Sentiment analysis in coronavirus related tweets

Ivan Jutamulia
MIT Course 6-3
Undergraduate, 4th Year
ivanj@mit.edu

Theo Sechopoulos
MIT Course 6-3
Undergraduate, 4th Year
theosech@mit.edu

6.864 Final Project - May 7, 2020

Abstract

The coronavirus pandemic and the government responses to it have been met with a wide range of reactions from the general public. We attempt to capture the polarizing perceptions on the pandemic from the general population by analyzing Twitter tweets and trying to find patterns. In this paper, we discuss our exploration of using different natural language processing models to train a sentiment classifier for tweets. We then leverage the models to look for interesting insights as they pertain to coronavirus related tweet sentiment.

Contents

I. Introduction

In the past couple months, the coronavirus pandemic has taken the world by storm, forcing countries and our state governments to take unprecedented actions. The responses of officials have been extreme relative to our recent history, and thus unsurprisingly have been met with extreme reactions from the general public. The wide range of reactions is best described as a spectrum, where one end consists of people who fully agree with the government's actions and are taking the pandemic seriously, and the other end people who don't think the virus is that serious and are treating the situation more as a joke.

We believe these sentiments are best captured on social media, as it gives a good representation of the perceptions that the general population seems to have. In particular, Twitter is one the most popular and accessible social media platforms for people to make posts. To briefly summarize Twitter, users can post "tweets" which other users will be able to see. Tweets can come in the form of retweets, in which a user echos what another user has tweeted, or replies, in which a user responds to another tweet.

We want to better understand how the world is responding to the Covid-19 crisis, and we think the best way to do that is to analyze tweets on Twitter. There are millions of tweets just in the past couple months that related to the coronavirus, so being able to draw insight about the public's sentiment should be an achievable task in our mind.

II. Overview

In this work, we seek to leverage Twitter data to perform sentiment analysis in coronavirus related tweets. In this case, the sentiment that we concern ourselves with is "serious" versus "joking". "Serious" tweets are ones that echo the idea that the virus is to be taken seriously, or just reflect facts of the pandemic itself. Contrast that with "joking" tweets, which include any tweets that poke fun at or make light of the situation.

We formulate this as a text classification task, in which a natural language processing machine learning model is needed to perform binary sentiment classification. There is plenty of NLP literature surrounding techniques to use when doing sentiment analysis of text, and we explore a couple different models in this work.

Our work primarily consists of three main components.

The first is how we ingest and preprocess the data in order to prepare it for our models. One of the most difficult aspects of this data preprocessing is the fact that all of our tweet data is unlabeled, meaning that none of the sentiments are given initially. To get around this, we use the semi-supervised approach of bootstrapping our training dataset, which will be discussed in more detail in section III.

The second component is actually building and training models to classify tweets according to their sentiment. We try various models and representations of the text, describing the model architectures in section IV and their performances in section V.

Lastly, we use our trained models to detect the sentiments of a larger set of tweets. We then analyze how different factors affect the sentiment and what kinds of patterns may exist with regards to the sentiment. We report our findings in section VI.

III. Data Preprocessing

The data we use is a collection of coronavirus related tweet IDs provided by Rabindra Lamsal on IEEE Data Port [1]. The tweets are filtered such that they contain keywords including "corona", "coronavirus", "covid", "covid19", and variants of "sarscov2". We limited our analysis to the period between March 19 and April 18, in which the entire dataset consisted of approximately 17 million tweets.

A. *Ingestion*

The raw data from this dataset in the form of tweet IDs as specified in Twitter's framework. The first step of preprocessing the data was converting the tweet IDs into their corresponding content as text. To do this, we utilized an application called Hydrator [2] to process the tweet IDs. This tool "hydrates" the tweet IDs, meaning that it leverages Twitter's API to extract the metadata and content of tweets and saves the results in a CSV file.

For a variety of computational reasons, we were not able to hydrate all 17 million tweets and use them effectively. The Twitter API has a limit on its usage rate, so running the hydration process on that many tweets would have taken weeks to run continuously. Furthermore, based on our estimates, the CSV file containing 17 million tweets and their metadata would require about 10 GB of storage, introducing a whole basket of storage and computational issues. Since we didn't have the resources to operate on that much data, we randomly sampled approximately 150,000 tweets from each day, and just hydrated those as a fairly good representation of the entire dataset. Thus, in total we hydrate a total of about 4.5 million tweets, which results in a 2 GB CSV file, much more manageable from our standpoint.

B. *Cleaning and Filtering*

After examining some of the tweets in our dataset, we found that it would be necessary to clean and filter through some of the tweets that we didn't want to confuse our models with.

We saw that a lot of the tweets were actually retweets of other users' posts. This resulted in there being a lot of duplicate entries in our dataset because the text in both cases for a retweet exactly match. It was easy to pinpoint these because retweets always have the text `RT` at the beginning of the tweet content. This was one of our filtering criteria, and we remove any tweets that begin with the token `RT`.

Something else that we noticed was that a lot of the tweets didn't have their full context self-contained in their own text. For example, many tweets were in fact responses to other users' tweets, and just by looking at the response tweet alone doesn't provide the context from the original tweet. These replies are characterized by the first character of the tweet being a `@` token, indicating that the tweet is in response to another user. Another example of not self-contained tweets is the containment of URL references in the tweet. Many times users will link to another source within the tweet, which is content that our models will not have access to. These tweets are also easy to find because they are characterized by containing the substring `http`. Because we wanted all the tweets to have their context self-contained, we filtered out these two kinds of tweets as well.

We found that these "bad" tweets actually made up a significant portion of the data. In particular, original tweets without any out-of-context references account for only about 7% of the total tweets we hydrated. However, given our large number of hydrated tweets, this wasn't a problem, as we were still able to get nearly 300,000 good, filtered tweets to use for our models.

C. *Bootstrapping*

Before we can train any kind of sentiment classifier, we have to address the major issue of not having any labeled data. While we have 300,000 tweets to use as training data, none of them are labeled with their sentiment as we had described earlier. Of course it is entirely infeasible for us to manually label all 300,000 tweets, so we had to come up with a clever way to minimize our manual work and still get accurate labels.

In order to fully label all 300,000 of these tweets in a reliable and methodical way, we used the semi-supervised approach of *bootstrapping* our training dataset. Our method of doing so is relatively simple and intuitive. The idea is to start with a small set of manually labeled data, and iteratively build up a larger and larger labeled dataset. At each iteration, we simply fit a classifier on our existing labeled data, make predictions on the unlabeled examples, and add a subset of those predictions to be "ground truth" labels for the next iteration. By the end, all the data should be labeled.

A detailed explanation of the exact bootstrapping procedure we utilized is as follows:

1) Manually label 2000 randomly selected examples: 1 for "serious", 0 for "joking".
2) Split labeled examples into 1000 training examples and 1000 validation examples.
3) Train classifier on set of training examples S.
4) Use classifier to make predictions for unlabeled data, add k most confident predictions as ground truth labels to S.
5) Repeat (3) and (4) until entire dataset is labeled.

We wanted k to be an exponentially increasing function of the iteration step i so that at each iteration we would be labeling a good portion of the unlabeled data without being too overconfident. We set k to be:

$$k = (i + 4)^4$$

to ensure this balance of quick convergence and maintaining high confidence in our label predictions. To fully label our dataset using the bootstrapping procedure required a total of 14 iterations.

Importantly, when we developed and tested our NLP models, we made sure that we were using the exact same model as the classifier to run this bootstrapping procedure, before evaluating on the 1000 validation examples. This was crucial because to accurately evaluate the model's performance we needed to make sure the labeling process followed the same method. Thus for each of the types of models that we describe, the training dataset is bootstrapped using the corresponding model.

IV. Models

In deciding on the appropriate sentiment classification model we considered the following attributes:

- Ability to train from relatively few examples: Due to our semi-supervised approach the model will uses must be able to learn from just 1000 samples on the first iteration.
- Training Speed: Our bootstrapping approach requires training a total of 14 different models and so we require sentiment classification models that can be trained quickly.
- Probabilistic Interpretation of output: We label joking tweets as negative and serious tweets as positive. This is a simplification on our parts and one can imagine that there is more of a spectrum between joking and serious rather than just two binary classes. Therefore, an ideal model would output a probabilities of belonging in each class allowing us to make more fine-grained judgement regarding the sentiment of a particular tweet which we can use later on in our analysis.
-

Sentiment classification models can be split into two main categories: Bag of Words models and Word Embedding Models. Bag of Words models represent documents by a vector of counts of the words they contain and thus only preserve the count information for each word, while word embedding models represent each word by its own vector which can either be learned or pre-trained. While bag of words models are rather simple and more restricting than word embedding models they are easier to train and can still yield good results. For the models that predict numbers between 0 and 1 we classify according to a 0.5 threshold (as we are just as interested in true positives and true negatives) but store the exact value and use it as a measure of seriousness in our analysis.

A. *Bag of Words Models*

Linear classification models tend to be faster, generalize better and more commonly have probabilistic interpretations compared to non-linear classification models [6]. These are all properties we desire for our domain and for this reason we choose to focus on this subclass of sentiment classifiers.

1) Ridge Classifier: This model converts the classification labels to -1 and 1 for negative and positive and then treats the problem as a regression task. The ridge regression model controls over-fitting by keeping the magnitude of the parameter vector small. This is especially important for our application given that we initially only train on 1000 tweets on the first iteration and have to learn weight values for each unique word which are in the order of hundreds of thousands. However one drawback of this model is that its output has no probabilistic interpretation and so we can only say whether a tweet is serious or joking but nothing in between.
2) Logistic Regression: Includes a lot of the benefits of a ridge classifier as we include a regularization parameter to control overfitting. Also, unlike the ridge classifier its output has a probabilistic interpretation which we can use to enrich our analysis.
3) Multinomial Naive Bayes: The easiest and fastest to train as it only requires one pass through the data. It models the joint probability distribution over features and labels and outputs the posterior of each label given a tweet's features (words). Its output has a really intuitive interpretation although it makes the naive Bayes assumption that the features

(words) are conditionally independent given the class label (joking or serious).

B. *Word Embedding Models*

In addition to the simpler linear models, we also experimented with feedforward neural networks to see if they could capture some more complex relationships between words in the tweets. For the neural neural networks, we decided that it would help the model if we provided word embeddings as input rather than the bag of words representations. We tried two different embedding schemes.

1) We first tried using the 1000 manually labeled examples to train our own word embeddings, and pass those into the training process. While it would have been ideal to retrain a new word embedding scheme at each iteration of the bootstrapping procedure, we found that it was far too computationally expensive to do so.
2) We also tried using a pre-trained word embedding scheme, namely the `glove-twitter-200` word embedding provided by Stanford NLP researchers [4]. This embedding was trained on 2 billion tweets, 27 billion tokens, resulting in a 1.2 million sized vocabulary, so we believed it would improve our accuracy.

Both models had relatively simple architectures, consisting of two hidden layers with 128 units each. We included dropout layers with dropout probability 0.5 to prevent overfitting as well. The hidden layer activation functions were ReLU units, and the output activation was a sigmoid to give us the probabilistic interpretation of the predicted sentiment that we were looking for.

V. Results

For each of the models above, we ran the bootstrapping procedure to generate training data using that specific model, and evaluated their performance on our held out validation set of 1000 labeled examples.

Our evaluation metrics are accuracy and ROC-AUC. We define accuracy as:

$$\frac{\text{\# correct predictions}}{\text{\# total predictions}}$$

where the predictions are the binary classifications 0 and 1. The ROC-AUC is a standard metric for evaluating binary classification tasks where the predicted values are continuous between 0 and 1. It is an overall measure of accuracy for varying thresholds of classification, and is suitable for this application.

The following table shows the performance of each model with respect to each of these metrics when evaluated on the held out validation set of 1000 labeled examples.

Model	**Accuracy**	**ROC-AUC**
NN	0.518	0.543
NN_pretrained	0.556	0.592
LR	0.662	0.711
RR	0.749	-
MNB	0.721	0.788

The linear bag of words models significantly outperform the neural network word embedding models which does slightly better than chance. This aligns with our expectations that these models would overfit and not generalize well from an initial seed of just 1000 labeled examples.

Of the linear bag of words models Ridge Regression and Multinomial Naive Bayes performed the best. However since the probabilistic output of Naive Bayes lends itself more easily to analysis we choose the predictions of this model for the following sections.

We qualitatively evaluate the learned model by examining the likelihood ratio (based on the likelihood estimates after training the Naive Bayes model) of individual features (words) and check whether they match with our intuitions. Displayed below are the highest and lowest likelihood ratio features (where a low likelihood ratio is evidence in favor of a serious tweet and a high likelihood ratio is evidence in favor of a joking tweet).

Word	Likelihood ratio (log-scale)
ima	4.74
bday	4.40
tattoo	4.31
dawg	4.25
gtf	4.17
needa	4.08
homies	4.08
⋮	⋮
coronavirusupdates	-4.45
maharashtra	-4.53
coronavid19	-4.56
indians	-4.74
washyourhands	-4.87
minister	-4.9
indiafightscorona	-5.03

VI. Tweet Analysis

We used our most performant model, the multinomial naive Bayes model, to then make predictions about the sentiment of a much larger dataset of tweets. In particular, using the same source of data [1], we randomly selected approximately 700,000 filtered tweets posted between March 19th and April 18th and made both continuous probabilistic predictions for their sentiment, as well as binary predictions for serious or joking. Using this data along with the tweet metadata, we were able to look at a few specific factors and how they relate to the sentiment of tweets.

A. *Average User Sentiment*

A significant aim of our investigation was to get a better sense of people's sentiment around Covid-19 is. To this aim, we use our trained sentiment classifier to predict the probability of a tweet being serious for each of the 700,000 randomly sampled tweets. We then average the predicted sentiment for each unique twitter user and plot that value as seen in seen in Figure 1.

The distribution of user sentiment is bi-modal with a really high right peak. Users are mostly split between those that tend to joke about the virus and those that tend to take it seriously with less users being in the middle of the spectrum. It is particularly interesting to note the sharp right peak corresponding to users who only tweet seriously about Covid-19 and whose tweets the model is very confident about being serious.

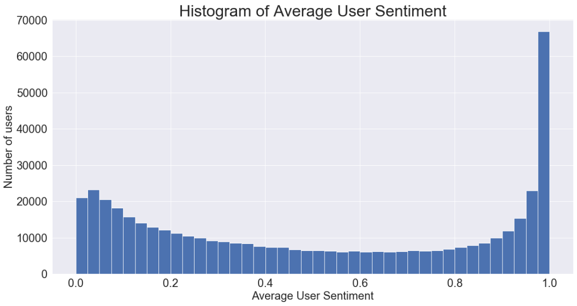

Fig. 1. Histogram of Average User Sentiment

B. *Time*

A very simple analysis was looking at how the average sentiment regarding the coronavirus evolved over time. We looked at the average sentiment from all the data on a day by day basis, which gave the plot seen in Figure 2.

Based on this, there doesn't seem to be any definitive increasing or decreasing trends in terms of the average sentiment. However, what we do see is spikes in the seriousness on numerous days. A possible explanation for this may be that these spikes line up with major news updates regarding the pandemic, and thus for a short period of time afterwards people tended to be more serious. Overall though, it seems as though the average sentiment remained relatively stagnant over time.

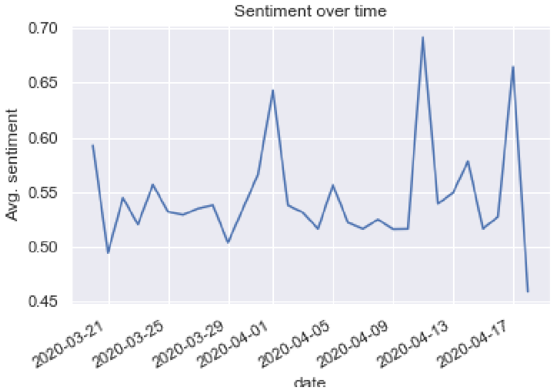

Fig. 2. Sentiment over time.

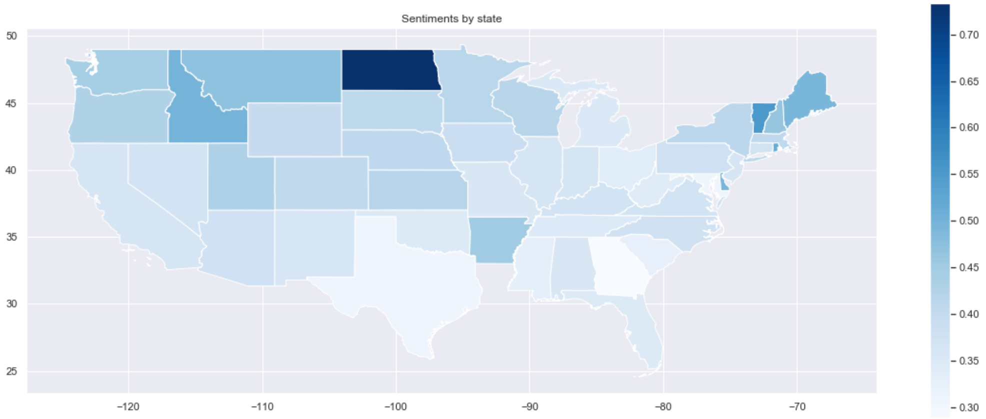

Fig. 3. Sentiment by state.

C. *Location*

While countries as a whole have been suffering from the coronavirus, the US is unique in that much of the government response has been conducted on a state by state basis. This has resulted in a lot of discrepancies between how different state governments have responded, and consequently how their constituents have reacted. We were able to leverage our sentiment analysis tool to break down the average sentiment on a state by state basis and measure how people in those states are reacting.

Figure 3 shows the average sentiment for every state. We immediately notice some geographical patters, as the pacific northwest region seems to be the most serious, and the southeast region seems to be the least serious. There are a variety of factors that could serve as an explanation for why this might be the case, including aspects such as culture, demographics, etc. What is interesting to note however, is that the less serious states also happen to be states that reopened earlier. In particular, Georgia has the least serious average sentiment, and was also the first state to start reopening their economy. Similarly, Texas was also not very serious in terms of sentiment, and they also were one of the first states to reopen. While we can't say anything definitive about causal relationships, it is at least interesting to observe these correlations and provide some insight as to how different states perceive the pandemic differently.

D. *Comparison with Cases and Deaths*

A factor that could explain some of the state to state differences regarding the sentiment towards Covid-19 is the difference in how much the virus has affected each state. To investigate this relationship we download daily counts of reported Covid-19 cases and deaths from the New York Times `covid-19-data` repository [5]. We then compute the average daily US sentiment based on all tweets from within the US on any given day and plot that against the daily new US Covid-19 cases.

No definite conclusion can be drawn, but based on Figure 4 we hypothesize that changes in average sentiment tend to follow changes in the number of daily new cases (and/or vice versa). It is important to emphasize that this is merely speculation and no causal inferences can be made. In addition, as we showed above, user sentiment as well as daily new case count varies a lot from state to state and so an aggregated plot for the whole country might not be the most illuminating plot.

With that in mind, we visualize the same plot but only for the state of New York instead of all of US as can be seen in Figure 5. One might argue that daily new death count is a better proxy for how widespread the virus is (as it is independent of varying testing capabilities) and so we also show a plot of daily new deaths overlaid with average user sentiment, this time for the state of Washington (see Figure 6).

Once again, there seems to be a correlation but

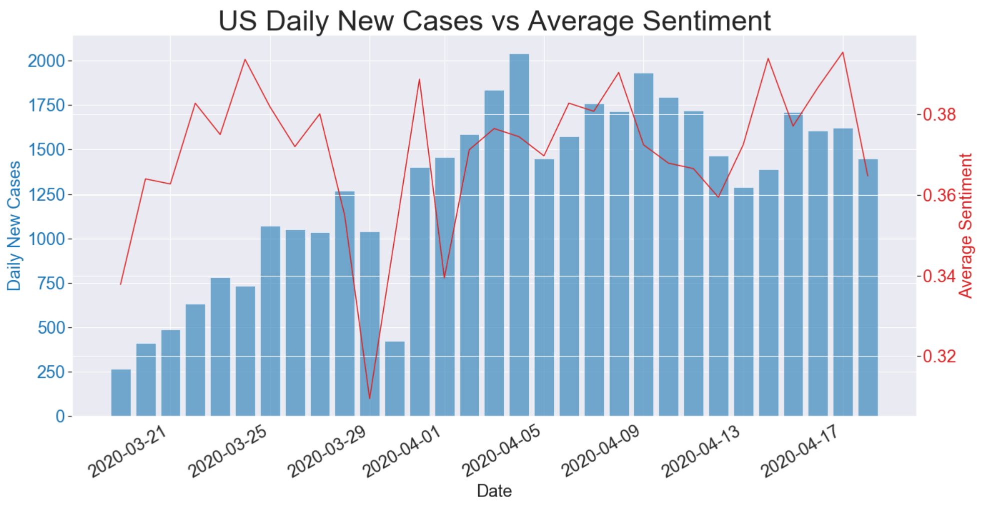

Fig. 4. Us Daily New Cases vs Average Sentiment

that is all we can really say. It is interesting to note that the causality could go both ways, i.e. that both the reported numbers could affect people's sentiment but also people's sentiment could affect how much the virus spreads and thus the numbers (assuming their sentiment is related to how much they comply with the social distancing measures). One could imagine a lot more ways in which these two quantities might interact and so our aim is primarily to merely present and not interpret the data.

VII. Conclusion

We were interested in understanding more about the sentiment surrounding the coronavirus pandemic by analyzing tweets related to it. We were able to devise a robust strategy for building up a training set to be used in training our NLP models, and were able to attain good results from those models. Our model did a good job distinguishing between serious and joke tweets, which allowed us to look for patterns and factors that may affect the people's sentiment towards Covid-19.

We believe that we have only scratched the surface in terms of what can arise from this kind of research. While our models worked pretty well, we did not get the opportunity to try more advanced classification models which might have been more performant. In addition, with more data, we could break down the sentiment analysis into more factors such as age, political affiliation, income level etc. These demographic factors would definitely give us some better insight as to how generally people are reacting to the pandemic.

Overall, we showed that the structural procedures of bootstrapping and doing sentiment analysis with tweets works quite well even with simple models, and provides a lot of promise for future work in the domain.

Acknowledgments

We would first like to thank the professors of 6.864 this semester, Professor Jacob Andreas and Professor Jim Glass, for their instruction and understanding during this unconventional semester. We would also like to thank all the TAs for their dedication to the course and guidance on the material.

References

[1] Rabindra Lamsal, "Corona Virus (COVID-19) Tweets Dataset", IEEE Dataport, 2020. [Online]. Available: http://dx.doi.org/10.21227/781w-ef42. Accessed: May. 05, 2020.

[2] Documenting the Now. (2020). Hydrator [Computer Software]. Retrieved from https://github.com/docnow/hydrator.

[3] Andrew Y. Ng, Michael I. Jordan. (2002). On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. NIPS2001_2020.

[4] Stanford NLP, GloVe, (2020), GitHub repository, https://github.com/stanfordnlp/GloVe.

[5] New York Times. covid-19-data. (2020). GitHub repository, https://github.com/nytimes/covid-19-data/

[6] Eisenstein. Natural Language Processing (2018). GitHub repository, https://github.com/jacobeisenstein/gt-nlp-class

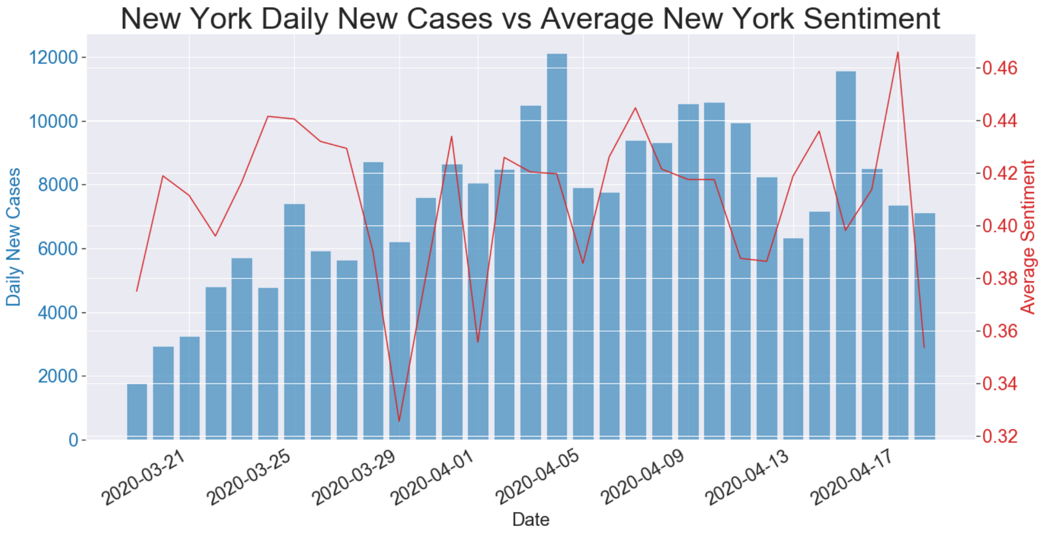

Fig. 5. New York Daily New Cases vs Average New York Sentiment

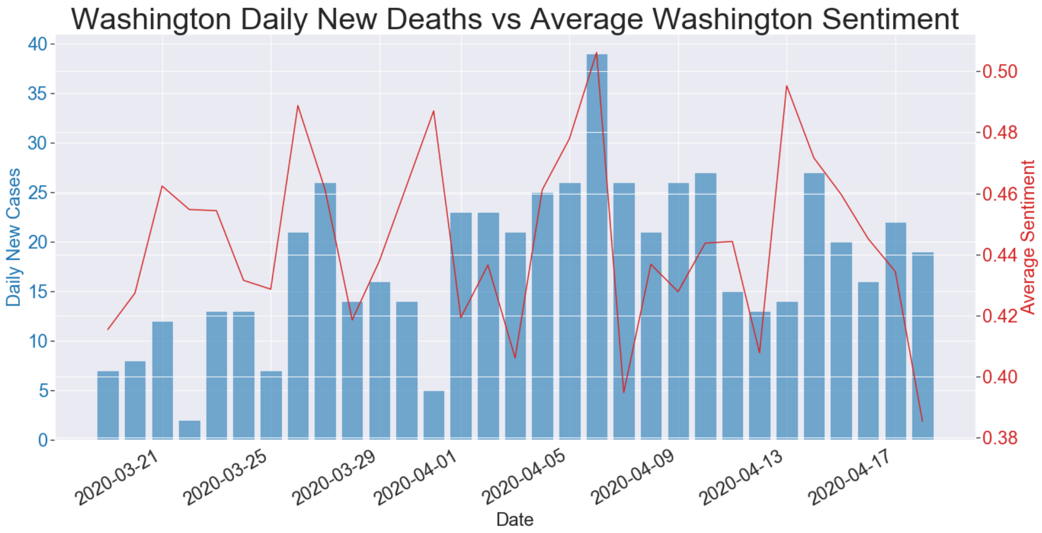

Fig. 6. Washington Daily New Deaths vs Average Washington Sentiment