Using Sentiment Analysis of Tweets to Gain Insight into the 2020 Democratic Nomination

Jiamin Huang, Michael Lu, Aakarshan Dhakal $\{jhuang13, mingci7, adhakal\}$@$stanford.edu$

1. Introduction

With the days news regularly dominated by the commander-in-chiefs Twitter commentary, the (now) 280-character tweet is the modern de-facto way to share political news, thoughts, and opinions. Given Twitters ascent in this sphere, the ability to identify the sentiment of tweets about political topics could provide a quicker temperature reading of public opinion than traditional polling. Our project tackles this natural language processing problem by building binary classifiers to automatically assign tweets a positive or negative sentiment. These classifiers can then be used on tweets about each of the potential 2020 Democratic presidential candidates, such as Joe Biden, Elizabeth Warren, and Bernie Sanders, to gain insight into their relative popularity. Our input is the text of a Tweet that has been pre-processed, a procedure that is detailed later in the paper. This text is then used in a Nave Bayes Classifier to output a predicted sentiment class, either positive (1) or negative (0). We trained our models with a set of general-topic tweets under the assumptions that human sentiment is not contingent on the topic (i.e. the features that make a tweet negative or positive are not wholly dependent on what the tweet is about) and that it is better to use a large training set of more general tweets than a much smaller training set of specifically political tweets. However, our model testing will be on both a set of general-topic tweets and a set of manually labeled political tweets.

2. Related Work

An overview of current popular sentiment analysis methods is given by Ribeiro et al. (2016), which outlines twenty-four commonly used methods covering both supervised learning models and lexicon-based approaches [1]. Many of these methods are commercially available and meant to be used off-the-shelf. Comparing our approach to the supervised learning models evaluated in this paper, we similarly developed our final models using a more general training set, which we then applied to a specific context. From the results of Ribeiro et al. (2016), we see this can have varying results that are both model- and context-dependent, with no one model unilaterally outperforming the others. Another overview of commercially available sentiment analysis tools, this time Twitter-specific, by Abbasi et al. (2014) evaluates the performance of twenty models across five different contexts, and again emphasizes the idea that various models will perform quite differently depending on the context. Unfortunately, both papers are unable to provide insight into how these off-the-shelf methods compare to ones actively developed for a particular purpose. Looking at related work done in our specific context, that of political tweets, Soelistio and Tunggawan (2017) provides an example of sentiment classification on political tweets, this time relating to the 2016 presidential election [3]. In particular, Soelistio and Tunggawan make use of a Naive Bayes Classifier, as we also do; however, they evaluate the tweets relating to fifteen different presidential candidates, training a model for each one. Given our time and resource constraints, as well as disadvantageous timing given that the 2020 presidential race is just in its nascency and real candidates are unclear, we chose to train our models using a general set of tweets rather than tweets specific to 2020 presidential politics.

3. Dataset

In order to train these models, we used the publicly-available Sentiment140 dataset, which contains 1.6 million general-topic tweets that were evenly distributed, with exactly half labeled with positive and half labeled with negative sentiment [5]. Of these 1.6 million tweets, 80 % (1.28 million) was used to train our models. The other 20 % (320,000 tweets) was held out to be used as a test set to evaluate the models. Additionally, we collected about 1500 tweets from Twitter from November 12-26, 2018, using potential Democratic presidential candidates names as queries. (The candidates chosen were those who received more than 1 % support in a November POLITICO poll [6].) We restricted our search so the tweets we collected all originated from the United States, and were consequently all in English. These tweets were then manually labeled with sentiments positive, negative, or unclear. All of the unclear tweets were subsequently removed, leaving us with 919 tweets of just positive or negative sentiment to be used as a context-specific test set to evaluate our models with. Of these 919 tweets, 285 were positive and 634 were negative - a slightly more uneven distribution that may be due to the overall negative slant of Twitter. Below are examples from both the Sentiment140

dataset and the collected political dataset.

	label	text
0	0	is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!
1	0	@Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds
2	0	my whole body feels itchy and like its on fire
3	0	@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.
4	0	@Kwesidei not the whole crew
5	0	Need a hug

Figure 1: Examples of entries from the Sentiment140 dataset, pre-processing

	label	followers	friends	message
0	4.0	57.0	177.0	And this is largely a result on the grassroots organization of the Beto O'Rourke campaign! Down ballot wins are BI… https://t.co/IWVHt8ZbAs
1	4.0	703.0	573.0	I want a man like Beto O'Rourke
2	4.0	798.0	355.0	Friends, runners, humans, read this. \n\nExcellent reflection-on-a-run from Beto O'Rourke.\n\nhttps://t.co/B360agNuMG

Figure 2: Examples of entries from the collected tweets dataset, pre-processing

All of the data underwent pre-processing to produce somewhat more uniform data to train and test the model with. This included removing hyperlinks and mentions of other users, removing extra whitespace and extraneous characters, and converting the text to lowercase. Stop words, words like and, is, and the, which are present in most text and are presumed to provide no extra information for classification, are then removed. With this normalized text, we then use the CountVectorizer function to transform the set of tweet texts into the bag-of-words matrix form, with each row of the matrix representing a tweet and each column representing a unique word. The entries of the matrix are either 1 or 0, representing whether or not a word is present in a tweet respectively.

4. Methods

As the problem we are tackling is one of binary classification, we tested various machine learning methods weve learned so far that handle classification.

4.1. Naive Bayes Classifier

Our initial baseline model is a Nave Bayes Classifier. The idea underlying the Nave Bayes Classifier is Bayes Rule, which calculates the posterior conditional probability given a set of prior probabilities that are more easily estimated (the equation is given below).

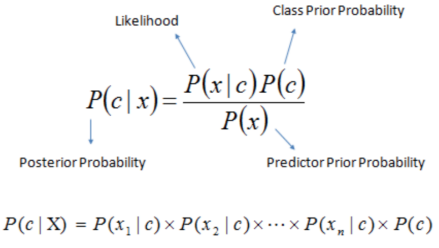

Figure 3: Illustration of Bayes' Rule

The Nave Bayes Classifier predicts the probability of an example belonging to class C given the values of its features X. What makes this classifier nave is its assumption that each feature is independent of the others (i.e. the presence or value of one feature has no bearing on the presence/value of another feature). Overall, while the underlying mechanisms differ, Nave Bayes is comparable to logistic regression in that both attempt to estimate $P(Y = 1|X)$ in order to make their predictions.

The Nave Bayes method by itself has a large weakness in that a word in the test set that was not present in the training set will be assigned a probability of 0, since it had zero frequency in the training data. In order to correct for this, we use Laplace smoothing, in which a small pseudocount alpha is incorporated into every probability estimation. Alpha, our smoothing parameter, is a hyperparameter that we need to determine the optimal value of.

4.2. Logistic Regression

The second model we worked on extensively was a regularized logistic regression model. Logistic regression is trained by minimizing the cost function shown below.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log\left(h_\theta(x^{(i)})\right) - \left(1 - y^{(i)}\right)\log\left(1 - h_\theta(x^{(i)})\right)] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

The model assigns classifications by estimating $P(Y = 1|X)$ (in the above equation, h(x)) for a singled example. If the probability is greater than 0.5, then the example is assigned to the positive class. Otherwise, it is assigned to the negative class. The regularization parameter, lambda, is to help prevent overfitting the training data. The larger lambda is, the more regularized the algorithm, meaning that we are introducing more bias in order to reduce the variance.

4.3. Determining Other Hyperparameters

Additionally, not specific to these models, we also needed to determine other choices of hyperparameters. These include choosing what value of n-grams should be extracted from the text for the final vocabulary (e.g. unigrams only, or unigrams and bigrams), and deciding whether or not to use TF-IDF weighting (weighting based on term frequency and inverse document frequency, which reflect the mportance of certain words in a tweet by finding their usage in a tweet and their overall usage in the dataset).

After training our models, we used 10-fold cross-validation to evaluate the performance of models with different hyperparameters, in order to select the best combination of hyperparameters. Then, we evaluated the best model on two test sets one made up of held-out data from the Sentiment140 dataset, and one made up of manually collected

and labeled political tweets to see how the performance compares across different methods and with different types of tweets.

5. Results and Discussion

To evaluate all of these models, we look at their overall accuracy first and foremost. Precision and recall are used to provide insight into the strengths and weaknesses of the model.

5.1. Results of Naive Bayes Classifier

For our Naive Bayes Classifier, we provided twenty possible combinations of the hyperparameters. Using 10-fold cross validation, we found the best performing model (i.e. had the highest overall accuracy) to be the model where both unigrams and bigrams were used, alpha = 1, and no TD-IDF weighting was implemented. The overall accuracy of this model on the general-topic test set was 78.02%. This is a fairly high overall accuracy, since according to Mozeti et al. (2016), even humans annotators will only agree with their own labels about 80% of the time [6]. Additionally, the precision was 79% and the recall was 77%, indicating that the model does a fair job at both identifying the actually positive tweets as positive and not mislabeling negative tweets as positive, at least for the general test set. Since our model performed similarly on the training data and the test data of the same type, overfitting is unlikely. However, when we ran our model on the focused test set of political tweets, we found that the model had an overall accuracy of just 58.2%. As we can see from the confusion matrix on the right, our model had a tendency to over-assign a positive label, leading to a precision of just 41% (recall was 77%). While this seems to be a general problem for sentiment analyzers, as noted by Ribeiro et al. (2016), we hypothesize that some of this error may be due to the fact that our training set is not as focused on the language that may be more common in political tweets. Additionally, the absence of a third neutral class may contribute to this error, as there may be more difficult, borderline tweets that possibly dont fit all too well as positive or negative, yet the classifier leans toward assigning them as positive.

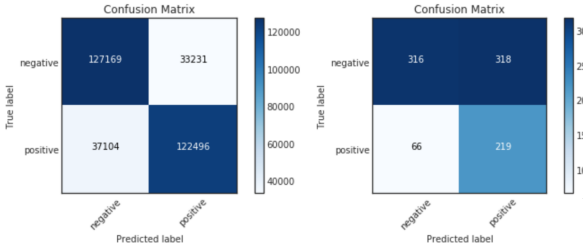

Figure 4: Resulting confusion matrices of the Naive Bayes Classifier. Left corresponds to the general test set, right to the political test set

5.2. Results of Logistic Regression

For our logistic model, we provided twelve possible combinations of the hyperparameters. After performing 10-fold cross validation again, we found the best model had the following parameters: both unigrams and bigrams were used, regularization parameter = 1, and TD-IDF weighting was implemented this time. Due to our use of a regularization parameter, overfitting is unlikely, and the results support that assertion. The overall accuracy of this model on the general-topic test set was 79.09%. Additionally, the precision was 78% and the recall was 80%. Overall, this model had slightly better performance than our Naive Bayes classifier. On the set of political tweets, we found that the model again performed similarly, with an overall accuracy of 58.8%. As we can see from the confusion matrix on the right, our logistic model also had a tendency to over-assign a positive label, leading to a precision of 41% (recall was 74%).

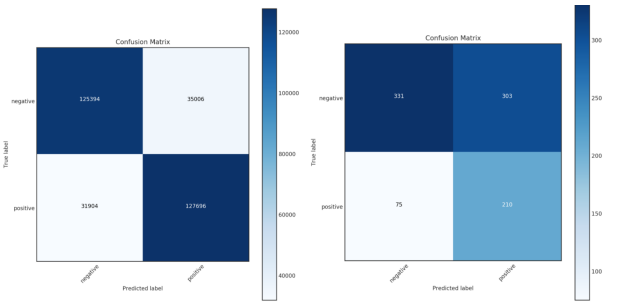

Figure 5: Resulting confusion matrices of the logistic regression model. Left corresponds to the general test set, right to the political test set

6. Conclusion and Future Work

We trained models using the Sentiment140 dataset in order to produce classifiers that could separate tweets into positive and negative sentiment classes. The Naive Bayes Classifier performed well on the test set of general-topic tweets, with an overall accuracy of 78%, but did less well on the politically-focused tweets, with an overall accuracy of 58.2%. The logistic regression model did similarly, with an overall accuracy of 79% on the general tweets and 58.8% on the political tweets. Furthermore, both models struggled with precision on the political test set, tending to overassign positive labels.

Looking forward, there are a number of ways we believe our work so far can be extended. For one, the collection of a semi-large set of accurately labeled political tweets could help us determine if training our models on topic-specific tweets will significantly improve their accuracy. Additionally, we could expand the number of classes to include a neutral class, for those tweets that dont have an overwhelmingly positive or negative sentiment and may be more difficult for our classifiers to identify. We could also refine the text pre-processing to include more complex or

potentially helpful features, by performing things such as parts-of-speech tagging or stemming (reducing words to their root form).

7. Contributions

Michael and Aakarshan did exploratory data analysis of the training set, text pre-processing, model training and evaluation. Jiamin gathered and labeled political tweets through the Twitter API, performed model training and evaluation, researched related literature, compiled figures and final write-up.

8. References

[1] F. N. Ribeiro, M. Arajo, P. Gonalves, M. Andr Gonalves, and F. Benevenuto, SentiBench - a benchmark comparison of state-of-the-practice sentiment analysis methods, EPJ Data Sci., vol. 5, no. 1, p. 23, Dec. 2016.

[2] A. Abbasi, A. Hassan, and M. Dhar, Benchmarking Twitter Sentiment Analysis Tools, in Language Resources and Evaluation Conference (LREC), 2014, pp. 2631.

[3] E. Tunggawan and Y. E. Soelistio, And the winner is...: Bayesian Twitter-based prediction on 2016 U.S. presidential election, in Proceeding - 2016 International Conference on Computer, Control, Informatics and its Applications: Recent Progress in Computer, Control, and Informatics for Data Science, IC3INA 2016, 2017, pp. 3337.

[4] A. Go, R. Bhayani, and L. Huang, Twitter sentiment classification using distant supervision, 2009.

[5] S. Shepard, Poll: Biden, Bernie, Beto lead 2020 Dem field - POLITICO, 2018. [Online]. Available: https://www.politico.com/story/2018/11/12/poll-biden-bernie-beto-lead-2020-dems-983995. [Accessed: 09-Dec-2018].

[6] I. Mozeti, M. Grar, and J. Smailovi, Multilingual Twitter Sentiment Classification: The Role of Human Annotators, PLoS One, vol. 11, no. 5, p. e0155036, 2016.

Thanks to Ronald Wahome for the use of some code which helped us implement Naive Bayes. Code used found at: https://github.com/RonKG/Machine-Learning-Projects-2/tree/master/3.