Text Classification using Facebook Political Ads

Mohammed Alneyadi, Jimmy McShane
Course: QAC386
05/21/2021

1. Introduction

The usage of political advertising has increased exponentially in recent years. In particular, it has become an essential component of campaign strategy for candidates in the United States. With the rise of social media, candidates nowadays can easily influence current political debates and attempt to shift voters' opinion to their benefit. This created a concerning political environment where there are really no "rules" governing the type of content candidates can advertise. This also gave rise to comparative advertising, in which candidates blatantly condemn their opponents. Previous studies have concluded that the negativity surrounding the political advertising space leads to decreasing levels of participation in politics because it creates an "avoidance mentality" (Ansolabehere and Iyengar, 1995).

In this paper, we seek to study the tone of political advertisements on Facebook. Specifically, we divide the tone of political advertisements on Facebook into three categories: ads which primarily contrast a candidate's positions with his opponent, ads which primarily serve to promote a candidate, and ads which primarily attack a candidate's opponent. We utilize the tone of political ads, all of which were published on Facebook in 2020, to build an effective text classification model that uses the text contained in the ad to predict its tone. We built multiple machine learning models that use different sets of features, including bag of words and word embedding, in trying to find the model that achieves the best performance.

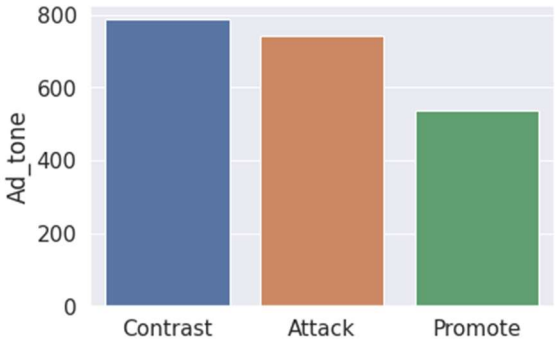
2. Data & Methods

a. Data

The data used in this project contained 3,334 observations, where each observation represents a political ad that was published on Facebook in 2020. The columns include the text of the ad, the tone of the ad, and the ad identification number. The text of the ad includes both the text shown on the body of the ad and the transcript of any media contained in the ad if the ad's content happened to feature text or audio that could be transcribed into a textual format. These two different text columns were combined by adding a space between them in a new third column.

To prepare our data for feature engineering, we cleaned and pre-processed the text column. The steps used in this part included filtering out punctuation marks and stop words, converting the text to lowercase, and grouping together inflected types of a word using Wordnet Lemmatizer. In addition, we also removed any text column that contained less than four words. After performing these steps, we now have 2,067 observations/rows in our dataset.

To examine our target variable, we looked at the class distribution of ad tone, which is shown in the figure below. Although there are more ads that contrast a candidate than attack or promote a candidate, we can see that the dataset is generally balanced. There are 538 ads that promote a candidate (26%), 742 ads that attack a candidate's opponent (36%), and 787 ads that contrast a candidate with her opponent (38%).

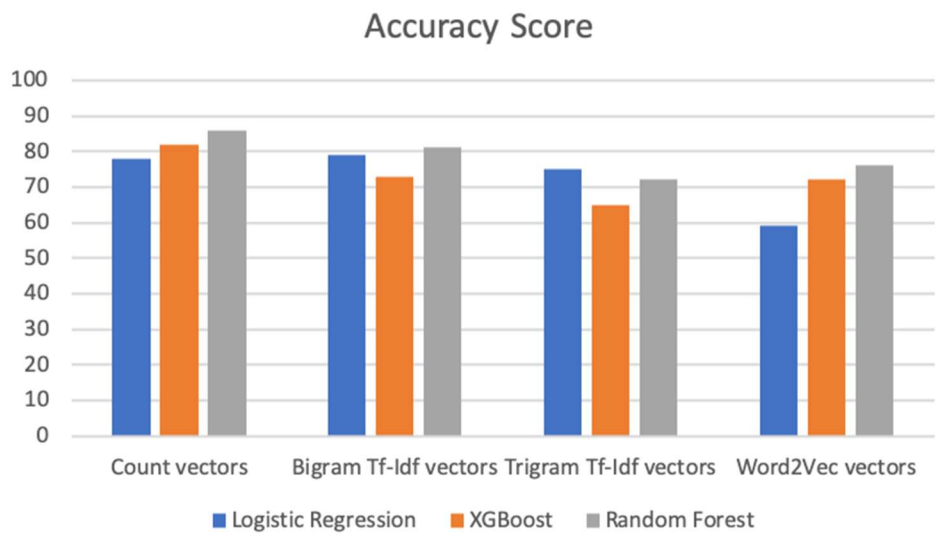


b. Methods

To build a machine learning model, we divided our dataset using an 80:20 split of training data versus testing data. We needed to first convert the text data to numerical data vectors. In doing so, we utilized multiple methods of vectorization, which included bag-of-words (with Count vectors & Tf-Idf) and word embedding (with Word2Vec). First, we converted our text data to Count vectors, which simply records a count of the number of times a word appears in each document. Second, we used Term Frequency-Inverse Document Frequencies (Tf-Idf) for converting the text to numerical data. We obtained the Tf-Idf scores for both bigrams and trigrams. Third, we used gensim to build a Word2Vec model to train it on our text corpus, in order to get a Word Embedding. We trained multiple machine learning models on each of the four vectorized datasets to evaluate the performance of each vectorization method.

We built three models using logistic regression, extreme gradient boosting, and random forest. For each model, we train four vectorized datasets, which include count vectors, bigram level Tf-Idf vectors, trigram level Tf-Idf vectors, and Word2Vec vectors, on it. For the random forest model, we set the number of parameters (n_estimator) to 200 and the criterion parameter to “entropy”. We evaluated the trained models based on their performance in predicting the labels in the testing set.

3. Results



The plot above shows the accuracy score for each model across the sets of features that were trained on it. Across the board models that were trained on bigram level Tf-Idf vectors and Count vectors had higher accuracy scores than models that were trained on trigram level Tf-Idf vectors and Word2Vec vectors. Using Logistic regression, count vectors and bigram Tf-Idf vectors achieved an accuracy score of 78% and 79%, respectively. In contrast, trigram level Tf-Idf vectors and Word2Vec vectors achieved an accuracy score of 75% and 59%, respectively. Also, while Word2Vec vectors didn't perform well using a logistic regression model, it achieved improved accuracy scores of 71% and 76% using the XGBoost model and the random forest model, respectively.

In terms of the models we developed, the random forest model performed the best across almost all sets of vectors. For count vectors and Word2Vec vectors, the XGBoost model performed better than the logistic regression model. However, for bigram level and trigram level Tf-Idf vectors, logistic regression performed better than the XGBoost model. Our best

performing model is the random forest model that was trained on Count vectors as its set of features. It achieved an accuracy score of 86%.

4. Conclusion

In this paper, we used Facebook political ads to build a text classification model that predicts the tone of the ad and classifies it in one of the three categories: ads which promote a candidate, ads which attack a candidate's opponent, and ads which contrasts a candidate's position with his opponent's. We used four methods of vectorization to convert text to numerical values, and we trained the vectorized dataset of each using different machine learning models. We found that Count vectors and bigram Tf-Idf vectors had better performance than trigram level Tf-Idf vectors and Word2Vec vectors. Our best performing model, which achieved an accuracy score of 86%, is the random forest model that was trained on Count vectors. Below, you can see the classification report of the results of this model. Overall, the model achieved well-balanced accuracy, precision, and recall metrics.

	precision	recall	f1-score	support
0	0.84967	0.90278	0.87542	144
1	0.86905	0.84393	0.85630	173
2	0.87097	0.83505	0.85263	97
accuracy			0.86232	414
macro avg	0.86323	0.86059	0.86145	414
weighted avg	0.86276	0.86232	0.86209	414

5. References

[1] Ansolabehere, Stephen, and Shanto Iyengar. (1995). "Going Negative: How Political Ads Shrink and Polarize the Electorate." *New York: Free Press.*