# Sentiment Analysis in Text

Richard Lettich, Carmen Gaver, Bryce Taylor, Mary Grace Oster

Fall 2018

**Abstract**

This paper introduces an approach for how to predict sentiment from idioms and text using Long Short Term Memory (LSTM) Neural Networks. In general, the neural networks learn from loops and this allows the method to predict the correct sentiment with high success. Following an overview of LSTM Neural Networks, we show it's success rate on the Plutchik's Emotion data and the Twitter data. After illustrating its performance on on two data sets, a set of labeled data according to Plutchik's wheel of emotion and a sample of labeled Twitter data. We then show how it can be used on raw Twitter data.

# 1 Introduction

The goal of this project is to create a method which will accurately analyze a given text and predict the correct emotion which is expressed in the text. For many companies there is a high need for programs which can analyze a customer review and determine the appropriate underlying sentiment. Customer satisfaction plays a huge role in how companies compete. Novak, Sparl, and Azman explain how "increased service quality positively influences customer satisfaction", which in turn leads to better financial results through increased customer loyalty [1]. Therefore companies need to know how well they are doing, how satisfied are the customers, where the company can improve, and much more. The goal of this project is to create a method which can accurately assign emotion to a given sentence.

It should be stated that Chick-fil-A is a customer service oriented brand. They desire to potentially use the methods described in this analysis as part of a larger project to discover how emotion relates to restaurant reviews in online comments and customer perception of service failure.

A spectrum of emotion (such as 'satisfaction') would likely be best for this purpose. Since the goal is to determine what service failures and successes cause the strongest customer emotions and predict the former using the latter, it would be best to rank the emotion quantitatively. Determining that long wait times make people unhappy would provide no novel information, but being able to say that people become an order of magnitude more upset over one service failure than another would be beneficial as a tool of analysis. In addition, a spectrum of satisfaction would allow for better predictive capability when being used as a component of another analysis [2].

Secondly, it would be ideal to have a confidence value for the emotional prediction. If this sentiment analysis is used as part of another meta-tool, being able to provide an estimate of confidence would allow the meta-tool to weigh the sentiment analysis appropriately with its other variables.

Succinctly, we can define our desired result as follows: for a given snippet of text, give a numerical rating on a spectrum of a single (or possibly multiple) emotions along with an associated confidence score for this rating. To accomplish this goal, we use machine learning.

# 2 Machine Learning

Machine Learning is the process of using a data set with known variables to create an algorithm which will predict a response variable [**Statlearning**]. There are a variety of different machine learning methods, such as Quadratic Discriminant Analysis, Neural Networks, Ridge Regression, Logistic Regression,

and much more [**Statlearning**]. In this data set each sentence has an assigned emotion so the method will be a supervised learning method [**Statlearning**]. Simply put a supervised learning method uses a data set which has both the predictor variable and the response variable to create an algorithm to predict the response variable [**Statlearning**]. In this situation, the predictor variables are the sentences and the response variables are the emotions associated with each sentence [**Statlearning**].

The product of a supervised machine learning process is a model function which yield (a) predicted value(s) based on input data $x$. For the function to be built, prelabeled input data must be provided (i.e., pre produced sets of corresponding points, $(x, (f(x)))$. While using $x$ and $f(x)$ oversimplifies the dimensionality of the input and outputs of the model, they will be continually used to explain machine learning conceptually.

In this paper, we utilize a supervising machine learning algorithm broadly categorized by the label of stochastic gradient descent[3]. The process is not fundamentally different to regression. There is an algorithm which attempts to find optimal parameters[4] [1] for the function f(x) to yield the best prediction. In the context of supervised machine learning, this algorithm is called the optimizer. For instance, if the optimizer was trying to produce a linear regression model,

$$f(x) = mx + b$$

The optimizer's goal would be to find the most ideal values of parameters $m$ and $b$.

While this task is not daunting for such a simple function, it becomes much more complex for larger models that may contain thousands of parameters. However, at the simplest level, the process involves three parts: the model $f(x)$ itself, an optimization algorithm $\alpha$, and a loss function $\lambda$.

## 2.1 Loss function

The loss function dignifies the error of the the predictions made by the model. It takes two parameters: the evaluation of our model at any given point$(y_p)$, and the actual value of the function at that given point, as provided by the input data set $(y_p)$. A simple error function would be the distance from our prediction to what our actual value is

$$\lambda(y, y_p) = |y - y_p|$$

The vast majority of loss functions are not so naive, but the moral this moral explanation holds.

[1] while the parameter of our model $f(x)$ is $x$, this is *not* the parameter we are addressing

## 2.2 Optimizer function – *Stochastic gradient descent*

In this paper, we use the *Adam* optimizer, which is a tuned up version of stochastic gradient descent optimization. As it name alludes to, stochastic gradient descent optimizes the model function by descending the gradient of the loss function.

Before explaining what that means, it should be known that SGD optimizers assumes that if the loss function is smooth and continuous *enough*, we are able to smooth it out, and the same rules of calculus still apply.

From this, if we take any point on the loss function, we are able to extrapolate that if we were modify our parameters $p$ (where $p = [m, b, ...]$), going in the direction opposite of the gradient of the loss function with respect those parameters ($\frac{\partial \lambda}{\partial p}$), we would be able to continuously re-evaluate and lower our error until we reach a gradient near zero, indicating we hypothetically found our optimal model parametrs

What does that mean? To illustrate simply, if we let $f(x)_{\text{act}}$ represent our actual data point, our loss function can be rewritten as

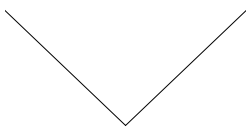
$$\begin{aligned} \lambda &= |f(x)_{\text{act}} - f(x)| \\ &= |f(x)_{\text{act}} - (mx + b)| \end{aligned} \tag{1}$$

for any particular point $x$ in our data set, and its corresponding $f(x)_{\text{act}}$. If we were to modify $m$ and graph it versus the error $\lambda$, we would see the resulting graph section 2.2.

SGD based optimizer functions find the gradient of this graph with respect to the optimization parameters (just $\frac{\partial \lambda}{\partial m}$ in this case). From this, they are able to accurately guess new parameters ($m$) in the direction of the gradient which will provide a lower error for our paticular test $x$ and $f(x)$ pair.

This process is iterative, and the change in the parameter per iteration is called step size. Too large of a step size, and new parameter would 'overshoot'. For instance if our $m$ was too small, 'overshooting' would lead to it being too large.

Figure 1: M vs. Error



## 2.3 Caveats

Of course, the model isn't as simple as linear regression. With hundreds of thousands to millions of possible data points, some are bound to disagree on how to adjust parameters. In addition, not all critical points where minima occur

are absolute minima, possibly leading to the optimizer getting 'stuck' in a local minima. Sometimes optimizers have issues 'settling' in the lowest valley, often requiring lower learning rates.[5] Going in the direction opposite the gradient isn't rocket science. The differences between SGD optimizers mainly concern how work around these caveats while navigating the $\lambda$ graph [5, 6, 7, 8].

# 3 Sentiment Analysis

In this project, "sentiment analysis, also referred to as opinion mining, aims to automatically extract and classify sentiments, opinions, and emotions expressed in text" is performed on the Primary Emotion Data created by Lowri Williams [2]. The purpose of Primary Emotion data set is to show the use-fullness of idioms in determining the emotion expressed in a given text. For example, an idiom might be "we'd fight like cats and dogs"; from this sentence we can deduce that whomever is speaking would have awful fights with someone else [2]. The labeled sentences were determined by contributors examined the sentences, rating their emotional content based on Plutchik's Wheel of Emotions and polarity categories (discussed in more detail below) [2]. Sentiment classification problems often use polarity categories to represent and classify sentiment [2]. Sentiment polarity can be positive, negative, and neutral. An example of sentiment polarity in the data is given by "Mr.Jones was grinning from ear to ear"; that is a positive idiom; "I was bored to tears at work today" is a negative statement. After deciding on the polarity it is very simple to then pick an emotion for the sentence [2].

# 4 Plutchik's Wheel of Emotion

Plutchik's Wheel of Emotion, (fig. 2), has eight basic emotions including joy, trust, fear, surprise, sadness, disgust, anger, and anticipation. Each of the eight emotions has three levels of activation represented by color boldness. The emotion space is represented so that combinations of basic emotions derive secondary emotions. For example: joy+trust=love, anger+anticipation=aggression [9].

The emotions on each leaf range in intensity. The emotions on the inner circle are the most intense and the emotions on the outer circle of each leaf are the least intense [10]. The emotions that fall between each leaf are a combination of each leaf's middle emotion [10]. For example, "Trust + Fear = Submission". The Primary Emotion Data we use in this project has emotions from the middle of each leaf, the combinations emotions, and categories "Neutral" and "Ambiguous" [2].


Figure 2: Plutchiks Wheel of Emotion [10]

# 5 Primary Emotion Data

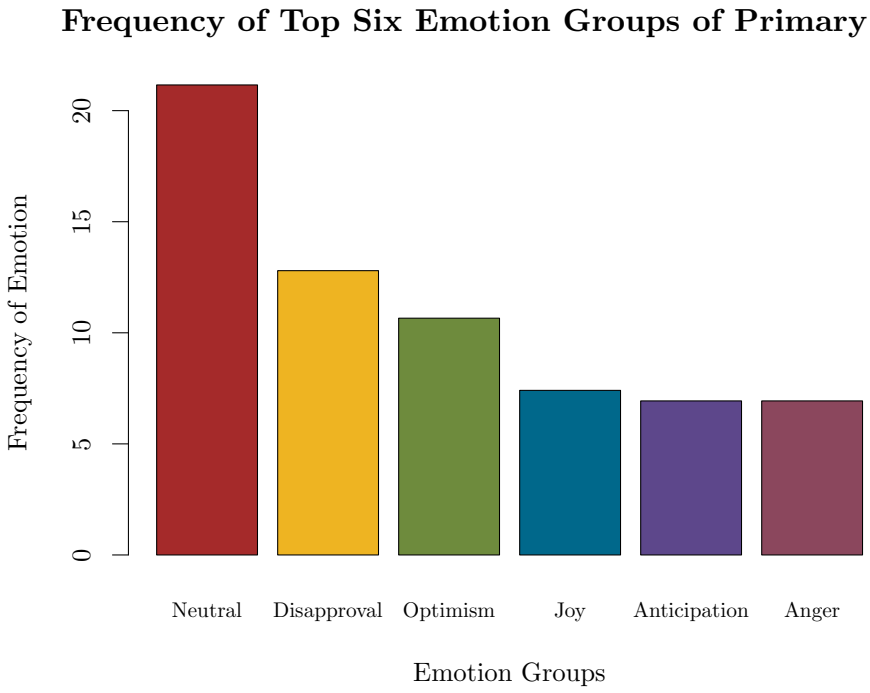
Table 1 gives a few examples of the sentences and labeled emotions in the data set which we will call the Primary Emotion data. The Primary Emotion data set contains around 2425 emotion labeled sentences. For this project the data will be used to develop a model which, when given a text, will assign an emotion to the text.

Table 1: Primary Emotion Data

| Emotion | Sentence |
|---|---|
| Neutral | Don't let him pick a fight now, we're almost home. |
| Optimism | You must sink your differences. |
| Aggression | We'd fight like cat and dog. |
| Disgust | For God's sake bury the hatchet. |

Williams hypothesized that the use of idioms in sentiment analysis would increase the accuracy of predicting the correct emotion or category [2]. Figure 3 shows a bar plot of the top six emotion groups, ranked by most frequently appearing groups in the data set. The emotion group "Neutral" occurs the most, it comprises just over 20 percent of the data set.

Figure 3: Top Six Emotion Groups




# 6 Methods

## 6.1 Text Mining

The goal of this project is to implement a method which when given a bit of text will sort the text into one of the emotions categories. The first step in this project is to do some basic text mining. Text mining is the process of "cleaning" a data set most raw data will contain punctuation, capitalizations, quotes, tenses, and much more [11]. For most text analysis, not all words are needed. Words like "the", "a", "an", and others, known as "stopwords", are not important to build a model which will assign emotion or predict a star rating [11].

Once the data is clean, each sentence must be broken down into "tokens". A token can be each sentence, each word, or even a set of words [11]. For this project each token represents a word in each sentence.

## 6.2 Term Frequency Matrix

The next step in the project is to create a term frequency matrix. Specifically, the term frequency matrix is a matrix with columns corresponding to the emotions we want to detect, and rows representing each word; the matrix itself will contain the number of times a word appears in the assigned emotion category. Since the data set contains over six thousand words, the frequency matrix will be large.

## 6.3 Adam Optimizer

The Adam optimizer is the de facto successor to a RMSProp [6]. In RMSProp initial values must be chosen for some variables in the optimization algorithm [6]. This can lead to bias. The Adam optimizer improves upon RMSProp by correcting this bias in later calculations. In addition, while RMSProp calculates its effective step size using momentum, Adam implements a 'Signal to Noise' method, changing it based on a smooth gradient moving average (moment estimation, $m_t$) and a moving variance estimation ($v_t$). This means the rate at which it navigates the gradient is a metaphorical function of confidence: the less steady the gradient, the slower it goes. The Adam optimizer has proved to produce consistently excellent results, therefore it was chosen as the benchmark optimizer for this project.

Algorithm 1 gives a pseudo-code representation of the Adam optimizer (Algorithm 1). Refer to Table 2 for an explanation of the components and notation in the description of the optimizer.

**Gradient** - The gradient of a function $\nabla_\theta f(\theta)$ evaluated at point $\theta$ is by definition the direction of steepest ascent at point $\theta$. We use this, as we are trying to reduce our inaccuracy/loss by moving in the direction of steepest *descent* (i.e. $-\nabla_\theta f(\theta)$).

**Gradient Moving Average** The Adam optimizer does not just naively use the gradient itself, but instead smooths it using the exponential moving average as originally described by J. Stuart Hunter[12]. This formula gives a much less noisy curve to descend, and helps prevent the optimizer from plateauing in local minima. Although used recursively here, a non recursive equivalent is given by

$$m_t = \sum_{t=0}^{t_f} (1-\beta_1)\,\beta_1^{t_f-t} \nabla_\theta f\left(\theta_t\right)^2 .$$

()Although purely semantic, in the original EMA formula, $\beta_1$ and $(1-\beta_1)$ are switched).

Table 2: Glossary of notation used in Adam optimizer

| | **Description** | **Elaboration** |
|---|---|---|
| $f(x)$ | Objective function | The term to be minimized. This is not the model function. If $g(x)$ function we wish to optimize, and $\lambda$ is our loss function, $f(x) = \lambda \circ g(x)$. See eq. 1 for clarification |
| $\alpha$ | Step Size | Size of increment of change of parameters per $t$. |
| $t$ | Time | Integer value to keep track of iteration of optimizer. Pragmatically, the iterator variable of the optimizer for loop. |
| $\odot$ | Component-Wise Multiplication | $[a, b, c, ..] \odot [d, e, f, ...] = [ad, be, cf, ...]$, When a non scalar $n$ value is raised to a power $p$, it implies $n \odot n^{p-1}$ |
| $m_t$ | Moment Estimation | Moving average of gradient – a smoothed version of the gradient function. Defined by first moment of gradient. |
| $v_t$ | Variance Estimation | Variance of gradient function as defined by second moment of gradient. |
| $\theta$ | Parameters of Model Function | Parameters of model. Earlier denoted $p$ and $m$. |
| $g_t$ | Gradient | Gradient with respect to optimization parameters. Earlier denoted $\frac{\partial \lambda}{\partial p}$, more appropriately denoted $\nabla_\theta f(\theta)$ |
| $\beta_1$ | Decay rate of $m_t$ | Multiplied by previous $m_t$ each iteration($t$) before adding new information.<br><br>Acceptable range [0,1), Default .9 |
| $\beta_2$ | Decay rate of $v_t$ | Multiplied by previous $v_t$ each iteration($t$) before taking into account<br><br>Acceptable range: [0,1), Default .999 |

**Data:** $\theta_0$, $f(x)$ (initial function parameters, unoptimized objective function),
**Initialize variables**: $\{m_0, V_0, t\} = 0$;

**while** *model is not fitted* **do**

$t := t + 1$

$g_t := \nabla_\theta f(\theta_{t-1})$ Find actual gradient.

$m_t := \beta_1 m_{t-1} + (1 - \beta_1) g_t$

Assign new gradient moving average, where previous gradients have weight $\frac{\beta_1}{1}$

$v_t := \beta_2 m_{t-1} + (1 - \beta_1) g_t^2$

Assign a new moving variance estimation in the same manner, but instead using $\beta_2$

$\widehat{m_t} := \frac{m_t}{1 - \beta_1^t}$

$\widehat{v_t} := \frac{v_t}{1 - \beta_2^t}$

Bias results from where we initiated $v_t$ with zero. The last two statements correct that bias. See Adam paper.

$\theta_t := \theta_{t-1} - \alpha \frac{\widehat{m_t}}{\sqrt{v_t} + \epsilon}$

Update the parameters going in the opposite direction of our new calculated gradient average. Weigh update by dividing by $\sqrt{\widehat{v_t}}$, (more noise, lower effective step size.)

**end**

**Algorithm 1:** Pseudo-code representation of Adam optimizer. The := ligature represents assignment.

**Moving Variance** The moving variance is used to modulate the learning rate of the function, depending on how noisy the slope is. For an intuitive sense, recall that standard deviation is given by:

$$\sigma = \sqrt{\sum (x_n - \bar{x})}.$$

If we replace $x_n$ with $\nabla_\theta f(\theta)$, and $\bar{x}$ with a gradient of $\vec{0}$, we get [2]

$$\sigma = \sqrt{\sum (\nabla_\theta f(\theta))^2}.$$

Applying the same exponential moving average to this 'variance from the zero gradient' yields a moving average in the same manner of the moving average of the gradient. Albeit, the variance is more conservative, as $\beta_2$ is typically larger than $\beta_1$.

## 6.4 Model Layers

A different approach was used for each model. Historically, Bidirectional LSTM has shown better results for textual analysis [13]. However, in testing, it performed equivalently or worse than regular LSTM, and significantly increased time to convergence. However, implementing Bi-directional LSTM on the idiom data has yielded a significant boost, raising accuracy approximately 5% over regular LSTM.

## 6.5 Dropout Technique

Dropout is a regularization technique for the Long Short Term Memory Neural Networks. It is a new approach that randomly picks neurons to be dropped from the training network [14]. Those neurons are completely neglected. The goal of using this approach is to keep the neural network from overfitting to the training set. The dropout rate that is used and experimented with is between 20-50% [14]. Dropout is a regularization technique for the Long Short Term Memory Neural Networks. It is a new approach that randomly picks neurons to be dropped from the training network [14]. Those neurons are completely neglected. The goal of using this approach is to keep the neural network from over-fitting to the training set. The dropout rate that is used and experimented with is between 20-50% [14].

## 6.6 Validation Accuracy

Figure 4 shows the change in validation accuracy by epoch, which is the number of times the model has seen the training data set, for a few different

[2] $f(\theta_t)^2 \implies$ component wise multiplication

methods. We used both mono-directional and bi-directional neural nets with up to three hidden layers. As we can see each method starts at approximately the same level of validation accuracy. The bidirectional method with two hidden layers appears to have the most consistent accuracy over the epochs; it levels off after five epochs without much fluctuation in validation accuracy.

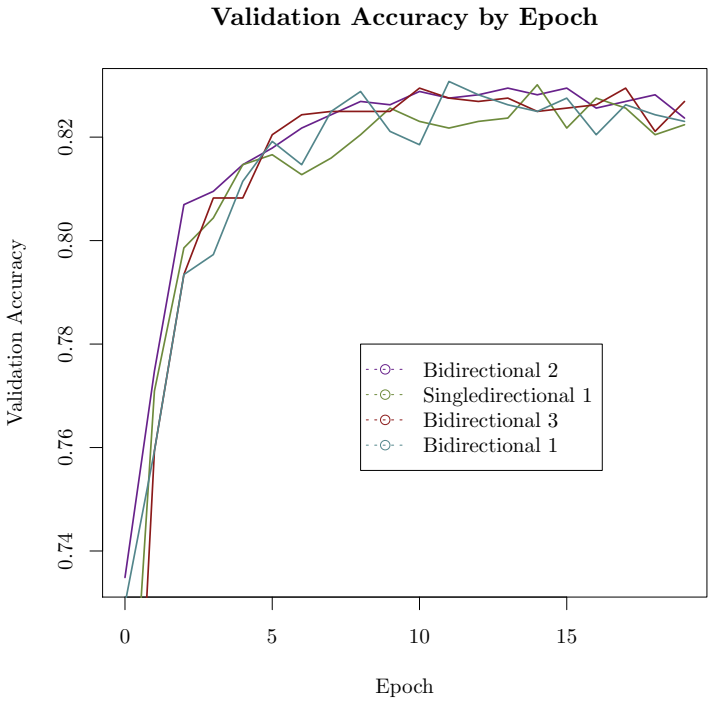



Figure 4: Validation Accuracy by Epoch

# 7 Positive and Negative Groups

For this project, we first ran our method on Primary Emotion Data [2], then we ran it on a sample of labeled Twitter data [15] data to determine if the method would work on both sets of data and accurately predict the underlying emotion of a text. In both cases the data was broken into two groups, positive and negative. Once the data was labeled as positive and negative, the data was then split into a training set and a testing or validation set. Next the data goes through the LSTM method which is programmed to go through each element of the data set and learn the appropriate response variable. In this project the LSTM network loops through each sentence and the corresponding emotion

category and then continues through each sentence. This LSTM network learns the proportion of each emotion category. Once the LSTM network has gone through the entire training data set, it will predict emotion in the test data set. The method will predict the same proportion of emotion categories from the test set as are represented in the training data set.

# 8 Results

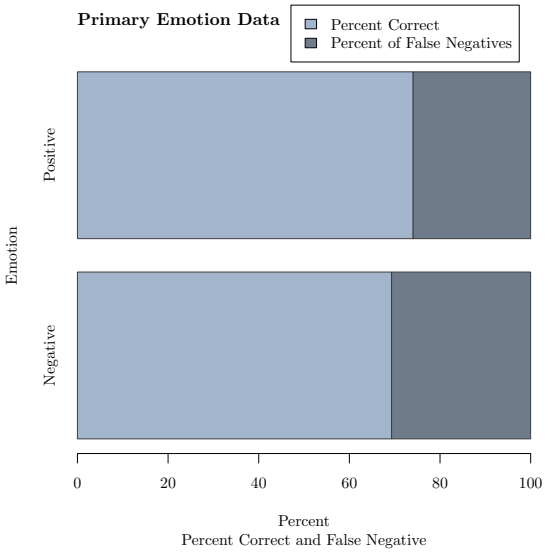
## 8.1 Primary Emotion Data Results

The Primary Emotion Data positive group contained the categories joy, love, optimism, trust, and awe; in the negative group we have anger, disgust, sadness, aggression, contempt, disapproval, and remorse. The rest of the emotion groups are disregarded since they do not fall into either positive or negative categorizations.

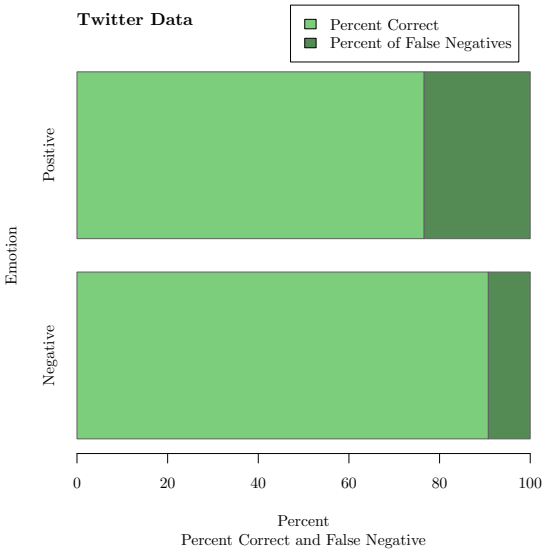
The results using this model on the training set, which was made up of 80 percent of the data, was an accuracy of around 89 percent. The testing set was composed of the remaining 20 percent of the data and had an accuracy of 74 percent. The gap in the training and testing accuracy suggests that the method is over-fitting to the training set. Put more simply, the method is learning the patterns in the training data set rather than learning the overall dynamics of the data. Consequently the method can accurately predict the training set, but it makes less accurate predictions for data it has not yet seen.

Figure 5a shows a barplot of both positive predictions and negative predictions for the Primary Emotion Data testing set. The light blue shows the percent of correct predictions for both categories. The dark blue bar indicates the percent of False Negatives, or sentences which should have been classified as positive for the positive bar, or negative in the negative bar. For the Primary Emotion data the model is most accurately predicting the positive emotions; meanwhile it isn't doing quiet as well with the negative emotion group. We would prefer more accuracy in the prediction of negative sentiment.

Companies want to know which areas they need to improve. A big part of that process is finding out where the company is going wrong with the customer. This information is found in reviews from the customer, whether through some social media website, Yelp!, or reviews posted directly to the company. At some point the company is going to see the comment, tweet, or review and have to analyze it. However a problem with this is the volume of reviews a company, especially a big chain restaurant will receive. A model like the one developed in this project could go through a large set of reviews and tell the company which reviews should be examined more closely, or which reviews are the most important. In other words, the model used in this project could potentially tell a company which reviews are the most negative and should be investigated further.




(a) Primary Emotion Accuracy




(b) Labeled Twitter Data Accuracy

## 8.2 Labeled Twitter Data Results

The Twitter data consists of about $40,000$ tweets, assigned emotion, and user identification. There are some cases of multiple tweets from the same person. The emotion categories are empty, sadness, enthusiasm, neutral, worry, love, fun, hate, happiness, surprise, and boredom. The twitter data was split into positive and negative categories. The positive category contains the emotion groups enthusiasm, love, and happiness. The negative category contains the emotion groups sadness and hate. The other emotions are left out since they do not fall into completely positive or negative categories.

The training set contained 90 percent of the data and had an accuracy of 82.29 percent. The testing set contained the remaining 10 percent of the data and had an accuracy of 83.3 percent. The testing accuracy was actually higher than the training accuracy which tells us the method was not over-fitting to the training data. The testing set contained a higher percentage of the data compared to the Primary Emotion data since the Twitter data is a bigger data set, and 10 of the data was more than and adequate size for testing the method.

Figure 5b is a barplot containing the results of the testing set for the Twitter data. There is a bar for both the negative and positive results. The light green shows the percent of correct predictions for each group. The dark green bar shows the percent of false negatives for each group. In other words, the percent of sentences that should have been predicted to be positive in the positive bar, or negative in the negative bar. From this barplot we can see that the model is doing a great job of predicting negative sentiment in the Twitter data but not as well of a job with the positive sentiment. However the percent of correctly predicted positive sentiment for the Twitter data is about the same as the percent of correctly predicted positive sentiment in the Primary Emotion data. We want the model to accurately find the negative sentiment in tweets, since that is what companies are interested in. Most companies want to know

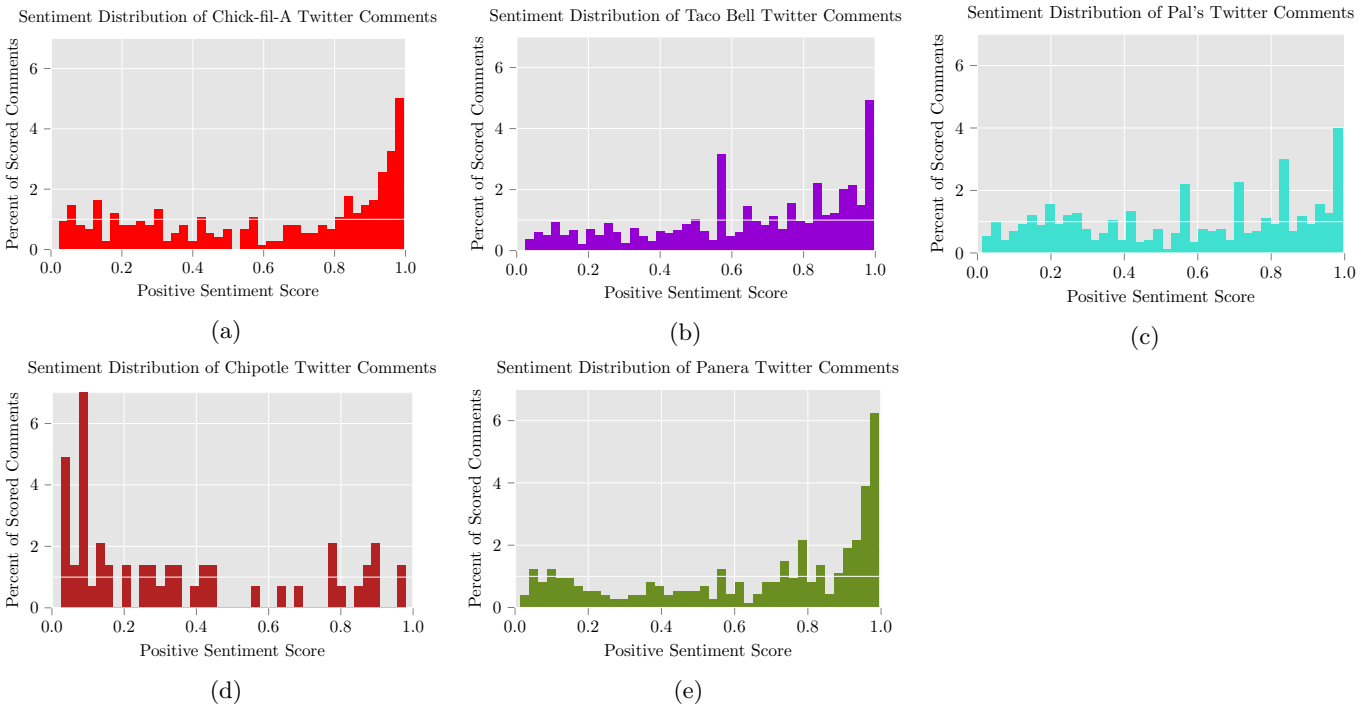



Figure 6: Sentiment Distribution of Tweets at Various Fast Food Companies

which areas they are succeeding in, but they need to know which areas they can improve. If our model can accurately find negative sentiment in a tweet or a comment, we can then make recommendations based on that output about which reviews the company should be taking seriously.

## 8.3 Unlabeled Raw Twitter Data Results

For the final step in this project we tested our model on a few different sets of raw unlabeled twitter data. The data sets were tweets at a few different restaurants. We chose Panera, Taco Bell, Chick-Fil-A, and Pal's. The model which trained on the labeled twitter data, then went through a set of the raw twitter data and for each tweet gave a percent positive score. We then made a histogram of this data. So we have a histogram of "positivity" for each restaurants tweets. First we look at the results from Panera tweets.

As we can see from the Panera Results (fig. 6e). The model predicted mostly positive responses. This is not surprising, Panera has a good reputation for great food and customer service. Obviously there are some less positive or negative reviews, but the majority of the reviews are positive.

Next we look at the results for tweets at the fast food chain Pal's.

From figure fig. 6a we can see that the majority of Pal's tweets have a positive sentiment score, which is good. This histogram is not as heavily skewed as the Panera histogram, so there may be slightly more negative tweets as compared to Panera, but overall Pal's is doing well from the customers standpoint.

Next we look at the results from Taco Bell tweets.

In figure fig. 6b we see that Taco Bell again is mostly in the positive score range, but the histogram does even off around .06. Overall it would appear that Taco Bell is meeting customer expectations.

Finally we look at the results for Chick-Fil-A.

From figure fig. 6a we see a high percentage of tweets with positive sentiment. This tells us that Chick-Fil-A is doing well overall. They are meeting customers expectations. They do have some negative tweets, just as the previous restaurants do, but that is to be expected. No business is perfect, so some less than positive reviews are bound to occur. It is interesting to note that there are not as many middling positive tweets, or tweets that fall in the middle of the graph.

Figure 7: Statistical Comparison of Twitter Restaurants

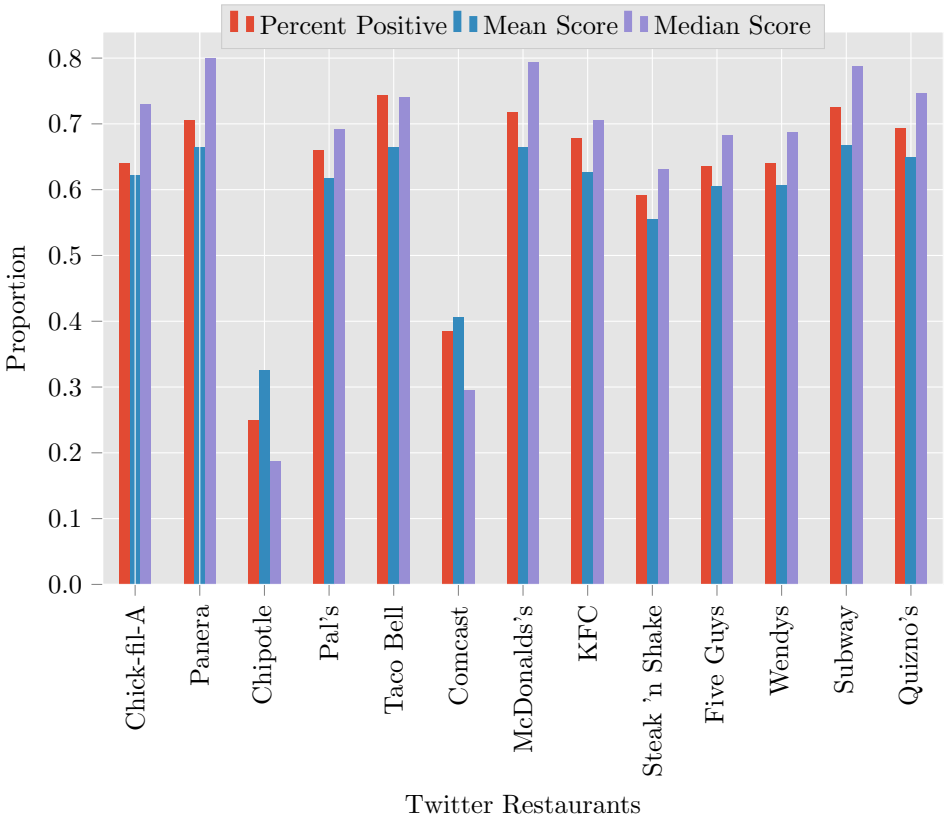
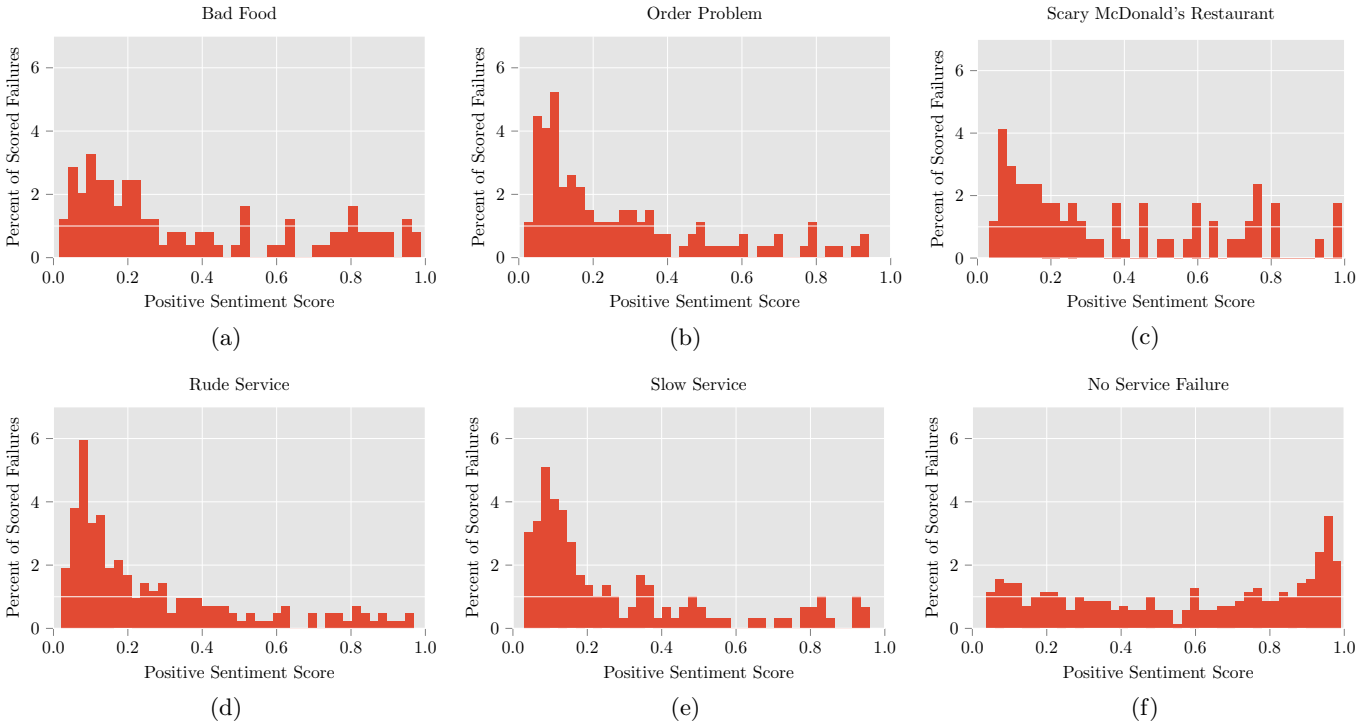


Figure 8: Sentiment of Various Labeled Service Failures




## 8.4 McDonalds Service Failure Sentiment Results

## 8.5 Analysis

# 9 Conclusion

# References


[1] S Azman A. Novak Petra Sparl. "Impact of Customer Satisfaction of Financial Results of Car Servicing Companies: Findings from Slovenia". In: 18.3 (2015), p. 113.

[2] Lowri Williams. "Pushing the Envelope of Sentiment Analysis Beyond Words and Polarities". MA thesis. Cardiff University, 2017.

[3] J. Kiefer and J. Wolfowitz. "Stochastic Estimation of the Maximum of a Regression Function". In: *Ann. Math. Statist.* 23.3 (Sept. 1952), pp. 462–466. DOI: 10.1214/aoms/1177729392. URL: https://doi.org/10.1214/aoms/1177729392.

[4] Michel Blay. "Le traitement newtonien du mouvement des projectiles dans les milieux résistants". fre. In: *Revue d'histoire des sciences* 40.3 (1987), pp. 325–355. ISSN: 0151-4105. DOI: 10.3406/rhs.1987.4061. URL: https://www.persee.fr/doc/rhs_0151-4105_1987_num_40_3_4061.

[5] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *CoRR* abs/1212.5701 (2012). arXiv: 1212.5701. URL: http://arxiv.org/abs/1212.5701.

[6] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014).

[7] T. Tieleman and G. Hinton. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning. 2012.

[8] Ning Qian. "On the Momentum Term in Gradient Descent Learning Algorithms". In: *Neural Netw.* 12.1 (Jan. 1999), pp. 145–151. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(98)00116-6. URL: http://dx.doi.org/10.1016/S0893-6080(98)00116-6.

[9] Robert Plutchik. "The Nature of Emotions: Human Emotions Have Deep Evolutionary Roots, A Fact that May Explain Their Complexity and Provide Tools for Clinical Practice". In: *American scientist* 89.4 (2001), pp. 344–350.

[10] Hokuma. *Plutchiks Wheel of Emotion*. 2017.

[11] Ted Kwalter. *Text Mining in Practice with R*. Wiley, 2017.

[12] J. Stuart Hunter. "The Exponentially Weighted Moving Average". In: *Journal of Quality Technology* 18.4 (1986), pp. 203–210. DOI: 10.1080/00224065.1986.11979014. eprint: https://doi.org/10.1080/00224065.1986.11979014. URL: https://doi.org/10.1080/00224065.1986.11979014.

[13] Keras Documentation. *K Keras*. 2018.

[14] Micheal A. Nielson. *Neural Networks and Deep Learning*. Determination Press, 2015.

[15] *Sentiment Anaysis: Emotion in Text*. 2016. URL: https://www.figure-eight.com/data-for-everyone/.