Sentiment Analysis of Tweets regarding Popular COVID-19 Vaccines

Sanchayan Bhunia[a](4849650)

[a] *bhunia.sanchayan@gmail.com*

1 Introduction

As the Covid-19 pandemic continues to persist in the world the only way out of this still seem to be vaccinating most of the population. Although many private and government organizations trying their best in order to convince people about the urgency of vaccinating themselves and their loved ones, hesitancy towards vaccine still present in the population sometimes due to the lack of knowledge or even in knowledgeable chunk of the population because of the fact that never in human history any vaccine have been discovered and rolled out for general purpose usage in the population. Different vaccines are developed in different ways by different manufacturers situated in different countries above all, their working principles vary widely. So, in an ideal world sentiment towards different vaccines should be different in the population.

Sentiment analysis on social media has been performed in multiple occasions earlier by organizations to find out opinion about many health related issues in past. Which brings us to the aim of this project which is to find out that difference in opinions from people living in different part of the world about popular vaccines in use.

2 Goal

The goal of this project is to collect tweets written in English, German and French from Twitter and applying some of techniques discussed during the course to clean the text and analyse the sentiment of those tweets, then by using some of the other techniques learnt from previous courses in my master's curriculum to find out the distributions of sentiments for different vaccines and also to find out any co-relation between different types of twitter users namely "verified" ones who are usually celebrates and well known people in their respective communities and non-verified ones, the general public. Other important goals of this project includes to find out heat-maps of vaccine hesitancy and vaccine decisiveness around the globe and if possible to find out which are the most popular vaccines in the different parts of the world.

3 Methodology

This project consists of a complete pipeline for not only analysing sentiment of tweets which is collected from Tweeter using its REST API but also statistical analysis of those tweets as well.

The project is divided into multiple self contained parts or modules starting from connecting with Tweeter's API, getting the Tweets, cleaning the Tweets, storing, sorting, analysing and tagging them with their sentiment and to further using them for statistical analysis.

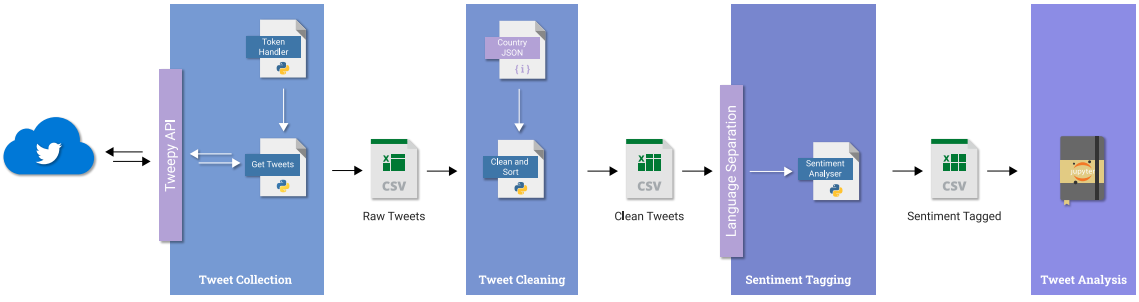


Figure 1: Project pipeline

The entire pipeline can be conceptualized from figure(1) where each colored rectangle represents each of the further steps in the process.

3.1 Structure of the Project Folder

The project folder consists of a "data" folder which contains raw data (`raw_tweets.csv`), cleaned data (`clean_tweets.csv`) and sentiment tagged data (`tweet\sentiment.csv`). The "key" folder contains my Twitter API tokens (`tokens.encrypted`), which is encrypted and the key to decrypt it will only be shared with the course instructor for security reasons.

Apart from the data and key files, the project folder contains *four* python scrips `apiToken.py`, `getTweets.py`, `cleanAndSort.py`, `sentimentAssignment.py` and a Jupyter Notebook file `Analysis.ipynb`. For each of the steps(1) in the project some libraries has been used which will be discussed further down the line.

3.2 Tweet Collection

The tweets are collected from Tweeter's API with the help of tweepy library which provides a Python wrapper on top of Tweeter's REST API. The first step is to authenticating with twitter API.

3.2.1 Authentication

The authentication is handled in `getTweets.py` script which upon run will ask the user if he/she is authorized to use 'Sanchayan's API Keys'.

- Upon `yes` entry in the console, the user will we asked for the secret key provided by Sanchayan which will be shared with the instructor.
- On contrary, upon `no` entry in the console, the user will be asked to get his own key from Twitter's Dev website and/or enter it manually through console.

Under the hood, the `getTweets.py` imports the "getToken" module and calls `getToken.getKey()` method which, in tern checks for the `tokens.encrypted` file inside "key" folder. If missing, it then prompts the user to enter the API key manually.

Upon success, either by decrypting the secret key or by user entering his personal tokens, it returns *four* keys-value pairs to `tweepy.OAuthHandler()` namely,

```
consumer_key: _____                    access_token: _____

consumer_secret: _____                 access_token_secret: _____
```

Which then creates an *authentication* object.

3.2.2 Creating API Object

In order to create an `api` object the `authentication` object needs to be passed into the argument of `tweepy.API()` method. An optional argument, `wait_on_rate_limit` is also expected to entered `True` for free account users.

3.2.3 Search for Tweets

Upon creation of the *api* object a search query can be made with `tweepy.Cursor()` by passing `(api.search, q = query, lang = "en", since = date_since)` these arguments and `.items(n)`. Which will return an iterable `tweet` object with `n` number of tweets in it.

Now we will make a list of our queries with a list of `search_words` which are the name of the vaccines e.g. `[''BioNTech'', ''Moderna'', ''Covishield'', ...]`. In order to get as many tweets as we can, we will also include synonyms of the vaccines namely the producing company or any given names as in different regions they might be referred by different names. We will also specify n to be $10,000$.

3.2.4 Collecting and Storing Tweets

With the list prepared, we will iterate through each of the vaccines in the list while making query with `tweepy.Cursor()` and will get iterable `tweet` objects in return. At the same time, we will iterate over `tweet` object related to each vaccine and store `'tweet.text'`, `tweet.user.location`, `tweet.user.verified`, `tweet.lang`, etc. in the `raw\tweets.csv` file, for all of the tweets where `tweet.text` is not blank.

3.3 Cleaning The Tweets

The methods for cleaning and sorting the tweets is implemented in the `cleanAndSort.py` module inside the project folder. In the previous step we have collected tweets about same vaccines with synonyms now we have to merge them to make the data-set compact and as in practice they are essentially the same vaccine, we will not lose any information in addition to that, we will perform two sub-steps, first step to clean the text of the tweet and another to fix the misspelled location names.

Every tweet and its related information is stored row-wise and `.csv` parser of python core package iterates over the file row-wise that is the reason why the main method, `cleanAndSort()` implemented in this module operates on an entire row, makes the necessary modifications and saves them in `clean_tweets.csv` file. The method also takes an argument `filter` which further filters out tweets without a minimum length, by default the value is set to 25 characters.

3.3.1 Merging Vaccine Names

The method that merges the vaccine names is `cleanAndSort.merge_vaccine_names()` which has the synonyms of vaccines stored in its local variable list. Upon passing an argument the method matches the value and returns the correct vaccine name.

3.3.2 Text Cleaning

The tweets often contains emojis, urls, email addresses, user names and numbers which is not necessary for the sentiment analysis process, that is why it is necessary to remove them from the tweets. The `cleanAndSort.clean_with_regex()` method cleans them using regular expressions and returns a cleaner text to be added to the `clean_tweets.csv` file.

Here is an example of an regular expression which removes `url` from the text.

```
    def remove_url(txt):
        return " ".join(re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", txt).split())
```

On the other hand, the emojis are sometimes used to analyse the sentiment of a tweet but the those analysers are not implemented in this project so it is necessary to remove the emojis which are usually represented in `utf-8` format and have their dedicated special characters. The following method removes them from the text.

```
    def deEmojify(text):

        text = str(text).replace(',',",")
        regrex_pattern = re.compile(pattern = "["
                    u"\U0001F600-\U0001F64F"  # emoticons
                    u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                    u"\U0001F680-\U0001F6FF"  # transport & map symbols
                    u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                                    "]+", flags = re.UNICODE)
        return regrex_pattern.sub(r'',text)
```

Similarly, the usernames, email addresses and the numbers are also removed from the text by the *cleanAndSort.clean_ with_ regex()* method.

3.3.3 Fixing Location Data

For a free account Twitter does not provide the location of the tweets rather we had to collect the location from the user's location which is manually entered by the user and sometimes does not correspond to an actual location. In that case it is necessary to remove those location entries for the statistical analysis to be fruitful.

On top of that, an user might enter the name of a street, the name of a city, the name of a province or some combination of all of those which makes it harder for hard code a location detector leave alone the fact that those names might have been misspelled. There are couple of choices while choosing python library for location tagging, geotext and geography. The main difference between these two libraries are the fact that geography uses NLTK for entity recognition and offers advanced features whereas, geotext uses regular expressions which is much faster than NER and easy to use. For our used case we would only require the name of the country and as further granularity being out of scope of the project, will use geotext.[Hu18]

The geotext library has a class called `GeoText` which given a sentence creates a dictionary object called `place` which has many keys, but we are interested in three of those keys `places.cities`, `places.countries` and `places.country\_mentions`. If a given string of words has name of a city, it will be stored as a string value of the `places.cities` key and similarly for the `places.countries`. But if it is a city, the corresponding country will not be stored in the value of the `places.country` key. In order to retrieve the name of the country we need to get the value of `places.country\_mentions` which is a set of (country abbreviate, occurrence) in the mentioned in the `string`.

A JSON file, `countries.json` in 'data' folder has been created with full names of all the countries along with their abbreviated forms as keys which loads into the main memory as a dictionary as soon the main program is launched. Which allows us to use the `country abbreviate` obtained from `places.country_mentions` as keys to get the full name of the counties and returned to the `cleanAndSort()` to be added to `clean_tweets.csv` file by the main method.

3.4 Sentiment Analysis and Tagging

The `clean_tweets.csv` file contains the cleaned tweet texts in German flagged with *de*, English with *en* and French with *fr*, which is then processed by the `sentimentAssignment.py` module in order to add sentiment tags to on each row which contains the tweets. Although The data is collected from `Tweeter` with search queries mentioning the language of choice and usually it returns tweets in that language itself, but it is always a great practice to cross verify the language of the tweets before performing sentiment analysis.

`sentimentAssignment` module performs two jobs first one being separating the tweets from pool of tweets with correct language and passing them to appropriate sentiment analyser.

3.4.1 Language-wise tweet Separation

Upon loading of `clean_tweets.csv` file, the main module `sentimentAssignment` looks for a list of target languages within the memory memory and then after it reads the `.csv` file row-by-row and whenever it finds a row flagged with the language in the list of targeted languages is asks for the `sentimentAssignment.confirmLanguage()` method to verify whether if the text of the tweet matches with the `flag`. And upon a `True` return, it passes the row to be processed by `addSentiment()` method.

`consirmLanguage()` is build using a python library for detecting language called langdetect which is a direct port of Google's language-detection library written in `java` which works on the principle of n-grams models with a naive Bayes classifier to classify the language of the argument string. Under the hood the language identification process is performed with in two steps, namely, language modeling and language classification.[Shu10] The library comes with pre-trained n-gram models for various languages. And whenever a string is entered it compares the occurance of the singlets in the string with its language n-grams and classifies the string based on most probable matching.

The `langdetect.detect()` method takes a string as an argument and returns the ISO code for the language of the string. This is used to build the `confirmLanguage()` method which takes two arguments `sentense`, `language` and returns `boolean`: `True` / `False` based on if the sentience matches the input language.

3.4.2 Sentiment Analyser

The project uses two different types of sentiment analysers namely, NLTK VADER sentiment analyser for texts in English language and TextBlob for texts German and French.

VADER: VADER, which stands for Valence Aware Dictionary for Sentiment Reasoning is a sentiment analyser which is sensitive to both polarity and intensity of sentiments. Under the hood, VADER relies on a dictionary words which maps lexical features to emotion scores. Emotion intensity or sentiment score of each word is measured on a scale from -4 to $+4$, where -4 is the most negative and $+4$ is the most positive. The midpoint 0 represents a neutral sentiment. Then the sentiment score of a sentience is calculated by summing over each of the sentiment score from the lexical dictionary. Then the score is normalized to assign a value in the range $(-1, 1)$.The sentiment analyser also takes into account the contextual elements of a sentence using *several* categorical heuristics namely, punctuation e.g. '!', capitalization of words and phrases, degree modifiers e.g. 'too much, so much etc.', shift in polarity using "but" and so on.[HG15]

TextBlob: The library is built on top of NLTK and assigns a polarity and subjectivity score to the input text. The numeric value for polarity describes how much the text is *negative* or *positive*. On the other hand, the subjectivity score describes how much the text is objective or subjective. Under the hood, TextBlob uses lexical sentiment dictionary like SentiWordNet.[BES10] which contains collection of words tagged with POS along with their polarity, subjectivity, intensity, and confidence.

For a sentence or phrase, TextBlob multiplies the subjectivity and polarity score and sums them up for each word in the text to calculate the polarity. But Whenever it finds an intensifier word in the text, it drops polarity and subjectivity and use the intensity to compute the sentiment of the sentence.

From a technical point of view, NLTK VADER is superior while analysing texts from social media because of the fact that it supports emoticons and some advanced features which is discussed in this article [BKJ19] which compares different Lexicon Based Approaches for Sentiment Analysis. That is the reason why *VADER* is used to analyse the sentiments in English. But *VADER* do not provide support for German and French. Where as *TextBlob* has a fork of the original original project for German textblob-de and textblob-fr French. The main advantage of analysing sentiments based on emoticons diminishes in our used case as we already have removed all emojis in the cleaning step.

In the project pipeline, upon confirming the language by using `confirmLanguage()`, the main method `sentimentAssignment()` passes the entire row containing the tweet as argument to `addSentiment()` which contains sentiment classifiers for each of the three languages.

- For English, VADER lexicon is downloaded using `nltk.download("vader_lexicon")` and the `SentimentIntensityAnalyzer()` method is used to get the sentiment score of the tweet.
- For German, `TextBlobDE()` method is used from `textblob_de` library to get the `polarity` score.
- For French, we first tag the POS with `PatternTagger()` and choose the `PatternAnalyzer()` imported from `textblob_fr` library and then use them as in `Blobber` imported from `textblob` to get the `sentiment` score from the `blob` object.

For each of the three languages the score we receive is in range from $(-1, 1)$ but for our analysis we will convert those values into absolute sentiment values $[-1, 0, 1]$ referring to *Negative*, *Neutral* and *Negetive* sentiments consecutively. This value is then added as a new entry to each row by the `addSentiment()` method itself. And finally, each of such rows is then added into `tweet_sentiment.csv` file by the main method.

3.5 Tweet Analysis

The next and final step in the project pipeline is actually analysing the tweets in order to fulfill the goals of the project. The statistical analysis has been performed in a Jupyter notebook named `Analysis.ipynb`. Various technical details, i.e. the python libraries and programming techniques.

3.5.1 Python Libraries

The data from `tweet_sentiment.csv` is then loaded into a pandas `dataframe`. Visualizations of the pi charts and bar charts are made using matplotlib library and the visualization on the maps is made using geopandas library. Although, using the `matplotlib` library is pretty standard, using the `geopandas` library requires special attention. Creating a geopandas `dataframe` object, a shape(`.shp`) of file of the world is needed along with various other files which can be found in `data/world_map` folder. Upon creation of the `datframe` object, it can be treated as a regular `pandas.dataframe` object.

3.5.2 Some Commonly used methods

- **Aggregations:** The dataframe object `df` contains *seven* columns and those are in the following order, vaccine, tweet, location, verified, language, follower and sentiment. In order to find the distribution of sentiments, tweets in various languages, distribution of language and location, distribution tweets about different vaccines etc. `dataframe.groupby()` method is used with appropriate arguments.
- **Filtering:** In order to filter out rows in a pandas dataframe, `dataframe.filter()` method is used along with the filter value.
- **Normalization:** For normalization, the following python function is used.

```
def normalize(df, feature_name):
    max_value = df[feature_name].max()
    min_value = df[feature_name].min()
    df[feature_name] = (df[feature_name] - min_value) / (max_value -
                                    min_value)
    return df
```

- **Joins:** If index of two dataframes are the same, a join can be made by using `dataframe.join()` method by mentionaing how the join is made (left or right).
- **Geopandas dataframe:** In order to create a geopandas map dataframe later to be plotted, a join has to be made on the `index="NAME"` with the target dataframe.

4 Observation

The data set consists of 166,761 tweets about 11 most popular vaccines e.g. Pfizer BioNTech, Moderna, AstraZeneca etc. and contains sentiment of three different categories positive (1), neutral (0) and negative (−1). The distribution of the tweets according to these categories are shown in the pie chart(2).

sentiment	-1	0	1
tweet	29664	84104	52993

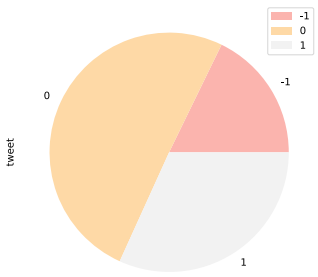


Figure 2: Distribution of tweets

The tweets are from all over the globe and tweeted in three different languages namely English, German and French. Any specific language is not tied to any specific country i.e. tweets form a country can be in different languages (figure 3a). The distribution of the tweets in languages are shown in the figure(3b).

language	de	en	fr
tweet	4614	158803	3344

(a) Tweet distribution in 3 language

```
language
de    [Germany, United States, Austria, Switzerland,...
en    [United States, Singapore, Canada, Japan, Arge...
fr    [France, United States, Tunisia, Morocco, Cana...
Name: location, dtype: object
```

(b) Tweeted in languages from countries

Figure 3: Distribution of tweets in different languages

The date-set also contains tweets tagged with the sentiments from bot "verified" and regular non-verified accounts flagged with **verified** (`True` / `False`), distribution of such is depicted in figure(4).

verified	False			True		
sentiment	-1	0	1	-1	0	1
tweet	27192	75671	47861	2472	8433	5132

Figure 4: Distribution of Tweets w.r.t. Verification Status

The distribution of tweets about individual vaccines are not uniform across the dataset. *Covaxin*, *Moderna* and *Pfizer BioNTech* has most of the tweets. The pi chart(5) shoes the distribution of every vaccine in the data-set.

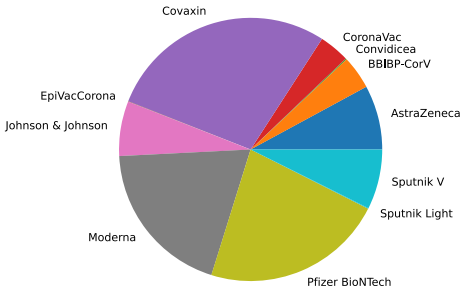


Figure 5: Distribution of Tweets w.r.t. Vaccines

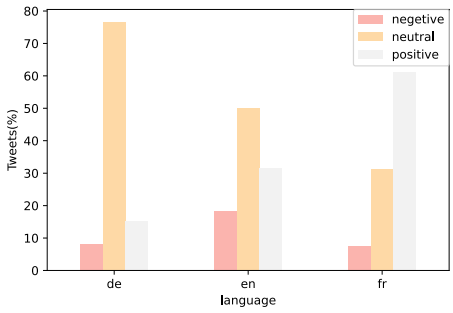
5 Analysis

5.1 Language Central Analysis

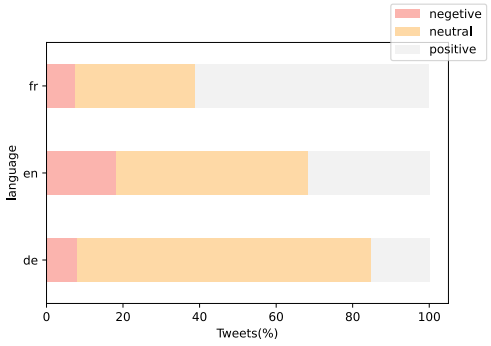
For tweets in each of the three languages in figure 3a we can further observe the distribution of sentiments depicted in figure 6a. Then we can calculate percentage of sentiments for each of the languages to find out if there is any language bias of sentiments, i.e. tweets in which languages express relatively greater vaccine hesitancy than others.

language	de			en			fr		
sentiment	-1	0	1	-1	0	1	-1	0	1
tweet	376	3537	701	29033	79522	50248	255	1045	2044

(a) Sentiment Distribution



(b) Percentage of tweets



(c) Relative sentiments

Figure 6: Sentiments in different languages

Figure 6b shows **percentage** of tweets expressing positive, negative or neutral sentiments where as, the relative sentiments are stacked for each language in figure 6c.

5.2 User Centered Analysis

The scope of this analysis is to find out what is there any shift in vaccine sentiments between the tweets from a user "verified" by Tweeter and of those who are not verified. Figure (7) shows the conditional probability of getting a positive sentiment form a `verified` and a non-verified account.

	sentiment
verified	
False	0.137131
True	0.165866

Figure 7: Cond. Probability

The conditional probability of getting a positive positive sentiment is $\frac{0.166}{0.137} \approx 1.21$ times higher in `verified` accounts than its non-verified counterpart.

5.3 Country Central Analysis

The scope of this analysis is to figure out the opinion about vaccination for each of the countries in the world, however the data-set has a collection of 150 countries.

5.3.1 Vaccine Hesitancy

In order to determine country-wise vaccine hesitancy, first we need to filter out the negative sentiments then calculate how much percentage of the tweets pose negative sentiments for each country. Then the value in normalized, made a join with geopandas dataframe and plotted as a heat-map.

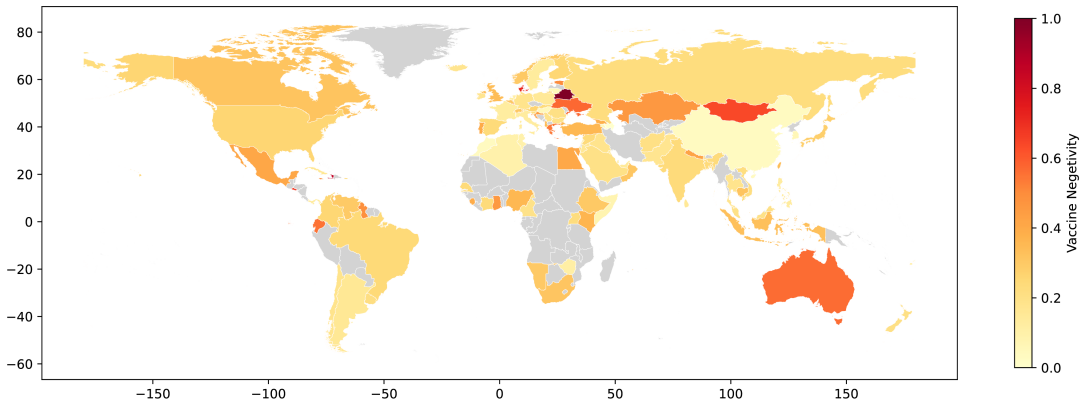


Figure 8: Heat-map of Hesitant Countries

Figure(8) shows the normalized value of vaccine hesitancy for available countries where, value ranges from $(0, 1)$ with 0 being the least hesitant represented by yellow and 1 being highly hesitant represented by red and every other colors are in-between.

5.3.2 Vaccine Decisiveness

For vaccine decisiveness we perform the same set of operations as performed with hesitancy but instead of filtering negative sentiment tweets, we filter out the positive tweets.

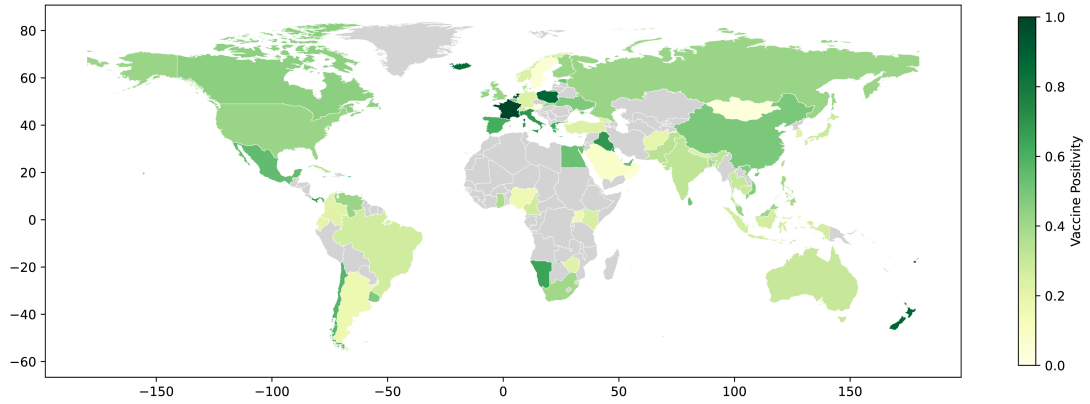
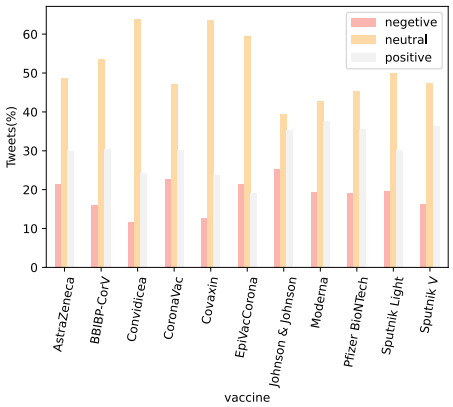


Figure 9: Heat-map of Decisive Countries

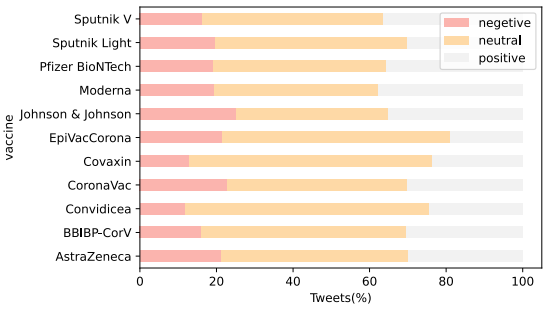
Figure(9) shows the normalized value of vaccine decisiveness for available countries. The value ranges from $(0, 1)$ with 0 being the least decisive represented by yellow and 1 being highly hesitant represented by green and every other colors are in-between.

5.4 Vaccine Specific Analysis

In this section we have focused on sentiments about each of most popular vaccines available in the data-set. For each vaccine we have calculated fraction of the positive, neutral and negative sentiments and multiplied it by 100 to get % of tweets and plotted the graphs using `matplotlib`.



(a) Percentage of tweets



(b) Relative sentiments

Figure 10: Sentiments about all vaccines

Figure(10a) depicts the percentile sentiment of each of the vaccines where as, the figure(10b) shows the relative sentiments on stacked bars.

5.4.1 Favourite Vaccine of Various Countries

In order to calculate the likelihood of any being the favourite vaccine of a country we need to calculate the marginal probability of that vaccine having positive or neutral sentiments in the concerned country multiplied by the frequency of tweets about that vaccine in the data-set. We perform this operation for all of the available vaccines in the dataset and obtain a pool of these probability values. From that pool, then we choose the vaccine with highest probability as the favourite vaccine of the concerned country.

Then we perform the above operation for every country to collect and eventually plot them using `Geopandas`.

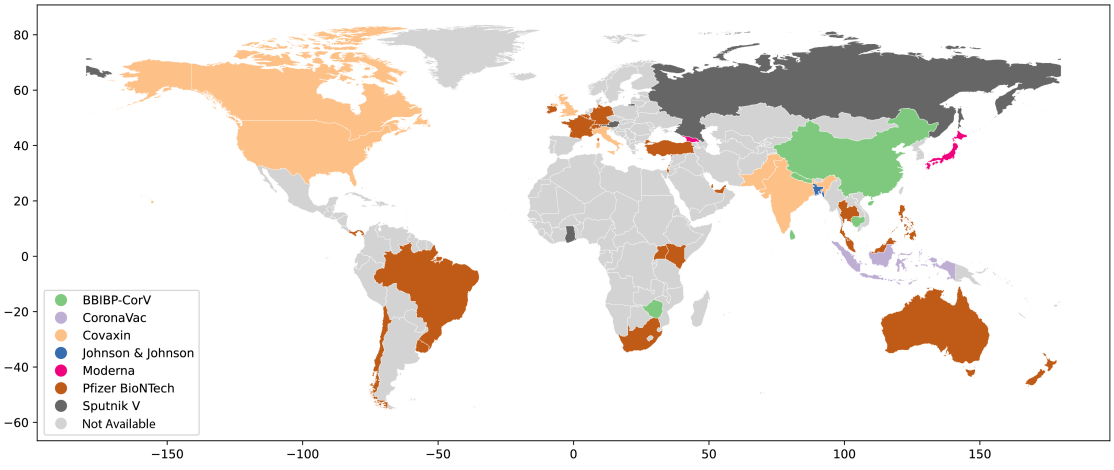


Figure 11: Favourite vaccine of Countries

Figure(11) shows the most probable popular vaccine of each country, i.e. the vaccine with highest probability of having positive and neutral sentiments.

6 Possible Sources of Error

There are two main types of error that has been encountered while doing this project based upon their point of origin.

6.1 Technical Error

The technical error mainly arises from the mis-labeling of the tweets.

- According to the observation made, the tweets about any vaccine do not represent sentiments about that vaccine all the time. Sometimes they are about availability about the vaccine in some hospital or some government policy about that vaccine which is barely pose any sentiment and thus classified as neutral statement. However, This problem can me solved, at least in theory by perspective analysis.
- Sometimes, the tweets are sarcastic and challenging for the sentiment analyser to analyse and categorize them. Although this problem is not very specific to this project rather a general problem for the Natural Language Processing field but this impacts the accuracy of the obtained results.
- As Tweeter does not provide tweet location information on free accounts, the location data has been collected from the user description where, people tend to be very creative sometimes quite figuratively, "Location: My Living-room" for example. In that case it is impossible to retrieve the location for any practical purpose.
- Often, even the location that contains actual places, are misspelled, although I have implemented some steps in order to collect even misspelled names of places, at times it is out of scope of the algorithm.
- One vaccine might have different names in different countries, although the method `getTweets()` implemented in this project, collects tweets about a vaccine in multiple popular alternative names, if a region has a different name for the vaccine which is not very popular, it flies out of the scope of the method.
- The sentiment analyser for English language is mature and has a huge user-base which allows it to be actively maintained and updated where as, the analysers, if any available for other languages like French or German, are not very mature and often buggy which might have mis-labeled tweets in those languages.

6.2 Statistical Error

Statistical error often arises from sampling bias and non-uniform availability of data of different categories. The reason of this non-uniformity is because even non-native English speakers among German and French speaking population prefer to tweet in English rather than their mother tongue.

Rater than taking a gross approach to determine the Vaccine Hesitancy or the Favourite vaccine of a country, the probabilistic approach has helped a lot to compute them with certain probability and avoid certain degree of bias during prediction. But this approach has its own limitation namely, the uncertainty coupled with the computation itself.

7 Conclusion

In conclusion, it can be said that the proposed pipeline was successful to fulfill the goals of this project while collecting about $150,000$ tweets in three different languages and with the help of `langdetect`, `geotext` and of two of the most popular semantic based sentiment analysis libraries `vader` and `textblob` to find out the distribution of the sentiments about most popular vaccines currently in use. We also managed to visualize out the hesitancy toward vaccines in different parts of the world and also figured out popular vaccines of several countries around the globe.

Further studies can be made on this topic by summarizing negative tweets for each of the vaccines and finding out the most common keywords in those tweets which made them to portray negative sentiments. This project gave me an opportunity to dive deeper into the field of contextual (multiple agent based) sentiment analysers which although I did not implement in this project but tried to build one with very limited success for the time being which, in future, might be implemented in this project by replacing the semantic based analysers.

7.1 Future Prospect

The proposed idea is, upon passing a sentence or a paragraph or tweets of related topics, initially an Universe Agent is created which keeps track of other agents in the system. The sentence in then tagged with POS tagger and based if there are words in the category of *"NN"*,*"NNS"*, *"NNP"*, *"NNPS"*, *"PRP"* and *"PRP$"* agents are created if they already do not exist in the system with a name similar to the word that has appeared.

Every new-born agent has a neutral sentiment believe about every other agents in the system, but the believe keeps evolving with addition of every new message from other agents. This is the reason why this algorithm is best if there are long paragraph of texts or tweets of similar topics. Later at some point the agent might be asked to tell about his believe about some other agent.

References

[BES10] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, volume 10, pages 2200–2204, 2010.

[BKJ19] Venkateswarlu Bonta, Nandhini Kumaresh, and N. Janardhan. A comprehensive study on lexicon based approaches for sentiment analysis. pages 1–6, 03 2019.

[HG15] C.J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. 01 2015.

[Hu18] Yingjie Hu. Geo-text data and data-driven geospatial semantics. *Geography Compass*, 12(11), September 2018.

[Shu10] Nakatani Shuyo. Language detection library for java, 2010.