Understanding Public Perceptions of the ObamaCare Program: A Sentiment Analysis Approach Utilizing Open-Source Data

Nadir Khan, Bob Surridge, Rob Mccormick, Kayecee Palisoc
CAPP-30254 (Spring 2023) Machine Learning for Public Policy

1 Abstract

Public policy analysts often encounter classification tasks in which there is a scarcity of labeled data to train their models. In the realm of sentiment analysis, however, there exist resourceful approaches to leverage large and publicly available datasets to train models for this purpose. In this study, our objective is to ascertain the prevailing opinion climate concerning one of the most widely discussed government programs in recent years: the Patient Protection and Affordable Care Act (PPACA), commonly referred to as ObamaCare. To achieve this goal, we will develop a customized model using open-source data (i.e. Yelp), to analyze sentiments expressed in tweets related to the ObamaCare program.

2 Data Preprocessing

To train our model based on a sample of Yelp reviews, three different samples ranging in the number of reviews (10,000, 100,000, and 1,000,000) were created.

The text for each review was cleaned through eliminating format, punctuation, and additional whitespace. Any uppercase characters from the dataset were set to lowercase. English stopwords - extremely common words with no informational value - were also removed based on a list from nltk.corpus. It's important to note that the sample of Yelp reviews only contain English words, but this is not necessarily the case with the test Twitter dataset.

To extract testable features, we need to count each unique word's frequency for each review. What ends up being created is a sparse training matrix, where the i-th row represents the i-th review, and the j-th column represents the j-th unique vocabulary word. The j dimension is equal to the total number of unique words in the entire review dataset.

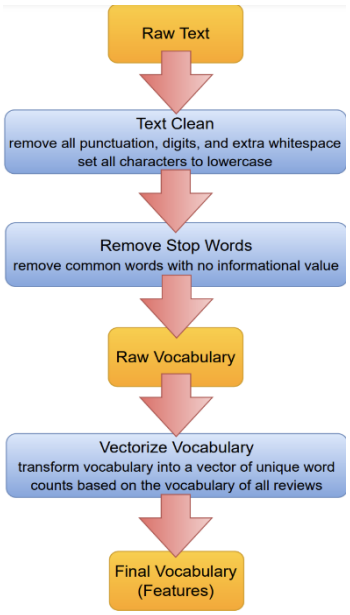


Figure 1. Data processing diagram

The (i, j)-entry represents the frequency of the j-th unique vocabulary word in the i-th review. All pre-processing steps were implemented in Python and can be located in the Jupyter notebook titled Yelp Review Sentiment Classification and Analysis. This whole process is shown in Figure 1.

These samples were then subsetted based on three different variation types for classification: star rating, standardized star count, and n-grams. In total, 18 variations of the Yelp dataset were constructed to be tested for optimal classification accuracy. These variations are outlined in detail in the next section.

3 Modeling

The data used in the modeling were divided into 3 classes. These are based on the star reviews and categorized as (1) All stars: 1-5 star reviews; (2) 1,3,5 stars: 1-negative, 3-neutral, 5-positive; (3) 1&5 stars: extremely negatives and extremely positives only. This choice of classes was a critical feature selection decision, based on the inference that 2- and 4-star may pose greater challenges in terms of interpretation and reduce the model's predictive accuracy, hence removing those data points in the succeeding classes. The initial models used were Multinomial Naïve Bayes, Random Forest, and Decision Trees, trained on 10,000 data points, with all_stars as baseline class. 80%-20% training-testing split was used across all models. For the baseline, 1_3_5_stars and 1_5_stars class, Multinomial Bayes performed best in terms of f1 scores. 1_3_5_stars has 6,663 data points while 1_5_stars has 5,524 data points. 1_5_stars combined are 55% of the data, noting that around 40% are 5. It makes sense to use f1 score to evaluate the models trained on this imbalanced data.

Baseline Class	Accuracy	F1 score	
		Macro avg	Weighted
Multinomial Naïve Bayes	**0.58**	**0.41**	**0.54**
Random Forest	0.53	0.34	0.45
Decision Tree	0.47	0.36	0.46

Table 1. Baseline Model Results

Given that most of the data is on the positive end, also noting that positive reviews typically have more words than negative reviews, the learning would be heavily biased towards positive reviews. Another feature engineering decision was to have some balanced training data where for all_stars, each star rating is 20% of the total count, for 1_3_5_star, each star rating is 33.3% of the total count, and each star rating for 1_5_star is 50% of the total. The minimum counts in the imbalanced data was used as a baseline count for all other star-rating sizes. For instance, for the 10,000 data points, the minimum rating count is 763. For the all-star balanced, there were 763 data for each star, summing up to 3,815. This reduced the total data points by 60%. This setup was applied to an even larger dataset, having 100,000 and 1 million data points in the end. Logistic Regression and Neural Network were added and planned to be trained on all data classes too. Due to computing power limitations, the other models were not validated. The complete training and validation process of all models was done only in the group where there were 100,000 balanced data. Accuracy score serves as the predominant evaluation metric due to the implementation of data balancing methods.

All the models mentioned above were using single words (unigram) as features. There were 76,896 possible unique words in the All-star class, 57,720 in 1_3_5_star class, and 45,501 in 1_5_star class. As an attempt to explore word associations, the contiguous sequence of two words (bigram) and sequence of 3 words (trigram) were extracted as well. The models were also trained with these features.

With all that, for the 100,000 data points, the models were trained on combinations of different classes (all_stars, 1_3_5 stars, 1_5_star), star-rating distribution (imbalanced, balanced), and word count as features (unigram, bigram, trigram).

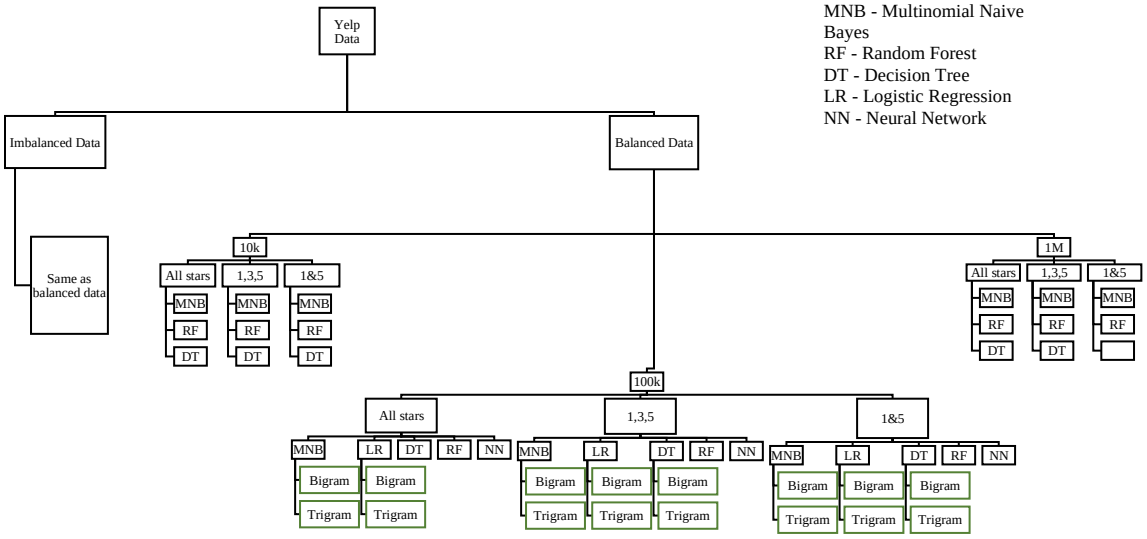


Figure 2. Process Map of all model, class, and feature combinations

It is worth noting here that there are significantly long runtimes with certain combinations, which is why certain models test fewer combinations. Specifically, for Neural Network, base model having hidden layer sizes = 100, activation = relu, solver = adam, L2 regularization = 0.0001, learning rate = 0.001 and max iterations/epochs = 200, tuning ended up at L2 regularization = 0.01, learning rate = 0.001 and number of hidden layers = 1, engaging in extensive hyperparameter optimization was not possible as some runs require about 4 hours or more to complete. This is the same case with trigram and some bigram datasets, as the number of unique word combinations dramatically increases with larger n-grams. This is also true with some all_stars hyperparameter variation, as the number of reviews increases dramatically. This is further elaborated on when discussing the Big O notation and overall runtime limitations. Consequently, while the potential for achieving improved accuracy existed, it was not feasible to pursue such hyperparameter tuning within the imposed temporal limitations.

RESULTS

Model	Hyperparameters	1&5 stars	1,3,5 stars	All stars
Logistic Regression	**L2; 100 iters**	**95.28**	**80.03**	**51.11**
Multinomial Naïve Bayes		94.96	77.91	51.38
Neural Network	Learning Rate = 0.001; HL: 100	94.77	79.03	56.65
Random Forest		94.43	78.51	49.8
Decision Tree		85.11	65.83	40.92

Table 2. Results of 100k group, balanced data, unigram features

From these results, Logistic Regression performed best across all classes. It is evident that as the number of rating classes decreases, the accuracy score increases. Models underperform with "neutral reviews." It is also worth noting that bigrams and trigrams did not decrease the model performance (see jupyter notebook). Specifically for Logistic Regression, increasing the number of iterations, ended up not improving the model significantly.

95.28% accuracy rate for Logistic Regression is relatively high and there is a big risk of overfitting in this case. As an attempt to reduce overfitting, a couple of hyperparameter tunings were done mainly on the regularization factor. Changing from the default Ridge Regression (L2) to Lasso Regression (L1) made sense as there were too many features. This reduced the complexity of the model somehow buy automatically removing less important features. Increasing the value of lambda (1/C) also added "penalty" to the model, as part of regularization.

Model	Hyperparameters	1&5 stars	1,3,5 stars	All stars
Logistic Regression	L1; C = 10	94.71	78.3	48.77
Logistic Regression	L1; C = 1	95.28	80.45	54.44
Logistic Regression	L1; C = 0.1	**93.46**	80.05	54.66

Table 3. Logistic Regression hyperparameter tuning

Final model is 1&5 class, with unigram features under Logistic Regression (Lasso Regression with lambda = 1/0.1)

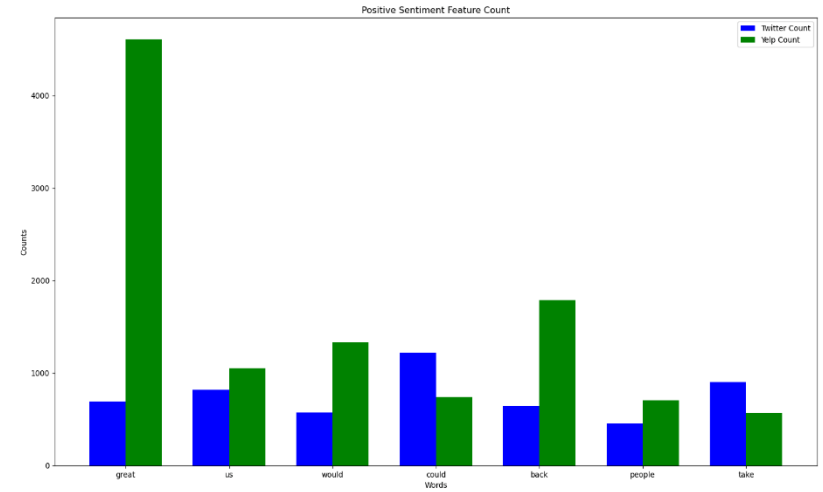
4 Application

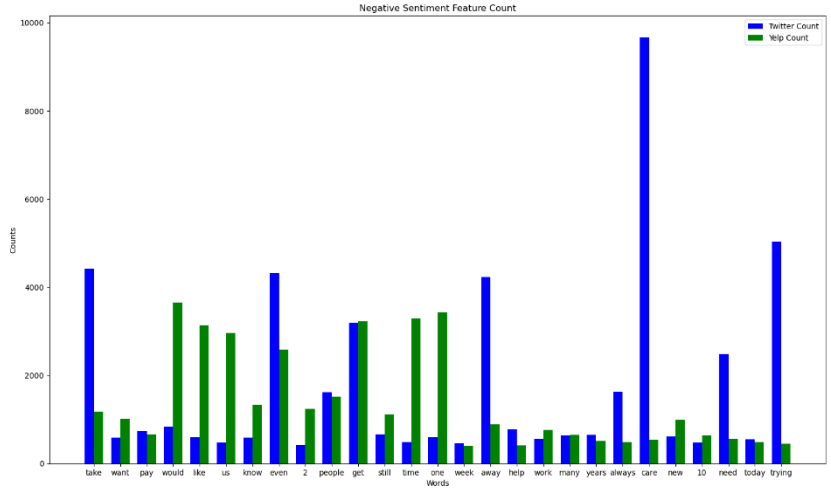
Our methodology for answering the question "what twitter features explain positive and negative sentiments over time?" is the following.

Our features include the unigram vectorization of the Yelp text tokens. The vectorization creates a matrix of all the reviews as rows and the unique unigrams for all the tokens in the Yelp training data as the columns or features. Each item in the sparse matrix is the count

of the unique word for that specific review. Then, we trained the Logistic Regression Model with the Yelp vectorized sparse matrix and applied it to the Twitter data for classification.

To explore the best features that explain positive and negative sentiments we subsetted the Yelp and Twitter data into positive and negative dataframes. From there, we created a dictionary for counts of each unique feature for the respective dataframes. To then evaluate which features explain positive sentiment, we created a function that makes a new dictionary of words that occur in both datasets at a certain threshold and plotted the results. The reasoning behind this is that features or words that occur most frequently in the positive Yelp dataset would have the most influence on classifying what tweets were positive. Furthermore, we wanted to see overlapping words in both positive datasets to see which words or features had the most influence in classification. We then applied the same process to the corresponding negative datasets.

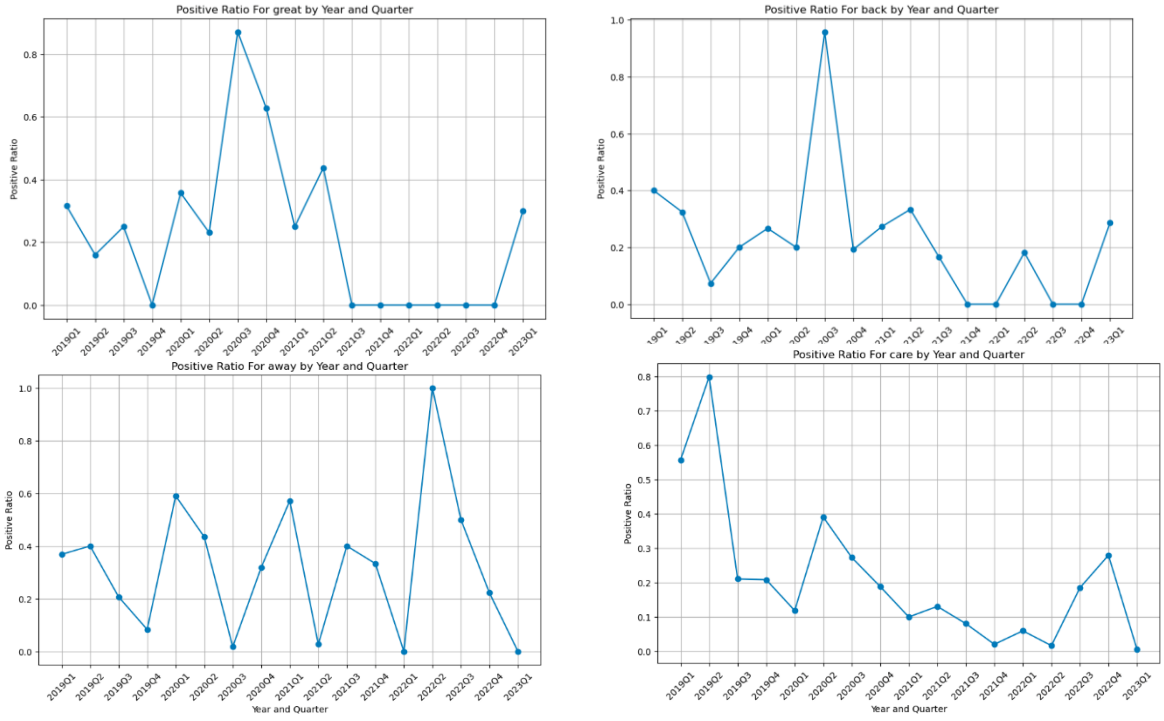




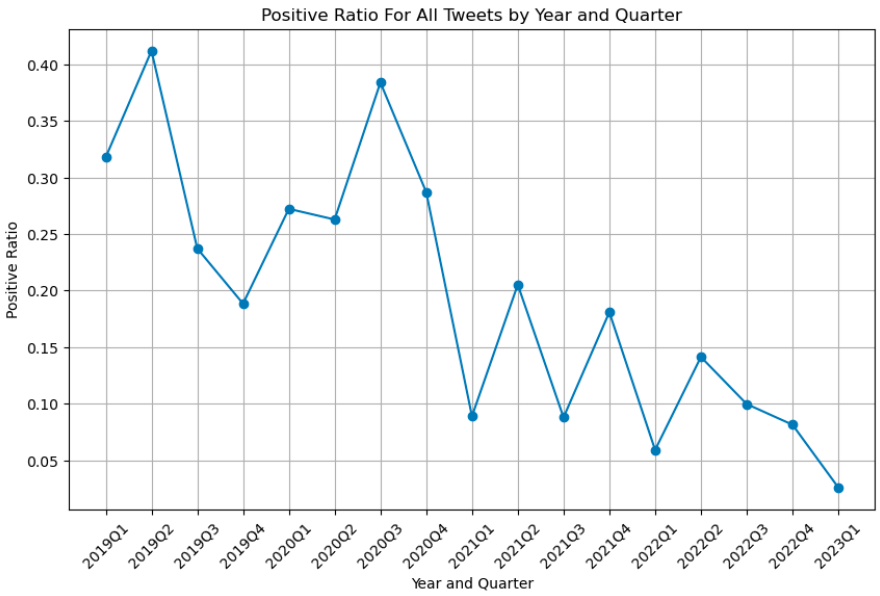
The next step is to evaluate how the sentiment of these features change over time. We approached this by selecting the features that seemed to have the most influence for positive sentiment based on the count thresholds. We then created a function that takes in the whole Twitter dataframe, including the positive and negative classifications, that was then subsetted into only tweets containing that feature. We then calculated the ratio of positive tweets over all of the tweets for that feature per quarter per year. This allows us to see how the ratio changes over time. In addition, we applied this function to see overall sentiment over time, selected words, mentions, and hashtags.

In the results we can see many overlapping words in both the positive and negative feature count plots like “take”, “people”", and “years”. These we will consider neutral since it appears in both datasets. We then selected ‘positive’ unique words like “great” and “back”, and “away” and “care” for the negative sentiment. These words, except for “great”, do not have a positive or negative connotation on their own but were selected based on uniqueness in the respective data set and more weight on occurrence in Twitter dataset.

We then plotted sentiment over time for these selected words. From what we can see, the results are quite noisy. We hoped to see a consistently high ratio of more positive or less positive for the selected positive and negative features, respectively. What is potentially happening is that selecting a single feature does not determine the classification and it is the cumulation of all the features that determines the classification. Potentially selecting trigrams or bigrams as a hyperparameter could provide more consistency on what features determine positive and negative sentiment over time.



In conclusion, as seen in the figure we can see that the overall sentiment for the Affordable Care Act was initially positive and has trended in the more negative directions over time.



5 Takeaways: Challenges, Limitations, and Learnings

5.1 Challenges

- Sampling Bias:

Time frame of twitter data is (Jan 2019 - Mar 2023). This temporal aspect of the data selection can lead to a bias in capturing sentiments that were prevalent during a specific period. Particularly, because this was the time of Presidential Elections in the US. By focusing on a specific time frame, the model might miss out on sentiments that were prevalent in the past. Selection of Yelp reviews and tweets for training and testing can introduce sampling bias. Biases in the selected samples can impact the sentiment analysis results.

- Sentiment Intensity and Context:

Yelp reviews and tweets may differ in terms of sentiment intensity. While Yelp reviews are detailed, tweets are often limited to short and concise expressions. Sentiment analysis models trained on Yelp data may struggle to accurately capture the intensity of sentiment in tweets due to the differences in the length and context of the text.

- Contextual Understanding:

Public program discussions on Twitter may involve complex contextual references, acronyms, or insider language that sentiment analysis models trained on Yelp data might not comprehend.

There is a fundamental difference between the content and context of Yelp reviews, which primarily focus on food-related experiences, and the sentiments expressed in discussions related to a public program. The lack of contextual understanding can lead to misinterpretation of sentiment in tweets.

- Twitter Application:

Not all features in twitter data can be found in the initial set of features created using Yelp data. Twitter dataset introduces unique features and characteristics that were not present in the initial set of features created using Yelp data. We had to remove those features from the testing data as the model was not trained on those features.

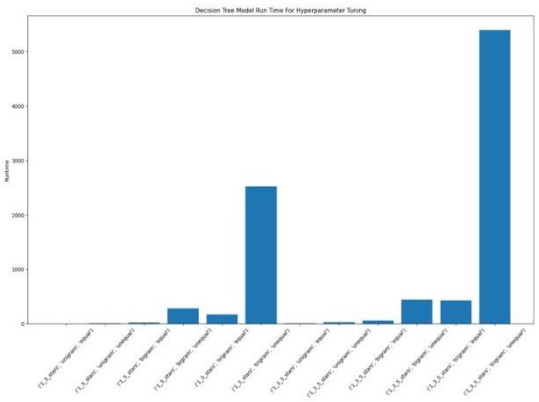
5. 2 Limitations

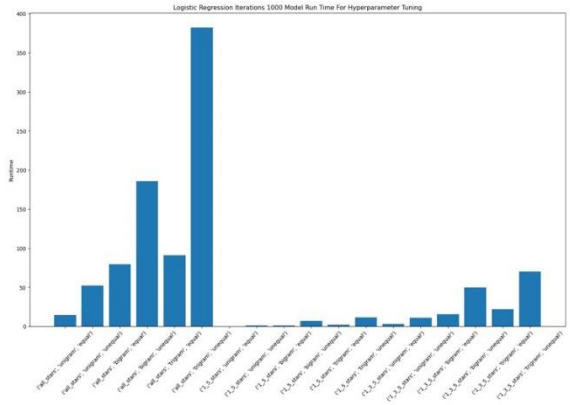
- Runtime and Computing Power Limitations:

One of the major limitations encountered in this project was the restriction imposed by runtime and computing power. The computational resources required to train and evaluate machine learning models, especially when dealing with larger datasets or incorporating additional classes, was substantial. With larger datasets, it became challenging to process and train the model efficiently, potentially compromising the model's performance or necessitating compromises in terms of sample size, model complexity or tuning of hyperparameters. Some of the models took as long as 15 hours to run. We also had to prune the dataset, hyperparameters and exclude complex models such as Neural Networks due to the time complexity and efficiency considerations

. Big O notation of each model's training time complexity.

Multinomial Naive Bayes	Logistic Regression	Decision Tree Classifier	Random Forest Classifier
$O(nd)$	$O(nd)$	$O(n \log(n)d)$	$O(n \log(n)dk)$





- Accuracy power with all-star classifications:

Model faces issues with neutral language and in a binary classification loses its accuracy power with all five stars classifications as it is not powerful enough to correctly decide binary classification of more neutral reviews.

- K-Grams Selection:

Another limitation can be seen in the selection of unigrams over a higher n-gram which lost the ability to capture the context of words. As an example, the feature 'care' in the context of food can be things like "the server didn't care" or "I don't care for this restaurant" which could potentially provide a negative weight for the word 'care'. This would then provide a negative rating for the phrase "health care", which is one of the most common features in the twitter data set and has no relation to the context of 'care' in those hypothetical examples.

5.3 Learnings & Conclusion

The Classification of the twitter dataset captures the overall sentiment of the tweet but lacks the complexity to understand context**.** As discussed, this is because twitter has limited number of characters, this limitation may affect choice of words and the ability to capture the whole context unlike yelp dataset.

Secondly, Yelp dataset is based on business reviews which reflect a person's personal experience and domain knowledge. This is unlike twitter where a person may not have the personal experience with the program or the domain knowledge that accurately reflects on the performance of the program.

More importantly, negative sentiments expressed in a tweet do not necessarily reflect on the performance of the program itself. The negative tweets in some cases even reflected positively on the program and vice versa. Model wasn't trained to understand sarcasm and lexical nuances and hence couldn't capture those nuances

Public Program Review can be affected by political bias. In our model, many tweets classified as positive use mostly the name of the program a.k.a Affordable Care Act, whereas, negative tweets refer to acronyms such as ACA or ObamaCare. Association with Obama may flare up negative sentiments reflecting political bias.

- Example **Positive:** "Biden lauds the Affordable Care Act for the progress it has made over the past 13 years and how beneficial it has been in transforming the health care landscape for Americans."
- Example **Negative:** "@POTUS BS Joe. Obamacare (ACA) made insurance expensive, costs higher, & caused many doctors to quit. The reality is that Obamacare is an utter failure for working America. Government subsidies never result in lower costs, Joe."

Finally, Sentiments may not necessarily imply negative performance of the program. Sentiments could be based on factors described above. Performance of public programs needs to be judged against more rigorous quantitative analysis.