Classifiers for Politically Charged and Normally Distributed Reddit Comments

Author
Elizabeth Anne Fitzgerald

March 9, 2020

Abstract

The research presented here began as a simple investigation of the sentiment of two politically charged subreddits, r/politics and r/The_Donald. This research models the sentiment of both subreddits as normal distributions. It is clear that r/The_Donald is the more negatively charged subreddit.From there, I move on to create four different classifiers to categorize comments as being from r/The_Donald or r/politics. The first is based solely on the distributions' normal curves. The second is based on the Federalist Papers example. The third is a combination of the previous two. Finally, the fourth is a Naive Bayes classifier. Each classifier had varying results.

1 Introduction

Youtube video at this link! https://youtu.be/sR6KUXf7VBY.

I grew up in a household that emphasized understanding a situation in full before making decisions. Although this applied to every facet of life at home, it was no more pertinent than during the 2016 US presidential election. I ended up becoming a true moderate because I felt I lacked the time to really understand what was happening politically (for better or worse). I maintain my neutrality to this day. However, this was not how most people reacted to the election. Most people became incredibly polarized and – especially in online communities – hostile towards those of differing opinion.

As I saw this happening, I was generally frustrated with it, and wondered: which side of the political spectrum was more vicious? That is the question that prompted this research, so it's first question I answer. To start, I collected Reddit comments from r/politics (liberal) and r/The_Donald (conservative). I took these comments, cleaned them, then calculated the sentiment of each using previous Stanford research completed by William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky (1). With these sentiment scores, I observed that the data for both subreddits could be well described by a normal curve.

From there, I proceeded to create classifiers. The first one, referred to as the Normal Classifier, uses the distributions of the two subreddits to figure out, given a new comment's calculated sentiment, which subreddit is more likely. The second classifier (the

Term Frequency Classifier) is based on the Federalist Papers example we reviewed in class. A mapping of words to probabilities is generated for both subreddits, and then based on that, the sum of log probabilities is calculated for both. The more negative sum is the one that receives the classification. The third classifier, the Combo Classifier, combines the two previous. Both classifiers make their guess, then the one with a greater confidence (based on the recall of the two models, and the confidence of the individual predictions) receives the classification. The fourth and final classifier is a Naive Bayes Classifier, and it follows the set-up of the spam email classifier in lecture 24. All four classifiers failed to perform as well as one would hope, although some worked well in particular categories.

2 Methodology

In reference to my code, all of it can be run from the MainCS109.py file. Simply answer the questions to run the appropriate parts of the program! In order to run the code, you will need to import the following packages: praw, pandas, pickle, numpy, requests, time, string, re, nltk, matplotlib, statistics, and math.

2.1 Scraping Data

In order to run my code from the start, with the web scraping, it's necessary to have a Reddit account with permission to scrape from the site. For privacy reasons, I'd like to not share my personal info, but it involves going to Reddit's app preferences and creating an app. With the right credentials (that my code will prompt you for), you can run the web scraping portion. However, if you prefer not to, the data I have scraped will be provided, and you can skip that part. The training data scraped is taken from 2016, 2017, 2018, and 2019, with test data coming from January, 2020. Comments are randomly selected using stratified random sampling, where the posts are groups. A post is randomly selected from a given time frame, and then all the comments from that group are selected.

2.2 Preprocessing

After data has been collected, the program will then ask if you'd like to preprocess the data. If you select yes, it will do the following for each comment collected:

1. Junk characters are removed (meaning non-alphabetic characters)
2. Stop words from the nltk corpus are removed
3. Remaining words are lemmatized (stemming proved to be too aggressive an approach, and lemmitazation maintained greater meaning in m the processed comments).

Again, if you would prefer to skip this step for efficient grading, the results of my preprocessing will be provided.

2.3 Calculating Sentiment and Normal Distributions

Next, I begin to examine the sentiment of the comment, which is where I relied on previous research (1). The SocialSent project created sentiment lexicons for specific subreddits, including r/politics and r/TheRedPill. Although r/TheRedPill is a separate subreddit from r/The_Donald, a quick review of the vocabulary used in both subreddits reveals that the lexicon is nearly identical and can be used without worry. Using these lexicons, the sentiments of comments from both subreddits were calculated. This calculation was done in the following manner for a single comment.

1. The appropriate lexicon was selected. This lexicon provided the sentiments of individual words in comments, as used in steps 2 and 3.
2. The number of words in the comment with positive sentiment was calculated, along with the sum of all the positive sentiment values (num_pos and sum_pos).
3. The number of words in the comment with negative sentiment was calculated, along with the sum of all the negative sentiment values (num_neg and sum_neg).
4. The percentages of positive and negative words were calculated (per_pos and per_neg).
5. The positive percentage was multiplied by the positive sum, and the negative percentage was multiplied by the negative sum ($per\_pos * sum\_pos = pos\_score$ and $per\_neg * sum\_neg = neg\_score$)
6. The positive score and negative score were summed to produce the overall sentiment of the comment ($pos\_score + neg\_score = sentiment$).

Once all the comments had gone through this process, they were shown graphically. They were plotted on histograms by year and overall for both subreddits. This can be seen in Figures 8 through 17. Based on these histograms, it was immediately apparent that the sentiments of both subreddits were normally distributed and could be modeled with normal random variables. After calculating the mean and variance for both the All Comments distributions, I found these two random variables, where D represents r/The_Donald and P represents r/politics:

$$D \sim \mathcal{N}(-.218,\, 1.112)\,.$$

$$P \sim \mathcal{N}(-.589,\, 2.00)\,.$$

2.4 Normal Classifiers

With these normal random variables, we can start discussing the classifiers, starting with the Normal Classifier. The Normal Classifier uses the PDFs of D and P to make a judgement about how much more likely a particular sentiment value is to fall on either distribution.

The PDF for D is: $f(x) = \frac{1}{1.056\sqrt{2\pi}} e^{-.5(\frac{x-(-.218)}{1.056)^2})^2}$
Similarly, the PDF for P is: $g(x) = \frac{1}{1.414\sqrt{2\pi}} e^{-.5(\frac{x-(-.589)}{1.414)^2})^2}$
When a new comment is fed into the classifier, its sentiment is calculated using both the r/politics and the r/TheRedPill lexicon, resulting in $p\_sentiment$ and $d\_sentiment$. These values are plugged into their respective PDFs to produce ratios. Mathematically, it looks like:

$$d\_ratio\_1 = \frac{f(d\_sentiment)}{g(d\_sentiment)}$$

$$d\_ratio\_2 = \frac{g(d\_sentiment)}{f(d\_sentiment)}$$

If $d\_ratio > d\_ratio\_2$, $difference = d\_ratio\_1 - d\_ratio\_2$ and $classification\_1 =$ donald otherwise $d\_difference = d\_ratio\_2 - d\_ratio\_1$ and $classification\_1 =$ politics

$$p\_ratio\_1 = \frac{f(p\_sentiment)}{g(p\_sentiment)}$$

$$p\_ratio\_2 = \frac{g(p\_sentiment)}{f(p\_sentiment)}$$

If $p\_ratio > p\_ratio_2$, $difference = p\_ratio\_1 - p\_ratio\_2$ and $classification\_2$ = donald otherwise $p\_difference = p\_ratio\_2 - p\_ratio_1$ and $classification\_2$ = politics

If $classification_1$ and $classification_2$ match, then that is the classification. Otherwise, if $d\_difference > p\_difference$, then the classification is r/The_Donald. Apart from that, it's r/politics.

2.5 Term Frequency Classifier

The next classifier I made was the Term Frequency Classifier. This classifier was based on the Federalists Papers example from lecture.

First, we make word-to-probability maps for both r/The_Donald and r/politics. Then, for a new comment, we find the sum of the log probabilities of its individual n words. Mathematically, if $f(word_n)$ returns the probability of the word being used in r/The_Donald and $g(word_n)$ returns the probability of the word being used in r/politics, it works like:

$$d\_sum = \sum_{n=1}^{n} \log f(word_n)$$

$$p\_sum = \sum_{n=1}^{n} \log g(word_n)$$

If $d\_sum > p\_sum$, the classification is r/The_Donald. Otherwise, the classification is r/politics. The Term Frequency Classifier also returns the confidence with which it made that decision, or $d\_sum - p\_sum$, or vice versa.

2.6 Combo Classifier

The Combo Classifier (creatively named) is the combination of the Normal Classifier and the Term Frequency Classifier. It functions by first getting the predictions of the two individual classifiers, which we'll denote as $normal\_classification$ and $tf\_classification$, which can either be 1 (r/The_Donald) or -1 (r/politics). It also gets the confidence with which those two classifications were made, which we'll denote to be $normal\_confidence$ and $tf\_confidence$. Finally, it uses the recall scores of the other two classifiers, which I call $normal\_recall$ and tf_recall. With these numbers, we can proceed to calculate our scores:

$$normal\_score = normal\_classification * normal\_confidence * normal\_recall$$

$$tf\_score = tf\_classification * tf\_confidence * tf\_recall$$

$$ultimate\_score = normal\_score + tf\_score$$

If $ultimate\_score > 0$, then the classification is r/The_Donald. Otherwise, if $ultimate\_score < 0$ the classification is r/politics. If $ultimate\_score = 0$, then no judgement can be made and the classification is essentially neither.

2.7 Naive Bayes Classifier

The final, and most legitimate, classifier is the Naive Bayes Classifier. It functions similarly to the Term Frequency Classifier in that the training features are words, but it works quite differently.
The classifier starts by taking comments and making a dictionary of all the words seen in those comments. Then, for each word in the dictionary, it finds the numbers of r/The_Donald and r/politics comments that contain the word. These numbers are then used to find the following proportions:

$$\hat{P}(word|donald) = \frac{\text{\# of r/The\_Donald comments containing word} + 1}{\text{number of r/The\_Donald comments} + 2}$$

$$\hat{P}(word|politics) = \frac{\text{\# of r/politics comments containing word} + 1}{\text{number of r/politics comments} + 2}$$

$$\hat{P}(donald) = \frac{\text{\# of r/The\_Donald comments} + 1}{\text{number of comments} + 2}$$

$$\hat{P}(politics) = \frac{\text{\# of r/politics comments} + 1}{\text{number of comments} + 2}$$

This process results in mappings from words to probabilities for both subreddits as shown above. With these mappings, we can begin to classify individual comments. Given a comment, the first thing the classifier will do is split it into individual words. Then, the following calculations are performed:

$$d\_sum = \log \hat{P}(donald) + \sum_{n=1}^{n} \log \hat{P}(word_n|donald)$$

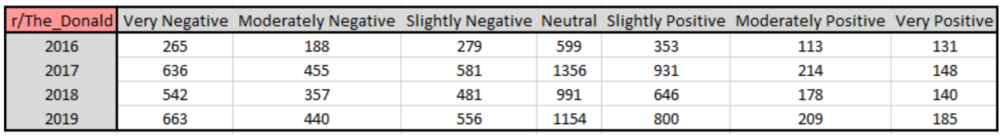
$$p\_sum = \log \hat{P}(politics) + \sum_{n=1}^{n} \log \hat{P}(word_n|politics)$$

If $d\_sum > p\_sum$, then the classification is r/The_Donald. Otherwise, the classification is r/politics.

3 Results

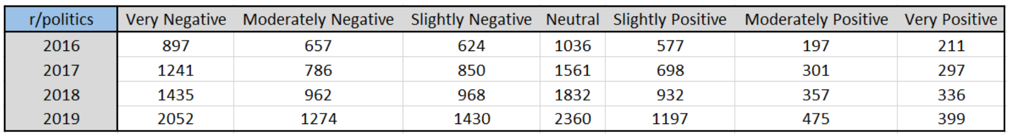
Quite a few results were gathered from this project! The graphical results are located at the end of the paper in Figures 1 through 10, but the numerical data is located here.

3.1 General Statistics Regarding Sentiment



r/The_Donald	Very Negative	Moderately Negative	Slightly Negative	Neutral	Slightly Positive	Moderately Positive	Very Positive
2016	265	188	279	599	353	113	131
2017	636	455	581	1356	931	214	148
2018	542	357	481	991	646	178	140
2019	663	440	556	1154	800	209	185

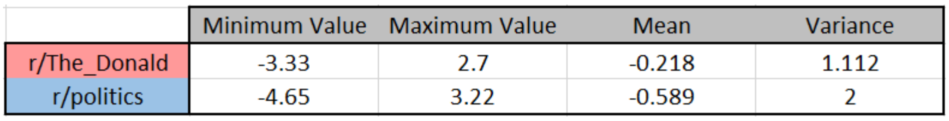
Figure 1: r/The_Donald Sentiment Distribution across Years and Categories



r/politics	Very Negative	Moderately Negative	Slightly Negative	Neutral	Slightly Positive	Moderately Positive	Very Positive
2016	897	657	624	1036	577	197	211
2017	1241	786	850	1561	698	301	297
2018	1435	962	968	1832	932	357	336
2019	2052	1274	1430	2360	1197	475	399

Figure 2: r/politics Sentiment Distribution across Years and Categories

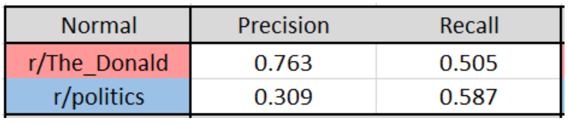
After removing outliers from both datasets (under the guidance of the Interquartile Rule), we find the following to be true:



	Minimum Value	Maximum Value	Mean	Variance
r/The_Donald	-3.33	2.7	-0.218	1.112
r/politics	-4.65	3.22	-0.589	2

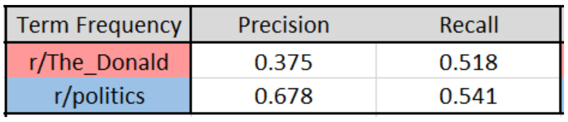
Figure 3: Overall Sentiment Statistics for both Subreddits

3.2 Classifier Results



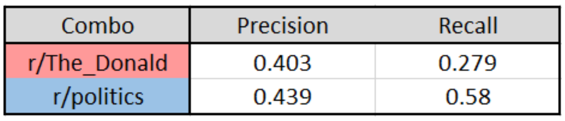
Normal	Precision	Recall
r/The_Donald	0.763	0.505
r/politics	0.309	0.587

Figure 4: Results of the Normal Classifier



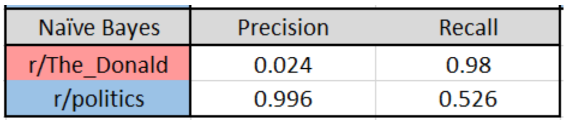
Term Frequency	Precision	Recall
r/The_Donald	0.375	0.518
r/politics	0.678	0.541

Figure 5: Results of the Term Frequency Classifier



Combo	Precision	Recall
r/The_Donald	0.403	0.279
r/politics	0.439	0.58

Figure 6: Results of the Combo Classifier



Naïve Bayes	Precision	Recall
r/The_Donald	0.024	0.98
r/politics	0.996	0.526

Figure 7: Results of the Naive Bayes Classifier

4 Discussion and Conclusions

This extra-credit project was a great learning experience, both as a statistician and a programmer. I ran into a lot of bugs along the way, as well as a lot of tough decisions about how to design these classifiers.

On the whole, I think with more time I could improve their results. I suspect a lot of the issues with this kind of categorization stem from the fact that there are not clear markers in the text between r/The_Donald and r/politics. The same vernacular is frequently used in both subreddits, which I think makes a classification problem difficult. Furthermore, the preprocessing could be improved to remove non-English words that result from the current preprocessing. That being said, I suspect that the largest issue came from the fact that the data were not balanced, or that there were significantly more r/politics comments than there were r/The_Donald comments. This caused significant issues with the Naive Bayes classifier, since the proportion of instances when a word was used was always bigger for r/politics, not because it was used more frequently, but because it was more times because there were three times as many comments. One solution for this might be to use oversampling in the r/The_Donald data set, and simply have enough repeated comments to increase the increase of comments to equal the number from r/politics. However, given that I discovered this issue twenty minutes before the deadline to submit this project, I did not have time to fix it.

All in all, I'm fairly pleased with how much I got to play with and design in this project, even if the results weren't perfect.

5 References

(1) *William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora. ArXiv preprint (arxiv:1606.02820). 2016.*

6 Figures

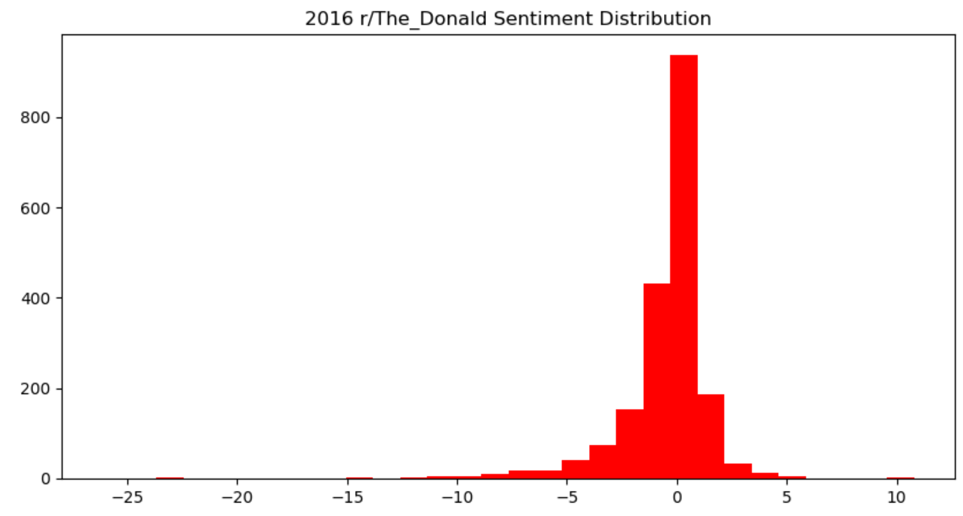


Figure 8: 2016 r/The_Donald Sentiment Distribution.

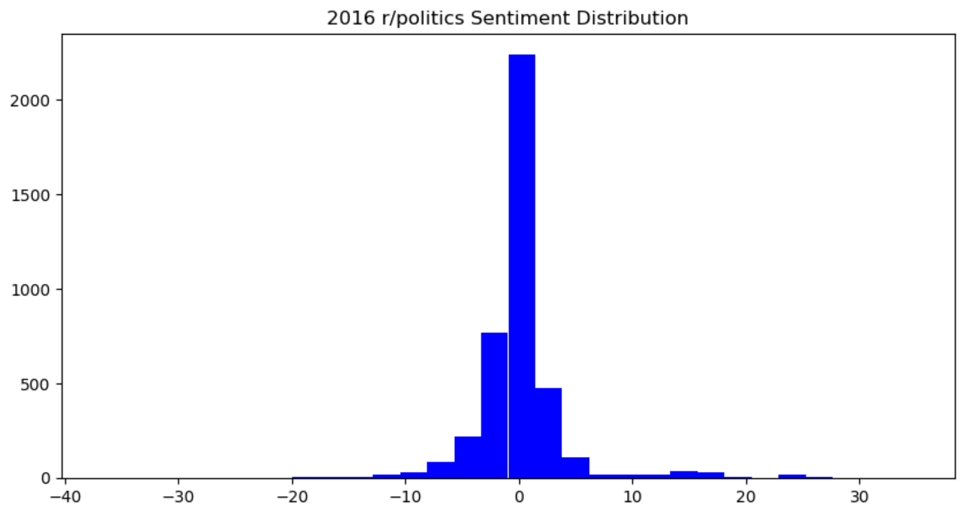


Figure 9: 2016 r/politics Sentiment Distribution.

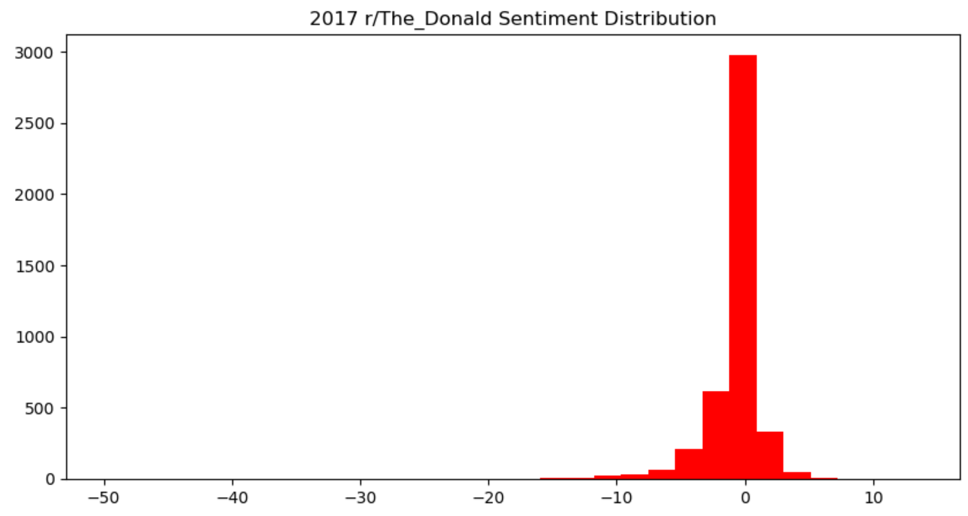


Figure 10: 2017 r/The_Donald Sentiment Distribution.

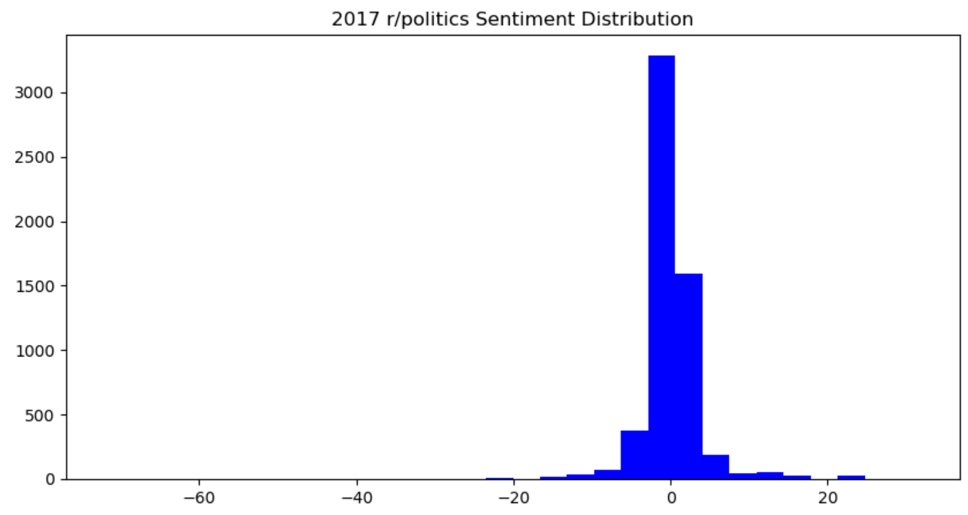


Figure 11: 2017 r/politics Sentiment Distribution.

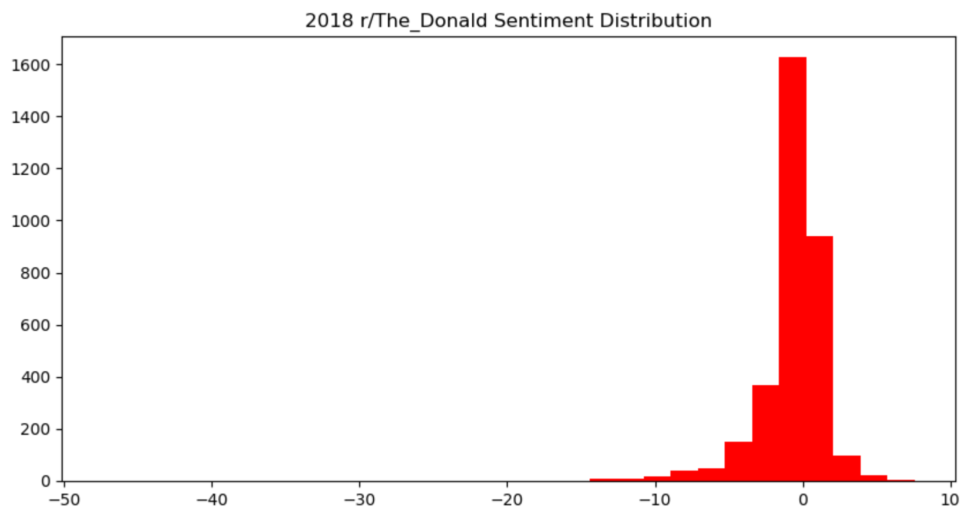


Figure 12: 2018 r/The_Donald Sentiment Distribution.

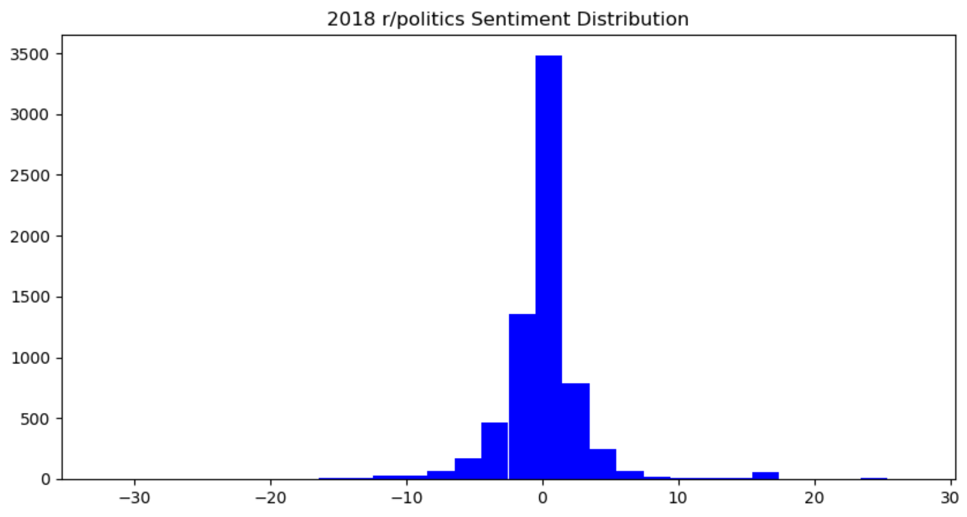


Figure 13: 2018 r/politics Sentiment Distribution.

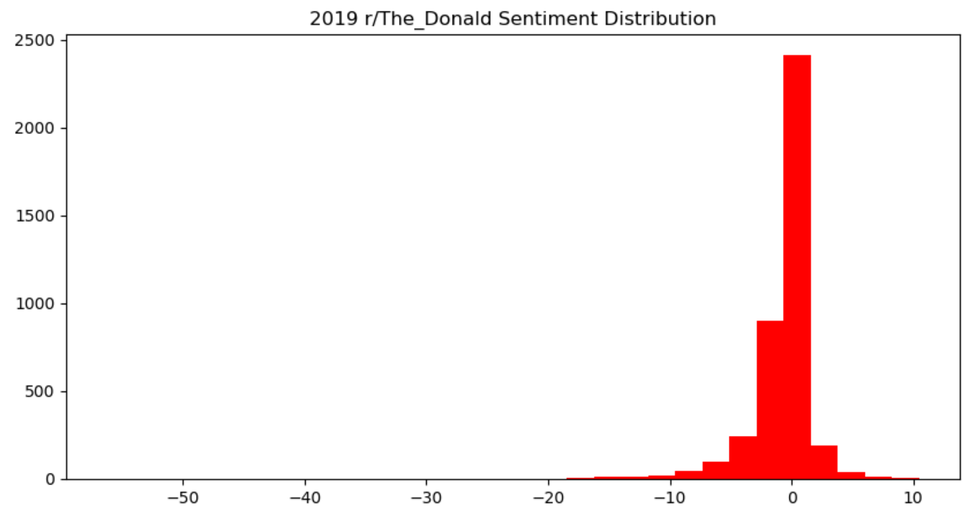


Figure 14: 2019 r/The_Donald Sentiment Distribution.

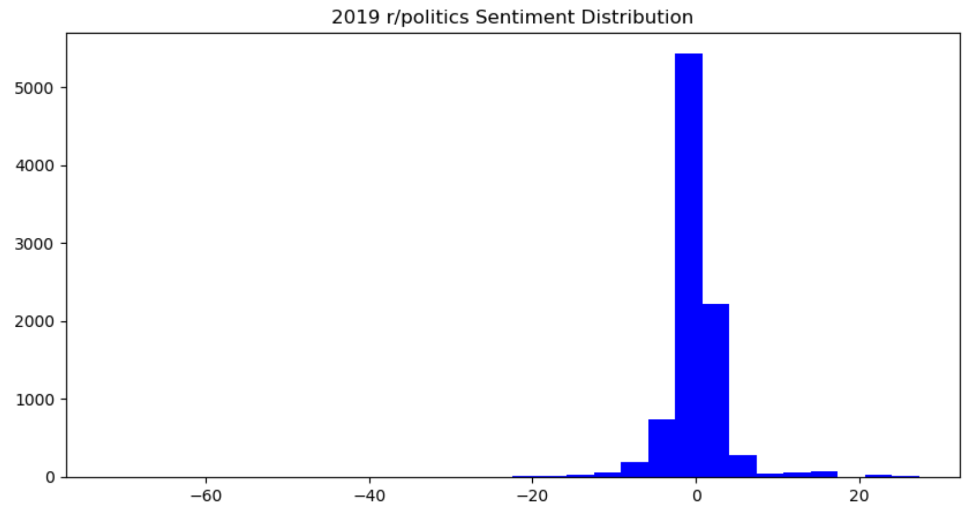


Figure 15: 2018 r/politics Sentiment Distribution.

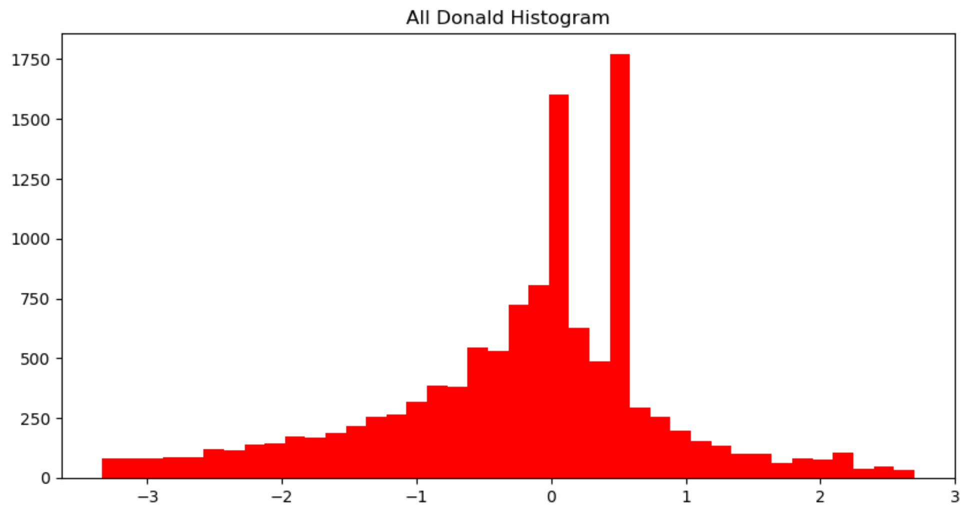


Figure 16: All r/The_Donald Sentiment Distribution.

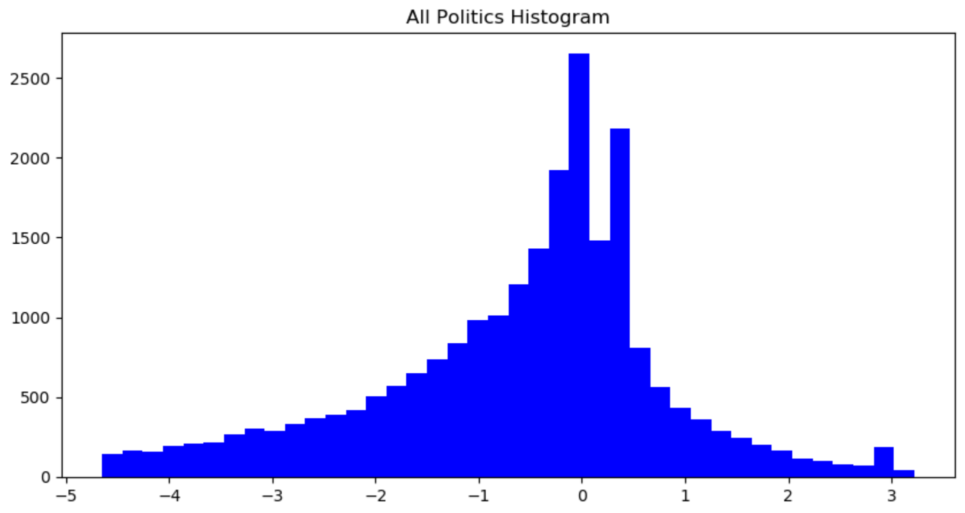


Figure 17: All r/politics Sentiment Distribution.