Emoji Sentiment Analysis

Angel Karafas, Dana Rubin, and Emily Stahle

November 2020

1 Introduction

According to Phill Agnew with Brandwatch Analytics, over 10 billion emojis are sent daily and the majority of internet users have used emojis (Agnew, 2018). Within the past several years, emojis have become a popular way for people to express their emotions in a compact way. It is increasingly more difficult for programs to decipher the meaning of emojis. Our analysis of emoji sentiment through Logistic Regression and Naive Bayes allows us to further understand emoji use and meaning. Our Github is located here.

2 Data Collection and Preparation

We used the EmojifyData-EN data set on Kaggle by Daniil Larionov (Larionov, 2019). The data set took 18 million tweets from January through April of 2018 from the ArchiveTeam Twitter Stream Grab. We used 10,000 tweets from this data set due to processing speed limitations. The data set was first processed by removing any non-English tweets, hashtags, mentions, and URLs. Each tweet contained at least one emoji. This data set did not include labels for positive and negative sentiment.

To not limit our analysis of tweet sentiment, we added labels to each tweet. We first took in a set of 118 labeled emojis from emoji-emotion with labels from -5 to +5 (Wormer, 2019). We then relabeled the set to have binary 1's and 0's for positive and negative sentiment. We did not include emojis that have neutral sentiment. We also took into account how some of these emojis are used in our society and on twitter in our relabeling process. Because there are around 3,000 emojis, future work for this project would be to use a larger set of labeled emojis to analyze. From this labeling process, we made a list for both positive and negative emojis.

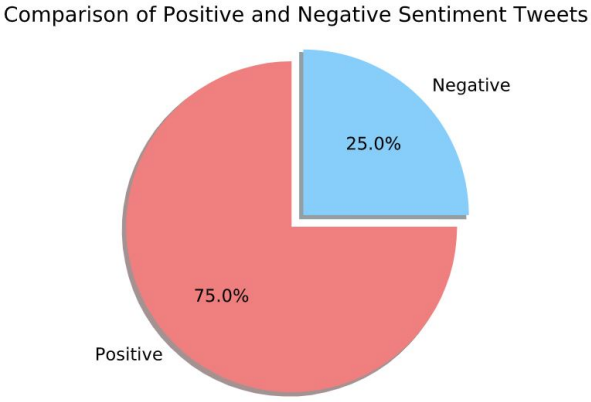
Sample of Positive and Negative Emojis

negativeEmojis = [😐,😩,🙃,😒,🙄,😫,😓,😭,😕,💀,💩,😱,😡,😈,😑,🤢,🖕,
☹,😳,😨,😑,😢,😟,😖, 😰]

positiveEmojis = [😉,💞,👍,😎,😜,😃,😻,😄,🙂,🤣,💋,💜,😆,😁,😘,😄,😇,😍,
❤,😁,💖,🤠,👏,😊,🖤]

To label the tweet data set, we used the positive and negative labeled emoji lists to see if the tweet contained a positive or negative emoji. If the tweet contained a positive emoji, it was labeled positive, the same for negative. If the tweet contained both positive and negative, we labeled it positive. Further work could contain a more rigorous labeling process, or a prelabeled tweet data set. Fortunately, there were not many tweets that contained both from our lists of positive and negative emojis. After the labeling process we ended with a data

set of 75% positive tweets and 25% negative tweets. This aligns with an observation from an analysis by Phillip Agnew with Brandwatch Analytics, "On average, 75% of emojis published on Twitter are positive and 25% are negative" (Agnew, 2018).



The tweets were further preprocessed before Logistic Regression and Naive Bayes for optimization and accuracy of learning. Tweets were turned into lowercase and tokenized into a list of strings. English "stop words" were removed using the Natural Language Toolkit and numbers, punctuation, emoticons, and Twitter specific language such as "rt" and "favorite" were removed (Bird et al., 2019). Groups of emojis were separated as distinct words in this process. The words in the tweets were also stemmed to their root form to further optimize learning.

3 Approach

For sentiment analysis of our tweets with emojis, we implemented Logistic Regression and Naive Bayes. This provides a robust analysis of our emoji data set with a comparison of two different supervised machine learning mechanisms. The Coursera courses Sentiment Analysis with Logistic Regression and Sentiment Analysis with Naive Bayes and their assignment solutions by Aman Chadha were utilized to guide our sentiment analysis on our emoji data set (Bensouda Mourri, 2020; Chadha 2020). Both approaches used the same labeling process, preprocessing, and split the training and test data set into 70 and 30 percent.

3.1 Logistic Regression

We first built a frequency dictionary for each word in our tweet data set including the label and count of how many times the word appeared in positive and negative tweets. The words were preprocessed in order to cut down on unnecessary words in our frequency dictionary. A single word may appear twice in the dictionary, once for both positive and negative sentiment.

We then used this frequency dictionary for feature extraction on our training set. This consisted of looping through each preprocessed tweet, checking our frequency dictionary for the word, and incrementing the positive and negative word counts for the tweet. Feature extraction resulted in a matrix; for each tweet there was a bias set to one, the positive word count, and the negative word count. This matrix created with feature extraction was used as our features for gradient descent.

Gradient descent took in features 'X', labels 'Y', an empty vector to store weights 'Theta', step size, and the number of iterations. Gradient descent was used to optimize the average loss for Logistic Regression of the training set.

$$average\ loss = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(pred(ith\ tweet)) + (1 - y_i)\log(1 - pred(ith\ tweet))$$

This is done in the log domain to avoid numbers going to zero and to increase accuracy. Each iteration of gradient descent calculates the average loss by using the prediction, a product of the weights and features passed through the sigmoid function.

$$sigmoid(x) = \frac{1}{1 + \exp(-x)}$$

This scales the linear function to our correct [0,1] range. The loss and weights eventually converges and reach a minimum. The final loss and weight vector are returned from gradient descent. Our final loss and weight vector was 51.3% and [9e-7, .00081509, .00018521] with a step size of 1e-9 and 2,500 iterations.

After training, we tested our model with the test set. For each tweet in our test set, we first did feature extraction using our tweet and frequency dictionary. To get our prediction for each tweet, we passed the product of the features and learned weights through the sigmoid function. Error was calculated by counting the number of incorrect predictions from our test set and dividing it by the number of tweets. The accuracy of the model, 1-Error, was 76.02%.

3.2 Naive Bayes

We first used an array containing all of the tweets with their corresponding labels to find the total number of positive and negative tweets in the data set. Since there are many repeated words in the tweet data, we looped through the frequency dictionary in order to find the total number of positive and negative words and the total number of unique positive and negative words.

Next, we found the probability that an individual tweet was positive or negative. To do this, we took the same labels 'Y' from the Logistic Regression model and cycled through that to find the total number of positive labels and negative labels. To get the probability that a tweet was positive, we divided the total number of positive labels by the total number of labels. We followed this same method to find the probability of a negative tweet. From this, we took the log of the prior probability, which is the log of the ratio of the probabilities.

$$log\ prior = \log(\frac{prob(tweet\ is\ positive)}{prob(tweet\ is\ negative)})$$

Following the calculation of the log prior, we began to find the log likelihood of each unique word. First, we needed to find the probability that a specific word was positive or negative. For each word in our set of unique words, we found the frequency of the word being positive. We used this to find the probability for a specific word being positive.

$$prob(word\ is\ positive) = \frac{frequency(positive) + 1}{total\ positive\ +\ num(unique\ words)}$$

We followed the same approach to find the probability of a specific word being negative. To find the log likelihood of the word, we took the log of the ratio of the probabilities and stored this value in a set.

$$loglikelihood(word) = \frac{prob(word\ is\ positive)}{prob(word\ is\ negative)}$$

Next we tested our model with the testing set. For each word in a tweet we found its log likelihood in our stored set, as long as it existed in the set. We then added up the log likelihoods of every word in the tweet plus the log prior and returned this value as the predicted sentiment. To test, we created our own "positive" and "negative" tweets. The positive tweet returned a positive value and the negative tweet returned a negative value, which were the values we expected. Our Naive Bayes model finished with an accuracy of 94.06

4 Results and Impacts

Logistic Regression did not have the accuracy that we expected. We took some steps to improve accuracy and did not see significant results. Increasing the step size of gradient descent did not bring down the loss function, where the minimum we found was at 51.3%. We tried 1,500, 2,500, and 3,500 number of iterations for gradient descent and did not see a significant increase in the optimization of the loss function. The best number of iterations was 2,500, where 1,500 provided an optimized loss of 1.3% higher and 3,500 was too slow to run. Changing the step size, although it impacted the loss and weights, did not have an impact on the accuracy of the model.

The Naive Bayes method had a much higher accuracy of 94.06%, which would be expected given the type of data that we are analyzing. The Naive Bayes model uses the assumption that the data is independent and thus works best on independent data sets. With our project, each tweet is being written by a different person and about a different topic so it is fair to assume that they are relatively independent. This explains the discrepancy in accuracy between the two approaches.

In running our Naive Bayes classifier, we also did some additional work to determine what tweets were classified as the most positive, most negative, and what types of tweets were misclassified. The most interesting finding in this analysis was that in each of the most positive tweets, there was a large number of positive emojis. The black heart emoji seemed to be the most prominent, but there were also hearts of various other colors as well that ultimately skewed each of the tweets to have the following highly positive sentiments: [86, 60, 56, 51, 50]. Of the most negative tweets, there was not the same amount of emoji overuse that was seen in the most positive ones. However, each of the five most negative tweets did include at least one negative emoji, though many included more than one. Altogether, this shows that emojis played an important role in determining not only the positive or negative emotion of a tweet but also the level of positivity or negativity.

To better understand the accuracy of both our Logistic Regression model and our Naive Bayes classifier, we did additional analysis to determine five tweets in each model that were misclassified. With both models, the five samples for incorrectly classified tweets were all misclassified as positive. It is interesting to note that each of these ten tweets also includes at least one negative emoji. Thus, the tweet may have been misclassified as positive based on the words of the tweet, as it is clear that the emoji sentiments match the overall tweet sentiment. If we were to further work on our classifier, we may have it place more weight on the emoji content of a tweet, as it seems that the emojis may be more effective in predicting the overall sentiment of the tweet than the words. This also shows one of the main driving factors behind our research, which is that we are living in a world in which emojis are being used more and more to express ourselves. So, it only makes sense to consider them as equally if not more important than words in determining the emotion evoked by a tweet.

5 References

1. Agnew, P. (2018, January 9). " 6 Facts About Emojis Found Using New Analysis ." Retrieved November 17, 2020, from https://www.brandwatch.com/blog/6-facts-about-emojis-found-using-new-analysis/.

2. Bensouda Mourri, Y., Kaiser, Ł, amp; Shyu, E. (2020, June). Sentiment Analysis with Logistic Regression and Sentiment Analysis with Naive Bayes. Lecture presented at Natural Language Processing with Classification and Vector Spaces in Coursera. Retrieved November 10, 2020, from https://www.deeplearning.ai/natural-language-processing-specialization/.

3. Bird, S., Klein, E., Loper, E. (2019, September 4). Accessing Text Corpora and Lexical Resources. Retrieved November 17, 2020, from https://www.nltk.org/book/ch02.html.

4. Chadha, A. (2020, August). Natural Language Processing Specialization on Coursera. Retrieved November 11, 2020, from https://github.com/amanchadha/coursera-natural-language-processing-specialization.

5. Larionov, D. (2019, May 26). "EmojifyData-EN: English tweets, with emojis." Retrieved November 17, 2020, from https://www.kaggle.com/rexhaif/emojifydata-en.

6. Titus, W. (2020, October 6). Emoji-emotion. Retrieved November 17, 2020, from https://github.com/words/emoji-emotion.