Phase 3 Report

March 10, 2021

1 Final Report

Team 7: Covid Sentiment Analysis Members: Lipsa J., Ji K., Yuanfeng L., Yu L.

1.0.1 Background and Motivation

The pandemic has uprooted the lives of every single person in the world. While it began as a minor inconvenience to many people, the harsh reality and severity of the virus were soon realized. In the beginning of enforcing protective measures to protect the public, many people's opinions on the virus, protective rules & procedures, and other topics relating to the pandemic have changed and continually do into 2021.

With such a slow response to protective measures in the U.S compared to other countries globally, we wanted to find out the public's stance on the matter over the period of nearly the entire pandemic.

With this in mind, we want to record and analyze these trends by looking at the metrics such as sentiment, LIWC metrics, and possibly more as we make further discoveries.

1.0.2 Goals and Objectives

Utilizing Twitter, an online social media platform for sharing content and microblogging, we've analyzed "tweets" (publicly posted messages) from everyday people about how they feel about the pandemic.

This has deviated from traditional sentiment gathering such as surveying and/or polling small sample sizes in localized areas by utilizing "social media big data" in the form of Twitter. By gathering a much wider demographic of people, we believe this approach has proven to be successful. Twitter has proven itself, historically, to be highly accurate, quick, and better reflect the perspectives of the everyday person since they're the ones whose data we're processing.

Here are the original objectives & research questions we've hoped to answer during our time working on the project and will go in-depth as the report goes on.

- **Goal 1**: Find out how many tweets sentiments changed on the regulation or rules about wearing a mask or taking a vaccine for the the year 2020 and current months in 2021 (January - March)
- **Goal 2**: Find out the sentiment of tweets relating to the COVID-19 virus for the year 2020 and the current months in 2021 (January - March)
- **Stretch Goal 1**: Find out the sentiments across geographical locations within the U.S about either protective measures (Eg. Wearing a mask) and the taking the vaccine. It's been shown throughout various news outlets and social media that different areas in the U.S have had

varying responses to these rules. If time & resources allow, we want to run the research experiment at a lower level - focusing on specific areas in the U.S - Perhaps areas with the lowest cases per capita vs. moderate vs. high.

- **Stretch goal 2**: Relate our findings to how misinformation & fake news on Twitter changed before and after the election; as well as its possible consequences on the public's sentiment on the topic of COVID-19 and its related topics (Eg. vaccines, lockdown, social distancing).

1.1 Summary of Phases 1 & 2

For the sake of coverage and to show the iterative process that is "Data Science", we thought it fitting to include our original approaches in the previous two phases.

1.1.1 Phase 1

Twitter's V2 API, and consequently the Twitter API wrapper for Python, Tweepy, only allows data querying for up to 1 week into the past. This was an immense barrier since our project revolved around analyzing data across a period of over a year. We got around this by utilizing IEEE DataPort's coronavirus tweets dataset. Since the very start of the COVID-19 pandemic, these dataset maintainers have collected thousands of tweets relating to a series of keywords relating to the pandemic. These keywords range from vaccines, PPE, lockdown measures, mask mandates, racial tensions, and so on. This was a series of CSV files containing only the tweet id - which Twitter's API and Tweepy allow you to look up regardless of the time. Our first problem was solved on data collection/

However, another issue came up. Twitter's standard API accounts have a strict rate limit of 900 Tweet-id requests per 15 minutes and searching tweets that have been or belong to accounts that have been suspended/blocked also count towards the rate limit cap. This equated to approximately scraping for hours to get ~5000 tweets which was our original goal per month. Highly inefficient but was enough to give our team the proof of concept we needed to continue the project.

1.1.2 Phase 2

A few months ago Twitter had a cycle for Twitter Research accounts and discontinued them shortly after. These accounts were how researchers have scraped for tweet id's relating to the pandemic on Kaggle and the IEEE DataPort dataset. Recently, they've started accepting again and we got approved.

With our elevated privileges, we changed our data scraping methodology that remains the same in the final phase.

1.2 Phase 3

We've divided our dataset into 3 categories based on keywords. 1. Tweets with keywords relating to COVID-19. 2. Tweets with keywords relating to the COVID-19 vaccines and related companies. 3. Tweets with keywords relating to protective measures (Eg. social distancing, quarantining, mask mandates, etc.).

Utilizing Twitter's API V2 and our research account permissions, we've collected about 500 tweets for each keyword category every 2-3 days. This equates to roughly 3000-4000 tweets per month per keyword category. (~3500 tweets) x (13 months) x (3 keyword categories)

Here is the code used to scrape tweets based on our methodology. This will only work if the account's bearer token is approved for research (our method) or enterprise (premium paid account).

```
[ ]: import json, requests, time, os
     import pandas as pd
     from collections import defaultdict

     date_ranges = [[("2020-02-01", "2020-02-03"),
                     ("2020-02-04", "2020-02-06"),
                     ("2020-02-07", "2020-02-09"),
                     ("2020-02-10", "2020-02-12"),
                     ("2020-02-13", "2020-02-15"),
                     ("2020-02-17", "2020-02-19"),
                     ("2020-02-21", "2020-02-23"),
                     ("2020-02-24", "2020-02-26"),
                     ("2020-02-27", "2020-02-29")]]
     """
     Each date range consists of a start date and end date.
     We went with this approach since you can only collect up to 500 tweets per␣
      ↪request,
     so breaking up each month into multiple requests allows us to get enough tweets␣
      ↪for analysis.
     The actual file repeats this pattern with every month but is significantly long␣
      ↪and does not
     add much to the report itself.
     """

     # Base URL endpoint.
     search_url = 'https://api.twitter.com/2/tweets/search/all'

     # Twitter API V2 utilizes the bearer_token instead of the access tokens Tweepy␣
      ↪utilizes
     # Grabs credentials from OS environment variables. If it's not in the␣
      ↪environment, can also manually copy & paste it.
     bearer_token = os.environ.get("BEARER_TOKEN")

     # Month titles to separate files
     months = ['feb_20', 'march_20', 'april_20', 'may_20', 'june_20',
               'july_20', 'august_20', 'sept_20', 'oct_20', 'nov_20', 'dec_20',
               'jan_21', 'feb_21']

     num_results = 500

     # Keyword terms to query by
     search_terms = {
```

```
    'covid' : "corona OR #corona OR coronavirus OR #coronavirus OR covid OR␣
↪covid19 OR #covid19 OR sarscov2 #sarscov2 OR covid_19 OR ncov19 OR pandemic␣
↪OR #pandemic",
    'vaccine' : "pfizer OR moderna OR vaccine OR #vaccine2021 OR #2021vaccine␣
↪OR covid vaccine OR vaccine against coronavirus OR corona vaccine OR␣
↪vaccines work OR #vaccineswork",
    'safety' : "socialdistancing OR #socialdistancing OR social distancing OR␣
↪wfh OR #wfh OR homeschooling OR #masks4all OR face shield OR quarantine OR␣
↪flattening the curve OR flatteningthecurve OR #flatteningthecurve OR␣
↪lockdown OR masks OR #masks4all OR wash ur hands OR wash your hands OR␣
↪#stayathome OR #stayhome OR#selfisolating OR self isolating OR hand␣
↪sanitizer OR covidiots OR #covidiots"
}

# Appends bearer token to HTTP Request headers
def create_headers(bearer_token):
    headers = {"Authorization": "Bearer {}".format(bearer_token)}
    return headers

# Sends HTTP 'GET' request with search params and our bearer token for␣
↪authentication
def connect_to_endpoint(url, headers, params):
    response = requests.request("GET", search_url, headers=headers,
                                params=params)
    print(response.status_code)
    if response.status_code != 200:
        raise Exception(response.status_code, response.text)
    return response.json()


"""
Note, this will not work unless the bearer token holder has been approved for␣
↪the research track.
The other accounts other than "Enterprise" and "Research" still cannot go past␣
↪the 7 day history.
"""
# Format HTTP request
headers = create_headers(bearer_token)
json_response = None
# 'key' = key word category and search_term = formatted query parameters.
for key, search_term in search_terms.items():
    print(f"Starting to scrape for {key} tweets")
    for date_range in date_ranges: # This represents the month
        tweet_df = defaultdict(list)
        print(f"Collecting {key} tweets for {months[month_index]}")
        # 'since' and 'until' are the beginning and end days within each month
        for since, until in date_range:
            # Format query parameters. Formatted in ISO8601 date time
            query_params = {'query': f"{search_term} -is:retweet",
                            'tweet.fields':'text,created_at,entities',
                            'start_time':f"{since}T00:00:00Z", # Standard format␣
↪uses Zulu time.
                            'end_time':f"{until}T23:59:59Z",
                            'max_results':num_results}
            # Unpack JSON response
            tweet_data = json_response['data']
            num_tweets = json_response['meta']['result_count']
            # Iterate through JSON array and extract data
            for i in range(0, int(num_tweets)):
                tweet_df['text'].append(tweet_data[i]['text'])
                tweet_df['id'].append(tweet_data[i]['id'])
                tweet_df['created_at'].append(tweet_data[i]['created_at'])
                # 'Entities' is left missing if the tweet had none.
                # Causes an exception if so.
                try:
                    tweet_df['entities'].append(tweet_data[i]['entities'])
                except KeyError:
                    tweet_df['entities'].append("None")
        # Save the file into each respective directory.
        df = pd.DataFrame(tweet_df)
        file_path = f"./{key}/{key}_{months[month_index]}.csv"
        df.to_csv(file_path)
        print(f"Saved to {file_path}.\n")
        month_index += 1
```

This code will only work given specific constraints and will not work as is. If interested to run the code, you can do the following steps in order to: 1. Must have a Twitter Developer account with research or enterprise privileges. 2. Bearer Token must be in the OS environment or copied pasted in the relevant field. 3. Respective directories for each keyword category must be in the same directory as the code.

1.2.1 How Does the Sentiment for COVID-related Topics Change Between February 2020 to February 2021?

There has been a cycle of public outcry against mask mandates along with a mob-mentality notion of protecting yourself and your families. With such opposite narratives being present in society at the same time; also extenuated by the fluctuations in support and opposition for these mandates over time, we wanted to find out the public's sentiment on this topic utilizing Tweets.

Furthermore, we also want to look at if significant/notable events had changed the public's sentiment such as when Donald Trump contracted COVID-19, the 2020 Presidential Election & Jan 20 Inauguration, and other events such as holidays.

Step 1 - Process Data

1. All the data is provided beforehand with the following fields: created_at, tweet_id, text, and entities.
2. Three major directions: Covid, safety measures, and vaccines.
3. Get the ratio of positive, neutral, and negative sentiments for visualization and analysis.

[1]:
```
# define a function that can reprocess csv file and find all the
# tweets having mask in it, and return a pandas' dataframe
import pandas as pd
import re
import os

def add_mask_tweets_in_df(target_file_path):
    """
    @param:
    target_file_path: string - usually a csv file
    target_dataframe - pandas.df

    @return:
    result_df - pandas data frame
    """
    df = pd.read_csv(target_file_path)
    # due to the missing values in hashtags
    # I have to process the data by my self
#     result_df = df[df['text'].str.contains('mask', case=False, na=False)]
    result_df = df
    return result_df

import os
# this function will iterate files in each folder and write dataframes
# into new csv files, and return the dataframes for future uses
def write_csv(target_folder_name):
    """
    ptype: folder_naem - str
    rtype: month_list - list
    """
#     month_list = []
    month_list = {}
    for file in os.listdir(target_folder_name):
        if file[-4:] == '.csv':
            df = add_mask_tweets_in_df(target_folder_name + file)
#             month_list.append(df)
            month_list[file[:-4]] = df
            print(df.shape)
            df.to_csv("%s_%s.csv"%(file[:-4], "Masks"))
    return month_list
```

How many data frames are there total and what are their sizes? Dataframe Shape: **(Number of records, Number of features)**

For COVID-19 related Tweets:

```
[2]: covid_dictionary = write_csv('./data/covid/')
```

```
(3691, 5)
(3726, 5)
(3724, 5)
(3703, 5)
(3871, 5)
(3710, 5)
(3615, 5)
(3651, 5)
(4052, 5)
(3618, 5)
(3249, 5)
(3716, 5)
(3627, 5)
```

For Vaccine related Tweets:

```
[3]: vaccine_dictionary = write_csv('./data/vaccine/')
```

```
(3237, 5)
(3578, 5)
(3148, 5)
(3323, 5)
(3245, 5)
(3160, 5)
(3488, 5)
(3613, 5)
(4009, 5)
(3533, 5)
(3618, 5)
(3205, 5)
(3619, 5)
```

Finally, for Safety measure related Tweets:

```
[4]: safety_dictionary = write_csv('./data/safety/')
```

```
(3930, 5)
(3589, 5)
(3574, 5)
(3533, 5)
(3563, 5)
(3611, 5)
(3543, 5)
(3518, 5)
(3604, 5)
(3734, 5)
```

```
(3584, 5)
(3197, 5)
(3625, 5)
```

Now that we have enough data, we do need to clean it up a bit. Different operating systems/platforms use different line endings, escape characters, and just the nuances of social media entail specific characteristics (Eg. hashtags, trends, etc.).

[5]:
```
# cleaning twitter specific data

def clean_tweets(row):
    row = row.lower()
    #remove urls
    row  = re.sub(r'http\S+', '', row)
    #remove mentions
    row = re.sub(r"(?<![@\w])@(\w{1,25})", '', row)
    #remove hashtags
    row = re.sub(r"(?<![#\w])#(\w{1,25})", '',row)
    #remove other special characters
    row = re.sub('[^A-Za-z .-]+', '', row)
    #remove digits
    row = re.sub('\d+', '', row)
    row = row.strip(" ")
    return row

round1 = lambda x: clean_tweets(x)

# Apply a second round of cleaning tweets specific cleaning
def clean_text_round2(row):
    row = row.lower()
    #remove urls
    row  = re.sub(r'http\S+', '', row)
    #remove mentions
    row = re.sub(r"(?<![@\w])@(\w{1,25})", '', row)
    #remove hashtags
    row = re.sub(r"(?<![#\w])#(\w{1,25})", '',row)
    #remove other special characters
    row = re.sub('[^A-Za-z .-]+', '', row)
    #remove digits
    row = re.sub('\d+', '', row)
    row = re.sub('rt', '', row)
    row = row.strip(" ")
    return row

round2 = lambda x: clean_text_round2(x)
```

TextBlob was utilized for calculating sentiments.

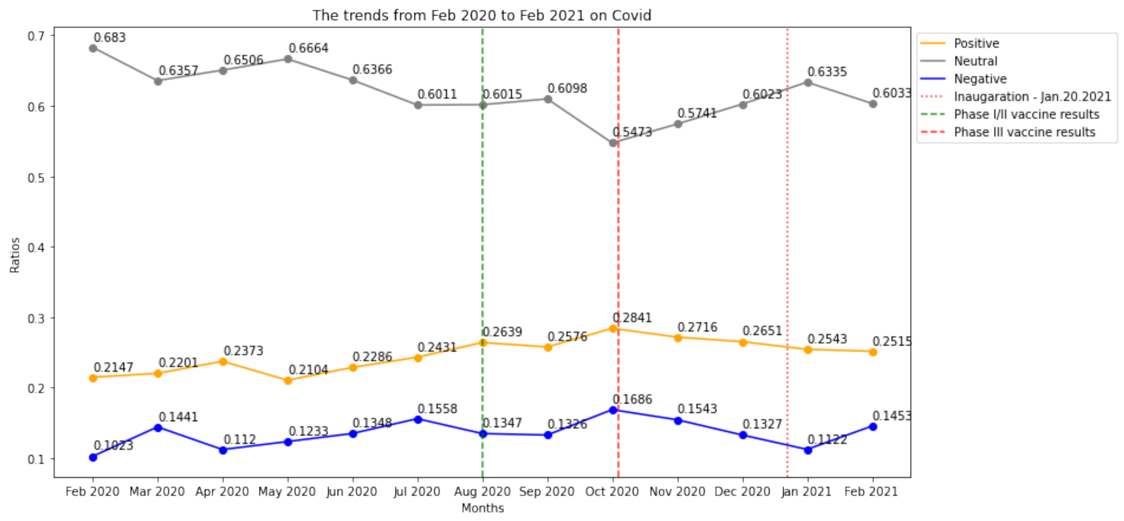
[6]:
```python
# we are using textblob
import nltk
from nltk.corpus import opinion_lexicon
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from textblob import TextBlob

def interpret_sentiment(score):
    if score <= -0.05:
        return 'Negative'
    elif score >= 0.05:
        return 'Positive'
    else:
        return 'Neutral'

def textblob_sentiment_analysis(content_list):
    """
    @parameters:
    content_list - is a list of content
    @ return:
    nothing, but print out outcome
    """

    list_sentiment = []
    for word in content_list:
        scores = TextBlob(word).sentiment
        score = scores.polarity
        list_sentiment.append(interpret_sentiment(score))
    return list_sentiment


import matplotlib.pyplot as plt

def get_ratios(target_textblobed_list):
    """

    rtype: p_ratio - float,
    """
    length = len(target_textblobed_list)
    p, neu, neg = 0, 0, 0
    # now lets count the ratio of the positive out of all data
    for sentiment in target_textblobed_list:
        if sentiment == "Positive":
            p += 1
        elif sentiment == "Neutral":
            neu += 1
        elif sentiment == "Negative":
            neg += 1
    p_ratio = round(p/length, 4)  #ratio for positive
    neu_ratio = round(neu/length, 4)   # ratio of neutral
    neg_ratio = round(neg/length, 4)   # ratio of negative
    #print(p, neu, neg, length)

    return [p_ratio, neu_ratio, neg_ratio]
```

Step 2 - Visualization Each keyword category will be line-plotted overtime to view the changes in sentiment as time passes.

[7]:
```
# put the info in a list in the chronological order
def in_order_lists(target_dict):
    """
    rtype: list - ord_df_l, dfs
    rtype: list - ord_month_l, list of strs
    """
    ord_df_l = [0] * len(target_dict.keys())
    ord_month = ["Feb 2020",
                 "Mar 2020",
                 "Apr 2020",
                 "May 2020",
                 "Jun 2020",
                 "Jul 2020",
                 "Aug 2020",
                 "Sep 2020",
                 "Oct 2020",
                 "Nov 2020",
                 "Dec 2020",
                 "Jan 2021",
                 "Feb 2021"]

    # Each data file is prepended to denote its month. "feb_20" means
    # February 2020. feb_21 = February 2021. And so on.
    for month in target_dict:
        if "feb_20" in month:
            ord_df_l[0] = target_dict[month]
        elif "march_20" in month:
            ord_df_l[1] = target_dict[month]
        elif "april_20" in month:
            ord_df_l[2] = target_dict[month]
        elif "may_20" in month:
            ord_df_l[3] = target_dict[month]
        elif "june_20" in month:
            ord_df_l[4] = target_dict[month]
        elif "july_20" in month:
            ord_df_l[5] = target_dict[month]
        elif "august_20" in month:
            ord_df_l[6] = target_dict[month]
        elif "sept_20" in month:
            ord_df_l[7] = target_dict[month]
        elif "oct_20" in month:
            ord_df_l[8] = target_dict[month]
        elif "nov_20" in month:
            ord_df_l[9] = target_dict[month]
        elif "dec_20" in month:
            ord_df_l[10] = target_dict[month]
        elif "jan_21" in month:
            ord_df_l[11] = target_dict[month]
        elif "feb_21" in month:
            ord_df_l[12] = target_dict[month]

    return ord_df_l, ord_month

def draw_in_one_plot(target_list_of_ys, target_list_of_xs, title):
    p_list, neu_list, neg_list = [], [], []
    annotating_list = [p_list, neu_list, neg_list]

    for i in range(len(list_of_df_of_months)):
        y = get_ratios(textblob_sentiment_analysis(target_list_of_ys[i]["text"].
 ↪tolist()))
        x = [list_of_month[i], list_of_month[i], list_of_month[i]]

        p_list.append(y[0])
        neu_list.append(y[1])
        neg_list.append(y[2])

    plt.figure(figsize=(13,7))
    plt.scatter(target_list_of_xs, neu_list, color = "gray")
    plt.scatter(target_list_of_xs, p_list, color="orange")
    plt.scatter(target_list_of_xs, neg_list, color = "blue")

    ## anotating the values
    for lists in annotating_list:
        y = lists
        x = target_list_of_xs
        for i, txt in enumerate(y):
                plt.annotate(txt, (x[i], y[i]),xytext=(x[i], y[i] + 0.
 ↪01),fontsize=10)

    plt.plot(target_list_of_xs, p_list, color="orange", label = "Positive")
    plt.plot(target_list_of_xs, neu_list, color = "gray", label = "Neutral")
    plt.plot(target_list_of_xs, neg_list, color = "blue", label = "Negative")
    plt.xlabel("Months")
    plt.ylabel("Ratios")
    plt.title("The trends from Feb 2020 to Feb 2021 on %s" % title)
    plt.axvline(x = 10.7, color = 'r', alpha=0.8, linestyle = ":",␣
↪label="Inaugaration - Jan.20.2021")
    plt.legend(bbox_to_anchor=(1, 1))
    plt.axvline(x = 6, color = 'g', alpha=0.8, linestyle = "--", label="Phase I/
↪II vaccine results")
    plt.axvline(x = 8.1, color = 'r', alpha=0.8, linestyle = "--", label="Phase␣
↪III vaccine results")
    plt.legend(bbox_to_anchor=(1, 1))
    # adding an line indicates the inaugaration & vaccine trials.
```

Trend for **COVID-19** related tweets.

```
[8]: list_of_df_of_months, list_of_month = in_order_lists(covid_dictionary)
     draw_in_one_plot(list_of_df_of_months, list_of_month, "Covid")
```



Discussion While there's a much more vast number of tweets, we believe the usage of over 100,000 tweets over this given time period did reflect the teend of sentiments over the past 13 months from February of 2020 to February of 2021.

- **Explanation of the Plots:** The Orange lines and dots represents the ratio of positive tweets in the month. We used the number of tweets that are analyzed as positive by textblob devided the total number of tweets to get the raios. And the same scheme apply to the blue lines - negative, and gray lines - which means the neutral.
- **Compare with hypothesis:** One of our research question is to see if there is any trend before the inauguration date for President Joe Biden or after, and we can see that the ratio of positive tweets has increased from Dec 2020 to Jan 2021. And the red dot line indicate the Jan 20th, which is the inaugurate day.

There might be a little confusion here, you might be confused why the red dot line is not between Jan 2020 and Feb 2021. It is because the dots whose x are "Jan 2021" has already included all the data in Jan 2021. So it would be more accurate to put the dividing red dot line in the position shown above.

One of our initial hypothesis was that the positive ratio might increase around the inauguration day, and it turns out our hypothesis was not quite right, as you can see from the plot above, the positive ratio is keeping steady with a little bit decrease. But I think it has meet our hypothesis partially, because the negative ratio has decreased around the inauguration date. Which means more people or tweets are having less negative sentiments in Jan 2021 than Dec 2020.

1.2.2 Extra Comparison (fun fact)

- As you can see the little different in the add_mask_tweets_in_df() function at the begining of my(Yuanfeng Li)'s part, I have commented the following line: 'result_df = df[df['text'].str.contains('mask', case=False, na=False)]', it was because we have used the keywords, such as #Wearmask, #mask, #masks4all to get our datas. However, I found that the mask keyword is not in all the tweets we collected, So I have added one more filter to filt out the tweets which contains 'mask' and write them in the new csv files. I have done the visulization as well for the filtered tweets, as you can see the plots below:

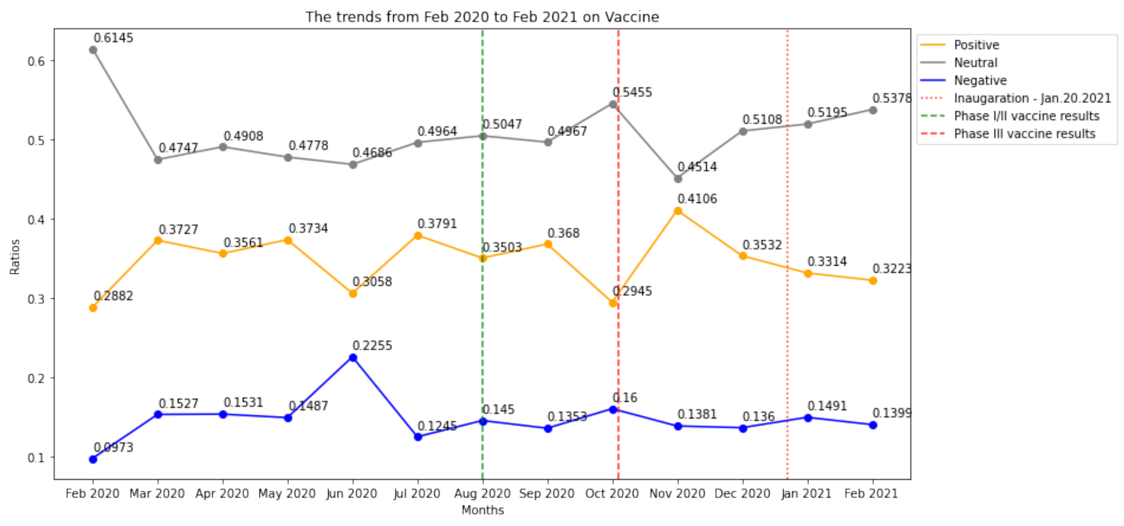
Note: For PDF version, we've removed the image since LaTex had difficulty converting it.

An interesting thing to note is that despite most of the positive and neutral sentiments having high volatility, 'negative' trend line has shown a normalization around July 2020 with a slow decrease up until the inauguration. It then, shows a sharp increase upwards of nearly 50% which signals signficance.

Now for **Vaccine** related tweets.

[9]:
```python
# using vader to do sentiment analysis
import nltk
from nltk.corpus import opinion_lexicon
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()

def interpret_sentiment(score):
    if score <= -0.05:
        return 'Negative'
    elif score >= 0.05:
        return 'Positive'
    else:
        return 'Neutral'

def vader_sentiment_analysis(content_list):
    """
    @parameters:
    content_list - is a list of content
    @ return:
```

```
    nothing, but print out outcome
    """
    list_sentiment = []
    for word in content_list:
        scores = analyzer.polarity_scores(word)
        score = scores["compound"]
        list_sentiment.append(interpret_sentiment(score))
    return list_sentiment

def get_ratios(target_list):
    """
    rtype: p_ratio - float,
    """
    length = len(target_list)
    p, neu, neg = 0, 0, 0
    # now lets count the ratio of the positive out of all data
    for sentiment in target_list:
        if sentiment == "Positive":
            p += 1
        elif sentiment == "Neutral":
            neu += 1
        elif sentiment == "Negative":
            neg += 1

    p_ratio = round(p/length, 4)  #ratio for positive
    neu_ratio = round(neu/length, 4)   # ratio of neutral
    neg_ratio = round(neg/length, 4)   # ratio of negative
    #print(p, neu, neg, length)
    return [p_ratio, neu_ratio, neg_ratio]

    # see trend in vaccine folders lipsa
list_of_df_of_months, list_of_month = in_order_lists(vaccine_dictionary)
draw_in_one_plot(list_of_df_of_months, list_of_month, "Vaccine")
```



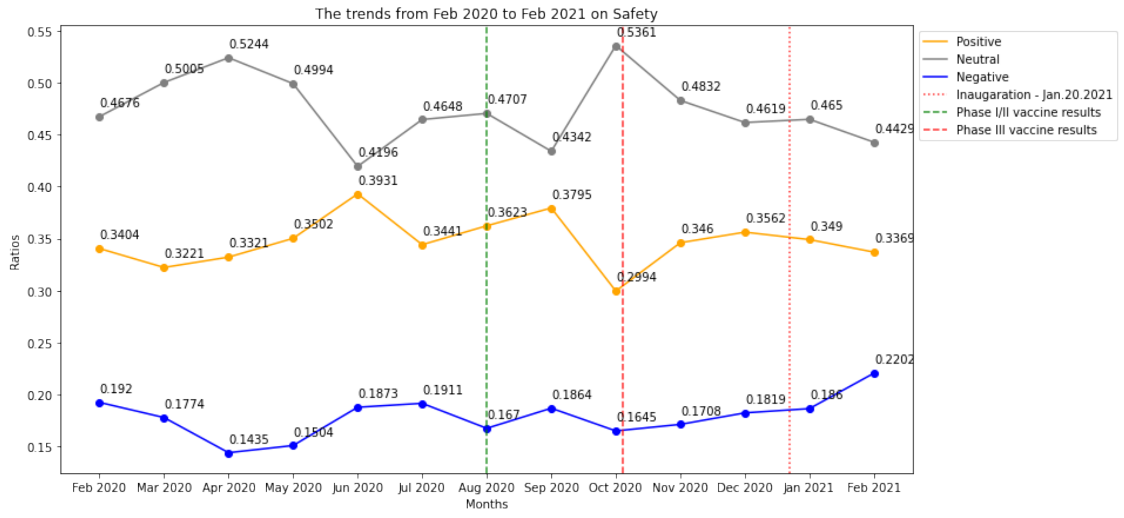
Discussion - This is a plot of Vaccine related sentiment analysis over 13 months from Feb 2020 to Feb 2021. - As you can see, the trend of negative sentiment towards vaccine has fallen down over all, also a trend of growing positive sentiment could be seen. - When clinical stage result came out, a upward positive sentiment could be noticed, but a slight growth in negative sentiment is also seen. - When Phase three results came out during early Nov 2020 which is also happens to be the election week, a huge drop in negative sentiment is noticed.

Compare with hypothesis: One of our research question is to see if there is any trend before and after important days in th last year. For example: in case of Vaccine - it makes sense to check around Vaccine phase outcomes day.

- As expected, when the vaccine companies like pfizer, moderna release their phase 3 result which was around 95% accurate, the negative sentiment dropped gradually in Nov 2020. The election also played an important role in our opinion.

Finally, for **Safety** related keywords:

```
[10]: # see trend in safety folders
list_of_df_of_months, list_of_month = in_order_lists(safety_dictionary)
draw_in_one_plot(list_of_df_of_months, list_of_month, "Safety")
```



Discussion The safety related data included keywords like "social distancing", "wash your hands", "medical workers", and so on. After plotting 13 months worth of sentiment data, it's clear the positive, negative, and neutral sentiment lines have been overall smooth. Some significant events such as the initial phase I/II vaccine results did seem to effect the overall negative sentiment. The trial results weren't seemingly positive and the public did feel disappointment that the initial results were not as promising as they had hoped.

In terms of answering the research questions, February 2021 does highlight something interesting. For the entirety of the pandemic, February 2021 was the only month in which the negative sentiment broke a ratio of 0.2 (over 1 in 5 tweets about safety measures were negative). While the event was extremely recent, possible COVID-19 lockdown procedures being lifted prematurely may have caused many vocal opinions about how some State's governments are handling the final stretch of this pandemic.

1.2.3 Visualize... Again!

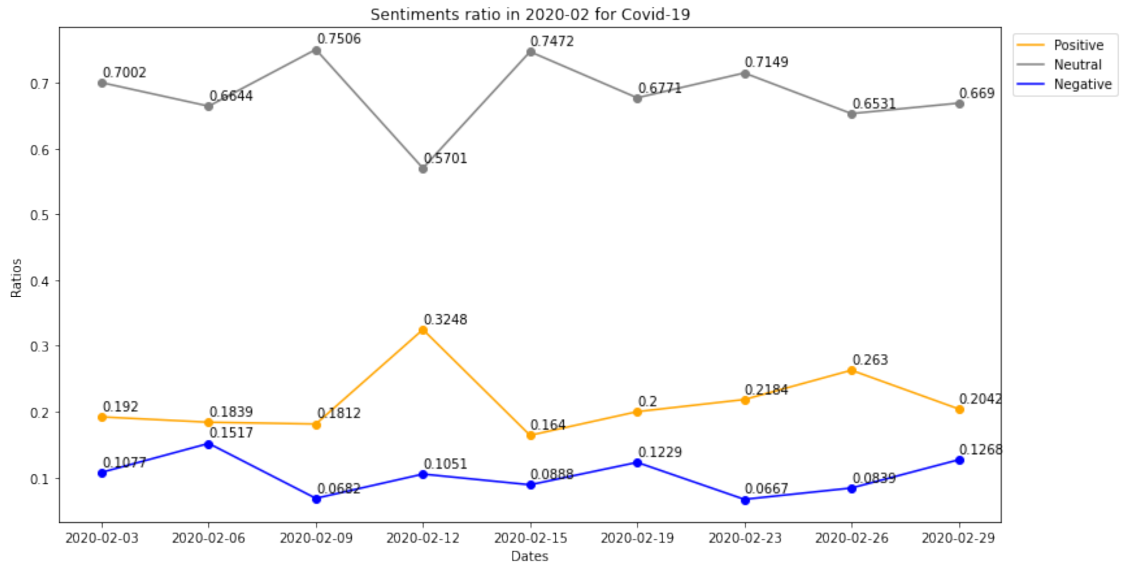
Data Science has shown itself to be a highly iterative process. The visualizations have shown us good results and we see a few general trends among the ratio of our data, however there seems to be many pitfalls with the existing plots. 1. The X-axis are months instead of dates which aggregate, or "squish" a lot of the information which does not show as much information as it could. 2. The vertical lines indicating important dates are not perfectly in line as a result of this aggregation. It does not fully show the effects of each of these significant events.

So, we believe a more detailed plot may serve well.

[11]:
```python
def draw_month_trend(target_file, topic):
    """
    type: str, target_file - "feb_2020.csv"
    rtype: list, date_list - ["2020-03-03", "2020-03-06",....]
    rtype: list, points_list : sentiment points related to the certain date -
                              [(positive, neutral, negative), (0.2, 0.5, 0.3)]
    """
    date_list, points_list = [], []
    # create a dataframe, and clean the created_at for future group by
    df = pd.read_csv(target_file)
    # get two lists
    created_at_list = df['created_at'].to_list()
    texts_list = df['text'].to_list()
    # modify the lists
    for i in range(len(created_at_list)):
        created_at_list[i] = created_at_list[i][:10]
        if created_at_list[i] not in date_list:
            date_list.append(created_at_list[i])
    # created new dataframe
    new_data = {'created_at':created_at_list, 'text':texts_list}
    new_df = pd.DataFrame(new_data)
    # set index of the dataframe
    new_df = new_df.set_index("created_at")

    total_txt_list = []
    for d in date_list:
        total_txt_list.append(new_df.loc[d]['text'].to_list())

    total_ratio_list = []
    for txts in total_txt_list:
        total_ratio_list.append(get_ratios(textblob_sentiment_analysis(txts)))

    p_list, neu_list, neg_list = [], [], []
    annotating_list = [p_list, neu_list, neg_list]
    for i in range(len(total_ratio_list)):
        y = total_ratio_list[i]
        p_list.append(y[0])
        neu_list.append(y[1])
        neg_list.append(y[2])

    plt.figure(figsize=(13,7))
    plt.scatter(date_list, p_list, color="orange")
    plt.scatter(date_list, neu_list, color="gray")
    plt.scatter(date_list, neg_list, color="blue")

    # anotating values
    ## anotating the values
    for lists in annotating_list:
        y = lists
        x = date_list
        for i, txt in enumerate(y):
                plt.annotate(txt, (x[i], y[i]),xytext=(x[i], y[i] + 0.
↪01),fontsize=10)
```

```
    plt.plot(date_list, p_list, color="orange", label = "Positive")
    plt.plot(date_list, neu_list, color = "gray", label = "Neutral")
    plt.plot(date_list, neg_list, color = "blue", label = "Negative")

    plt.xlabel("Dates")
    plt.ylabel("Ratios")
    plt.title("Sentiments ratio in %s for %s"%(date_list[0][:7],topic))
    plt.legend(bbox_to_anchor=(1.13, 1))


    return [date_list,p_list,neu_list,neg_list]

# the following should draw a scatter plot, and returns two lists
# date_list, points_list = draw_month_trend("../data/covid/covid_feb_20.csv")

draw_month_trend("./data/covid/covid_feb_20.csv", "Covid-19")
;
```

```
[11]: ''
```



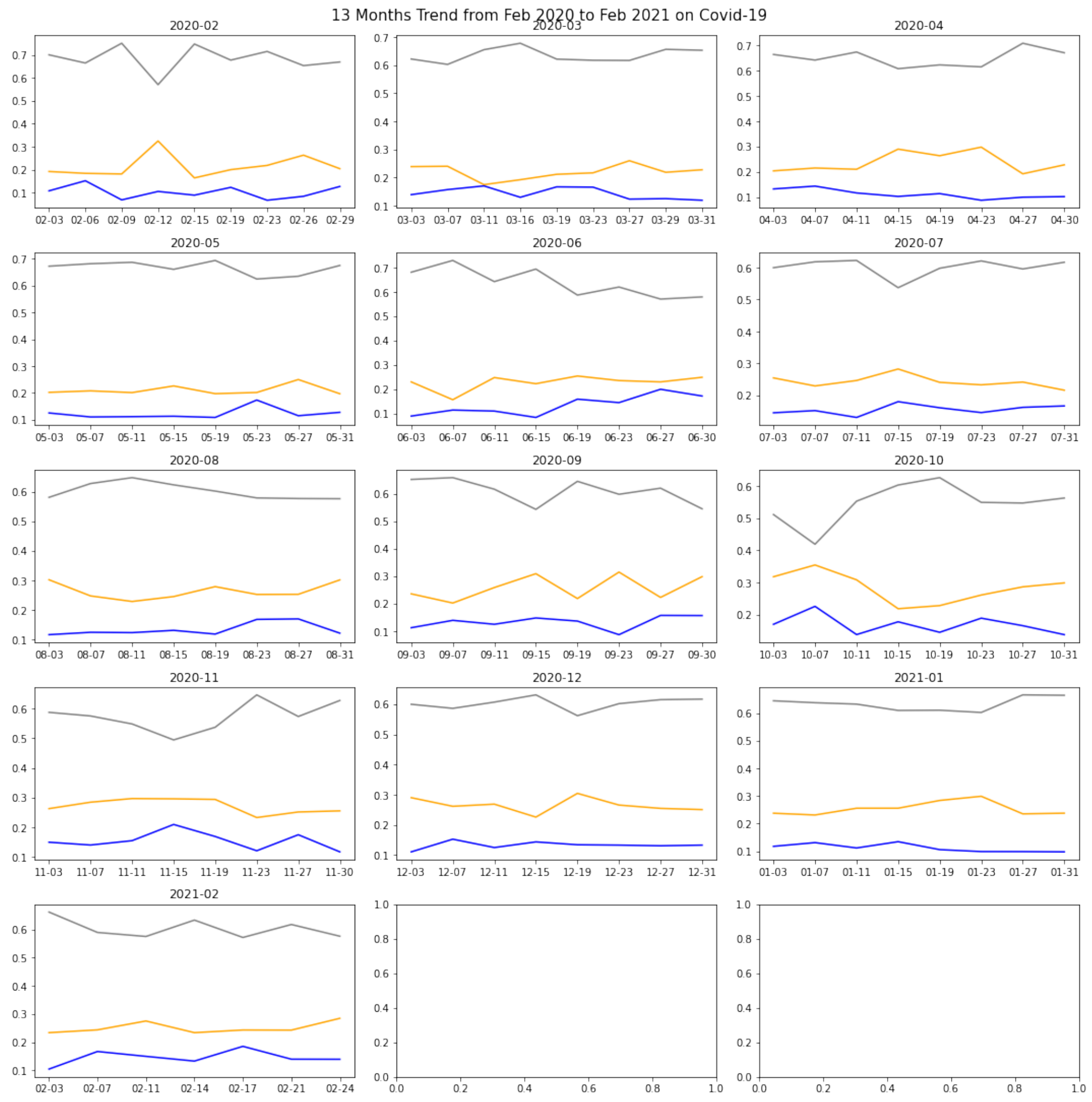
1.2.4 Improvement:

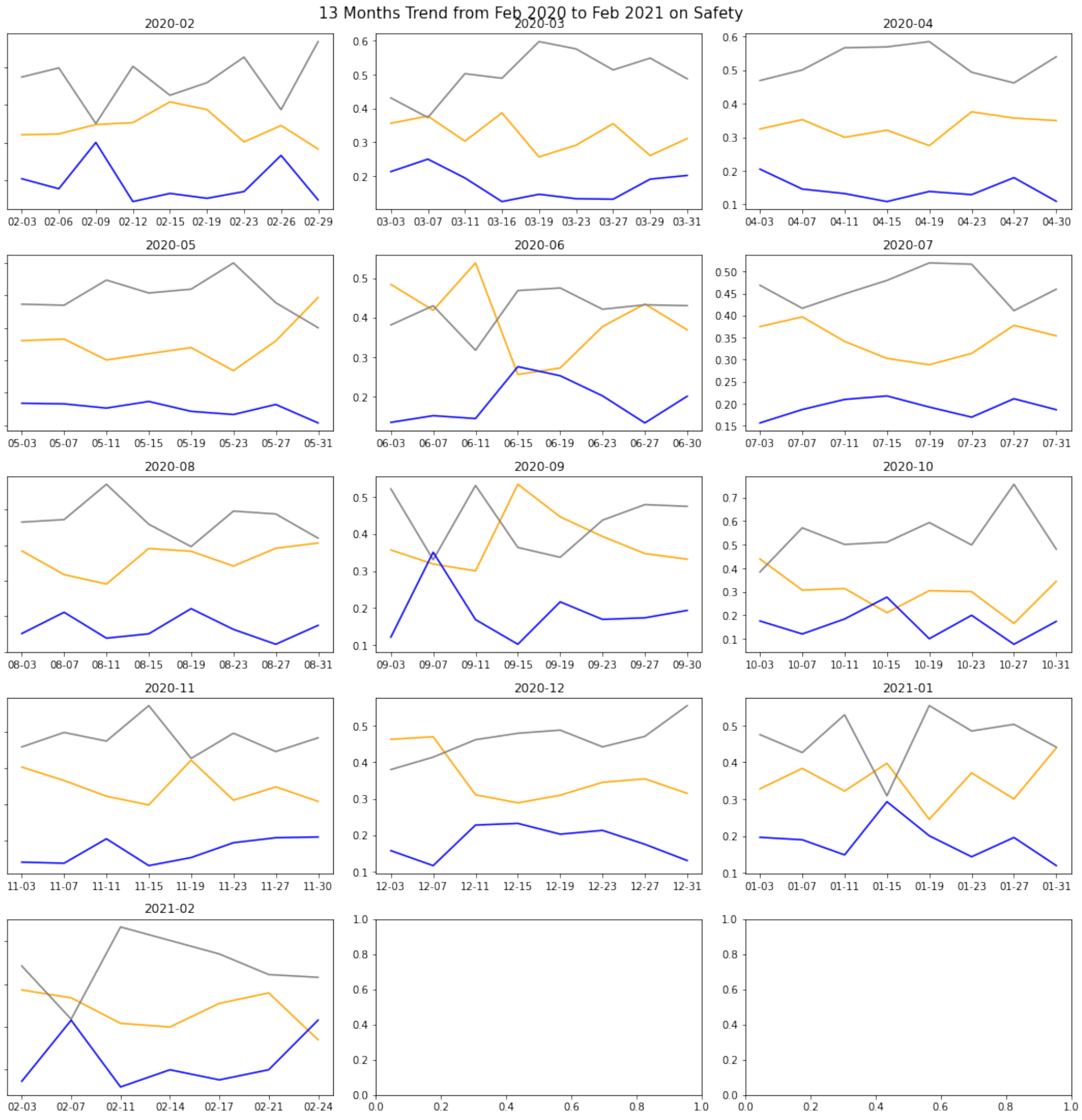
- As you can see from the image above, we get a more detailed view of the ratios of different sentiments on different days instead of seeing the average data for the month.
- Next, we'll repeat this process for all 13 months and utilize our divided datasets.

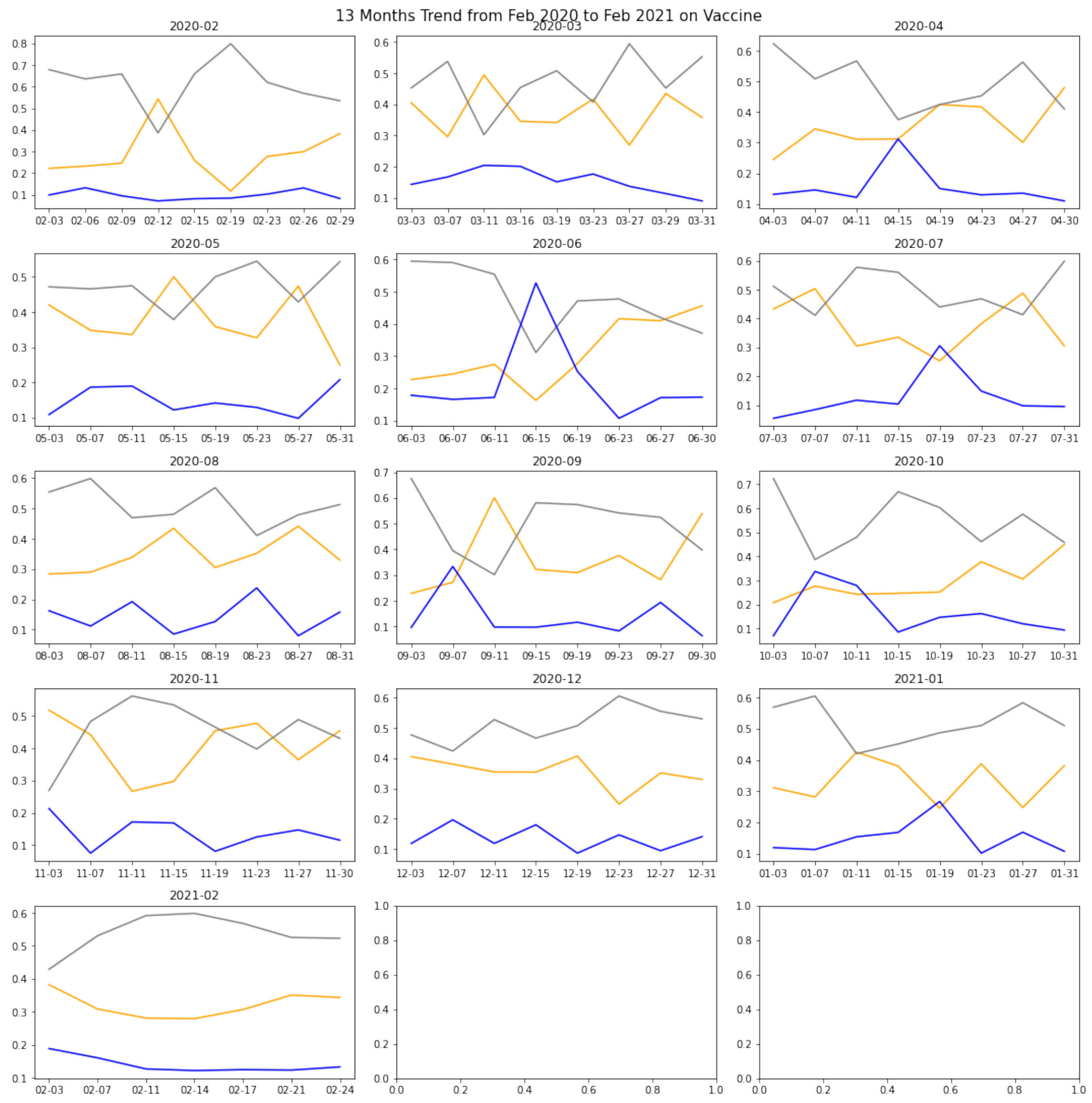
[12]:
```python
import pandas as pd


def get_month_lists(target_file):
    date_list, points_list = [], []
    # create a dataframe, and clean the created_at for future group by
    df = pd.read_csv(target_file)
    # get two lists
    created_at_list = df['created_at'].to_list()
    texts_list = df['text'].to_list()
    # modify the lists
    for i in range(len(created_at_list)):
        created_at_list[i] = created_at_list[i][:10]
        if created_at_list[i] not in date_list:
            date_list.append(created_at_list[i])
    # created new dataframe
    new_data = {'created_at':created_at_list, 'text':texts_list}
    new_df = pd.DataFrame(new_data)
    # set index of the dataframe
    new_df = new_df.set_index("created_at")

    total_txt_list = []
    for d in date_list:
        total_txt_list.append(new_df.loc[d]['text'].to_list())

    total_ratio_list = []
    for txts in total_txt_list:
        total_ratio_list.append(get_ratios(textblob_sentiment_analysis(txts)))

    p_list, neu_list, neg_list = [], [], []
    annotating_list = [p_list, neu_list, neg_list]
    for i in range(len(total_ratio_list)):
        y = total_ratio_list[i]
        p_list.append(y[0])
        neu_list.append(y[1])
        neg_list.append(y[2])


    return [date_list,p_list,neu_list,neg_list]
```

[13]:
```python
import numpy as np
import matplotlib.pyplot as plt
import os
import copy

def draw_in_calender(target_folder_name, title):
    """
    """

    yearly_date_list, yearly_p_list, yearly_neu_list, yearly_neg_list = [], [],␣
↪[], []
    y_d, y_p, y_neu, y_neg = [], [], [], []
    counter = 0
    file_list = []
    for file in os.listdir(target_folder_name):
        if file[-4:] == '.csv':
            counter += 1
            file_list.append(file)
    for file in sorted(file_list):
        d, p, neu, neg = get_month_lists(target_folder_name + file)
        yearly_date_list.append(d)
        y_d += d
        yearly_p_list.append(p)
        y_p += p
        yearly_neu_list.append(neu)
        y_neu += neu
        yearly_neg_list.append(neg)
        y_neg += neg

    title_list = copy.deepcopy(yearly_date_list)
    #     clean yearly dates
    for i in range(len(yearly_date_list)):
        for j in range(len(yearly_date_list[i])):
            yearly_date_list[i][j] = yearly_date_list[i][j][5:]

    fig, axs = plt.subplots(5, 3, figsize=(15, 15))

    for i in range(counter):
        axs[int(i/3), int(i%3)].plot(yearly_date_list[i], yearly_p_list[i],␣
↪color = "orange")
        axs[int(i/3), int(i%3)].plot(yearly_date_list[i], yearly_neu_list[i],␣
↪color = "gray")
        axs[int(i/3), int(i%3)].plot(yearly_date_list[i], yearly_neg_list[i],␣
↪color = "blue")
        axs[int(i/3), int(i%3)].set_title(title_list[i][0][:7])
    plt.tight_layout()
    plt.suptitle("13 Months Trend from Feb 2020 to Feb 2021 on %s" % title, x=0.
↪5, y=1, fontsize = 15, weight="medium")
    plt.show()

#     return [yearly_date_list, yearly_p_list, yearly_neu_list, yearly_neg_list]
    return [y_d, y_p, y_neu, y_neg]
```

```
print("")
d1, p1, neu1, neg1 = draw_in_calender("./processed_data/covid/", "Covid-19")
print("")
d2, p2, neu2, neg2 =draw_in_calender("./processed_data/safety/", "Safety")
print("")
d3, p3, neu3, neg3 =draw_in_calender("./processed_data/vaccine/", "Vaccine")

# hide the outputs
;
```




13 Months Trend from Feb 2020 to Feb 2021 on Safety
2020-02
2020-03
2020-04
2020-05
2020-06
2020-07
2020-08
2020-09
2020-10
2020-11
2020-12
2021-01
2021-02



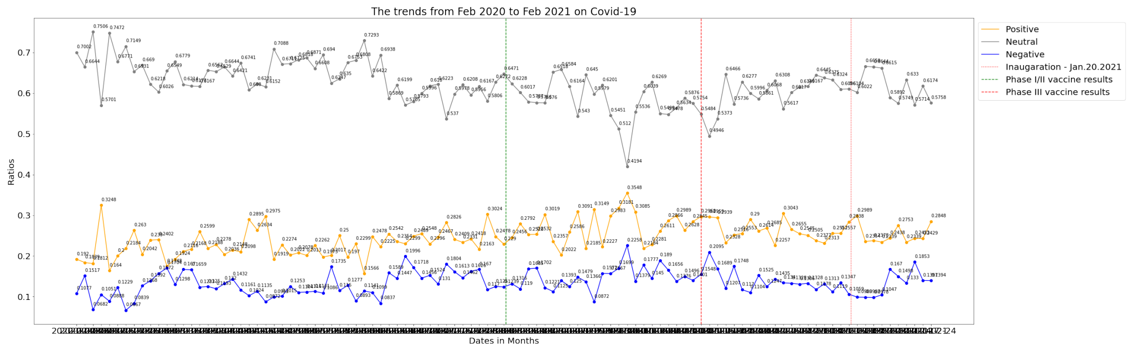
```
[13]: ''
```

Now We Can See Each Month's Trend for Each Category **Orange** signifies the positive ratio, **blue** represents negative, and **gray** is neutral * From the improvement, our visualization can give you a deeper understanding of each month's data by plotting each month overall. * So far so good, however seeing the whole trend for the year detailed can also bring better perspective.

1.3 Detailed 13-Month Plot

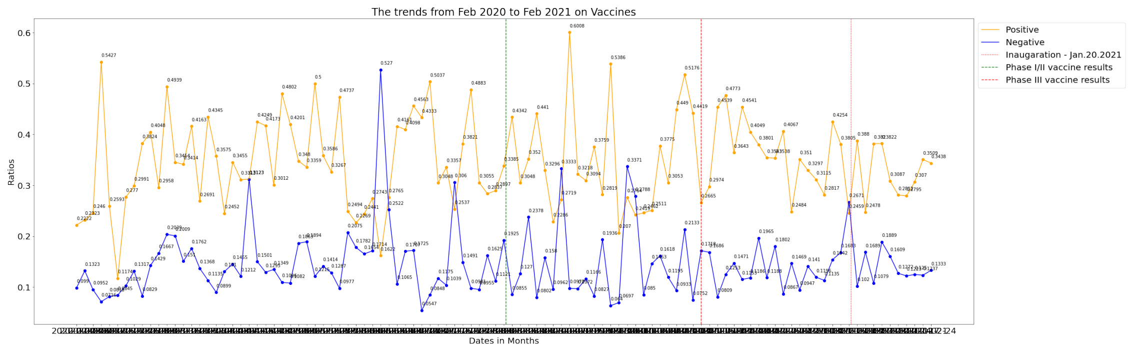
One of the initial reasons why we didn't have a overal plot was it was hard to read but after iterations, we believe it provides a nice visual.

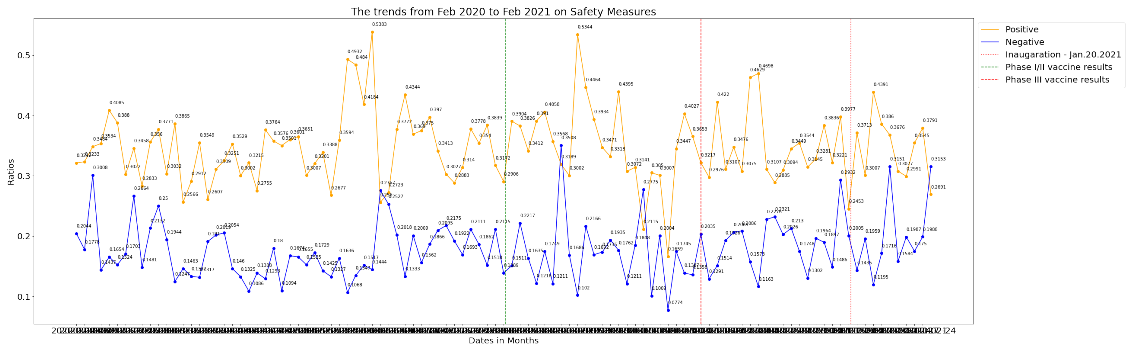
```
[14]: def draw_yearly_plot(target_list_of_xs, p_list, neu_list, neg_list, title,␣
       ↪with_neutral=True):
          """
          type: with_neutral - {False, True}, false means witout neutral
          """
          annotating_list = [p_list, neu_list, neg_list]
          if with_neutral is False:
              annotating_list = [p_list, neg_list]
          # plt.figure(figsize=(13, 7))
      #     plt.figure(figsize=(26, 9))
          plt.figure(figsize = (39, 13))
      #     plt.figure(figsize=(52, 28))
      #     plt.figure(figsize=(75, 25))
      #     plt.figure(figsize=(99, 33))
          plt.rcParams.update({'font.size':20})
          if with_neutral is True:
              plt.scatter(target_list_of_xs, neu_list, color = "gray")
          plt.scatter(target_list_of_xs, p_list, color="orange")
          plt.scatter(target_list_of_xs, neg_list, color = "blue")

          ## anotating the values
          for lists in annotating_list:
              y = lists
              x = target_list_of_xs
              for i, txt in enumerate(y):
                      plt.annotate(txt, (x[i], y[i]),xytext=(x[i], y[i] + 0.
       ↪01),fontsize=10)

          plt.plot(target_list_of_xs, p_list, color="orange", label = "Positive")
          if (with_neutral is True):
              plt.plot(target_list_of_xs, neu_list, color = "gray", label = "Neutral")
          plt.plot(target_list_of_xs, neg_list, color = "blue", label = "Negative")

          plt.xlabel("Dates in Months")
          plt.ylabel("Ratios")
          plt.title("The trends from Feb 2020 to Feb 2021 on %s" % title)
          plt.axvline(x = 94.25, color = 'r', alpha=0.9, linestyle = ":",␣
       ↪label="Inaugaration - Jan.20.2021")
          plt.axvline(x = 52.25, color = 'g', alpha=0.9, linestyle = "--",␣
       ↪label="Phase I/II vaccine results")
          plt.axvline(x = 76, color = 'r', alpha=0.9, linestyle = "--", label="Phase␣
       ↪III vaccine results")
          plt.legend(bbox_to_anchor=(1, 1))

      draw_yearly_plot(d1, p1, neu1, neg1, "Covid-19")
```



Following the same fashion, we can visualize the other two categories

```
[15]: draw_yearly_plot(d3, p3, neu3, neg3, "Vaccines", False)
      draw_yearly_plot(d2, p2, neu2, neg2, "Safety Measures", False)
```





Some aspects are still very hard to visualize such as the individual dates but the markets have also been added to provide some context.

COVID-19 The overall sentiment levels of COVID-19 have been very volatile, albeit hovering around the same ratio values. The events themselves do not seem to have any meaningful impact on the overall public sentiment but the trend lines seem to smoothen out as time passes. We believe it's because COVID-19 had essentially become a part of everyday life. While the news of a highly contagious and lethal virus alarmed many people in the beginning months, after 13 months, many people have normalized it.

Vaccines Some interesting events are extenuated. In the beginning of the pandemic, the anti-vax community has been very vocal about their opposition to any vaccine that comes out. While the topic has been one of debate, many who typically vaccinate themselves have also shared similar sentiments. Over half the tweets in the beginning of the pandemic have shown to be negative towards vaccines and the notion of taking a vaccine developed in less than a quarter of the average time did scare many people. One solace is that after the repeated vaccine phase results, overall negative sentiment has normalized with some spikes.

We tend to see much more spikes either directly before and after a trial results. Most phase results of the vaccine have shown very promising results which restore the faith in the public but talk about vaccine supply vs demand may be a result. Another aspect we wish we could've researched more into.

Safety Overall the positive and negative sentiments have been extremely volatile throughout the entire period but it can be visually seen that the negative sentiments had a smoother trend in the first half of the timeline than in the second. We're not entirely sure for what reason this had happened - where public sentiment towards safety measures seem to flip-flop very frequently. One hypothesis we have is due to the different states having different lockdown/quarantine measures enforced and repealed at different times. If we had more time, we could break this further by state and viewing each state's trends along with each state's safety mandates.

1.3.1 So What's Next?

The plot visualizations have provided good insight and context for what the public feels overall and how specific events in the timeline had affected it. However, not all negative sentiment tweets may be directly attacking the category itself. An example are some tweets we've found during exploratory data analysis that had shown to be a negative tweet but its meaning did not appear that way overall. A tweet complaining, criticizing, or attacking people for not wearing a mask will have a high negative sentiment score but can be seen as a overall positive in terms of its meaning.

1.3.2 What Specific Things are Being Said About Each Keyword Category?

After seeing that there was significance in sentiment trends over time for each keyword categories, we want to explore further and go lower-leveled to see why. In this context, we thought it fitting to utilize topic modeling and word clouds.

[16]:
```
import pickle, re, os
import pandas as pd
from collections import Counter
from gensim import matutils, models
import scipy.sparse

"""
Slightly modified to only examine text. We know which month & year
```

```
each tweet is from. Additionally, we also know which keyword category each
tweet belongs to by how we organized our dataset directories.
"""
def add_tweets_in_df(target_file_path):
    """
    @param:
    target_file_path: string - usually a csv file
    target_dataframe - pandas.df

    @return:
    result_df - pandas data frame
    """
    df = pd.read_csv(target_file_path)
    df = df[["text"]]
    tweetlist = df['text'].tolist()
    return tweetlist

# this function will iterate files in each folder and write dataframes
# into new csv files, and return the dataframes for future uses
def write_csv(target_folder_name):
    """
    ptype: folder_naem - str
    rtype: month_list - list
    """
    month_list = {}
    for file in os.listdir(target_folder_name):
        if file[-4:] == '.csv':
            df = add_tweets_in_df(target_folder_name + file)
            month_list[file[:-4]] = df
    return month_list
```

Now we only need the text for each tweet for further analysis.

[17]:
```
vaccine_dictionary = write_csv('data/vaccine/')
safety_dictionary = write_csv('data/safety/')
covid_dictionary = write_csv('data/covid/')
```

[18]:
```
dates = ['2020_02', '2020_03', '2020_04', '2020_05', '2020_06', '2020_07',␣
 ↪'2020_08', '2020_09',
             '2020_10', '2020_11', '2020_12', '2021_01', '2021_02']

names = ['_feb_20', '_march_20', '_april_20', '_may_20', '_june_20',␣
 ↪'_july_20', '_august_20',
            '_sept_20', '_oct_20', '_nov_20', '_dec_20', '_jan_21', '_feb_21']

dictionaries = [vaccine_dictionary, safety_dictionary, covid_dictionary]
keywords = ['vaccine', 'safety', 'covid']
```

```python
for category, dictionary in zip(keywords, dictionaries):
    for date, month_name in zip(dates, names):
        dictionary[date] = dictionary.pop(f"{category}{month_name}")
```

1.3.3 Word Clouds

To see the most common words being used for each month, we believe we can gain insight into what the usual feeling towards these topics are.

We cannot utilize sentiment or LIWC metrics alone for this. Previously, we have found that a tweet that had high marks for anger and negative emotion (negemo)

[19]:
```python
def combine_text(list_of_text):
    '''Takes a list of text and combines them into one large chunk of text.'''
    combined_text = ' '.join(list_of_text)
    return combined_text

combined_data = {'vaccine' : {}, 'safety' : {}, 'covid': {}}
keywords = ['vaccine', 'safety', 'covid']
combined_dfs = {}
for keyword, dictionary in zip(keywords, dictionaries):
    combined_data[keyword] = {key: [combine_text(value)] for (key, value) in␣
 ↪dictionary.items()}
    data_df = pd.DataFrame.from_dict(combined_data[keyword]).transpose()
    data_df.columns = ['transcript']
    combined_dfs[keyword] = data_df.sort_index()
```

Now to clean the data through our round 1 and round 2 cleaning methods. We will also save corresponding pickle files that can be unpacked and used later.

[20]:
```python
for k, v in combined_dfs.items():
    combined_dfs[k] = pd.DataFrame(v.transcript.apply(round1))
    combined_dfs[k] = pd.DataFrame(combined_dfs[k].transcript.apply(round2))
```

[21]:
```python
# We are going to create a document-term matrix using CountVectorizer, and␣
 ↪exclude common English stop words
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(stop_words = 'english')
combined_cvs = {}
for k, v in combined_dfs.items():
    data_cv = cv.fit_transform(v.transcript)
    data_dtm = pd.DataFrame(data_cv.toarray(), columns=cv.get_feature_names())
    data_dtm.index = combined_dfs[k].index
    combined_cvs[k] = data_dtm
    data_dtm.to_pickle(f"dtm_{keyword}.pkl")
```

[22]:
```
for k,v in combined_cvs.items():
    combined_cvs[k].to_pickle(f"{k}_dtm_tweets.pkl")
    combined_dfs[k].to_pickle(f"{k}_clean_tweets.pkl")
    pickle.dump(cv, open(f"cv_{k}_tweets.pkl", "wb"))
    # Saved in each corresponding file based on keyword
```

[23]:
```
# Find the top 30 words in all the months for each keyword's related tweets
top_dict = {'vaccine': {}, 'covid': {}, 'safety':{}}
combined_data = {}
for keyword in keywords:
    combined_data[keyword] = pd.read_pickle(f"./pickles/{keyword}_dtm_tweets.
 ↪pkl").transpose()
for keyword in keywords:
    for c in combined_data[keyword].columns:
        top = combined_data[keyword][c].sort_values(ascending=False).head(30)
        top_dict[keyword][c] = list(zip(top.index, top.values))
```

Let's pull the top 30 words for each month for each keyword category

[24]:
```
words = {'vaccine': [], 'covid': [], 'safety':[]}

for k,v in combined_data.items():
    for c in v.columns:
        top = [word for (word, count) in top_dict[k][c]]
        for t in top:
            words[k].append(t)
```

[25]:
```
add_stop_words =␣
 ↪['just','dont','el','que','en','por','del','los','vaccine','se','es','like','da','al','una'
                   ␣
 ↪'casos','la','covid','coronavirus','amp','mask','masks','got','says','say','need','y','para
                    ,'o','las']
```

Now we're going to be adding our stop words back into our document-term matrix

[26]:
```
# Let's update our document-term matrix with the new list of stop words
from sklearn.feature_extraction import text #text contains the stopword list␣
 ↪for sklearn
from sklearn.feature_extraction.text import CountVectorizer

combined_clean = {}
combined_stop = {}
# Add our own stop words
stop_words = text.ENGLISH_STOP_WORDS.union(add_stop_words)

# Recreate document-term matrix
cv = CountVectorizer(stop_words=stop_words)
```

```
for keyword in keywords:
    combined_clean[keyword] = pd.read_pickle(f"./pickles/{keyword}_clean_tweets.
 ↪pkl")
    data_cv = cv.fit_transform(combined_clean[keyword].transcript)
    combined_stop[keyword] = pd.DataFrame(data_cv.toarray(), columns=cv.
 ↪get_feature_names())
    combined_stop[keyword].index = combined_clean[keyword].index # New document␣
 ↪term matrix
    pickle.dump(cv, open(f"{keyword}_stop_tweets.pkl", "wb"))
```

1.3.4 Wordclouds!

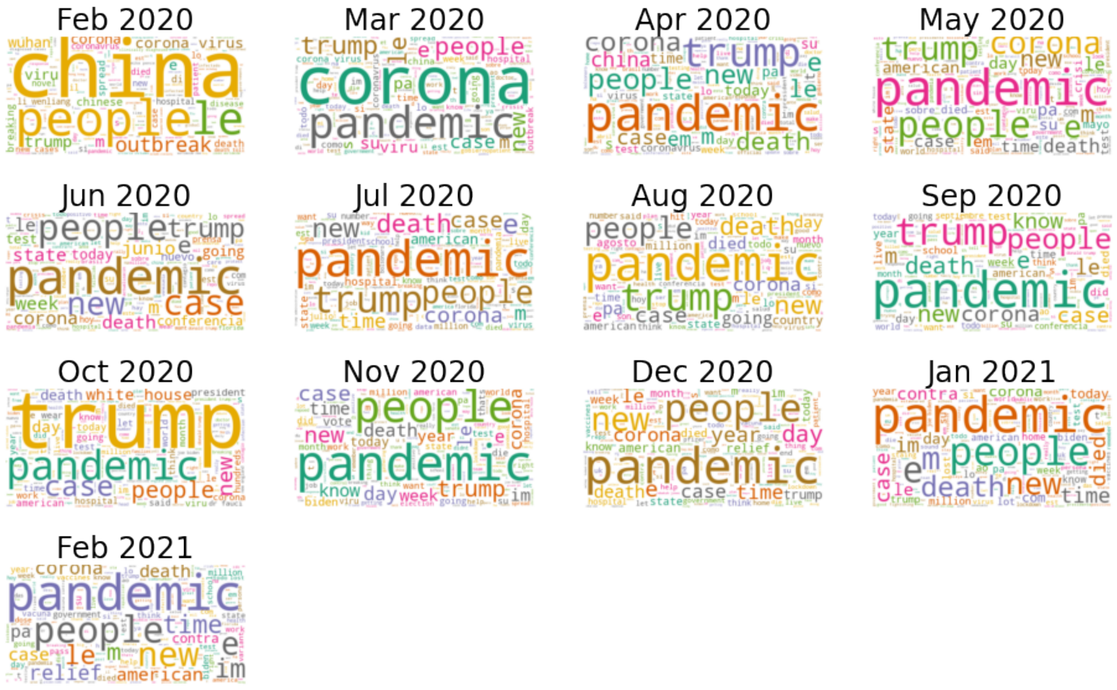
Long awaited results incoming!

[27]:
```
from wordcloud import WordCloud
plt.rcParams['figure.figsize'] = [16, 10]
full_names = ['Feb 2020', 'Mar 2020', 'Apr 2020', 'May 2020', 'Jun 2020', 'Jul␣
 ↪2020',
              'Aug 2020', 'Sep 2020', 'Oct 2020', 'Nov 2020', 'Dec 2020', 'Jan␣
 ↪2021','Feb 2021']
#create word cloud object
wc = WordCloud(stopwords=stop_words, background_color="white", colormap="Dark2",
               max_font_size=150, random_state=42)
#Read in our data

for keyword in keywords:
    if keyword != 'covid': # Mismatched format. Not sure where it misformatted␣
 ↪but won't transpose.
        combined_data[keyword] = pd.read_pickle(f"./pickles/dtm_{keyword}.pkl").
 ↪transpose()
    else:
        combined_data[keyword] = pd.read_pickle(f"./pickles/dtm_{keyword}.pkl")

def gen_wordcloud(keyword):
    for index, c in enumerate(combined_data[keyword].columns):
        wc.generate(combined_clean[keyword].transcript[c])
        plt.subplot(4, 4, index+1)
        plt.imshow(wc, interpolation="bilinear")
        plt.axis("off")
        plt.title(full_names[index])
    plt.show()
```

1.3.5 Covid-19 Word Cloud

We've removed some of our querying keywords such as “covid” and “coronavirus” to get better results for the word cloud.

```
[28]: gen_wordcloud('covid')
```



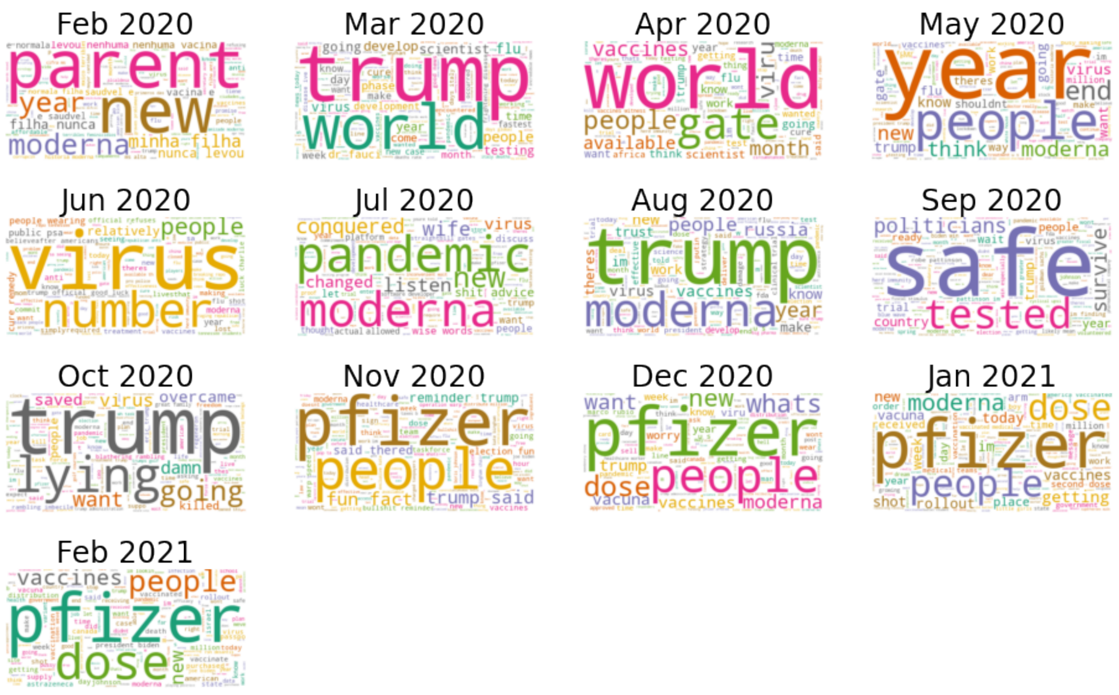
1.3.6 Observations from word cloud of COVID

- **Feb 2020**: As the pandemic had just started, we can see alot of mention about **China, wuhan** and **outbreak**.
- **Pandemic** is common across all the 13 months. similarly, **Trump** seems to be there until 2020 and starts to disappear from 2021. Much of the controversy surrounding the virus has been a result of Trump so without him in power or giving him publicity, people do not mention him as often in relation to the pandemic (and our categories).
- **Death** is one of the highlights in the month of April, May and september as expected. April was the month when covid death count started to increase. In september, COVID-19 related deaths repeatedly broke the previous day's record. New daily death count records were being met nearly everyday during this month.
- **Nov 2020**: This month has a mention of **Vote** reflecting on the election month
- **Feb 2021** : Jan has a mention of **relief** , as the relief packages were in news as well due to the presidential inauguration but Biden's name does not seem to appear frequently.

1.3.7 Vaccine Word Cloud

We've removed a few keywords like "vaccine" and "vaccines" to get better results in our word cloud.

```
[29]: gen_wordcloud('vaccine')
```

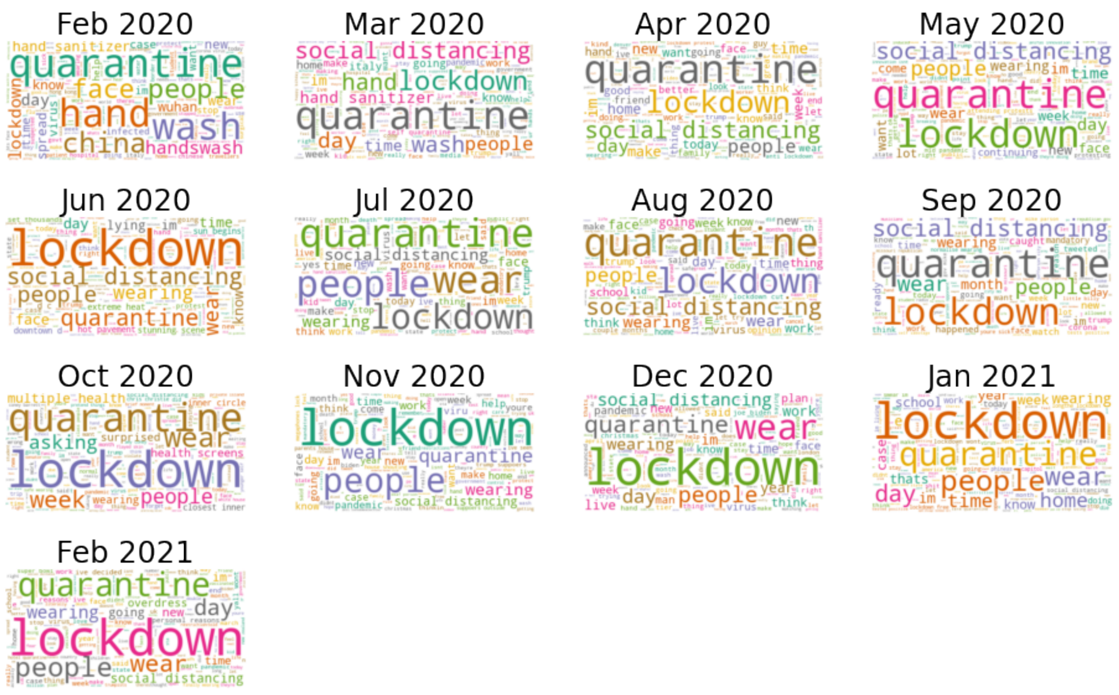


1.3.8 Observations from word cloud of Vaccine

- **Feb 2020**: As news of the virus first starts becoming widespread, we see concerns over people's **parents**, **moderna** who announced their work towards the vaccine, and some mentions of **anti** - signalling the anti-vax community.
- **Mar 2020**: March 2020 shows words like develop and testing as the world governments were starting with vaccines preperations.
- **Moderna** was the highlight before and after phase 2 results - July and August 2020.
- **Oct 2020** highlights **lying**. This is the time when former President Trump contracted COVID-19 and played off the results. Many in the public suspected it to be a farce; and as a result, **lying** has shown up quite frequently.
- **Pfizer** Once the vaccine phase 3 result came in november, there is mention of pfizer in almost all the months. Also, **dose** is the highlight in past three months as people took the vaccines.

1.3.9 Safety Word Cloud

[30]:
```
gen_wordcloud('safety')
```



1.3.10 Observations from word cloud of Safety

- As we collected tweets with keywords like social distancing, safety, and quarantine. There is nothing negative or interesting about these tweets.
- The tweets are similar across all the 13 months but seem to become even more similar as time goes on. In February 2020, we see a wide array of different words relating to safety - Eg. "hand sanitizer", "handswash", "wash", "face", "wear masks", etc. However, as the months go on, the variety in safety-related keywords diminish to just "social distancing", "lockdown", and "quarantine".
- As time progressed, these safety-measures have become a behavioral norm and does not have as many mentions anymore. We believe, since it's been over 13 months, that it's become a normal part of life for many; and as a result, have fewer mentions of those keywords as time progressed.

1.3.11 Some Expected Words are Missing...

We've had our own list of words to expect to have high frequency but the word cloud and its random state nature may not fully show it. Luckily, we've saved all of our **pkl** files so we can utilize topic modeling as well. **Note**: Putting our topic modeling method has had problems printing to standard output which is why we have repeating code blocks.

COVID-19 Topic Modeling

[31]:
```
from gensim import matutils, models
data = pd.read_pickle('./pickles/dtm_stop_tweets.pkl')
# One of the required inputs is a term-document matrix
```

```
tdm = data.transpose()
# We're going to put the term-document matrix into a new gensim format, from df␣
↪--> sparse matrix --> gensim corpus
sparse_counts = scipy.sparse.csr_matrix(tdm)
corpus = matutils.Sparse2Corpus(sparse_counts)
# Gensim also requires dictionary of the all terms and their respective␣
↪location in the term-document matrix
cv = pickle.load(open("./pickles/cv_stop_tweets.pkl", "rb"))
id2word = dict((v, k) for k, v in cv.vocabulary_.items())
# Now that we have the corpus (term-document matrix) and id2word (dictionary of␣
↪location: term),
# we need to specify two other parameters as well - the number of topics and␣
↪the number of passes
lda = models.LdaModel(corpus=corpus, id2word=id2word, num_topics=5, passes=10)
lda.print_topics()
```

[31]:
```
[(0,
  '0.006*"pandemic" + 0.004*"trump" + 0.004*"people" + 0.002*"new" +
0.002*"corona" + 0.002*"cases" + 0.002*"died" + 0.001*"deaths" +
0.001*"americans" + 0.001*"positive"'),
 (1,
  '0.008*"pandemic" + 0.005*"people" + 0.004*"new" + 0.004*"trump" +
0.003*"cases" + 0.003*"corona" + 0.002*"im" + 0.002*"health" + 0.002*"deaths" +
0.002*"contra"'),
 (2,
  '0.008*"trump" + 0.007*"pandemic" + 0.004*"new" + 0.004*"people" +
0.004*"cases" + 0.003*"president" + 0.002*"realdonaldtrump" + 0.002*"white" +
0.002*"house" + 0.002*"day"'),
 (3,
  '0.008*"china" + 0.006*"cases" + 0.005*"corona" + 0.005*"virus" + 0.004*"new"
+ 0.003*"wuhan" + 0.003*"outbreak" + 0.003*"trump" + 0.003*"people" +
0.003*"le"'),
 (4,
  '0.008*"pandemic" + 0.005*"trump" + 0.005*"people" + 0.004*"corona" +
0.004*"new" + 0.003*"cases" + 0.002*"virus" + 0.002*"im" + 0.002*"health" +
0.002*"deaths"')]
```

Vaccine Topic Modeling

[32]:
```
data = pd.read_pickle('./pickles/dtm_stop_vaccine.pkl')
# One of the required inputs is a term-document matrix
tdm = data.transpose()
# We're going to put the term-document matrix into a new gensim format, from df␣
↪--> sparse matrix --> gensim corpus
sparse_counts = scipy.sparse.csr_matrix(tdm)
corpus = matutils.Sparse2Corpus(sparse_counts)
```

```
# Gensim also requires dictionary of the all terms and their respective␣
↪location in the term-document matrix
cv = pickle.load(open("./pickles/cv_stop_vaccine.pkl", "rb"))
id2word = dict((v, k) for k, v in cv.vocabulary_.items())
# Now that we have the corpus (term-document matrix) and id2word (dictionary of␣
↪location: term),
# we need to specify two other parameters as well - the number of topics and␣
↪the number of passes
lda = models.LdaModel(corpus=corpus, id2word=id2word, num_topics=5, passes=10)
lda.print_topics()
```

[32]:
```
[(0,
  '0.007*"trump" + 0.006*"pfizer" + 0.005*"people" + 0.005*"moderna" +
0.003*"vaccines" + 0.003*"doses" + 0.003*"realdonaldtrump" + 0.003*"virus" +
0.003*"im" + 0.003*"getting"'),
 (1,
  '0.009*"moderna" + 0.005*"trump" + 0.005*"people" + 0.004*"virus" +
0.004*"gates" + 0.004*"new" + 0.003*"world" + 0.003*"vaccines" + 0.003*"flu" +
0.003*"year"'),
 (2,
  '0.007*"people" + 0.006*"moderna" + 0.006*"pfizer" + 0.004*"vaccines" +
0.003*"virus" + 0.003*"americans" + 0.003*"biden" + 0.003*"trump" +
0.003*"public" + 0.003*"im"'),
 (3,
  '0.008*"trump" + 0.006*"people" + 0.005*"moderna" + 0.005*"pfizer" +
0.004*"virus" + 0.004*"vaccines" + 0.003*"new" + 0.003*"im" + 0.003*"going" +
0.003*"flu"'),
 (4,
  '0.010*"trump" + 0.005*"people" + 0.005*"pfizer" + 0.005*"moderna" +
0.004*"said" + 0.004*"im" + 0.004*"safe" + 0.004*"realdonaldtrump" +
0.003*"wont" + 0.003*"tested"')]
```

Safety-Measure Topic Modeling

[33]:
```
data = pd.read_pickle('./pickles/dtm_stop_safety.pkl')
# One of the required inputs is a term-document matrix
tdm = data.transpose()
# We're going to put the term-document matrix into a new gensim format, from df␣
↪--> sparse matrix --> gensim corpus
sparse_counts = scipy.sparse.csr_matrix(tdm)
corpus = matutils.Sparse2Corpus(sparse_counts)
# Gensim also requires dictionary of the all terms and their respective␣
↪location in the term-document matrix
cv = pickle.load(open("./pickles/cv_stop_safety.pkl", "rb"))
id2word = dict((v, k) for k, v in cv.vocabulary_.items())
# Now that we have the corpus (term-document matrix) and id2word (dictionary of␣
↪location: term),
# we need to specify two other parameters as well - the number of topics and␣
↪the number of passes
lda = models.LdaModel(corpus=corpus, id2word=id2word, num_topics=5, passes=10)
lda.print_topics()
```

```
[33]: [(0,
  '0.001*"quarantine" + 0.001*"lockdown" + 0.001*"people" + 0.000*"distancing" +
0.000*"wear" + 0.000*"social" + 0.000*"wearing" + 0.000*"im" + 0.000*"face" +
0.000*"time"'),
 (1,
  '0.009*"quarantine" + 0.008*"lockdown" + 0.008*"wear" + 0.007*"people" +
0.007*"social" + 0.006*"distancing" + 0.006*"wearing" + 0.004*"face" +
0.003*"im" + 0.003*"new"'),
 (2,
  '0.019*"lockdown" + 0.016*"quarantine" + 0.009*"people" + 0.006*"wear" +
0.005*"social" + 0.005*"distancing" + 0.005*"wearing" + 0.004*"im" +
0.004*"face" + 0.004*"hands"'),
 (3,
  '0.019*"lockdown" + 0.018*"quarantine" + 0.008*"people" + 0.007*"social" +
0.006*"distancing" + 0.005*"wear" + 0.005*"im" + 0.004*"wearing" + 0.003*"new" +
0.003*"day"'),
 (4,
  '0.012*"quarantine" + 0.011*"lockdown" + 0.008*"social" + 0.007*"distancing" +
0.007*"people" + 0.004*"wearing" + 0.004*"wear" + 0.003*"face" + 0.003*"im" +
0.003*"new"')]
```

On initial glance, **safety**'s topic modeling has not provided much meaningful output. One insight is the COVID-19 topic modeling where you can see some notable events occur. 'Trump' being a direct word for many of these tweets. As racially divisive narrative has circulated, many blamed China and not the United State's poor handling of the pandemic. As time progressed however, Trump's name has become mentioned less and less and general news has shifted towards reporting on just the pandemic itself.

Vaccines is showing non-conclusive output. Only solid insight we could make is the mention of "Biden" towards the more recent tweets which can be a direct effect of his pledge to provide vaccines to every person in the U.S.

1.4 Conclusion & Afterthoughts

As we've progressed in this project, we were able to track the sentiment of various categories across the entirety of the pandemic to-date. Through this process, we were able to see definite correlation of specific events to the public's sentiment towards each specific topic. By visualizing the plots with certain markets, it shows a clear and concise timeline for each keyword as well as their trends for the future.

1.4.1 Limitations

We recognize that Twitter is only one of many social media platforms that people voice their concerns, frustrations, and opinions on. Furthermore, *Big Questions for Social Media Big Data:*

Representativeness, Validity, and Other Methodological Pitfalls by Tufekci have highlighted the extrapolative nature of utilizing data from Twitter. While its benefits include transparency, querying, and ease-of-use, Twitter data isn't necessarily representative of the whole public. With approximately ~20% of the population having used or are currently using it, our findings may not correctly represent the whole population.

One thing we originally planned on doing was doing the same experiment with data from Reddit but for time purposes, we were unable to. The data collection, cleaning, and visualization all took considerable amount of time. We believe it'd provide even more fascinating insights. Most accounts on Twitter are tied to people's real-life identities; and thus, people behave much differently since their online behavior is in a sense, a reflection of their real behavior. Reddit, by contrast, has a layer of anonymity which may introduce more "troll" comments and posts but also provides a sense of some people's more extreme or honest opinions on topics.

Lastly, we started off utilizing the Linguistics Inquiry and Word Count (LIWC) tool for exploratory data analysis and had planned to do similar work in our data visualization steps. We decided as a team that looking at the qualitative data would help us gain a better sense of the context. Utilizing LIWC, we found not all negative or angry tweets were about masks or mask mandates but because people refused to wear masks. It required much more data handling and cleaning for that information to be useful.

1.4.2 Ethical Considerations

One of the widely regarded question of ethics in data mining is consent. While users agree to have their posts be public when they set their account as "public", most users are unaware their information may be used to generate projects such as these.

A concern of participants whose information is in a database is the question of anonymity. Could any individual be singled out from an aggregation of many users. The way we collected data and processed it, we did not take into consideration the Twitter account which it was from; and as a result, cannot tie the processed data to any single individual. The corpus information was translated into quantitative metrics and those quantitative metrics were used for all of the analysis.

Twitter is majority Gen-Z and millenials who are proportionately left-leaning and/or liberals. The politicized nature of the pandemic may disproportionately show the younger generations and not accurately reflect the older generations.

1.4.3 Future Work

As mentioned, because of the amount of additional work LIWC would've required, we had removed it from our final phase plans. If we had more time, we want to utilize subsets of LIWC metrics such as: clout, authority, posemo, negemo, anger, grammer, and ppron. Combining our existing work with these additional metrics may help gain lower level insights. An example to help illustrate this would be correlating the various emotional metrics from LIWC to the common words shown in the word cloud which provides the context in which the words are being said.

Previously mentioned, coupling the Twitter data with Reddit and possibly other platforms can help gain a larger sense of public sentiment. Twitter is only a small subset of the overall population and does serve well for data analysis but we cannot be over-reliant on it.

1.5 Credit Listings & Notes

Note 1: All the data pertinent to analysis is in the directory **data**. All of our pickle (.pkl) files used for word clouds and topic modeling are in the directory **pickles**. Lastly, we have a slightly modified version of our data where the files are organized by month and year. The data itself is the same as the data in the first directory but the organization is different for easier handling. This data is in **processed_data**.

Note 2: Our code for producing every aspect of the project is included in this Jupyter Notebook. This also includes all of our code to generate pickle (.pkl) and cleaned csv files. It will generate and save the .csv and .pkl files in the same directory as the program. It is not a huge amount of space but we thought we should warn the reader.

1.5.1 Yuanfeng Li

Phase 1: * Code for scraping current data (2021) * Collected tweets relating to masks based on keywords (Eg. wearamask, masks4all, n95, respirator, etc.)

Phase 2: * Processed data into new CSV files for future manipulations with different sizes: Original and filtered. Filtered files contain tweets which explicitly mention "mask" in them. * Calculated the average ratio for positive, negative, and neutral tweets, and plotted them. Added visual markers to indicate events such as the presidential inauguration day. * Analyzed the tweets sentiments toward the 'Mask' related keywords and hashtags, compared with the results with one of our initial hypotheses.

Phase 3: * Revised the visualization from Phase 2, digged more data points by different dates in each month, added more data points in the visualizations, upgraded the month to dates for X-axis of the final visualization. * Implemented more accurate visualizations, and performed better analysis from them! * Proved one of our initial hypotheses is correct! Extracted few insights from the upgraded visualization!

1.5.2 Ji Kang

Phase 1: * Collected past Twitter data using IEEE and Kaggle Tweet-ID datasets * Ran data through LIWC to generate LIWC metrics * Exploratory data analysis on LIWC to content of tweets * Project Manager

Phase 2: * Refactored scraping program to utilize Twitter API V2 * Scraped tweets, separated by month and keyword category, for months February 2020 to February 2021. * Cleaned CSV files for team use * Project Manager

Phase 3: * Aggregated findings into report * Refactored visualization code - Plots & Word cloud * Project Manager

1.5.3 Lipsa Jena

Phase 1: * Code for scraping current data (2021) * Collected tweets relating to vaccine (Eg. keywords = covidvaccine, vaccineswork, coronavaccine, vaccine2021, #vaccines, etc.)

Phase 2: * Processed data into new CSV files for future manipulations with different sizes: Original and filtered. Filtered files contain tweets which explicitly mention "vaccines" in them. * Calculated the average ratio for positive, negative, and neutral tweets, and plotted them. Added visual markers to indicate events such as the presidential inauguration day, vaccines phase I, phase II and phase III release dates as well. * Analyzed the tweets sentiments toward the vaccines related keywords and hashtags, compared with the results with one of our initial hypotheses.

Phase 3: * WordClouds for our topics. So in the process of creating word clouds for all the tweets collected so far, did data cleaning for all the processed data * Did analysis of the words on cloud and correlated them to the events taking place over last year (2020) and events of the current year as well. * Did topic modeling for all the three segments of our project.

1.5.4 Yu Ling

Phase 1: * Code for scraping current data (2021) * Collected tweets relating to masks (Eg. keywords = socialdistance, social distancing, quarantine, lockdown, wash hands, etc.)

Phase 2: * Processed the data for Safety file, it includes 13 months related Tweets data, from Jan 2020 to Feb 2020. Read the CSV files and filter the keyword for social distancing. * Calculated the average ratio for positive, negative, and neutral tweets, and plotted them. Added visual markers to indicate events such as the presidential inauguration day. * Analyzed the tweets sentiments toward the 'social distancing' related keywords and hashtags, compared with the results with one of our initial hypothesis.

Phase 3: * Search on some visualization approaches that are best related to our current and future goals. (Interactive Visualization) * Trying to use machine learning models to make predictions on the data. * Explore future expansion on our topic for COVID.