Sentiment Analysis in Text

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

Fall 2018

Contents

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 Labeled Twitter data. After illustrating the models performance on the two data sets, a set of emotion labeled data according to Plutchik's wheel of emotion and a sample of emotion labeled Twitter data, we show how the model can be used on unlabeled Twitter data and a data set from McDonalds with labeled Service Failures. This paper concludes with how this model may be useful for businesses.

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.

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 [3]. There are a variety of different machine learning methods, such as Quadratic Discriminant Analysis, Neural Networks, Ridge Regression, Logistic Regression, and much more [3]. In this data set each sentence has an assigned emotion so the method will be

a supervised learning method [3]. 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 [3]. In this situation, the predictor variables are the sentences and response variables are the emotions associated with each sentence [3].

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 [4]. The process is not fundamentally different to regression. There is an algorithm which attempts to find optimal parameters [5] [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 α, and a loss function λ.

2.1 Loss function

The loss function denotes 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). 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 [6], which is a tuned up version of stochastic gradient descent optimization [6]. As it name alludes, stochastic gradient descent (SGD) 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 [6].

From this, if we take any point on the loss function, we are able to extrapolate that if we modify our parameters p (where $p = [m, b, ...]$), 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

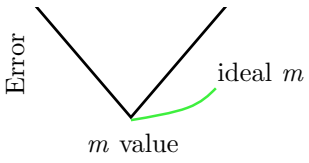
$$\begin{aligned} \lambda &= |f(x)_{\text{act}} - f(x)| \\ &= |f(x)_{\text{act}} - (mx + b)| \end{aligned} \tag{1}$$

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 λ, we would see the resulting graph fig. 1.

SGD based optimizer functions find the gradient of graph fig. 1 with respect to the optimization parameters (just $\frac{\partial \lambda}{\partial m}$ in this case) [7] 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 particular 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 the new parameter could 'overshoot'. For instance, if our m was too small, 'overshooting' would lead to the error 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 [8]. Going in the direction opposite

the gradient isn't rocket science. The differences between SGD optimizers mainly concern how to work around these caveats while navigating the λ graph [8, 6, 9, 10].

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 examining the sentences and 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 inner circle of emotions are the most intense while the outer circle of emotions are least intense. The emotion space is represented so that combinations of basic emotions derive secondary emotions. For example: joy+trust=love, anger+anticipation=aggression [11]. 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].

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

Figure 2: Plutchiks Wheel of Emotion [12]

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.

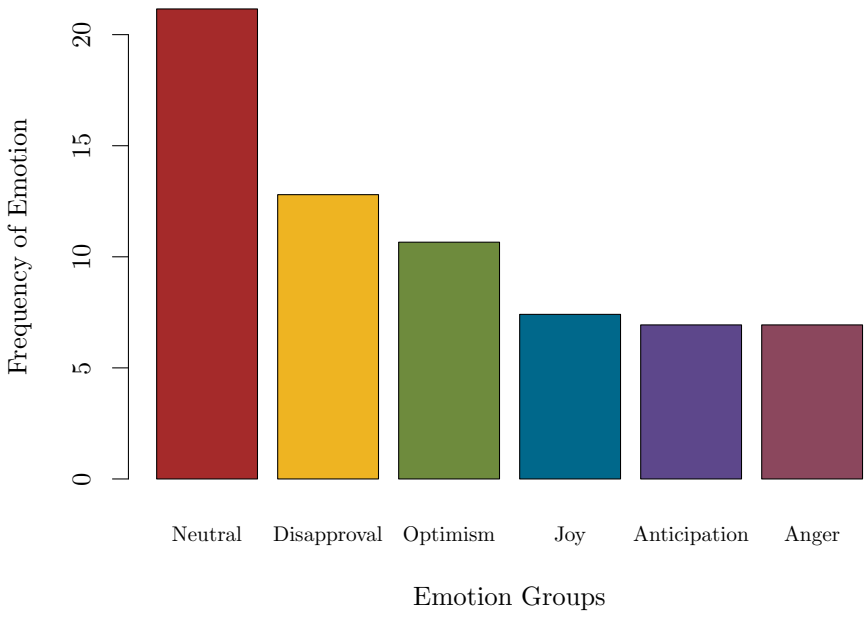
Williams hypothesized that the use of idioms in sentiment analysis would increase the accuracy of predicting the correct emotion or category [2]. Figure 3 shows a bar plot of the top six emotion groups, ranked by most frequently appearing groups in the data set. The emotion group "Neutral" occurs the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

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

Figure 3: Frequency of Top Six Emotion Groups in the Primary Emotion Data



"cleaning" a data set most raw data will contain punctuation, capitalizations, quotes, tenses, and much more [13]. 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 [13].

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 [13]. 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

Building on it, the Adam could be viewed as the theoretical successor to RMSProp [6]. While RMSProp keeps an average square update velocity or 'momentum'[9], Adam keeps a exponential moving or running average of update velocity squared to control the rate at which the gradient is descended, in which old values are decayed and outweighed by new values [6]. The original Adam paper justifies this as a 'signal to noise ratio'.[6].

While theoretically superior, it is sometimes beat out by other optimizers, even its parent optimizer RMSProp. However, the differences are small, and most deficits can be removed by adjusting hyper-parameters. Adam, being theoretically advanced and generally performing well in empirical testing, was chosen as the optimizer for this project. We decisively stuck with it as a matter of practicality [7]. In this project, our model averaged from ten to twenty minutes to fully train. Hence, we chose to focus on developing one optimizer well, as optimizer to optimizer performance among top contenders don't dramatically differ [7].

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

[6]

6.3.1 Further Comments on Components

Gradient - The gradient of a function $\nabla_\theta f(\theta)$ evaluated at point θ is by definition the direction of steepest ascent at point θ. We use this, as we are trying to reduce our inaccuracy/loss by moving in the direction of steepest *descent* (i.e. $-\nabla_\theta f(\theta)$). [14]

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 J. Stuart Hunter.[15] 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, β_1 and 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 λ is our loss function, $f(x) = \lambda \circ g(x)$. See eq. 1 for clarification
α	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.
θ	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 $\nabla_\theta f(\theta)$
β_1	Decay rate of m_t	Multiplied by previous m_t each iteration(t) before adding new information. Acceptable range [0,1), Default .9
β_2	Decay rate of v_t	Multiplied by previous v_t each iteration(t) before taking into account Acceptable range: [0,1), Default .999

Data: θ_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 β_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 it. For an intuitive sense, recall that standard deviation is given by

$$\sigma = \sqrt{\sum (x_n - \bar{x})^2}$$

. 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 it in the same manner of the moving average of the gradient. Albeit, the variance is more conservative, as β_2 is typically larger than β_1[6].

6.4 Model Layers

Although earlier referred to as the objective function, the predictive model is not actually a single function, but instead a composition of functions. Each element function is henceforth referred to as a layer. Diagrammed in fig. 4, our model is composed of three main layers, an embedding layer, a single or multiple (Bi-)LSTM layer(s), and a dense output layer.

Data is transported through the layers in tensors[16]. Tensors are a generalization of linear algebra, matrices and vectors to higher dimensions. The essentials are that, a rank zero tensor is a scalar, a rank one tensor is a vector, rank two is a matrix, and three is a 'three dimensional' matrix, and so on. [17]

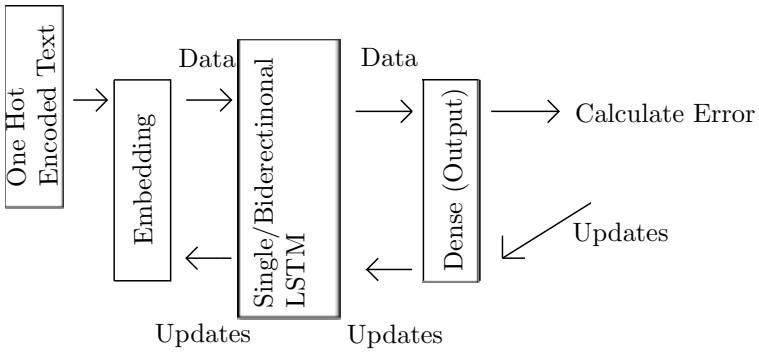
6.4.1 Embedding Layer

The embedding layer takes one hot encoded texts, and returns a collection of words, and yields a tensor composed of 128 dimensional vectors, each of which represents a single word. While it would have been possible to use input one hot encoded vectors directly, using an embedding layer has a couple of unique advantages [19]. Namely, rather than representing words in a discrete manner (i.e. enumeration), the words are viewable on a continuous spectrum [20].

Just as with the other layers, when in training, the inner workings (parameters) of this layer may change. This gives our model the ability of choosing how to represent the words as it sees ideal. Rather than each just being portrayed by a label, the model is able to place the words in a vector-space [21]. This can yield additional information to later layers. For example, it could be possible that semantically similar words would be placed close together, allowing for one to gain benefit from information learned about the other.[20, 21]

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

Figure 4: Model Layers with Data Flow and Back Propagation of Updates [18]



6.4.2 LSTM Layer(s)

The LSTM layer is the brains of the model, which is explained in greater detail in **??**.

While a single LSTM neural networks is only illustrated in fig. 4, two bidirectional LSTM layers were used in when developing a model for the Primary emotion data. This change inspired an increase of approximately five percent validation accuracy. This is not surprising, as historically, multiple Bidirectional LSTM has shown better results than single layer standard LSTMs [19, 22].

However, for the Labeled Twitter Dataset, changing from a single-layer single-directional LSTM to any number of bidirectional layers only lowered performance, and increasing time to convergence. We hypothesize this may be due to Twitter's character count limit, resulting in tweets being less semantically complex than other text documents.

LSTM Neural nets, or Long Short Term Memory Neural Networks are type of recurrent neural network (RNN). RNNs are designed for processing sequentially organized data. In our case this is text. Keeping θ as our training parameter, and x_n as the n^{th} term in our sequential 'array', the change of 'state' s of information of a RNN can be described as

This method recurrence allows for the context of previous information to be used process later information, and allows learning to be shared among states, as the parameters for all states are the same. Moreover, bidirectional neural networks allow information from both directions, allow for context from words later on and previously in a sentence. For a more in depth explanation of

$$n := 0$$

while $n \leq last$ **do**

$$n := n + 1$$

$$s^{(n)}(f(), x, \theta) := f(s^{(n-1)}(...), x_n, \theta)$$

end

RNN and LSTM layers, see Nielson [23] and Graves [24].

6.4.3 Dense Layer

The Dense Layer is the final layer of the model. Our LSTM layer has 196 output cells, but we simply want a single number to define the sentiment of a comment. The Dense layer takes every output from the (last) LSTM layer, and applies a transformation and shift based on what the Adam Optimizer found Ideal during training. It can be expressed as

$$\text{Output} = \text{softmax}(W \cdot I + B)$$

. In the equation above W is our weight tensor, I is our input tensor, and B is a bias tensor that simply shifts the output. While regular dot products are defined to always produce a scalar, tensor dot products produces a tensor who's rank is dependent on the ranks of its parent tensors. In our case, we get in a tensor of rank two, with dimensions of $2 \times b$. The first index corresponds to either positive and or negative correlation of our input, and the second corresponds to the position any particular documents we inputted. [17, 16]

6.5 Dropout Technique

Softmax is our activation function. It modifies our output to from a collection of two vectors with arbitrary scaling of positive and negative correlation to a collection of two vectors that each add up to one. We do desire a probability for our predictions, but simply summing to one does not make this our output a probability in itself. However, combined with our loss function that optimizes for probabilities (as described in section 6.8), our outputs theoretically ought to be one [23, 16].

For any particular non normalized raw input $r(x)$ into the softmax function, with indices $i = 0$ and $i = 1$ representing positive and negative sentiment

respectively, the output would be [23]

$$f(x)_i = \sigma(r(x))_i = \frac{e^{r(x)_i}}{\sum_{k=0}^{1} e^{r(x)_k}}$$

.

6.6 Dropout

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 [23]. 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% [23].

6.7 Cross-entropy Loss

As describe before, the loss function can be anything, it is used to find the magnitude of the error, and to descend the gradient. We used Cross-entropy loss, which is a measure the divergence between two probabilities. It yields how much information would be required to transmit the distribution $f(x)_{\text{act},i}$ in a system perfectly tuned for $f(x)_i$ for all instances i. Cross-entropy It is defined as [23]

6.8 Cross-entropy Loss

As describe before, the loss function can be anything, and is used to find the magnitude of the error, and is used to descend the gradient. We used Cross-entropy loss, which is a measure the divergence between two probabilities. It yields how much information would be required to describe the distribution $f(x)_{\text{act},i}$ in a encoding perfectly tuned for $f(x)_i$ for all instances i. Cross-entropy It is defined as [23]

$$L = -\sum_{n=0}^{i} f(x)_{\text{act},n} \log f(x)_n$$

. For example, in a system where event $f(x)_1$ never occurs, a perfect encoding would never allow for the possibility of expressing $f(x)_1$, as that would translate to inefficiency. Logically the Cross-entropy loss for any actual occurrence (i.e. $f(x)_{\text{act},1} = 1$) of event $f(x)_1$ would be undefined,.

$$L = -1 \cdot \log 0$$

We use this over some other loss function, such as root mean square error, because that we are trying to optimize the accuracy of our *probability* prediction of sentiment being positive or negative. Thus, this method of measuring accuracy is more ideal in our case .

For example, an insurance adjuster who accounts for no probability of a flood for a 20 year period, when there is in fact a 1% probability has made a much worse prediction than one who accounts for 5% probability of flooding when there really is a 6% chance. The root mean square error does not distinguish between these cases [23].

We have two probabilities, sentiment being positive or negative. If $f(x)$ is our algorithms probability of the sentiment being positive, then the algorithm's prediction for the negative probability is $1 - f(x)$.

Therefore our total loss for a single training example is

$$L = -f(x)_{\text{act}} \log f(x) - \sigma - (1 - f(x)_{\text{act}}) \log(1 - f(x))$$

.

6.9 LSTM Neural Networks

For example, an insurance adjuster who accounts for no probability of a flood for a 20 year period, when there is in fact a 1% probability has made a much worse prediction than one who accounts for 5% probability of flooding when there really is a 6% chance. The root mean square error does not distinguish between these cases. Although abstract, one of the most elegant explanations of Cross-entropy loss is as a measure of surprise. [23]

We have two probabilities, sentiment of the comment being positive and sentiment of the comment being negative. Since they must add up to one, If $f(x)$ is our algorithms probability of the sentiment being positive, then the algorithm's prediction for the negative probability is $1 - f(x)$.

Therefore our total loss for a single training example is

$$L = -f(x)_{\text{act}} \log f(x) - (1 - f(x)_{\text{act}}) \log(1 - f(x))$$

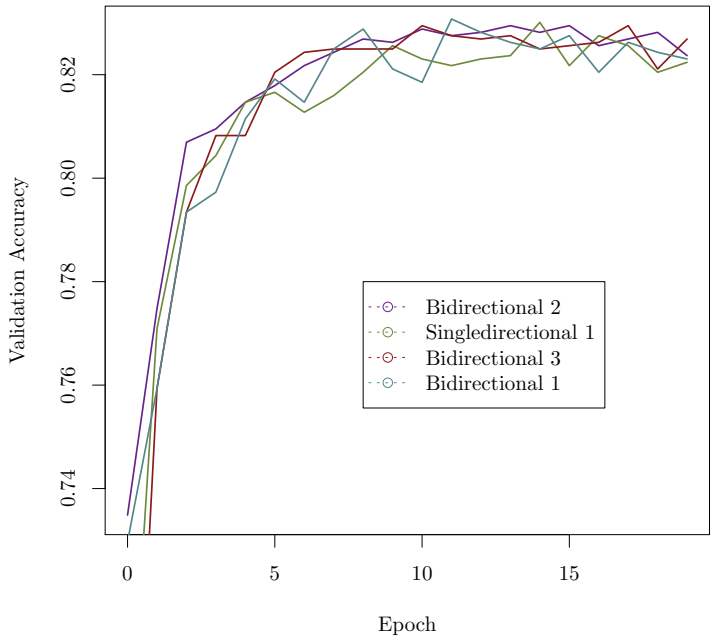
.

6.10 Validation Accuracy

Figure 5 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 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 5: 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 [25], 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 the emotion in the test data set. The model will predict the same proportion of positive sentiment from the test set as is represented in the training data set, and the model will do the same for negative sentiment.

8 Results

8.1 Primary Emotion Data Results

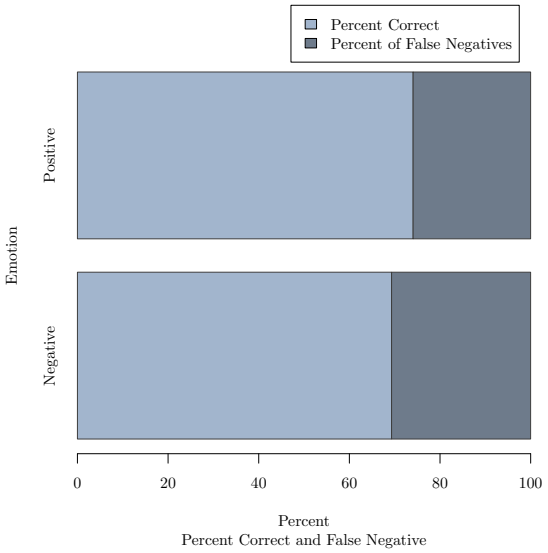
The Primary Emotion Data positive group contained the categories joy, love, optimism, trust, and awe; in the negative group we have anger, disgust, sadness, aggression, contempt, disapproval, and remorse. The rest of the emotion groups are disregarded since they do not fall into either positive or negative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

The results using the predictive 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.

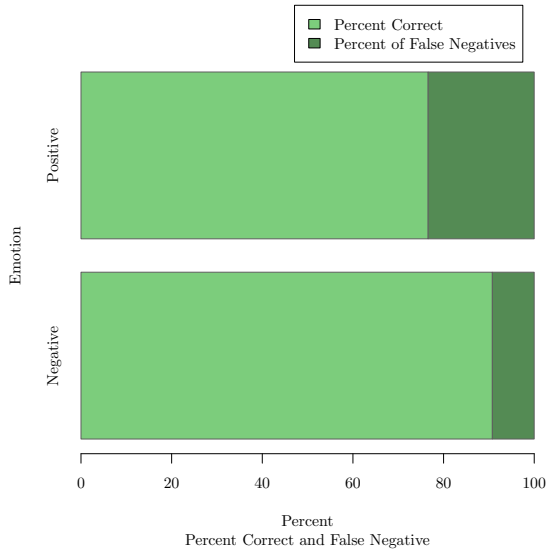
Figure 6a shows a barplot of both positive and negative predictions for the Primary Emotion Data testing set. The light blue shows the percent of correct predictions for both categories. The dark blue bar indicates the percent of False Negatives, or sentences which should have been classified as positive 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.

The accurate prediction of negative sentiment in review is important to companies so they can 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 potentially the most important. In other words, the model used in this project could tell a

(a) Primary Emotion Accuracy



(b) Labeled Twitter Data Accuracy



company which reviews are the most negative and should be investigated further.

8.2 Labeled Twitter Data Results

The sampled Twitter data was collected by Figure Eight [25] and 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, where the positive category contains the emotion groups enthusiasm, love, and happiness, and the negative category contains the emotion groups sadness and hate. The other emotions are ignored since they do not fall into completely positive or negative categories.

Table 3: Labeled Twitter Data

Emotion	Tweet
happiness	mmm much better day... so far! it's still quite early. last day of uds
worry	sucks not being able to take days off of work or have the money to take the trip so sad
hate	dammit! hulu desktop has totally screwed up my ability to talk to a particular port on one of our dev servers. so i can't watch and code
love	@RobertF3 correct! I ADORE him. I just plucked him up and put him under my arm cuz he was cryin. All better now! Hahaha

The training set contained 90 percent of the data and had an accuracy

of 82.3 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 and both values were very close which indicates the method was not over-fitting to the training data. We note that the testing set of the Labeled Twitter data contained a higher percentage of the Labeled Twitter data compared to the testing set of the Primary Emotion data since the Labeled Twitter data is a much bigger data set, and 10 percent of the data was more than an adequate size for testing set.

Figure 6b gives a barplot containing the results of the testing accuracy for the Twitter data; again 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 quiet as well when predicting 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. Most companies want to know which areas they are succeeding in, but they also need to know in which areas they can improve. If the 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 more seriously.

8.3 Unlabeled Raw Twitter Data Results

For the next 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 was trained on the labeled twitter data, then was applied to a set of the raw twitter data and outputted a percent positive score, between 0 and 1, with a score of 0 indicating negative sentiment and a score of 1 indicating positive sentiment. Figure 8.3 shows the distribution of positive sentiment scores for the various restaurants in the Unlabeled Twitter data.

A look at Panera's result (fig. 7e) reveals that 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. From figure fig. 7c we can see 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 tweets. In figure fig. 7b we see that Taco Bell again is mostly in the positive score range, but the histogram does even off around .06.

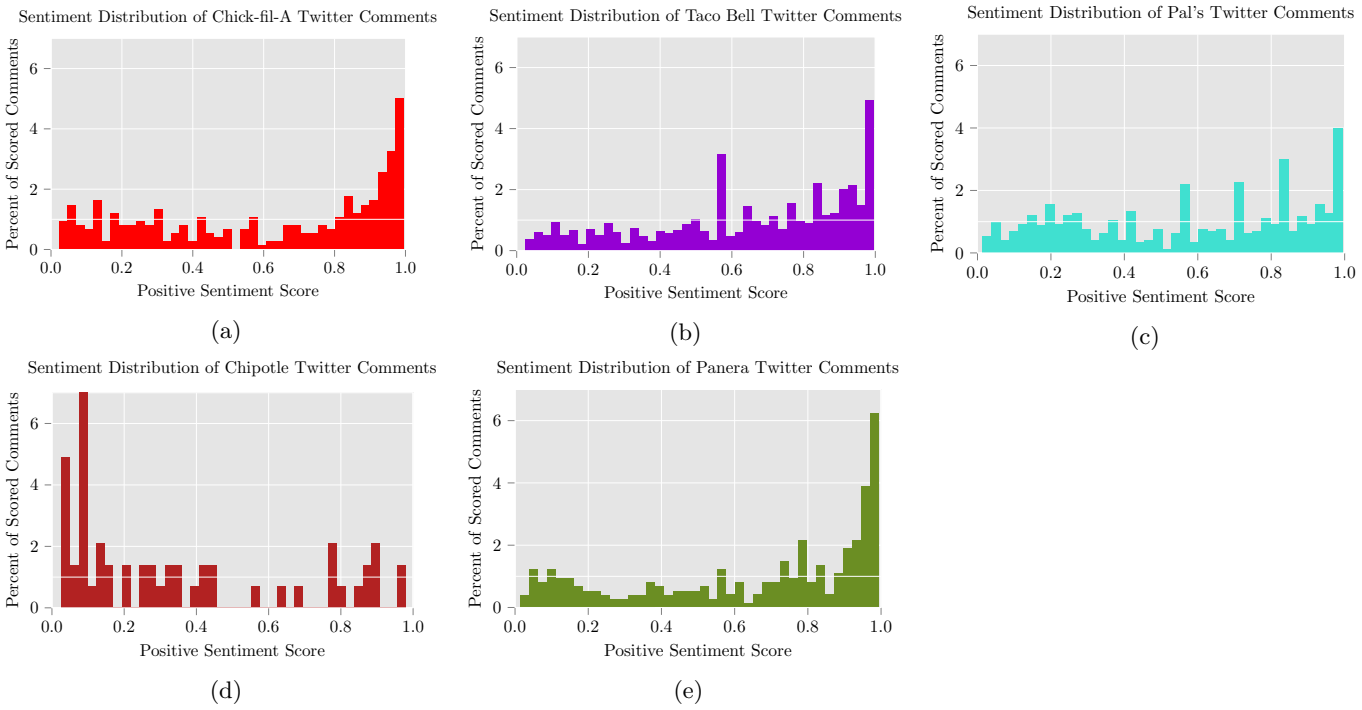


Figure 7: Sentiment Distribution of Tweets at Various Fast Food Companies

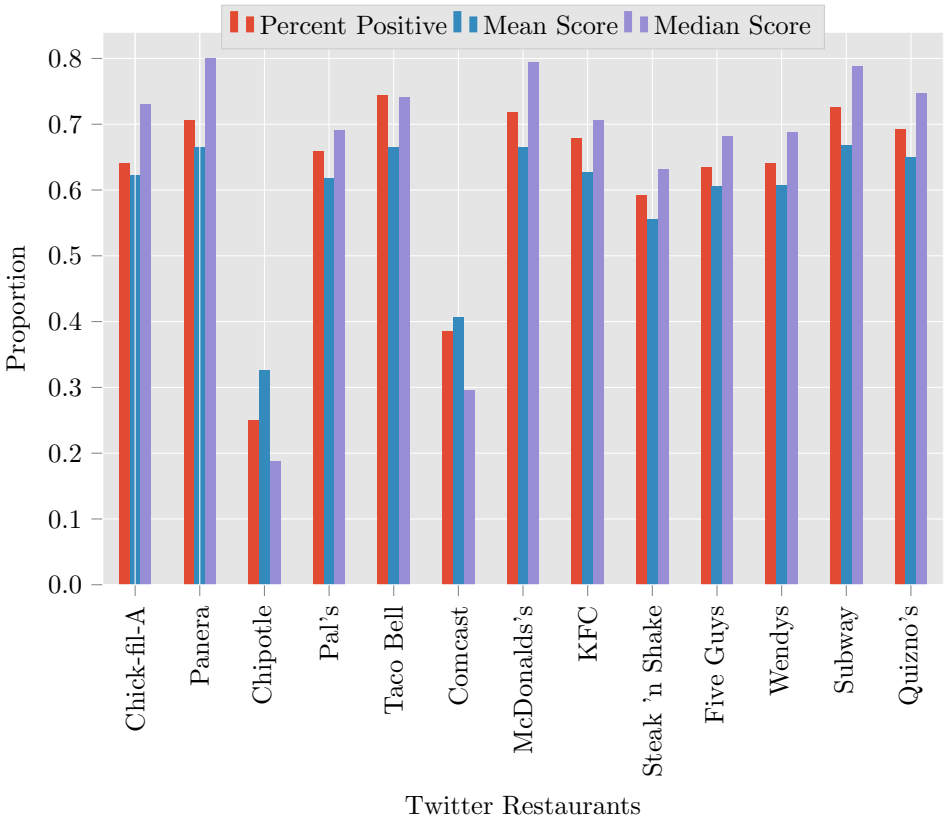
Overall it would appear that Taco Bell is meeting customer expectations. We also looked at Chipotle fig. 7d, which as we can is is pretty negative, this could be due to many factors. Finally we look at the results for Chick-Fil-A. From figure fig. 7a we see a high percentage of tweets with positive sentiment. 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 8 displays some basic statistical analysis on each restaurant from the raw unlabeled Twitter data. The red bar gives the proportion of positive reviews which were over 0.5 on the positive sentiment score, the blue bar gives the mean positive sentiment score, and the purple bar shows the median positive sentiment score for each restaurant. As we can see most of the restaurants are positive, each of the bars falls over 0.5 in this bar graph. The only two exceptions are Chipotle and Comcast, which both have means that are higher than their medians. This is not surprising since we know that a mean will be pulled toward the outliers. Panera appears to have the most positive statistics of the restaurants, with a median score of 0.8.

8.4 Mcdonalds Service Failure Sentiment Results

The McDonalds Service failure is a labeled data set that contains just over 1500 reviews. The reviews are of Mcdonalds restaurants in different locations

Figure 8: Statistical Comparison of Twitter Restaurants



around the United States. The Mcdonalds data set aims to classify reviews into different service failure categories. Service failures describe an instance when a customer at a given restaurant had an unsatisfactory experience due to something specific with either the food, staff, or location. An example might be a customer enters a restaurant and places an order, however it's over fifteen minutes before the order is filled. This would be an example of an Order Problem. The Mcdonalds data set has six different service failure classification groups, Bad Food, Order Problem, Scary McDs, Rude Service, Slow Service, and a category that contains No Service Failures (NA). Table table 4 gives a few examples of the Mcdonalds Service Failure data.

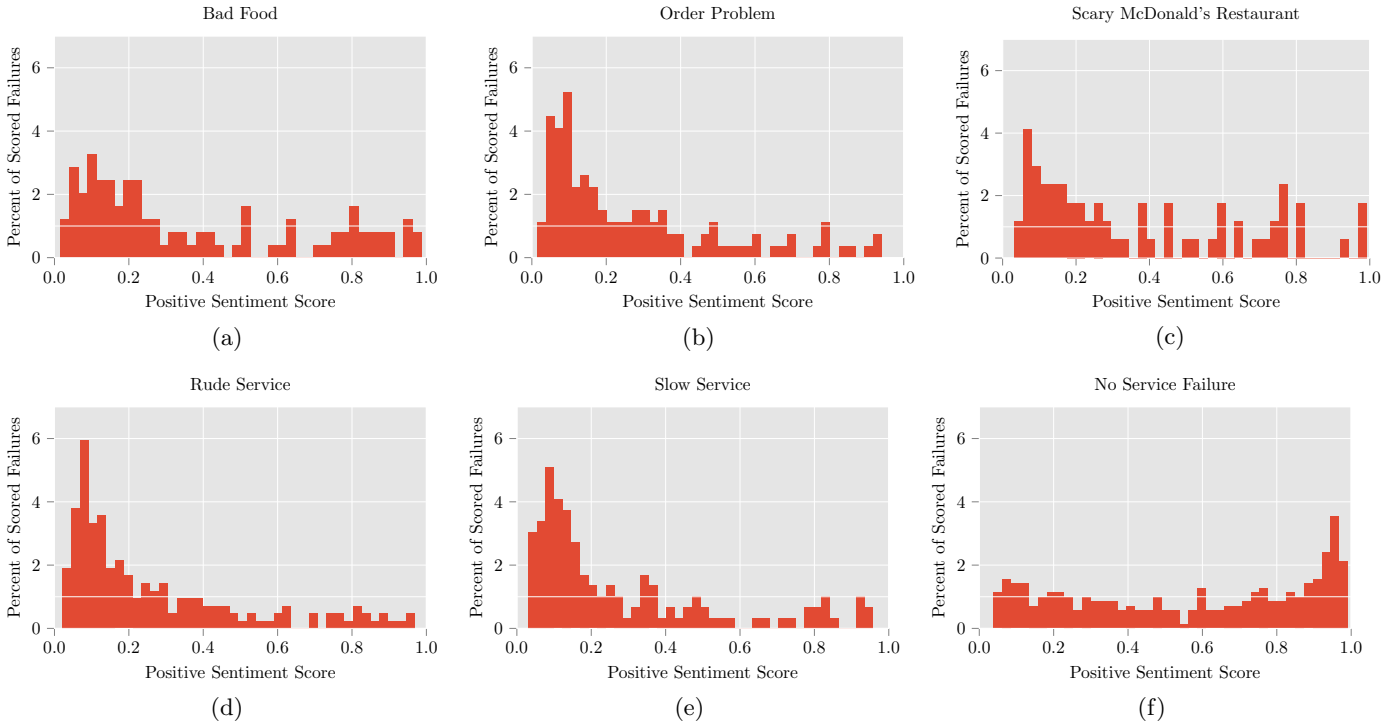
Table 4: Mcdonalds Service Failure Data

Service Failure	Review
na	I see I'm not the only one giving 1 star. Only because there is not a -25 Star!!! That's all I need to say!
SlowServiceScaryMcDs	25 minutes in drive through line. Gunshots from the apartments behind. Worst McDonald's ever!
RudeService	Disorganized. Didn't even get to order. Employees were hanging out with friends out front.
BadFood	So horrible no matter what time of day it is, you'll always get food that has been prepared so messy.

For the purposes of this project, we wanted to see how our model would classify the sentiment in each review. Figure 9 gives the sentiment results of the six different Service Failure groups. As we can see, most of the groups contain negative sentiment, which is what we would expect. The NA group works as a control group, showing that our model doesn't just predict negative sentiment. Rather the NA group has almost a uniform distribution, neither overly positive or negative, it also contains a lot of middling sentiment (review that fall between 0.4 and 0.6 on the Positive Sentiment Score).

The results from Figure 9 show that service failures are expressed in text that contains negative sentiment. Knowing that a service failure will have highly negative sentiment tells businesses which comments they need to investigate. While it's important to know where a business is doing well, businesses are able to improve once they know where they are failing. Not only does this model tell a business which reviews they need to look at to improve, this model also speeds up the process. Most companies could just pay someone or multiple people to read through every review, tweet, yelp! post, and etc, but that is incredibly time consuming and expensive. A model that looks through all that text data and then tells you which texts contain the most important information saves both

Figure 9: Sentiment of Various Labeled Service Failures



time and money.

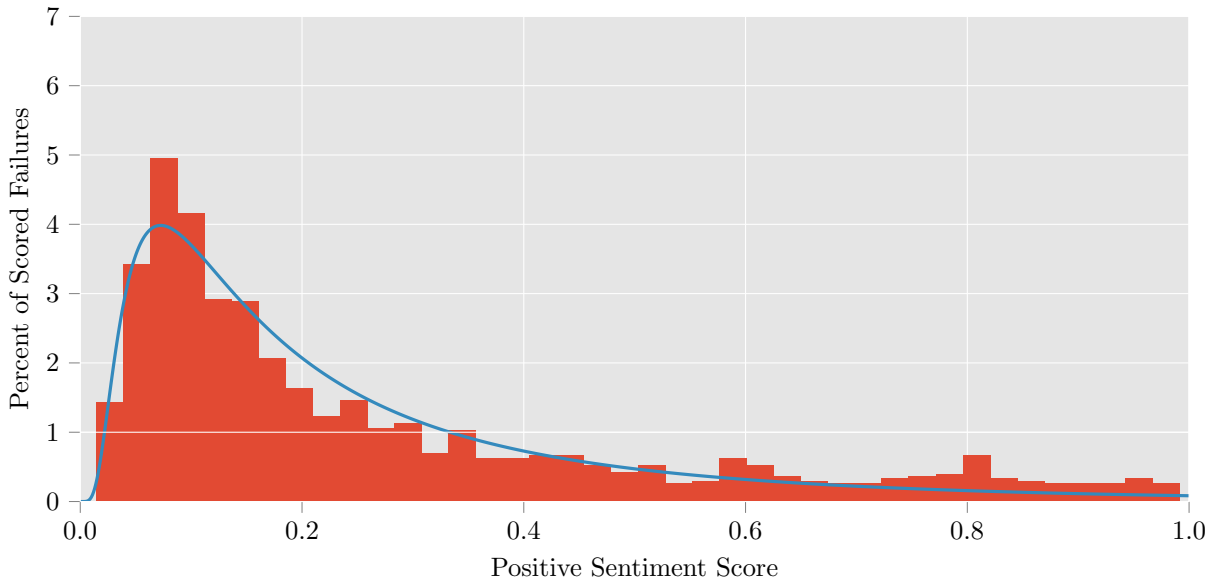
8.5 Analysis

Once we ran the Mcdonalds Service Failure data through our model we were able to develop a function which models the distribution of the service failures. Given in fig. 10a is a histogram showing the distribution of all the Service Failures combined. As we can see it is heavily right-skewed showing mostly negative sentiment, just as we would expect. The blue line shows the function which models the distribution of the service failures. From this model we were able to calculate the Cumulative Distribution Function (CDF), essentially the area under the curve, or a function which gives the area under the curve for any value of x (in this case x is the positive sentiment score). For example, we want to know the percent of service failures that fall below a 0.4 positive sentiment score, we can simply plug in 0 to 0.4 into the CDF and we will get the percentage!

Figure 10b gives the function of the percent of Service Failures which fall under a certain positive sentiment score. So if we want to find 80 percent of all Service Failure reviews, we simply need to look at all reviews that fall under the 0.4 positive sentiment score.

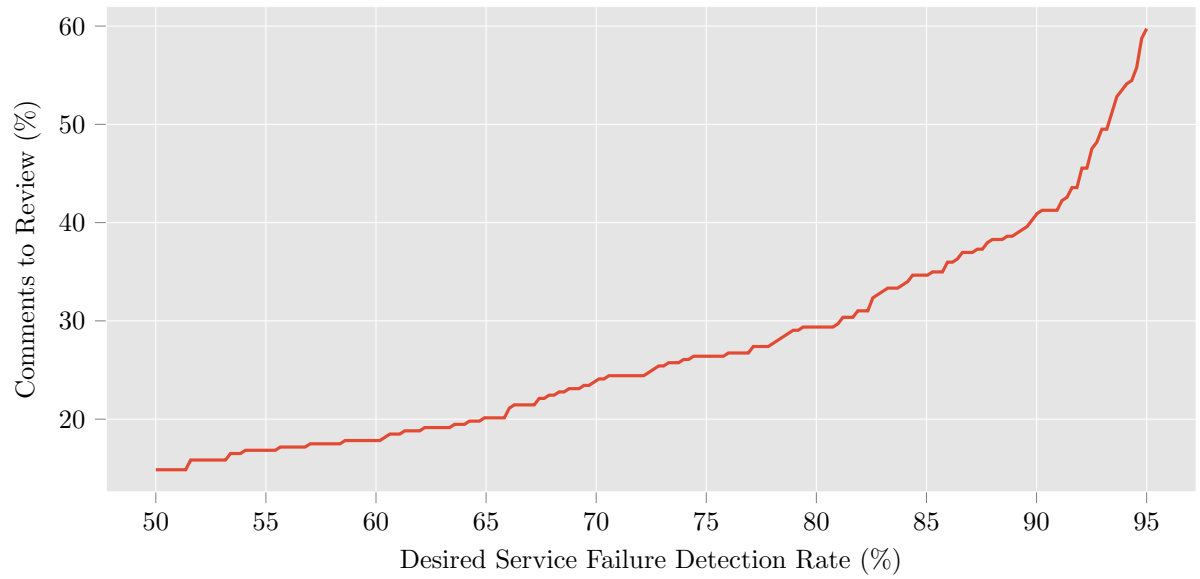
Figure 10c shows the positive sentiment scores that contain the desired percentage of Service Failures. The red line models the Chick-Fil-A Twitter

(a) Sentiment Distribution of All Service failures



(b) Sentiment Vs. Cumulative Service Failure Occurrence

(c) Required Comments to view Catch Service Failures



distribution. While this method of determining the amount of Service Failures within a specific positive sentiment score assumes that Mcdonalds service failures are similar to Chick-Fil-A Service Failures, which may not be the case, this is more simply a proof of concept for how this data and model could be used to help businesses improve.

9 Conclusion

This project developed a model which when given a bit of text, predicted the underlying emotion. The model was trained on one of two data sets, either the Primary Emotion Data [2] or the Labeled Twitter data [25]. The model was tested for accuracy on both the Primary Emotion and Labeled Twitter Data, while our method performed better on the Labeled Twitter data, overall its performance exceeded our expectations. Once the model was fully developed and tested, we ran the model on two unlabeled data sets. First we scraped Twitter for tweets at various restaurants and ran that data through the model. The results of the unlabeled Twitter data showed that most of the restaurants such as, Panera, Taco Bell, and Pal's, contained mostly positive tweets. To test that our model was not biased or just predicting positive sentiment all of the time, we ran tweets at Comcast through the model. The results for Comcast were mostly negative, as we expected, which shows that the model is not unduely biased and does predict negative sentiment.

The final data set we ran through our model was Mcdonalds Service Failure data. The purpose was to determine the underlying sentiment in those reviews. The data was labeled with several Service Failures, we ran each category through the model and again got the expected negative results. The Mcdonalds data did contain one category labeled "NA" for no service failures, this category had results that were uniform accross the positive sentiment score. The NA group served as a control group for the Mcdonalds data ensuring that the model was not only predicting negative sentiment.

Given the results of each data set we were able to develop a function which modeled the distribution of the Service Failure data. Given that function we then found the CDF which enabled us to show the percentage of Service Failures that fall under a given positive sentiment score. With this information we were able to recommend which reviews or tweets are most urgent and contain the most information for improvement of a business. The use of our model for a business would allow that business to not only learn which areas it most needs improvement, but our method would also give this information quickly and accurately.

The business applications of a model like this are numerous. Businesses in the United States run in a competitive market for customers. With reviews on social media and other sites consumers can easily find the ideal business to meet

their needs. These same reviews can also help the business to learn where they are struggling, where they are succeeding, and how to improve. For businesses, quicker improvement times lead to better customer loyalty and in turn better profits.

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] Gareth James Daniela Witten Trevor Hastie Robert Tibshirani. *An Introduction to Statistical Machine Learning with Examples in R*. Springer, 2013.

[4] 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.

[5] 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.

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

[7] Kaifeng Lv, Shunhua Jiang, and Jian Li. "Learning Gradient Descent: Better Generalization and Longer Horizons". In: *CoRR* abs/1703.03633 (2017). arXiv: 1703.03633. URL: http://arxiv.org/abs/1703.03633.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] P. Debye. "Näherungsformeln für die Zylinderfunktionen für große Werte des Arguments und unbeschränkt veränderliche Werte des Index". In: *Mathematische Annalen* 67.4 (Dec. 1909), pp. 535–558. ISSN: 1432-1807. DOI: 10.1007/BF01450097. URL: https://doi.org/10.1007/BF01450097.

[15] 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.

[16] Martıén Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[17] United States. National Aeronautics and Space Administration. *AN INTRODUCTION TO TENSORS FOR STUDENTS OF PHYSICS AND ENGINEERING... NASA/TM–2002-211716... NATIONAL AERONAUTICS AND SPACE ADMINISTRATION.* 2003. URL: https://books.google.com/books?id=-NsqHAAACAAJ.

[18] Jeff R Knisley. *Personal Interview*. Dec. 2018.

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

[20] Yarin Gal and Zoubin Ghahramani. "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks". In: *arXiv e-prints*, arXiv:1512.05287 (Dec. 2015), arXiv:1512.05287. arXiv: 1512.05287 [stat.ML].

[21] Jon Ezeiza Alvarez. "A review of word embedding and document similarity algorithms applied to academic text". In: 2017.

[22] Zhiyong Cui, Ruimin Ke, and Yinhai Wang. "Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction". In: *CoRR* abs/1801.02143 (2018). arXiv: 1801.02143. URL: http://arxiv.org/abs/1801.02143.

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

[24] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Vol. 385. Jan. 2012. DOI: 10.1007/978-3-642-24797-2.

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