours. However, in my search I could not find something publicly available.

## 7.2.2 Complete Evolution

On the other hand, there are a lot of ways to not just improve my work but evolve it into something more, something more complicated, more advanced and more productive in every way.

### 7.2.2.1 Sentiment Analysis

There are multiple articles and research done on this famous topic, both for general text but also specifically for COVID-19 and its topic such as the vaccines. One prime example is through the use of the amazing and very interesting "Deep Convolutional Neural Networks" as [6] did in their own research, going much deeper on how to train this unstructured data. It seems through [6]'s work and many others' that neural networks are truly the way for sentiment analysis to progress even further into the future.

However, apart from the classification algorithms idea, there has also been work done around lexicon based approaches such as with [2] and [1] who used algorithms such as VADER, OpinionFinder and many more to collect the texts' polarity.

### 7.2.2.2 Fake News Detection

Fake News Detection is a newer and upcoming topic of interest. Compared to sentiment analysis, not so much research has been done but there is still plenty of published work about it. [10] on their own research do not just try to connect the text from the article to a potential true or fake story but to further understand who created it, what type it is and for what reason this story was published, which is also very similar, in its core idea, with [9]'s work.

On the other hand, a lot of people have tried creating their own algorithms such as in the case of [13] that is graph based and tries to do something similar to our approach but with a completely different model and functions.

These are just some examples of how differently we can approach fake news detection compared to sentiment analysis. It is much more complicated the deeper we dive into it but that makes it so much more interesting!

# Bibliography

- [1] F. Bravo-Marquez, M. Mendoza, and B. Poblete. Combining strengths, emotions and polarities for boosting twitter sentiment analysis. In *Association for Computing Machinery*, New York, NY, USA, Aug. 2013.
- [2] S. Yousefinaghani, R. Dara, S. Mubareka, A. Papadopoulos, and S. Sharif. An analysis of covid-19 vaccine sentiments and opinions on twitter. *International Journal of Infectious Diseases*, 108:256–262, 2021.
- [3] Real-time twitter sentiment analysis for brand improvement and topic tracking chapter 1. <https://towardsdatascience.com/real-time-twitter-sentiment-analysis-for-brand-improvement-and-topic-tracking-chapter-1-3-e02f7652d8ff>.
- [4] Real-time twitter sentiment analysis for brand improvement and topic tracking chapter 2. <https://towardsdatascience.com/real-time-twitter-sentiment-analysis-for-brand-improvement-and-topic-tracking-chapter-2-3-1caf05346721>.
- [5] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan. A system for real-time twitter sentiment analysis of 2012 u.s. presidential election cycle. In *Association for Computational Linguistics*, USA, Jul. 2011.
- [6] A. Severyn and A. Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *SIGIR '15: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, Aug. 2015.

- [7] X. Wang, F. Wei, X. Liu, M. Zhou, and M. Zhang. Topic sentiment analysis in twitter: A graph-based hashtag sentiment classification approach. In *Association for Computing Machinery*, New York, NY, USA, Oct. 2011.
- [8] Multi-class text classification with lstm. <https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590be1bd17>.
- [9] J. Reis, A. Correia, F. Murai, A. Veloso, and F. Benevenuto. Explainable machine learning for fake news detection. In *Association for Computing Machinery*, New York, NY, USA, Jun. 2019.
- [10] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu. Fake news detection on social media: A data mining perspective. *Association for Computing Machinery*, 19(1):22–36, Jun. 2017.
- [11] D. Khattar, J. Goud, M. Gupta, and V. Varma. Mvae: Multimodal variational autoencoder for fake news detection. In *Association for Computing Machinery*, New York, NY, USA, May 2019.
- [12] A. Agarwal and P. Meel. Stacked bi-lstm with attention and contextual bert embeddings for fake news analysis. In *IEEE2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, Mar. 2021.
- [13] S. Gangireddy, D. P. C. Long, and T. Chakraborty. Unsupervised fake news detection: A graph-based approach. In *Association for Computing Machinery*, New York, NY, USA, Jul. 2020.

# Appendix

## GitHub

This project is aimed to analyze the sentiment of tweets that contain the word "vaccine" in two different ways and detect any possible misinformation inside them.

First of all, all files mentioned below must be placed in the same folder for the notebooks to properly work. It should look like this.



















	Collecting.ipynb	IPYNB File	5 KB
	Connecting.ipynb	IPYNB File	11 KB
	Fake News Detection.ipynb	IPYNB File	13,415 KB
	Libraries.ipynb	IPYNB File	3 KB
	Sentiment Analysis.ipynb	IPYNB File	13,739 KB
	Sentiment Prediction.ipynb	IPYNB File	9,502 KB
	airline	Microsoft Excel C...	3,619 KB
	coachella	Microsoft Excel C...	641 KB
	corona1	Microsoft Excel C...	10,255 KB
	corona2	Microsoft Excel C...	979 KB
	dilemma	Microsoft Excel C...	6,926 KB
	fake_news	Microsoft Excel C...	61,319 KB
	mytweets	Microsoft Excel C...	85,317 KB
	news	Microsoft Excel C...	96,317 KB
	true_news	Microsoft Excel C...	52,328 KB
	vaccine_vial	PNG image	7 KB
	wordcloud_usa	PNG image	86 KB
	wordcloud_world	PNG image	88 KB

Figure .1: All Files

In the GitHub repository they are separated in different folders only for the purposes of clarification and order.

## 1 Contents

This [GitHub repository](#) contains all the Jupyter Notebooks described in this report as well as any additional necessary files. More specifically it contains:

1. a folder named "Notebooks" containing the "Libraries", "Connecting", "Collecting", "Sentiment Analysis", "Sentiment Prediction" and "Fake News Detection" notebooks
2. a folder named "Datasets" containing all the datasets used in the "Sentiment Analysis" and "Sentiment Prediction" notebooks as well as the 500000 tweets dataset, all in separate sub-folders
3. a folder named "Images" containing three additional png images used in the "Sentiment Analysis" notebook

Here is a quick summary of the notebooks:

1. Libraries: Here we have all the imported libraries used in all the notebooks. This notebook is executed in every other notebook in order to not import again and again all the necessary libraries in each notebook. The user can change, add or delete any libraries in this notebook and this change will be transferred in all the notebooks.
2. Connecting: In this notebook we connect with the Twitter App and MySQL Workbench. The user does not need to change anything in this notebook as it is executed through "Sentiment Analysis" in order to just connect with the tools. The only tool he needs to download himself is the Workbench only in the case that he wants to access it. If the user does not want to use the Workbench to download new tweets and wishes to use a dataset like mine, he does not need the Workbench at all. If the user wants to connect another Twitter App instead of mine, all he has to do is change the keys at the start of the notebook. This notebook contains questions, that will be asked in "Sentiment Analysis", about how to navigate his Workbench, if of course he chooses to use it, such as if he wants to use existing databases and tables or new ones.



3. Collecting: This notebook is used to stream and collect new tweets by saving some specific features of the tweets such as their text, time and place of creation, the user's id and another feature is always extracted which is the polarity of the text through the use of the TextBlob algorithm. The polarity is always one of the three major sentiments, neutral, positive or negative. All these features create the new tweets dataframe object that will be used the rest of the project. All these downloaded tweets share some common words in their texts that the user chooses on his own, which in our case is the word "vaccine". It is again executed through "Sentiment Analysis" and the user does not have to do anything about it, only in the case that he wants to download new tweets and possibly extract even more features from the tweets. However for this to happen, the user must not only save the features in new variables but also change the code both in "Connecting" and "Collecting" notebooks in the respective SQL queries.
4. Sentiment Analysis: This notebook is the first of the three main processes. It involves the use of the TextBlob algorithm to take the polarity of the tweets and the display of this knowledge in many ways such as with a time series graph and pies for example in order to show the three sentiments. The whole purpose is to see the polarity that TextBlob extracted from these tweets. The user has to answer some questions in the first two cells but after that everything else can be executed without any interruption. A dataset of 500000 tweets, downloaded with this program, is also provided in case the user does not want to download new ones and it can be found in the "Datasets/Tweets" sub folder. Additionally, there are three png files in "Images" folder that are used in this notebook. The two wordclouds are generated throughout the whole process, saved in the current folder where this notebook is and then re-used in the final plot as images. In every execution of the cells that generate the wordclouds, new wordclouds are generated and saved on top of the old ones for any potential changes, which are then used in the final plot as already mentioned. The tweets are again used in the same graphs as before but this time only tweets made in the US are used. In the end we compare everything together to see potential differences between the world against only the US.
5. Sentiment Prediction: This notebook on the other hand uses a lot of classification algorithms and an LSTM model to predict, rather than extract, the polarity of the tweets. The input data for the classifiers are the datasets inside the "Datasets/Sentiment Prediction Datasets" sub folder. The data is then pre-processed, transformed into numerical

arrays, split into training and testing datasets and the training ones are inputted in the models. The models are then trained and each one predicts the polarity of the testing data to get their accuracy scores, while displaying some useful information for each algorithm. The user is free to add or delete any of the classification models or even change their parameters for potentially different results. The user is then asked to choose one of them and finally use it to predict the polarity of our tweets, insert it into the polarity column of a copy of the original dataset, display it again with some graphs, both for the world and US tweets, and finally compare the results with the results of "Sentiment Analysis", in order to see if TextBlob's output is any different from the classification algorithm's output, always regarding the polarity of the tweets.

6. Fake News Detection: Lastly this notebook is exactly the same with "Sentiment Prediction". The only differences are the input datasets, which are found in the "Datasets/Fake News Detection" sub folder, and that the purpose of this notebook is not to predict the polarity of the tweets but to find if they contain any misinformation. After the training of the algorithms and the user's choice for one of the them, the tweets get valued as fake or true and all the fake ones are discarded. Once again we display in graphs, both for the world and US tweets, the polarity of the tweets and compare the results with "Sentiment Prediction's" results, in order to see if the deleted tweets were part of a specific sentiment group or not.

And here is a quick summary of the datasets:

1. Tweets: As explained above it is a collection of 500000 tweets, always containing the word "vaccine", with the four extracted tweet features, all downloaded in August 2021, as well as the polarity the TextBlob algorithm has outputted.
2. Sentiment Prediction Datasets: All datasets in this sub folder are full with tweets of different topics with various columns of which we are only interested in their text as well as in their polarity that humans have valued tweet by tweet. We want to use these tweets as input data for the classification models and we needed texts that have been valued as negative, positive or neutral by real humans and not another algorithm or process. In that way the models train to somewhat understand which words output which emotion and then be able to predict the polarity of our own tweets.

3. Fake News Detection Datasets: All datasets in this sub folder are the same with the datasets in "Sentiment Prediction Datasets" in their basis. They are short articles about news around the world that instead of a polarity attached to them they have a label for being a fake or not story, which is also handled by a human. In the same notion we feed these datasets to the same classification algorithms and get trained models that can predict if our tweets spread misinformation or not.

## 2 Final Note

The main three notebooks of "Sentiment Analysis", "Sentiment Prediction" and "Fake News Detection" are, as described above, the core of all the project. They use a variety of graphs and algorithms for a lot of purposes but all these processes take time. In my own computer, which is heavily outdated, it took around ten hours for everything to fully be executed with a sample of 50000 tweets out of the original 500000. Smaller samples make the procedures faster and bigger ones not only slow it down but can even terminate the process due to ram limitations.

The most time consuming parts are the training of the algorithms or at least some of them, such as SVM. The same applies when the user wants to download new tweets. It is a slow and steady process and noone should expect thousands of tweets in mere minutes. Every notebook should be left to execute all cells and when it fully finishes to proceed to the next one, because "Sentiment Prediction" needs some information taken from "Sentiment Analysis" when it fully finishes the execution and "Fake News Detection" needs the same from "Sentiment Prediction", which means that not all three can run at the same time even if someone had a computer to do such a heavy task.

Feel free to experiment with any algorithm, any process or even use totally different datasets from what I used!