CSC110 Project : A Sentiment Analysis Oriented Investigation of the Impact of COVID-19 on the Political Satisfaction of the General Public

Aviraj Newatia, Mishaal Kandapath, Rudraksh Monga, Taylor Whatley

Tuesday, December 14, 2021

Problem Description and Research Question

Politics is a very polarizing field of study - even before COVID-19, political opinions varied vastly among differing political ideologies, but ever since the coronavirus went viral the onset of political mistrust, misinformation, and fake news has made the political landscape even more difficult to navigate. The public political opinion in countries such as Bosnia and Herzegovina, for example, was largely negative as many blamed the government for prematurely opening the border, leading to increased death rates, while the public's political opinion in other countries such as China, for example, was largely positive due to the efficacy with which it handled the pandemic [1]. Considering how many countries were affected by COVID-19, our group decided to narrow our focus to just one country, the US, because of its economic importance in the global economy and because of how much the country has stood out during the pandemic (due to things such as Donald Trump's infamous "disinfectant" cure). Given how governmental decisions could cost the lives of hundreds during the pandemic, our group decided that it would be best for us to analyse the US public's opinions with the government during this pandemic.

Reddit [https://www.reddit.com/] is a social media website revolving around user-led communities. It has been typically perceived as a digital platform for people to express their opinions, especially for younger populations with access to the Internet. Subreddits such as r/usapolitics have been integral platforms for the community's expression of their political opinions, especially during periods of political unrest or public disapproval (such as during elections or pandemics). Being active members of the Reddit community ourselves, our group decided to look at how people's political opinions, as expressed on various different subreddits, changed during the progression of COVID-19. Research has shown that people tend to be more honest on online forums where they can remain anonymous [2], so a public platform such as Reddit was an ideal location to ground this investigation. Research similar to what we are investigating use sentiment analysis to analyse comments on different subreddits suited for their investigations, and this illuminates the approaches we should take to analyse comments for our investigation as well [3], [4]. As such, to find answers to our questions our group has decided to answer the question of **"How Has the Political Satisfaction of the General Public Changed During the COVID-19 Pandemic?"** using sentiment analysis of comments on political subreddits before and during COVID-19.

Dataset Description

The dataset we used, 'The Reddit Covid Dataset' [https://www.kaggle.com/pavellexyr/the-reddit-covid-dataset][5], was sourced from the online machine learning data repository Kaggle, a subsidiary of Google LLC, which serves as an online community of data scientists and machine learning practitioners. It is a CSV(comma-separated values) file of 17777472 unique comments related to COVID that were collated from the online forum platform Reddit.

The dataset contained the following columns:

- 'type': Type of the datapoint
- 'id': Unique Base-36 ID of the comment
- 'subreddit.id': Unique Base-36 ID of the comment's subreddit
- 'subreddit.name': Human-readable name of the comment's subreddit

- 'subreddit.nsfw': Is the comment's subreddit NSFW?
- 'created_utc': Timestamp of the comment's creation
- 'permalink': Permalink to the comment on Reddit
- 'body': Comment's body text
- 'sentiment': Analyzed sentiment for the comment
- 'score': Comment's score

However, out of all the columns available in the dataset, we only used 'created_utc' and 'body' in our computation. While there already was a column containing the analysed sentiment for each commend ('sentiment'), we chose to not use this in our analysis and create our own sentiment analyzer instead because we needed a standardized analyzed sentiment that would be valid across all the datasets we used in this investigation; we were unsure as to how accurately the analyzed sentiment provided by the dataset would compare if applied to comments from other datasets.

Additionally, we filtered out a sub-dataset containing the comments from political subreddits. Further details about this can be found in the computation overview.

We also used Sentiment140[6], a dataset containing sentiment labels for 1.6M unique tweets. It has 1.6M rows and 6 columns with the first column giving the sentiment as an integer (0=negative, 4=positive), and the last being the tweets themselves. Additionally pre-processing was not conducted for this dataset. This dataset and these two columns are used to train the sentiment analysis model.

Computation Overview:

Cleaning Dataset:
First, we used the 'The Reddit COVID Dataset' to extract a sub-dataset containing comments from political subreddits such as r/uspolitics. This was done by filtering the dataset to select rows that contained comments from the subreddits that were suited to our investigation (to be specific, subreddits related to US government activity, such as r/uspolitics). Once we had the datasets, we used the Pandas library to preprocess the dataset so it could be used by the ML model. Out of the 10 columns of the datasets extracted, we used only 2: the timestamp of the comment ("created_utc") and the comment itself ("body"). All other columns were dropped using Panda's pandas.DataFrame.drop() function, with its inplace parameter set to True so that it mutates the dataframe instead of returning a new copy. Then, we added an 'id' column to the dataframe, which used integers to represent the id of rows to allow for row indexing, if needed. This was done using Panda's pandas.DataFrame.insert() function to insert an 'id' column at the 0^{th} index, and then we used pandas.DataFrame.set_index()[7] to make the new column an index for the rows. Finally, to remove rows with any empty values, we used pandas.DataFrame.dropna(), with its inplace parameter set to True to mutate the dataset.

Gathering Pre-Covid Reddit Comments:
To obtain pre-covid reddit comments relating to the political nature of our investigation, we wrote a module utilising the Reddit API and the Python praw module to scrape the online forum website for comments before a certain date. To do this we initialised an instance of the praw.Reddit class with details for the reddit API and used this instance to grab comments from the submissions (posts) of a given subreddit. We also filtered to only retrieve comments from before timestamp 1577817000 which is the beginning of 2020. Additionally we utilised the datetime.datetime.fromtimestamp() function from the python datetime module to convert the timestamp provided by reddit into a datetime object so that we could easily understand when the comment was posted and the circumstances surrounding that time.

Statistical Approach:
Our statistical model is set up to detect trends in our dataset. In particular, we need to be able to model our comment, normalize our data and present it in a format that is easy to analyze and manipulate by our frameworks, and graph the data in a visual way that is easy to draw conclusions from.
The key classes involved in the representation and normalization sections are "StatisticsCommentInfo" (that holds a date and a sentiment) which gets normalized into the more general "StatisticsPoint" (that holds an x and y

coordinate) by the statistics_normalize method. By doing this, we can turn a complicated structure like comments into a more digestible collection structure like "StatisticsNormalizeResult" which can then be passed to the graph and statistics_analyze methods. StatisticsNormalizeResult holds all StatisticsPoint structures along with essential data for analysis and graphing, such as range of held dates along with min and max sentiment values.
Our final step in our statistical analysis is to infer trends from the points in StatisticsNormalizeResult, which is done by statistics_analyze. This function uses the statistics module's mean, median and mode functions to calculate primitive statistics, along with pandas's DataFrame and corresponding DataFrame.corr() concepts to extract trend data. This will give us an 'r' correlation value that can tell us in what direction sentiment is going over a period of time and how strong time is correlated with sentiment. In addition, we pass the data frame over to numpy's polyfit method to get a 2^{nd} degree polynomial fit on the data[8]. This will provide a fairly sane model of our sentiment over time (in terms of trend) and allow us to predict values of sentiment at points we don't necessarily have data for if necessary. A StatisticsAnalysisResult is then produced with all calculated metrics.
Our graph method takes our StatisticsNormalizedResult object, invokes statistics_analyze and presents the data alongside our fitted polynomial using plotly.express's line and scatter methods to generate and show a figure. We have also provided methods to our main.py method to take more control over statistics. For example, process_average_comments uses the datetime.datetime objects arithmetic operators to average out each day's sentiment and provide a smoother graph. In addition, filter_comments allows users to remove comments that are not within a time period. We use the graph method to present our results from both pre-covid era and covid era datasets in main.py. Together, these provide a strong framework of visualization and analysis tools that can help us get a better picture of the trends related to sentiment over the course of the pandemic.

Sentiment Analysis:
Our sentiment analysis model is based off Bayes Law of Probability:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

If X is a string and C a class, then the probability of the given string belonging to sentiment class C (hereby referred to as posterior); $P(C|X) = \frac{P(X|C)P(C)}{P(X)}$.
Here, $P(X|C)$ is the probability of the string given a class (as in how likely is it for the given string to be outputted given this class). This term will hereby be referred to as the likelihood factor.
$P(C)$ is the probability of the class, that is, given the sentences utilized in training, how any of those belong to the class under consideration. Since our dataset has a balance of positive and negative sentiments, this value is very close to 0.5. This value will hereby be referred to as the prior factor.
The term $P(X)$ is termed as the probability of evidence by Bayes' theorem.
The model does both classification and regression. For classification, it chooses the class with the highest probability of association with the given string. Regression and the overall return type of the model classifier will be elaborated on later. Since we seek to find the maximum value of $P(X|C)$ across all the 'k' classes. Since here we are only considering 2 different class, k = 2. If h is the predicted class of string X:
h = $argmax_k(\frac{P(C_k|X)P(C_k)}{P(X)})$. Since the probability of evidence is a constant factor as it does not depend on the class, and as we seek to consider only the maximum values, we can remove the probability of evidence from the equation as it will not influence in the maximum. Thus we get: h = $argmax_k(P(X|C_k)P(C_k))$.
Since log is an increasing function, we can apply log to the input to get: h = $argmax_k log(P(X|C_k)) + logP(C_k)$.
Now, given that string X consists of n words $x_1, x2, ...xn$, we can re-write h as: $argmax_k log(P(x_1, x2, ...xn|C_k)) + logP(C_k)$. Assuming that all words are independent of each other, we get:
$argmax_k log(\prod_{i \in n} P(x_i|C_k)) + logP(C_k)$
$= argmax_k((\sum_{i \in n} log(P(x_i|C_k))) + logP(C_k))$

The sentiment analysis model has two components to it: Training and Classifying.

- **Training**:
  The model was trained using the Sentiment140 dataset developed by Stanford University utilizing data from Twitter.
  For each sentiment class that we want to classify into (positive and negative in this case), we must calculate the prior factor of that class. Then for the purpose of calculation of the likelihood factor, we must calculate the frequency of all words that occur in all sentences that have been pre-labeled as belonging to the current sentiment class under consideration. Since the calculation of the probability of a word given a class $P(x_i|C_k$

is determined by the number of occurences of the word by the total number of occurrences of all words in the dataset that are in sentences that belong to the sentiment class under consideration, we must also calculate this sum in the train function to avoid calculating it every time the classification algorithm runs (this value is clearly a constant depending on the training examples).
This is all that is required for training the sentiment analysis model.
The function for this is the train method in the SentimentAnalyzer class.

- **Classification and Scoring**:
  Given a sentence we want to calculate $argmax_k((\sum_{i \in n} log(P(x_i|C_k))) + logP(C_k))$ for classification. Thus, we calculate the posterior for each sentiment class.
  Here, the probability of each word in the sentence belonging to that statement is calculated. It is worth noting that the number of times a word occurs in the training examples may be 0. Since zero probability is not favorable for calculation, we add a constant factor 1 to every number of times a word appears in a particular sentiments trained examples. The denominator for calculating probability is adjusted accordingly.
  In order to obtain the actual probability values from 0 to 1, further calculations are required:
  We used the log-sum-exponent rule to obtain this real score value. The classify function returns the class with the maximum posterior factor, the maximum value h, and a dictionary mapping sentiment classes to their real probability scores. The function for this is the classify method in the SentimentAnalyzer class.

How we trained our model:

- We obtained the data from Sentiment140 dataset with 1.6 million unique labeled datapoints. We split the training set into testing and training sets (1:4 ratio) and ran the training algorithm used in our sentiment analysis model. We measured an average accuracy of 77% across 10 tests we conducted. It is worth noting that by specifying the pretrained parameter in our model's initializer to True, pretrained values may be directly imported for direct classification purposes.

Simulation:
In order to demonstrate our project, we decided to simulate the impact that the posting of a single comment could have on the sentiments of individuals within the population of a society. This simulation was designed in a way that used the sentiment analyser model we built to compute the sentiment of an input comment and see how the sentiment of that comment changed people's overall opinions and sentiments.

The OpinionSimulation class was designed as a subclass of the abstract PopulationSentimentSimulation class and initialises with a value for the number of "people" in the population of the society being simulated. Treating OpinionSimulation as self, the main simulation loop is run by the self.run_simulation() function which works by:

- Taking in the comment whose impact will be simulated as a parameter called self.comment
- Computing the sentiment value of that comment using the self.compute_comment_sentiment function and storing it into self.comment_sentiment
- Running the self.generate_responses_sentiment function which does the following
  - Randomly selecting population sentiment impact values (using random.uniform to return float values as to model the entropy of human emotional/sentimental responses) based on the sentiment of the comment (self.calc_impact) and random people within the population to be impacted
  - Impacting these people's sentiments, and generating the sentiment of their responses, which then impact others

This process repeats until the change in sentiment caused by a comment is below a certain value or if the process iterates too many times. This process was originally achieved by using recursion however, after hitting a recursion limit, we changed it to an iterative process utilising a for loop. The sentiment spread was achieved by ensuring that no single person could be impacted repeatedly in a disproportionate manner.

The function then takes all of the values generated throughout the simulation and transforms it into a pandas dataframe (using pandas.DataFrame.from_dict() and pandas.DataFrame.transpose()) before calling and animating a plotly graph (using plotly.express.bar) to produce an html file (using plotly.offline.plot[12]) with an interactive bar chart (with one bar per person in the population) which displays how each person's general sentiment changed over the course of the simulation.

We extended the simulation by writing a manager class, OpinionSimulationManager, as a subclass of the abstract class SimulationManager. This managerial class implemented the functionality of aggregating OpinionSimulation instances with different properties and populations. We also implemented the ability to run each instance on multiple comments simultaneously to simulate the impact of consecutive differing opinions on a populace. Treating OpinionSimulationManager as self, the main managerial simulation process was as follows:

- Instantiating an instance object of OpinionSimulationManager, for purpose of explanation let the object be named object
- Calling object.add_sim_instance to add a singular instance of OpinionSimulation with a certain population as many times as instances aggregated desired by appending each instance to the self.instances list
- Calling object.add_comments which
  - Iterates through each index in self.instances
  - Takes user input for a comment to simulate for the particular instance until the termination keyword ("FIN") is entered at which point the flag_done variable is updated and the while loop terminates
- Calling object.run_simulation which
  - Iterates through each index in self.instances
  - Iterates through each of the comments for each instance
  - Runs the simulation of that instance (instance.run_simulation) on the comments one by one
  - Appends the final sentiments of the populace to the self.results list

This procedure and implementation allows the visualisation of the input of sequential comments and their sentiments on different populaces and also allows us to model the entropy of the impact of an opinionated comment.

Instructions for Obtaining Datasets and Running Program:

In order to obtain the datasets required to run this project, navigate to send.utoronto.ca and click "Pick-up" to collect the files. Use the following claim information to access the files:

Claim ID: Nhc2iJhQNCgnmsws
Claim Passcode: GAV8nvaCohUoUeKn

An alternative option is to use this link (https://send.utoronto.ca/pickup.php?claimID=Nhc2iJhQNCgnmsws&claimPasscode=GAV8nvaCohUoUeKn&emailAddr=aviraj.newatia%40mail.utoronto.ca) to access the files via send.utoronto.ca.

Upon accessing the files, they should be downloaded and placed within a directory called "datasets" (which should be created) within the project folder.

After running main.py the TA should expect to see two plotly graphs opening in the browser, one mapping the sentiment of the comments pre-covid, and one mapping the sentiment of the comments during covid. These plotly graphs will be interactive, and the TA will be able to pan and zoom, download the plot as a png, and select a portion of the graph.

Afterwards, the TA should expect to see another interactive plotly graph in the browser (which is also saved locally as an html file) demonstrating the simulation of the impact of the comment "I really like python" on a population of 100. This plotly char, in addition to the interactive features mentioned above, will have a slider at the bottom, allowing the TA to view the change in the sentiment of the population as the iteration of the simulation changes.

Description of Changes:

Following our project proposal, we have made a few changes. Firstly, we implemented the changes suggested in our TA feedback and changed our title to be more reflective of approaches used in our investigation, and in our

computations we compared pre-COVID comments on political subreddits with comments posted on said subreddits during COVID. Additionally, we decided not to use the Twitter API[10] anymore but rather used PRAW (the Python Reddit API Wrapper)[11] to get comments off of Reddit directly. Instead of using NLTK[9] to perform sentiment analysis, we created our own Naive Bayes regression and classification model, and we added a simulation to better understand and visualize how people's political opinions changed over the progression of the Coronavirus pandemic.

Discussion Session:

For the purposes of our investigation, we define the pre-COVID era to be from July 13th 2019 through January 1st, 2021, as it's the lengthiest period of time before the pandemic in which we have a high enough resolution of data that allows us to draw better conclusions about the trends surrounding that period of time, while the duration of time following January 1st, 2021 to present day is defined as the current COVID era.. From our investigation, we discovered that for the pre-COVID era there is a very light upward trend (r = 0.080) in government sentiment over time. It's important to note that at this point, the trend line is always greater than 0 (our neutral sentiment rating), meaning most discussion around US government politics in our dataset is slightly positive.

However, once we enter the COVID era, **we see an immediate drop in comment sentiment over the span of a few days**, which could be credited to the consecutive implementations of lockdowns amidst the rising COVID cases, along with the exposure of government and judicial system corruption that motivated the BLM protests during this time. To be more specific, the trend line for this period of time always stays ¡ 0, which is a very distinctive fact about this period. During this period, our correlation value nearly switches signs, from +0.08 to -0.08, which **implies a light negative correlation between time after the pandemic has started and overall sentiment about comments towards the government**. However, it should also be noted that the rate of decrease of negative sentiment decreases over the progression of the pandemic, implying that, while people still hold negative sentiment towards the government, fewer people are developing a negative sentiment over time. This is likely due to the rise of acceptance of vaccination among the general public, as well as the increased transparency in communications with the government regarding the pandemic.

Throughout the investigation, we discovered many limitations. A major limitation of our sentiment analyzer was its inability to factor in the context of a comment. It viewed each comment as a "closed system". Similarly, it was unable to identify the use of things such as sarcasm and irony in the comments too. We believe that this was because we used a relatively simple model to classify comments; if we were to use an RNN (Recurrent Neural Network) instead, our model would have been able to take previous comments into consideration when analyzing a comment, allowing it to better determine its sentiment.

We had some difficulty extracting pre-covid era dataset information. In particular, the datasets we found were either too heavy for our needs (our hardware could not reasonably parse it) or did not contain the relevant data we needed to perform our analysis (timestamp, content body). We decided to write our own program to extract data from the reddit API directly, which allowed us to exercise more control over what data we collected. Unfortunately, this made us subject to heavy rate limiting, which limited the amount of data we could collect for this period. Although we are happy with the resolution of the data we did end up collecting, we would like to spend more time collecting additional data as next steps.

During our simulation, we see a nice visual representation of how ideas spread in a society. Often, after initial impressions are made, it is rare for opinions to change. In particular, we see strong and quick change on early stages of the simulation, when in later stages we see the simulation calm down. This creates a polarization effect, which represents quite well what we see in real life (from personal experience, political parties can often divide the internet hard and fast- people are stubborn about political views). In conclusion, our simulation shows how important initial impressions are, which explains why the weak initial response to the COVID-19 pandemic created such a wave of negativity towards governments that we see in the covid era trendline.

For an extension for this project, we believe that using an RNN (Recurrent Neural Network) to analyze the sentiment of comments would lead to a greater accuracy in their classification, and would provide better insights into the change of the sentiment of the general public towards the government.

References

[1] B. Howell, "The countries who've handled covid-19 the best and worst," MoveHub, 04-Oct-2021. [Online]. Available: https://www.movehub.com/blog/best-and-worst-covid-responses/. [Accessed: 05-Nov-2021].

[2] M. Stanger, "People are actually more honest online than in person," Business Insider, 28-Dec-2012. [Online]. Available: https://www.businessinsider.com/people-are-more-honest-online-2012-12. [Accessed: 06-Nov-2021].

[3] C. Yan, M. Law, S. Nguyen, J. Cheung, and J. Kong, "Comparing public sentiment toward covid-19 vaccines across Canadian cities: Analysis of comments on Reddit," Journal of Medical Internet Research, vol. 23, no. 9, 2021.

[4] V. Basile, F. Cauteruccio, and G. Terracina, "How dramatic events can affect emotionality in social posting: The impact of covid-19 on reddit," Future Internet, vol. 13, no. 2, p. 29, 2021.

[5] "The Reddit COVID dataset", Kaggle.com, 2021. [Online]. Available: https://www.kaggle.com/pavellexyr/the-reddit-covid-dataset. [Accessed: 06- Nov- 2021].

[6] A. Go, R. Bhayani, and L. Huang, "For academics - sentiment140 - a Twitter sentiment analysis tool," Sentiment140, 01-Jan-2009. [Online]. Available: http://help.sentiment140.com/for-students. [Accessed: 14-Dec-2021].

[7] "pandas documentation — pandas 1.3.4 documentation", Pandas.pydata.org, 2021. [Online]. Available: https://pandas.pydata.org/docs/. [Accessed: 06- Nov- 2021].

[8] "Overview — NumPy v1.21 Manual", Numpy.org, 2021. [Online]. Available: https://numpy.org/doc/1.21/. [Accessed: 06- Nov- 2021].

[9] "NLTK :: Natural Language Toolkit", Nltk.org, 2021. [Online]. Available: https://www.nltk.org. [Accessed: 06- Nov- 2021].

[10] "Twitter API Documentation", Developer.twitter.com, 2021. [Online]. Available: https://developer.twitter.com/en/docs/twitter-api. [Accessed: 04- Nov- 2021].

[11] "The python reddit api wrapper", PRAW. [Online]. Available: https://praw.readthedocs.io/en/stable/. [Accessed: 06- Dec- 2021].

[12] Plotlygraphs, Plotly Express, 03-Jul-2019. [Online]. Available: https://plotly.com/python/plotly-express/. [Accessed: 09- Dec- 2021].