Sentiment Analysis and Racism Detection in Tweets related to Covid-19 and Covid-19 vaccines

CIS 400 Principles of Social Media and Data Mining

Project Report

Chenmeinian Guo
Jinchao Zhao
Timothy Liu
Jinzhi Cai
Xinyue Mao

Contents

1 Introduction

This project intends to do sentiment analysis related to the COVID-19 vaccines in Twitter with traditional methods and classification methods (logistic regression and support vector machine). With the exponential growth of social networks, the technique of sentiment analysis has been frequently applied to analyze the user's opinions. With the usage of linguistic mechanisms, sentiment analysis contributes to processing data in social networks by providing proper computer-based definitions of the words. The topic of the COVID-19 vaccine is pretty popular these days, and there is tons of data available on Twitter. We would like to find what attitudes the users have, what the geological distribution, timeline, and other potential features look like, how the sentiment of users will affect the current trend, and what is the logic prediction based on the sentiment information we have collected.

2 Gather User Information

2.1 Use Crawler to collect information

The first stage for this project is to collect data from the tweet. In tweet, it provide a lot of different user identity in the return data. The key for this part is to find the correct part of data that should be use for this project and help the other team member to reduce the possible noise from the data source. In order to gather the fresh information about the topic the group discuss, the stream API is required. The stream API from twitter allow program to gather information on the topic [**TwStream**]. The second step is to explore the relative user more. However, most of the information that create by user can be faked. Therefore, the following count and friend count is the few of the data that user can not fake on. On the other hand, program also collect tweet the user sent. By applying natural language analysis can better improve the accuracy of the speech the person do on the topic.

TweetID	USERID	Source	Data	CreatedDate
105	470	Searched	RT @normanswan: Isra	2021-04-26 06:36:13
106	214	Searched	RT @sanchezcastejon:	2021-04-26 06:36:13
107	930	Searched	@DAupperlee @Michiga	2021-04-26 06:36:15
108	153	Searched	RT @JKJAVMY: 91-year	2021-04-26 06:36:16
109	556	Searched	University of Queens	2021-04-26 06:36:18
110	499	Searched	Lol.why lobby is not	2021-04-26 06:36:18
111	656	Searched	RT @JILWorldwide: We	2021-04-26 06:36:19
112	134	Searched	Over 14.19 crore Cov	2021-04-26 06:36:20
113	544	Searched	RT @DrEricDing: NEW—	2021-04-26 06:36:23
114	279	Searched	RT @tvgurl: Today, o	2021-04-26 06:36:24
115	386	Searched	RT @AskAnshul: A lob	2021-04-26 06:36:24
116	255	Searched	RT @rising_serpent:	2021-04-26 06:36:24
117	208	Searched	(Johnson & Johns	2021-04-26 06:36:25
118	391	Searched	I walked into my gar	2021-04-26 06:36:27
119	415	Searched	RT @amritabhinder: W	2021-04-26 06:36:29
120	204	Searched	I had a fever, chill	2021-04-26 06:36:34
121	134	Searched	RT @BrankoMilan: If	2021-04-26 06:36:36
122	246	Searched	RT @LawrenceSellin:	2021-04-26 06:36:36
123	776	Searched	idc what the cdc say	2021-04-26 06:36:37
124	564	Searched	RT @Kishoreciyer1: J	2021-04-26 06:36:38
125	443	Searched	RT @JKJAVMY: 91-year	2021-04-26 06:36:39
126	769	Searched	RT @cov19treatments:	2021-04-26 06:36:40
127	139	Searched	RT @JonahBlank: This	2021-04-26 06:36:44
128	232	Searched	RT @adam_tooze: Vacc	2021-04-26 06:36:45
129	322	Searched	RT @ParisDaguerre: I	2021-04-26 06:36:48
130	144	Searched	RT @Reuters: Israel	2021-04-26 06:36:49
131	226	Searched	RT @Kishoreciyer1: J	2021-04-26 06:36:51
132	584	Searched	RT @wtfex: Adar Poon	2021-04-26 06:36:52
133	641	Searched	RT @Kishoreciyer1: J	2021-04-26 06:36:52
134	162	Searched	RT @lindyli: Got my	2021-04-26 06:36:54
135	401	Searched	RT @LqLana: I know s	2021-04-26 06:36:55

Figure 1: Collected Data

2.2 Database Design

Due to COVID-19 restriction, a well design database is require for this design. It provide data access to all of the member in the team. Base on the previous analysis, the database design is as following.

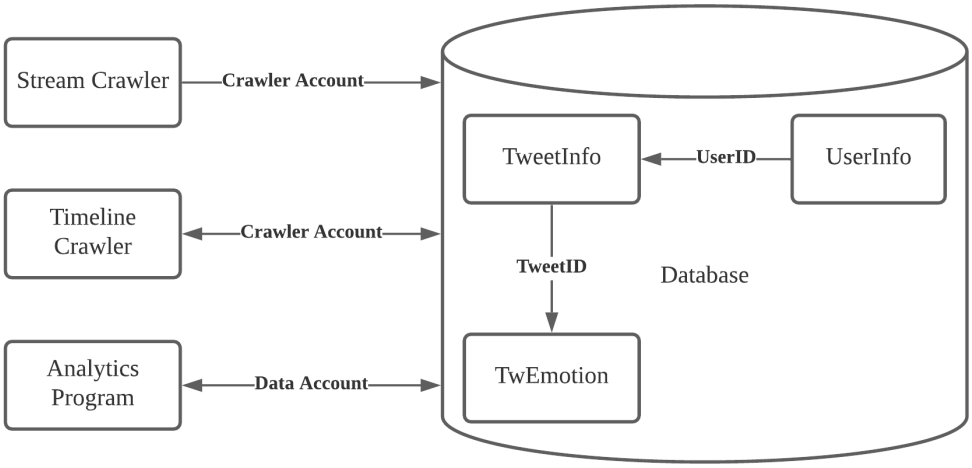

Figure 2: Database Design

3 Data preprocessing

3.1 Background

When we talk about data, we usually think of some large datasets with huge number of rows and columns. Raw data collected from first step cannot be process by machine language easily. Machine language doesn't understand human's natural language. In order to process the data. I'll be doing the data pre processing in this project.

3.2 Import the methods we need for this part

```
[ ]: # !pip install pymysql
     # !pip install tweet-preprocessor
     import nltk
     from nltk.tokenize import word_tokenize
     from nltk import PorterStemmer
     from nltk import WordNetLemmatizer
     from nltk.corpus import stopwords
     import re
     import pymysql
     import pandas as pd
     import preprocessor as p

     from sklearn.feature_extraction.text import TfidfVectorizer
     nltk.download('punkt')

     nltk.download('wordnet')
```

Figure 3: Methods

Most of methods here I used is from NLTK. It's called Natural Language Toolkit. I also imported pymysql for import our raw data from the database on AWS.

3.3 Connect to the AWS

```
[ ]: db=pymysql.connect(host="database-1.ct9nsqndasrh.us-east-1.rds.amazonaws.com",
                        port=3306,user="DataUser",
                        password="v9fyJ2JwcWcEnb6a",
                        database="FinalProject")
     cursor = db.cursor()
     sql = "select * from TweetInfo"
     cursor.execute(sql)
     # 获取所有记录列表
     results = cursor.fetchall() # catch tweetinfo
```

Using the pymysql. Connect to the server by put the host, port, username, password, and the name of the data base in to collect the data. The sample result looks like this:

[5]:

	text	date
0	RT @normanswan: Israel has just done a deal wi...	2021-04-26 06:36:13
1	RT @sanchezcastejon: Spain is commited to equa...	2021-04-26 06:36:13
2	@DAupperlee @MichiganHHS Covid according to th...	2021-04-26 06:36:15
3	RT @JKJAVMY: 91-year-old Dato Dr Gurmukh Singh...	2021-04-26 06:36:16
4	University of Queensland scientists have been ...	2021-04-26 06:36:18

Figure 4: Sample result

3.4 Process the data

```
[ ]: remove_non_alphabets = lambda x: re.sub(r"[^a-zA-Z']", " ", x ) # rmove non alphabets

     tokenize = lambda x: word_tokenize(str(x)) # tockenizer

     ps = PorterStemmer()
     stem = lambda w: [ps.stem(x) for x in w] # stemmer


     ler = WordNetLemmatizer()
     lemmatizer = lambda x: [ler.lemmatize(word) for word in x] # lemmatizer
```

```
[ ]: %%time
     df['text'] = df['text'].apply(lambda x: p.clean(x)) # remove twitter speicialy chars
     df['text'] = df['text'].apply(remove_non_alphabets)
     df['text'] = df['text'].apply(tokenize)
     df['text'] = df['text'].apply(stem)
     df['text'] = df['text'].apply(lemmatizer)
     df['text'] = df['text'].apply(lambda x: " ".join(x)) # merge cleaed list
```

Figure 5: Create and apply the functions

3.4.1 Remove special characters

After we import all of the methods and raw data we are going to need. We can start to process the data. The first thing here i did is remove all the special characters that we don't need to sentiment analysis. Function 'remove non alphabets' here will remove all that characters that is not A-Z or a-z. By apply this function. It will be much easier to process the data.

3.4.2 Tokenization

Tokenization means Breaking a stream of characters into tokens. Tokenization is important for computer because this is a important step let computer read the sentence by breaking down each

words. I am using NLTK here doing the tokenization. Simply apply the tokenize package to the sentence I need to process.

For example: Someone post a tweet on Twitter and the content is: "The weather today is good"

After we did tokenization, the sentence will be: "['The', 'weather', 'today', 'is', 'good']" much easier for computer to process the data.

3.4.3 Remove the stop words

```
[ ]: from sklearn.feature_extraction.text import TfidfVectorizer # tokenizer
vectoizer = TfidfVectorizer(stop_words= "english", max_features = 100) # remove stopwords, max features set as 100
vectoizer.fit(df.text)
vecs = vectoizer.transform(df.text) # get vector
```

Figure 6: Remove the stop words

In computing, stop words means the words which are filtered out before or after processing of natural language data. Though "stop words" usually refers to the most common words in a language.

Example of stop words in English: "Is, are, my, was, were, the, etc."

We want to make the sentence we are going to process shorter. We would not want these words to take up space in our database, or taking up valuable processing time. NLTK has a list of stop words of 16 different languages inclues English.

Example of remove stop words: "['The' 'weather' 'today' 'is' 'good']"

will becomes "['weather' 'today' good']"

3.4.4 Deal with stemming words

The idea of stemming is a sort of normalizing method. Many variations of words carry the same meaning, other than when tense is involved. The reason why we stem is to shorten the lookup, and normalize sentences. Consider this: "I was taking a ride in the train" and "I was riding in the train". The almost means the same. Someone was taking the train but the ways in natural language display this could be quite different. In order to normalize, let computer understand it more clearly. I used function from NLTK to deal the the stemming words.

3.4.5 Lemmatize

There are tons of ways we could express good in natural language such as English, we can say: "good, awesome, perfect, wonderful, etc." In order to let computer understand what is each word means. I am going to do the last step of data pre-processing here which is lemmatize. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meaning to one word. For example: Plural nouns in English always ends in 's' or 'es'. String 'books' and 'book' means the same in English. Lematize can make 'books' become 'book'. Friendly for data processing like sentiment analysis. I used the module from NLTK here you can see from the picture.

3.5 Output

At the end of my part — data pre-processing. I'll output a cleaned data that twitter speical characters is removed, Tokenized, and lemmatized.

```
df['text'][:5] # check cleaned data
```

```
0    israel ha just done a deal with moderna for a ...
1    spain is commit to equal and univers access to...
2    covid accord to the law of averag will be rd o...
3    year old dato Dr gurmukh singh receiv hi first...
4    univers of queenslandscientist have been re en...
Name: text, dtype: object
```

Figure 7: Sample Output

4 Label Data

4.1 Catch data from Database

Catch the required data from the database and receive the returned results. Catching data from the database created before. I achieve this goal through using functions from Python MySQL. First, using the function pymysql.connect to the database to get the data that we need. Second, during the entire analysis process, we do not need to analyze the data information reflected in each piece of data because of the large amount of data, then I used the function db.cursor() and picked part of the dataset as our dataset TweetInfo. Finally, using function cursor.execute() to query the data in the TweetInfo table, and use function fetchall() to receive and return all the results.

```
db=pymysql.connect(host="database-1.ct9nsqndasrh.us-east-1.rds.amazonaws.com",
                   port=3306,user="DataUser",
                   password="v9fyJ2JwcWcEnb6a",
                   database="FinalProject")
cursor = db.cursor()
sql = "select * from TweetInfo"
cursor.execute(sql) # get a list of all records
results = cursor.fetchall() # catch tweetinfo
```

Figure 8: 4.1

4.2 Dataframe

Observe and select the desired columns. After fetching, the data is transformed into tabular data through the function pd.DataFrame to make the dataset more stratforward.

```
                0           1           2  \
0             105   470797902    Searched
1             106  2147483647    Searched
2             107   930163998    Searched
3             108  1539896714    Searched
4             109   556834642    Searched
...           ...         ...         ...
3284988   3285094         246  ByTimeLine
3284989   3285095         246  ByTimeLine
3284990   3285096         246  ByTimeLine
3284991   3285097         246  ByTimeLine
3284992   3285098         246  ByTimeLine

                                                         3                   4
0        RT @normanswan: Israel has just done a deal wi... 2021-04-26 06:36:13
1        RT @sanchezcastejon: Spain is commited to equa... 2021-04-26 06:36:13
2        @DAupperlee @MichiganHHS Covid according to th... 2021-04-26 06:36:15
3        RT @JKJAVMY: 91-year-old Dato Dr Gurmukh Singh... 2021-04-26 06:36:16
4        University of Queensland scientists have been ... 2021-04-26 06:36:18
...                                                    ...                 ...
3284988  @brad_frost @NikkitaFTW @colmtuite The caveat ... 2021-03-12 00:47:35
3284989  @brad_frost @NikkitaFTW @colmtuite Even for pe... 2021-03-12 00:44:45
3284990  @brad_frost @NikkitaFTW @colmtuite Respectfull... 2021-03-12 00:42:09
3284991  @infil00p I wish they'd prioritize "essential ... 2021-03-12 00:24:04
3284992  @infil00p All the science I've seen is pointin... 2021-03-12 00:22:57

[3284993 rows x 5 columns]
```

Figure 9: 4.2

Next, by observing the data in the table, getting rid of the useless columns will make the analysis more straightforward. I removed the useless columns and only left the 3 columns that shows the details about Tweet Information. Set the names of these 3 columns to ['id', 'text', 'date'], the result as Figure 4.3.

```
              id                                                text  \
0            105  RT @normanswan: Israel has just done a deal wi...
1            106  RT @sanchezcastejon: Spain is commited to equa...
2            107  @DAupperlee @MichiganHHS Covid according to th...
3            108  RT @JKJAVMY: 91-year-old Dato Dr Gurmukh Singh...
4            109  University of Queensland scientists have been ...
...          ...                                                ...
3284988  3285094  @brad_frost @NikkitaFTW @colmtuite The caveat ...
3284989  3285095  @brad_frost @NikkitaFTW @colmtuite Even for pe...
3284990  3285096  @brad_frost @NikkitaFTW @colmtuite Respectfull...
3284991  3285097  @infil00p I wish they'd prioritize "essential ...
3284992  3285098  @infil00p All the science I've seen is pointin...

                        date
0        2021-04-26 06:36:13
1        2021-04-26 06:36:13
2        2021-04-26 06:36:15
3        2021-04-26 06:36:16
4        2021-04-26 06:36:18
...                      ...
3284988 2021-03-12 00:47:35
3284989 2021-03-12 00:44:45
3284990 2021-03-12 00:42:09
3284991 2021-03-12 00:24:04
3284992 2021-03-12 00:22:57

[3284993 rows x 3 columns]
```

Figure 10: 4.3

4.3 Analysis & labeling

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

The sentiment property returns a named tuple that contains polarity and subjectivity. For here, polarity score will only be used to make objective analysis. The polarity score is a float number within the range of [-1.0, 1.0]. The closer the score is to -1.0 means that people are extremely against the vaccine, and the closer the score is to 1.0 means that people are very willing to accept the vaccine. Also, it is still possible that the polarity score is 0, which means neutral and people neither disagree nor accept the vaccine.

This process can be achieved through code df['text'].map(lambda x: TextBlob(x).sentiment.polarity). The polarity score for the text in each tweet is as Figure 4.4

	id	text	date	score_Textblob
0	105	RT @normanswan: Israel has just done a deal wi...	2021-04-26 06:36:13	0.000000
1	106	RT @sanchezcastejon: Spain is commited to equa...	2021-04-26 06:36:13	0.000000
2	107	@DAupperlee @MichiganHHS Covid according to th...	2021-04-26 06:36:15	-0.062500
3	108	RT @JKJAVMY: 91-year-old Dato Dr Gurmukh Singh...	2021-04-26 06:36:16	0.250000
4	109	University of Queensland scientists have been ...	2021-04-26 06:36:18	0.200000
5	110	Lol.why lobby is not working for Moderna or J&...	2021-04-26 06:36:18	-0.200000
6	111	RT @JILWorldwide: We heard you ask,so here's o...	2021-04-26 06:36:19	0.000000
7	112	Over 14.19 crore Covid-19 vaccine doses admini...	2021-04-26 06:36:20	0.000000
8	113	RT @DrEricDing: NEW—Biden WH is sending suppli...	2021-04-26 06:36:23	0.000000
9	114	RT @tvgurl: Today, on my 41st birthday, I just...	2021-04-26 06:36:24	0.000000
10	115	RT @AskAnshul: A lobby is urging to import Pfi...	2021-04-26 06:36:24	0.000000
11	116	RT @rising_serpent: Can you point to a single ...	2021-04-26 06:36:24	0.107143
12	117	(Johnson & Johnson COVID-19 Vaccine Side E...	2021-04-26 06:36:25	0.516667
13	118	I walked into my garden yesterday and thought ...	2021-04-26 06:36:27	0.000000
14	119	RT @amritabhinder: While US had restricted exp...	2021-04-26 06:36:29	0.000000

Figure 11: 4.4

4.4 Data upload to database

By using the functions in Figure 3.5, dataset structure can be viewed, and the analyzed and marked data will also be uploaded to the database at the same time.

```
db=pymysql.connect(host="database-1.ct9nsqndasrh.us-east-1.rds.amazonaws.com",
                   port=3306,user="DataUser",
                   password="v9fyJ2JwcWcEnb6a",
                   database="FinalProject")
cursor = db.cursor()

sql = 'show tables;'
cursor.execute(sql)
rowList = cursor.fetchall() # get connection
```

```
db=pymysql.connect(host="database-1.ct9nsqndasrh.us-east-1.rds.amazonaws.com",
                   port=3306,user="DataUser",
                   password="v9fyJ2JwcWcEnb6a",
                   database="FinalProject")
cursor = db.cursor()

sql = "SHOW CREATE TABLE `TwEmotion`;" # see table structure
st = cursor.execute(sql)
cursor.fetchall()
```

```
# Create a new record
for i, row in df.iterrows():
  sql = "INSERT INTO `TwEmotion` (`TweetID`, `TwEmotionScore`) VALUES (%s, %s)"
  cursor.execute(sql, (int(row['id']), row['score_Textblob'])) # to mysql

db.commit() # commit
```

Figure 12: 4.5

5 Data Analysis

Chenmeinian Guo is responsible for data analysis. Two types of analysis were conducted in this part. Codes and datasets for my part could be found in my github page:
https://github.com/guochenmeinian/CIS400-Project

5.1 Data

With the data collected and pre-processed by my teammates, along with Python package, csv, I converted datasets to dictionary data structure for convenience. I further added one dataset for emotion value, which is explained in 5.2.

```
# tweets_sentiments: {'TweetID': ['UserID', 'Source','Text', 'CreatedDate', 'Sentiments']}
tweets_sentiments = load_data("tweetsentiments.csv")

# tweens_emotions_NLTK: {'TweetID': ['emotion_score']}
tweets_emotions_NLTK = load_data("tweetemotion.csv")

# users: {'UserID': ['UserName','followers_count', 'friends_count', 'FriendID']}
users = load_data("userinfo.csv")
```

Figure 13: Three Datasets

5.2 Comparison of algorithms

In this part I compared results between scoring method(from part 3) and transformer classification method on sentiment analysis. I used python package, matplotlib, to provide a visual representation and give the readers a better sense of how much these two methods differ.

5.2.1 NLTK Algorithm

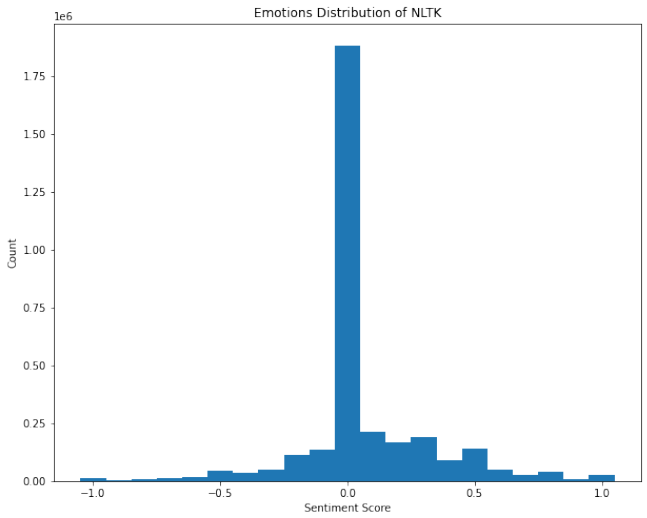

Figure 14: Emotion Distributions of NLTK

This is covered in **Part 3** by my teammate **Jinchao Zhao**. For further details, please check part 3 above. Using this algorithm, most of the tweets have 0 sentiment scores. It is very hard to study correlation between sentiments and other features with this algorithm.

5.2.2 Transformer Algorithm

The transformer library downloads pretrained models for Natural Language Understanding (NLU) tasks, such as analyzing the sentiment of a text, and Natural Language Generation (NLG), such as completing a prompt with new text or translating in another language. Using this algorithm, most of the tweets are classified into either 'positive' or 'negative' with high confidence. One thing worth noting is that NTLK and Transformer are actually completely different methods (Scoring and Classification), and the latter algorithm (binary classification method) is more suitable for my later work rather than scoring methods.

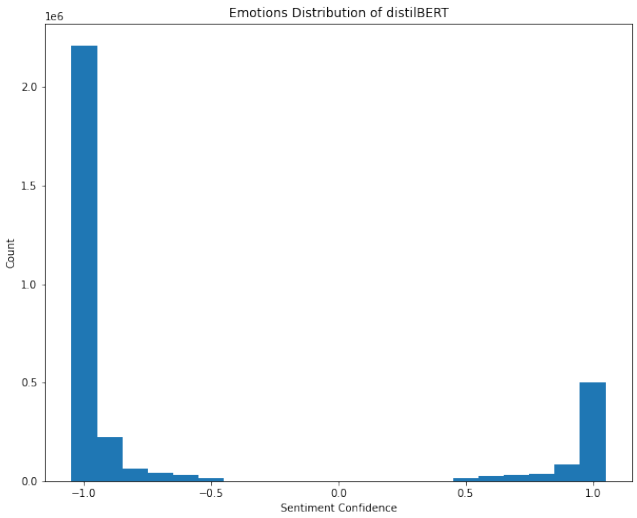

Figure 15: Emotion Distributions of Transformer

5.2.3 Example

```
tweet: RT @AskAnshul: A lobby is urging to import Pfizer vaccine since January.
They just want India to buy Pfizer. They don't care about its cost…
NLTK_score: 0.0
distilBERT_Result: {'label': 'NEGATIVE', 'score': 0.998418927192688}
```

Figure 16: Performance Example

5.3 Time Series Analysis

5.3.1 Before Plot

```python
def judge_sentiment(score):
    if score > 0.95:
        return 'Positive'
    elif score < -0.95:
        return 'Negative'
    else:
        return 'Neutral'

def count_sentiment(tweets_sentiments, sentiments_distilBERT):
    result = {}
    for k in tweets_sentiments.keys():
        d_t = dt.datetime.strptime(tweets_sentiments[k][3], '%Y-%m-%d %H:%M:%S')
        day = d_t.date()
        score = sentiments_distilBERT[k]
        if day not in result.keys():
            result[day] = {'Positive': 0, 'Negative': 0 ,'Neutral':0}

        result[day][judge_sentiment(score)] += 1

    return result
```

Figure 17: Python Code

As shown from Figure 3, the distribution of Transformer emotion value, most tweets are categorized as positive or negative. Therefore, I set a threshold to consider tweet with confidence score > 0.95 as **neutral** tweet, confidence score < -0.95 as **positive** tweet, and the rest being **netural**. Then, in order to analyze time, I extracted time property of each tweet and imported datetime package to store these data. Finally, I plot the sentiment over time graph, with a yellow regression line indicating the tendency of the sentiment ratio since Covid-19 (from 2020-3-1 to 2021-3-1).

5.3.2 Graph

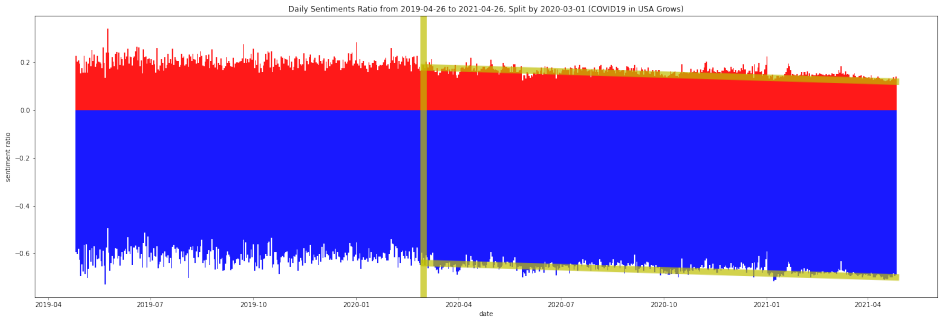

Figure 18: Sentiment Over Time Analysis

5.4 Clustering Analysis

5.4.1 Procedure

Machine learning is a method of data analysis that automates analytical model building, and it enables analysis of massive quantities of data. K-means clustering, as one of the most widely used upsupervised machine learning technique, tries to group similar kinds of items in form of clusters. In this part, I used K-means of euclidean distance to clustering users. However, since K-means is super sensitive to distances, I log the followers/friends counts to mitigate this. My aim for this part of analysis is to determine whether sentiments are related with user's popularity(followers/friends).

5.4.2 Elbow Method

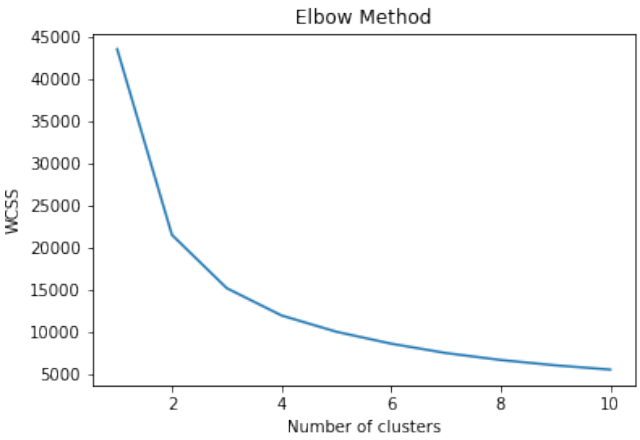

Figure 19: Elbow Graph

Elbow is one of the most famous methods by which you can select the right value of k and boost your model performance. According to Wikipedia, in cluster analysis, the elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use. It calculates the sum of the square of the points and calculates the average distance.Based on the figure on the left, we could conclude that k=4 since it seems to be an elbow.

5.4.3 Implementation

```python
# elbow method to select k
# We measure Within Cluster Sum of Squares (WCSS) for different k, then select a k at 'elbow' of plot
def elbow_k_selecting(points):
    X = np.array(list(points.values()))

    wcss = []
    for i in range(1, 11):
        kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
        kmeans.fit(X)
        wcss.append(kmeans.inertia_)
    plt.plot(range(1, 11), wcss)
    plt.title('Elbow Method')
    plt.xlabel('Number of clusters')
    plt.ylabel('WCSS')
    plt.savefig('elbow k means.png')
    plt.show()
```

```python
# K-mean clustering
def preform_Kmeans(points, k = 4):
    results = {}

    X = np.array(list(points.values()))
    kmeans = KMeans(n_clusters = k, random_state = 0).fit(X)
    labels = kmeans.labels_
    userids = list(points.keys())
    for i in range(len(userids)):
        results[userids[i]] = labels[i]

    return kmeans.cluster_centers_, results
```

Figure 20: Machine Learning Python Code

5.4.4 Graph

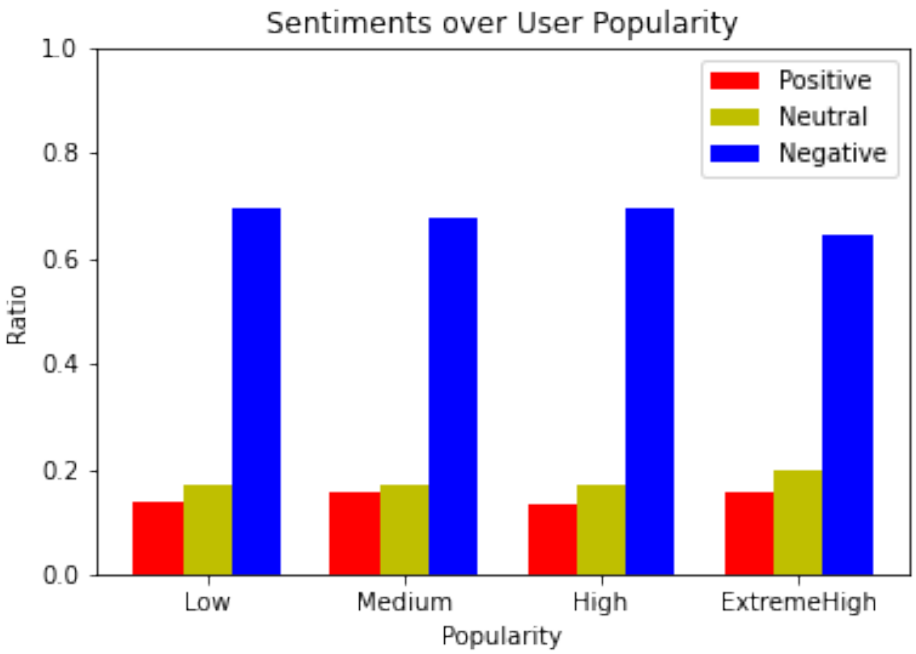

Figure 21: Clustering Analysis Graph

6 Racism Speeches Detection using Classification

Xinyue Mao is responsible for this part.

Background: Because of the gene differences, different races have different physical responses to vaccines. Also, as the recent research showed, Black participants living in the U.S. were less likely to receive a vaccine than White participants [**ra2**]. I thought there might be some racism problems related to the vaccines. Since it is not easy to get the race information from a user profile in Twitter, I decided to create a classification to detect racism speeches in Twitter.

The task is to classify racist tweets from other tweets. We use supervised learning with labeled data to train the model using Support Vector Machine and Logistic Regression. Then apply the model to the collected Tweets and get racism-scores for each Tweet.

```
In [11]:
# split training and testing data
tweetMatrix = np.array(tweetMatrix)  # shape (31962, 15915)
tweetMatrix = tweetMatrix/255
X_train, X_test, y_train, y_test = train_test_split(tweetMatrix, label, test_size=0.3, random_state=42)
```

Figure 22: split data

6.1 data for training and testing the model

The data used for training and testing the model (included in the zipped folder) is retrieved from Kaggle [**datawebsite**]. The data includes 31962 labeled Tweets (2242 racism Tweets and 19720 normal Tweets) in total.

I use the code in Part2 (by Timothy Liu) to pre-process the data, including cleaning, tokenization, stopwords, lemmatization, etc.

After the data pre-processing, I vectorize the sentences to get a big matrix. The big Tweet matrix is (31962*15915) which means there are 31962 samples (i.e. Tweets) and 15915 features (i.e. different words) in the data set. The data is splitted randomly into a training set and a testing set with ratio 7:3. Formally, given a training sample of tweets and labels, where label '1' denotes the tweet is racist and label '0' denotes the tweet is not racist.

6.2 Logistic Regression

The logistic model (or logit model) is used to model the probability of a certain class or event existing such as racist or not racist.

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable. In regression analysis, logistic regression is estimating the parameters of a form of binary regression. Mathematically, a binary logistic model has a dependent variable with two possible values, which is represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables; the independent variables can each be a binary variable or a continuous variable.

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

p = probability

$\frac{p}{1-p}$ = corresponding odds

Figure 23: logit function

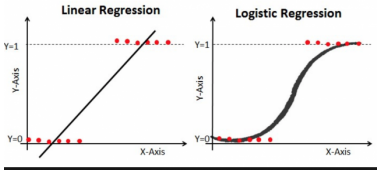

Figure 24: logistic regression

6.3 Support Vector Machine

Support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

```
In [16]:
%%time
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
model2 = LogisticRegression(random_state=0)
parameters2 = {'C':[0.001, 0.0001, 0.000001, 0.000000001, 0.00001, 0.1, 0.01, 1, 10, 100, 1000, 10000, 100000],'penalty':['l1','l2']}
clf2 = GridSearchCV(model2, parameters2)
clf2.fit(X_train, y_train)

...

In [17]:
clf2.best_score_

Out[17]: 0.9421622434275754

In [18]:
clf2.best_params_

Out[18]: {'C': 100000, 'penalty': 'l2'}
```

Figure 25: logistic regression

```
In [12]:
%%time
from sklearn.model_selection import GridSearchCV
parameters = {'kernel':['linear'],'C':[1,10]}
model1 = SVC()
clf1 = GridSearchCV(model1,parameters)
clf1.fit(X_train,y_train)

Wall time: 1min 2s

Out[12]: GridSearchCV(estimator=SVC(), param_grid={'C': [1, 10], 'kernel': ['linear']})

In [13]:
clf1.best_score_

Out[13]: 0.9303625016545004

In [14]:
clf1.best_params_

Out[14]: {'C': 1, 'kernel': 'linear'}
```

Figure 26: SVM

6.4 Train the model

In terms of the logistic regression, using grid search cross validation, we choose the best-performing parameter from parameters2 = 'C':[0.001,0.0001,0.000001,0.000000001,0.00001,0.1,0.01,1,10,100, 1000, 10000, 100000], 'penalty':['l1','l2'] and get the best parameters 'C': 100000, 'penalty': 'l2'.

Applying the model to the testing set to get the prediction racism-score and compare with the labels in the testing set, we can figure out the accuracy rate. We tried the process for more than 10 times and the accuracy rate is always around 0.94-0.95. It takes 24.9 ms to predict the result and get the accuracy rate.

With respect to the Support Vector Machine, we set the parameters 'C': 1, 'kernel': 'linear'. Applying the model to the testing set to get the prediction racism-score and compare with the labels in the testing set, we can figure out the accuracy rate. We tried the process for more than 10 times and the accuracy rate is always between 0.92 and 0.95. It takes 24.9 ms to predict the result and get the accuracy rate.

From the information given above, the logistic regression seems to perform better than SVM, therefore, we decided to use the model with 'C': 100000, 'penalty': 'l2'.

6.5 Apply the model to the collected Tweet data

After collecting the new Tweets, we applied the same pre-process function mentioned in the 5.2. We also need to care about the out-of-vocabulary problem in the current larger dataset. We create a list of all the words (called word-sample) used in the training and testing dataset and then take the common set of word-sample and each Tweet to make sure we extract the words that appear in the features and delete other words which are out of vocabulary. After that, we get a big data matrix (3284993*15915) and use the logistic model to do the prediction.

6.6 Analysis

Based on this model, no tweets are classified as racism. I conclude that, based on this model, all tweets are not racism. People do not have hate speeches related to vaccince.

Another possibility is that the possible "racism" related to vaccine is different from the general racism, so we cannot use the classifier of general racism to find the racism tweets related to vaccine.

6.7 Limitation of the model

There are some limitations and implementation ideas as follows:

The size of the labeled dataset is too small, including only 15915 different words in total, which is much smaller than the real world situation. If the dataset could be larger, the accuracy rate will be better.

The label was created by the author of [**datawebsite**]. The decisions are purely based on subjectivity. Everyone might have different criterions for racism and not racism. Therefore, it is better to have a comprehensive way to label the training and testing set, such as taking the average of scores from the public or more people than just one person.

The out-of-vocabulary is a big topic in sentiment analysis. In this project, due to the time span and limited knowledge, we decided to ignore the words that were out of our vocabulary. However, those out-of-vocabulary words also have a lot of information that is useful to detect racism. If we could better handle the OOV problem, the prediction result will be implemented a lot.

7 Conclusion

Overall, our project is to collect and process tweets related to the COVID vaccine, analyze the public emotion (positive/negative) related to vaccine, and detect racism speeches related to vaccine.
Based on figure 5.3.3 and figure 5.4.3, We could conclude:
1. People on twitter are mostly negative (roughly 20 % positive / 60% negative)
2. Before the pandemic people's sentiments are relatively stable
3. Through the pandemic people's sentiments become more and more negative
4. Accounts with higher popularity are slightly more positive. One possible explanation may be that because all official accounts are in the 'ExtremeHigh' group.
Based on Part6, we could conclude that: most of tweets are not racism. Race differences does not affect the speeches related to vaccine. People around the world can work together to improve the vaccine and fight with COVID-19.