# Models for the Multi-label Emotion Classification of Tweets

Steve Hof
*Undergraduate - Computer Science and Statistics*
*University of Victoria*
*stevenhof@uvic.ca*

***Abstract*—The purpose of this project (a directed study at the University of Victoria) was to build machine learning models that could properly classify the emotions present in tweets. In doing so, it was to serve as a learning space for natural language processing, as well as both the theory behind and python implementation of algorithmic classifiers. The classification problem was of an eleven way multi-label structure and proved to be a difficult but rewarding first project in Machine Learning. The project was inspired by a 2017 CodaLab competition and was suggested by Dr. Alona Fyshe, the instructor overseeing the directed study.**

**Several models were built in two general styles: those with a Bag of Words input structure, and those using GloVe embeddings. Both required the use of pre-processing techniques discussed in this paper, and while the Bag of Words models were decidedly simpler and easier to implement, the accuracies achieved by the varying models were surprisingly similar.**

**Final results on the multi-label classification were evaluated by Jaccard Similarity and both micro and macro F1 scores. Five different neural networks were designed using Tensorflow and Keras, and many other models were built using Sci-kit Learn. The accuracies (Jaccard Similarity scores) achieved during this project would not have been awarded a medal, but were not too far away from the projects that did.**

**This paper introduces beginners to some of the tools available and theories behind a few of the models being used in natural language processing research, and strives to show that a complicated problem can be a great choice for a first project in machine learning.**

## 1. Introduction

Twitter is a web based social network specializing in microblogging style text updates. Users upload 'tweets' no longer than 280 characters (formerly 140) along with images, video and other media. Compared with regular blogging, or even Facebook, "microblogging fulfills a need for an even faster mode of communication. By encouraging shorter posts, it lowers the users requirement of time and thought investment for content generation" [1].

Twitter has become especially popular during live events [2] where users log in to share their thoughts and feelings in real time. Both the low thought and time investment, as well as the popularity of tweeting during live events make for user updates that especially represent the user's opinions and feelings on things like products [3], movies [4], [5] and elections [6] and therefore serves as a good choice for classifying emotions.

The motivation for this project stems from an online CodaLab competition [7]. The competition consisted of five subtasks but only subtask five - *Task E-c: Detecting Emotions (multi-label classification)* was attempted here. The specific task was phrased as: "Given a tweet, classify the tweet as 'neutral or no emotion' or as one, or more, of eleven given emotions that best represent the mental state of the tweeter".

As a first project in machine learning, the CodaLab competition provided much of the structure necessary to aid in the completion of the project. Such structure included three direct downloads for training, validation and testing data, the ability to compare results with other competitors to provide both baselines and standards of success, and a wealth of research and results from previous years of the competition. As emotions can be defined and classified in several different ways [8], [9], it was also helpful that the emotions to use as labels were already decided. (see section 2 for more details.)

The remainder of this paper is separated into sections further describing the data used, other projects of a similar nature, an explanation of the methods used for this project and their final results, concluding thoughts and some ideas for future work.

## 2. Data

As mentioned in section 1 the training, validation and testing data for this project was supplied in CodaLab's *SemEval-2018 Affect in Tweets (AIT-2018)* competition [7]. This project only tackles one of the five subtasks of the competition. Therefore, the purpose was not to compete, but (as also explained in section 1) use the structure provided as a solid base from which to complete a first project in machine learning.

The training, validation and testing data contained 6,838, 886, and 3,259 samples respectively. Each sample contained the tweet, a unique ID (not used) and eleven binary labels for emotions, where a zero indicated that emotion was not present and a one indicated it was. The emotion categories chosen to predict for were the eight basic emotions as per

[10] as well as pessimism, love and optimism. The emotion labels and their frequencies (in the 6,838 training samples) are shown in Table 1. Note also that the labels are far from balanced, ranging from *anger* with 2602 occurrences to *trust* with only 357.

The labels for this dataset were all obtained via manual annotation, specifically Best-Worst Scaling. Annotators were given four tweets at a time and asked to rank them in order of how strongly a given emotion was present. Best-Worst Scaling is a relatively new style of comparison annotation but has proven to be very consistent, even with multiple annotators [11].

Although the nature and size of tweets provide emotionally rich messages, their short length makes classification difficult. Fewer words mean fewer inputs for the model to gain information on. In addition, unlike binary or multi-class classification, multi-label classification often requires the prediction of more than one label per sample, thereby exponentially increasing the inherent difficulty of the task. For instance, this task, with eleven class labels, creates more than two thousand ($2^{11}$) possible labels.

A quick scan of the labels, as demonstrated in Table 2 shows that a tweet may be classified as containing several different labels, some of which may or may not be contradictory. In addition, as Table 4 shows, many of the most frequent words in one emotion are also the most frequent in others. Because of this, a heat map (Figure 1 was generated from the labels to see which emotions were correlated with each other and whether or not those correlations made sense. Referring back to Figure 1, it's plain to see that there isn't a great deal of correlation between any of the labels except for anger-disgust and optimism-joy, both of which make perfect sense.

## 3. Related Work

Much research and experimentation has been done on the use of machine learning to classify text. Compared with newspaper articles, movie reviews and entire novels, tweets have a far smaller input space and, therefore, present a unique problem to be tackled. Because of this, the focus of this section will be on research projects done on classifying tweets.

Until recently much of the tweet classification work has been focused on binary or multi-class sentiment classification, with 'multi' often representing only three or four different (ordinal) labels. In fact this is the first year the multi-label emotional classification (E-c) subtask was included as a part of the competition [7]. For instance, an interesting project with a paper titled *EMOTEX: Detecting Emotions in Twitter Messages* achieved a phenomenal accuracy of better than 90%, nearly 30% better than the other multi-class emotion classifiers in this section [12]. However, the project used quite a different emotional classification system called the Circumplex model which considers *Happy-Active*, *Happy-Inactive*, *Unhappy-Active*, and *Unhappy-Inactive* its only categories. While, as the paper points out, the Circumplex model is well supported, its classification categories are too dissimilar for comparison with the task of this paper.

Of the research done on classifying tweets as different sentiment categories (as opposed to an ordinal classification of intensity) or emotions, one notable paper paid particular attention to feature pre-processing [13]. The authors then classified tweets as one of seven different sentiment categories using a Random Forest model and achieved an accuracy of 60.2% (over 80% accuracy was achieved after removing neutral tweets). Another achieved similar results using Naive Bayes, Support Vector Machine, and K-Nearest Neighbours models to classify tweets into one of five different categories [14].

In a 2017 Github project Timothy Liu designed an ensemble neural network comprised of both a Convolutional Neural Network (CNN) and a Long Short Term (LSTM) Recurrent Neural Network (RNN) (both described in Section 4) to achieve a multi-class accuracy of 62%. In his write-up he states part of the motivation for the project was, "Classifying short sequences of text into many classes is still a relatively uncommon topic of research".

As mentioned, however, the task of this paper was multi-label classification, not multi-class. While research has been done on multi-label tweet classification, nearly all of that research employed methods to change the multi-label problem into a multi-class or series of binary classification problems [15].

A study by Colneri and Demsar published shortly after the research portion of this project was thought to have been completed, did serve as a near perfect comparison, however [16]. Colneri and Demsar focused on using deep learning models to classify tweets in up to eight different multi-label categories across three different emotion classification models. Although nearly identical in scope of topic, a comparison of resources available and author expertise does provide a colossal distinction between their project and that of this paper's. Along with some baseline Bag of Words (BoW) models (also discussed in Section 4, Colneri and Demsar had access to 73 billion tweets (more than 17 TB of hard drive space) and trained RNNs and CNNs on over 4.7 million of them while saving nearly 3 million more for validation and testing. Their study achieved macro and micro F scores of over 64 and 67 respectively. For these reasons, Colneri and Demsar's work inspired much of this paper's Future Improvement ideas in Section 6.

## 4. Methods & Results

The simplest way to categorize the methods used during this project is to separate them based on their input methodology. The classifiers developed used either a Bag of Words (BoW) or Global Vectors (GloVe) model to transform the tweets into the numerical values necessary for them to be consumed by machine learning models [17], [18].

TABLE 1. LIST OF EMOTIONS TO CLASSIFY

| Labeled Emotion | Also Includes | Frequency in Training Data |
|---|---|---|
| disgust | disinterest, dislike, loathing | 2602 |
| anger | annoyance, range | 2544 |
| joy | serenity, ecstasy | 2477 |
| sadness | pensiveness, grief | 2008 |
| optimism | hopefulness, confidence | 1984 |
| fear | apprehension, anxiety, concern, terror | 1242 |
| anticipation | interest, vigilance | 978 |
| pessimism | cynicism, lack of confidence | 795 |
| love | affection | 700 |
| surprise | distraction, amazement | 361 |
| trust | acceptance, liking, admiration | 357 |

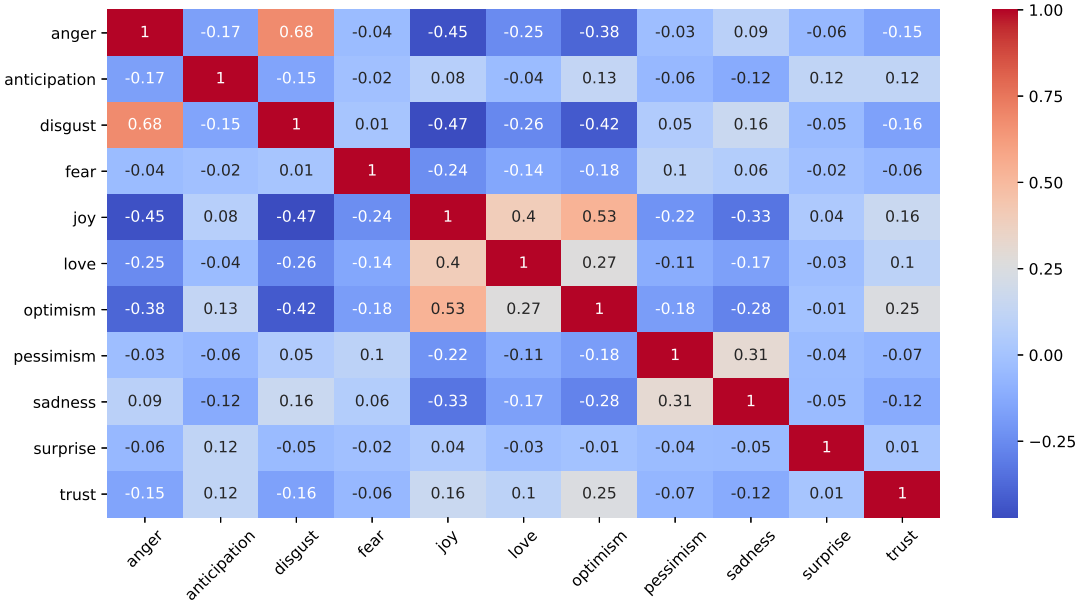



Figure 1. A Heat Map showing the correlation between any two emotions

TABLE 2. TWEETS WITH CORRESPONDING EMOTION ANNOTATIONS

| Tweet | Emotions |
|---|---|
| ”star trek online has a update to download oh fuming yay” | anger, disgust, joy, sadness |
| ”@tampabayray I cried with laughter” | joy, love, sadness |
| ”When will the weeks full of Mondays end?? #disheartened” | pessimism, sadness |
| ”The best revenge is massive success.” | anger, joy, optimism |

## 4.1. Twitter Pre-processing

Tweets have a common format unlike that of common text or even other social media sites. They often make use of *hashtags* (words following the ‘#’ symbol that summarize the topic and can be searched by other users). When mentioning other users on Twitter an ‘@’ symbol is placed in front of their *handle* (their Twitter username). This ensures the tweet will show up in the other user’s feed and is often referred to as *tagging*. In addition to hashtags and tagging, tweets differ from regular (long form) text in that they often contain an abundance of emojis [19]. Emojis are pre-packaged, pictorial versions of emoticons (textual portrayals of emotions or facial expressions in the form of punctuation and other keyboard symbols). Some common emoticon symbols and the emotions / facial expressions they represent are listed in Table 3.

Whether using a BoW or GloVe model, dealing with hashtags, tagging and emojis ahead of time was deemed necessary. A script found on Github [20] was slightly modified then used to go through each tweet looking for ‘#’s ‘@’s and symbols representing emoji eyes and noses. Regular expressions were then used to turn a ‘#’ into ‘hashtag’ followed by the tag itself, remove user handles completely,

TABLE 3. COMMON EMOJIS AND THEIR SYMBOLIC STRUCTURE

| Symbols | Emotions / Facial Expressions |
|---|---|
| :-) :-3 :o) | happy or smiley |
| :-( :-¡ :c | frowning or sad |
| :'( : | crying |
| :-O :O 8-O | suprise or shock |
| ;-) ;^) | winking or smirk |

and turn some basic emojis into the emotion they tend to represent such as 'smile', 'sadface' or 'heart'.

## 4.2. Bag of Words Models

**4.2.1. Pre-processing.** Bag of Words is the simpler of the two input models used for this project. Machine learning algorithms require that any text input be transformed into numbers so the algorithm can perform mathematical operations. BoW models determine all of the unique words in the entire dataset, in this case all of the tweets, and create a column for each word. Each tweet is then represented by a count of the number of times each word in the entire corpus is used in that particular tweet. BoW input models, therefore, do not retain any information regarding context or the ordering of the words in any way.

It is easy to see that large or even medium sized datasets can lead to inputs of enormous dimensions. It is, therefore, necessary to reduce the number of unique words in the corpus as much as possible without losing too much of the information each word carries with it. To that end, this project made use of *word stemming*, removed *stop words* and punctuation, and turned all words into lower case.

Word stemming is the process of removing prefixes and suffixes to trim words to their core meaning. For example, if one tweet contained the words 'worry', 'worrying', and 'worried', the latter two words would be trimmed to 'worry' and the column representing 'worry' for that tweet would be changed to a three.

Stop words are words considered to have low informational value. Words such as 'the', 'r', and 'is' tend not to provide much insight about the theme or emotional state of the tweet. This is especially true for BoW models which don't retain any information about context.

Before the pre-processing mentioned above, the training set contained 24,351 unique words. Once processed that number fell to 16,231; a reduction of more than 8,000 dimensions.

Since BoW bases its classifications on word counts, it was useful to know which words were used most frequently throughout the entire corpus. Figure 2 shows a histogram of those words with their corresponding counts and Table 4 shows the top few words for each individual emotion.

**4.2.2. Models / Algorithms.** While the task of this project was a multi-label classification, to simplify the coding and make use of Sci-kit Learn's (SKL) vast library of machine learning modules (which were utilized throughout the project), the initial focus was to separate each emotion and perform binary classifications with several different modules. In retrospect, this was perhaps misguided because, as discussed in Section 2, many of the tweets are labeled with opposing emotions. The focus for labeling was to include all emotions possibly present in the tweet, not just the most prominent one.

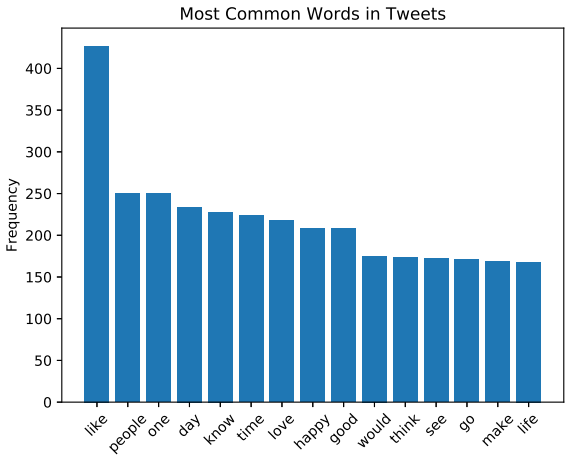



Figure 2. Histogram of most frequently used words across all tweets

TABLE 4. FREQUENT WORDS FOR EACH EMOTION

| Emotion | Most frequent words |
|---|---|
| anger | 'people', 'angry', 'one', 'awful' |
| anticipation | 'know', 'start', 'want', 'really' |
| joy | 'happy', 'love', 'good', 'smile' |
| love | 'love', 'happy', 'smile', 'day' |
| optimism | 'happy', 'good', 'love', 'time' |
| sadness | 'sad', 'depression', 'people', 'one' |
| surprise | 'shocking', 'know', 'watch', 'amazing' |
| trust | 'good', 'god', 'love', 'always' |
| fear | 'fear', 'anxiety', 'nervous', 'nightmare' |
| disgust | 'people', 'one', 'often', 'even' |
| pessimism | 'sad', 'depression', 'know', 'life' |

The models employed from (SKL) during the binary classification attempts included Gaussian, Multinomial, and Bernoulli Naive Bayes models, Random Forest Classifiers, Logistic Regressors and Support Vector Machines. The most successful are shown in Table 5. The *Choose No Emotion* row represents a classifier that classifies every tweet as not containing the emotion in question and will be referred to as the baseline for the remainder of this paper. Table 5 shows that for emotions such as love, surprise, and trust the data is so unbalanced that besting the baseline was nearly impossible.

Keras (https://keras.io/) is a high-level neural networks API capable of running on top of TensorFlow. While using Tensorflow directly allows for the direct manipulation of all aspects of the neural network, Keras simplifies the Tensorflow library by encapsulating layers, pre-processing utilities and other pieces of a network with code that's quick to implement and easier to understand. The Keras API was used to implement all of the GloVe multi-label models

discussed below.

For binary classification, accuracy is a standard and simple calculation:

$$Accuracy = \frac{\text{number of tweets labeled correctly}}{\text{total number of tweets}} \quad (1)$$

Logistic Regression slightly bettered Bernoulli Naive Bayes as the most accurate. It is also worth noting that, although Support Vector Machines are often extremely successful at classification tasks, the one trained on this data simply classified almost every tweet as not containing the emotion in question.

As previously mentioned, multi-label classification is inherently more difficult. In the binary case, the algorithms chose between two labels, and if the data were labeled as multi-class (each tweet labeled as only one of the emotions), the algorithms would have chosen between twelve labels (the twelfth being the absence of any emotion). In the multi-label case the algorithms had 2,048 choices for each a.787, but in the multi-label case the baseline classifier came in with an accuracy of .027 and a classifier that chose randomly managed to achieve .012.

There is no single appropriate definition for accuracy when performing multi-label classification, however, and it is generally accepted that several differing methods should be chosen for comparison. [21]. The CodaLab competition chose to use Jaccard Index accuracy (JI) as its official metric for the emotion classification sub-task, and micro-averaged F-score and macro-averaged F-score as its two secondary metrics.

JI is calculated for each tweet $t$, and then averaged over all tweets in the dataset $T$:

$$Accuracy = \frac{1}{T}\sum_{t \in T}\frac{|G_t \cap P_t|}{|G_t \cup P_t|}$$

where $G_t$ is the set of the gold labels (true labels) for tweet $t$, $P_t$ is the set of the predicted labels for tweet $t$, and $T$ is the set of tweets.

Micro-averaged F-score $(miF)$ is found by first calculating micro-averaged Precision $(miP)$ and micro-averaged Recall $(miR)$ and then combining them as follows:

$$miP = \frac{\sum_{e \in E} \text{tweets correctly assigned to emotion } e}{\sum_{e \in E} \text{tweets assigned to emotion } e}$$

$$miR = \frac{\sum_{e \in E} \text{tweets correctly assigned to emotion } e}{\sum_{e \in E} \text{tweets in emotion } e}$$

$$miF = \frac{2 \times miP \times miR}{miP + miR}$$

where $E$ is the given set of eleven emotions.

Macro-averaged F-score $(maF)$ is calculated as:

$$\text{Precision } (P) = \frac{\text{tweets correctly assigned to emotion } e}{\text{tweets assigned to emotion } e}$$

$$\text{Recall } (R) = \frac{\text{tweets correctly assigned to emotion } e}{\text{tweets in emotion } e}$$

$$\text{F-score } (F) = \frac{2 \times P \times R}{P + R}$$

$$maF = \frac{1}{|E|}\sum_{e \in E} F$$

SKL only offers eight algorithms that support multi-label classification. Four are forms of Decision Trees (one being a Random Forest), two are forms of Nearest Neighbours, and the other two are a Multi-Layer Perceptron (MLP) and a Ridge Classifier. Only two of the eight achieved a JI over .300. One was the Random Forest and the other MLP.

A Random Forest is an ensemble algorithm composed of several decision trees trained on different subsets of the dataset that are then averaged to give the final decisions. The subsets were sampled with replacement on the training data but, as is the case with all results in this paper, accuracies were calculated on unseen testing data.

An MLP is a simple feed forward neural network. The best results were achieved using a network with two hidden layers of 120 neurons each. Both of the layers were fully connected using the Rectified Linear Unit (Relu) activation function and, and were optimized with a stochastic gradient descent algorithm named Adam. The learning rate was set to a constant 0.0005 and the max number of epochs was set to 200.

Results for both the Random Forest and MLP are shown in Table 6

### 4.3. GloVe Embeddings Models

Unlike BoW vectors, GloVe, short for Global Vectors for Word Representations, does take context into account. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Specifically it trains by comparing the non-zero elements of word-to-word co-occurrence matrices [18]. The idea is that each dimension of the vector represents a measurement of the difference between words. The details of the algorithm are far beyond the scope of this paper, but a commonly used example to help visualize how the algorithm works is as follows: since "king is to queen as man is to woman" is a widely accepted analogy, when these words are represented mathematically, Equation 2 should hold.

$$king - queen = man - woman \quad (2)$$

Pre-trained GloVe embeddings are available in various sizes (number of dimensions) and are trained on different

TABLE 5. BINARY CLASSIFICATION ACCURACIES

| Model / Algorithm | Anger | Anticipation | Disgust | Fear | Joy | Love | Optimism | Pessimism | Sadness | Surprise | Trust |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Gaussian NB | .573 | .664 | .561 | .710 | .603 | .814 | .602 | .737 | .558 | .860 | .855 |
| Multinomial NB | .799 | .820 | .743 | .852 | .814 | .866 | .769 | .842 | .732 | .920 | .919 |
| Bernoulli NB | .769 | .858 | .735 | .827 | .795 | .894 | .758 | .874 | .743 | .944 | .948 |
| Logistic Regression | .808 | 853 | .764 | .903 | .819 | .900 | .784 | .868 | .783 | .950 | .944 |
| Baseline | .630 | .857 | .620 | .818 | .638 | .898 | .710 | .884 | .706 | .947 | .948 |
| Support Vector Machine | .628 | .855 | .631 | .815 | .637 | .892 | .703 | .876 | .708 | .944 | .948 |
| Random Forest | .791 | .851 | .743 | .896 | .786 | .893 | .766 | .862 | .774 | .947 | .940 |

TABLE 6. MULTI-LABEL RESULTS FOR BOW MODELS

| Model / Algorithm | Jaccard Index | Micro F | Macro F |
|---|---|---|---|
| Random Forest | .323 | .450 | .326 |
| Multi-layer Perceptron | .429 | .560 | .443 |

corpuses. For this project, embeddings trained on two billion tweets with a 1.2 million vocab were used, and at that time GloVe twitter embeddings were available in 25, 50, 100 and 200 dimensions. Initially, the 200 dimension vectors were used until it became apparent that for this dataset the higher dimension embeddings led to overfitting. After much trial and error, the 50 dimension vectors were found to perform best and those were used for all the GloVe embedding models / algorithms discussed in Section 4.3.

**4.3.1. Pre-processing.** Aside from the methods discussed in Section 4.1, the pre-processing of GloVe embeddings didn't involve any stemming or stop word removal. All of the tweets in the corpus were left the same but transforming the tweets into numerical embeddings was more involved than for the BoW models.

The pre-trained GloVe embeddings were downloaded as a .txt file and the 20,000 most common words and corresponding embeddings were turned into an embedding dictionary with the words as keys and the vectors as values. Then, similar to what was done in the BoW pre-processing, each unique word in the corpus of tweets was mapped to a corresponding unique integer. Using the two dictionaries an embedding matrix was created where the index of each row represented a word in the total vocabulary and each column represented one of the corresponding 50 dimensions of the real valued GloVe embeddings. The tweets were then turned into a list of lists where each sublist was a sequence of integers that represented the words in one of the tweets. Finally, using built in Tensorflow functions, rank-3 tensors (see Figure 3) were created to use as the input neurons for the neural networks explained in Sections 4.3.2, 4.3.3 and 4.3.4.

**4.3.2. Neural Networks.** As previously mentioned, BoW input models ignore the context and ordering of words in a tweet. With an input model that maintains that information (GloVe), the algorithms / models used for classification also needed to be updated. There is no point in using GloVe embeddings if the model using them isn't capable of using the extra information they contain.

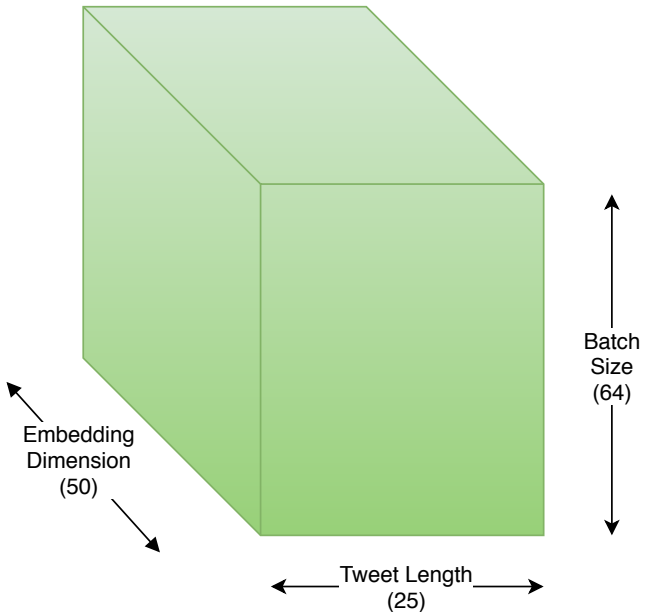



Figure 3. Rank-3 Input Tensor Dimensions

Neural networks come in many shapes and sizes. The most basic is a fully connected **Feed Forward Neural Network** (FFNN) where every neuron in one layer is connected to every neuron in the next. As the inputs move their way through the network from left to right, they are multiplied by *weights* the network learns by repeatedly comparing the ground truth labels with the predicted labels that the network outputs. The difference between the two is called an *error* or *loss*. In a process termed backpropagation, the network works its way back through the layers from output to input adjusting the weights in an attempt to minimize the error that will be calculated the next time through the network.

Other networks are defined by adjusting both a multitude of hyper-parameters and redefining the make up of the neurons themselves. This paper briefly discusses three such networks: Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and Long Short Term Neural Networks (LSTMs).

**CNNs** are often used for image classification and computer vision. They work by taking an input matrix (often a matrix of pixels but in this case a matrix of word embeddings) and running a series of filters over its partitions to extract the most important elements in each. From layer to layer this process continues to break up the input matrix into

its most essential pieces. A *pooling* layer combines these pieces into one feature vector that is then fed through a softmax (or some other function) to produce the network's output.

Because CNNs break the input matrix into smaller and smaller pieces with each successive layer, they are eventually (depending on the size of the filter) only taking into account elements that occur very close together. With images this tends to make sense because pixels that are right next to each other are likely to be related to each other (ie. part of the same object in the image). With text, however, this isn't always the case. For instance, tweet number 960 in the dataset is "Refuse to let myself get discouraged". The words 'Refuse' and 'discouraged' are essential to classifying its emotion but they are not close together in the sequence, and a CNN would likely misclassify the tweet as containing fear, pessimism or sadness when the ground truth labels are joy, optimism and trust. For this reason, CNNs are rarely used as the sole type of neural network in natural language processing problems. Rather, they are used in tandem with RNNs and LSTMs to aid against overfitting and help handle inputs of varying length [16].

CNNs were employed early on this project with extremely poor results and were, therefore, written off as a poor choice. Since reading [16], however, CNNs will be an import piece of any future work done on this project as discussed in Section 6.

Unlike CNNs, **RNNs** were specifically designed to handle data such as time series and text where the ordering matters [22]. In a Feed Forward model the network assumes all inputs, outputs and all calculations in between are independent of one another. This obviously isn't the case for classifying tweets. The order and arrangement of words in a tweet matter a great deal when classifying the underlying emotions. The word *recurrent* is used because RNNs execute the same operations on every element of a sequence and are able to remember the information contained in words from earlier in that sequence. They do so by maintaining a hidden state that accumulates the information in a sequence (tweet) from start to finish. Because RNNs execute the same operation at each layer of the network they can be trained much more efficiently than FFNNs which have a different weight to learn at every cell. This is great for training times but can lead to a different problem called the *vanishing* gradient.

As mentioned above, a neural network *trains* or *learns* the weights it should use during a process called backpropagation. Backpropagation works by calculating the partial derivatives of the loss function with respect to each of the weights in the network in order to find the gradient or optimal direction for adjusting those weights. In a FFNN this process works well because while calculating the derivative with respect to one weight, the other weights are treated as constants. In an RNN, however, because each hidden state depends on the same weight calculation, the chain of derivatives to calculate are dependent on one another and, therefore, are not held constant. While there is much more to this than what has been explained here, suffice to say this causes the gradients to quickly tend toward zero, thereby limiting the amount of training that can be done on the network. To overcome this hurdle, a specific type of RNN called an LSTM is often used in natural language processing.

Instead of having one hidden layer state that accumulates information, **LSTMs** have four [23]. These four layers in the hidden state make use of gates to limit the amount of information that is preserved between elements in the sequence. The weights used in the functions that decide which information to keep and which to forget are then learned over time by the network. In this way LSTMs allow for sequences of much greater length to be fed into the network without it falling victim to the vanishing gradient.

**4.3.3. Binary Classification GloVe Models.** Similar to the BoW models, the first GloVe model was built for the binary classification of a specific emotion. The Tensorflow library was chosen to implement what ended up being a multi-layer LSTM. The phrase 'ended up being' is used here because the process involved a great deal of trial and error. As mentioned at the beginning of this paper, part of the motivation for this project was as a learning space for the understanding behind and implementation of machine learning algorithms. Much like SKL, Tensorflow provided an excellent framework for learning and helped to fuel a desire for many more projects in the future.

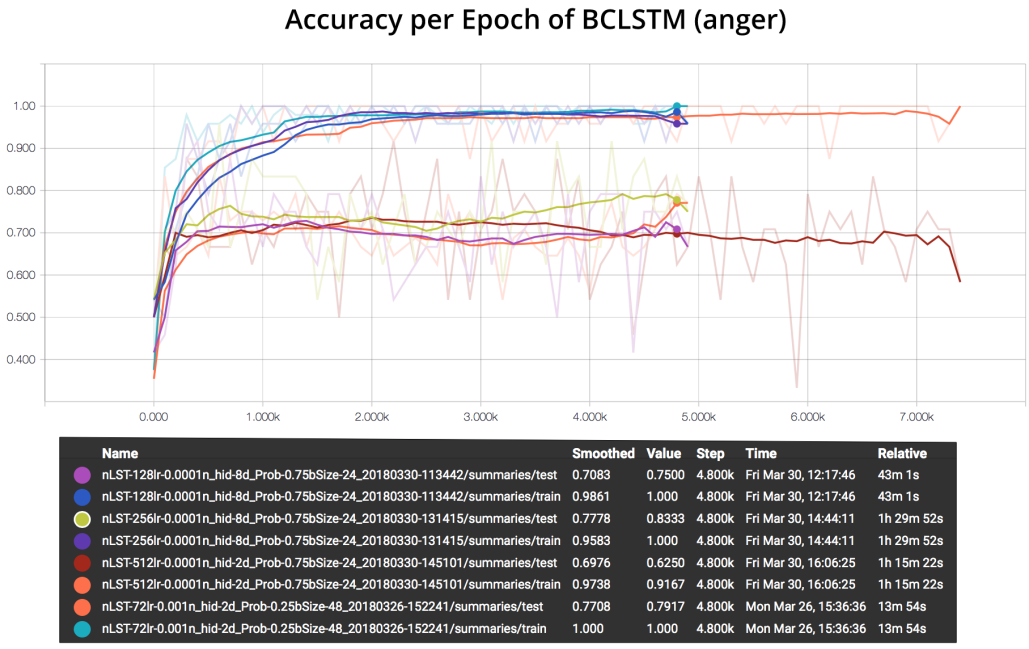
While building the Binary Classification Tensorflow LSTM Model (BCLSTM), numerous different hyper-parameters and settings were tested to find the combination providing the highest accuracy possible. Following is a list of some of the configurations tested on and Figure 4 shows the learning curves for a subset of those configurations. Table 7 then relates the hyper-parameters to their short form in the legend.

- Twitter pre-processing: no, yes
- GloVe embedding dimensions: 25, 50, 100, 200
- Number of LSTM neurons per layer: 12, 24, 48, 64, 72, 128, 256, 512
- Number of hidden layers: 1, 2, 3, 4, 8, 12, 24
- Dropout layer keep probability: 0.2, 0.4, 0.5, 0.75
- Activation functions: relu, tanh, sigmoid, softmax
- Optimizers: sgd, adam

TABLE 7. HYPER-PARAMETER REFERENCES

| Hyper-parameter | Short form |
|---|---|
| LSTM Nodes per Layer | nLST- |
| Number of Hidden Layers | n_hidd- |
| Dropout Layer Keep Probability | dOut- |
| Unseen Testing Data | test |
| Training Data | train |

Figure 4 shows a textbook case of overfitting. Somewhere around 500 epochs the BCLSTM's accuracy on the training data spikes toward 100%, while the testing data accuracy plateaus between 70% - 80%. The model was learning characteristics too specific to the training data and




| Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|
| nLST-128lr-0.0001n_hid-8d_Prob-0.75bSize-24_20180330-113442/summaries/test | 0.7083 | 0.7500 | 4.800k | Fri Mar 30, 12:17:46 | 43m 1s |
| nLST-128lr-0.0001n_hid-8d_Prob-0.75bSize-24_20180330-113442/summaries/train | 0.9861 | 1.000 | 4.800k | Fri Mar 30, 12:17:46 | 43m 1s |
| nLST-256lr-0.0001n_hid-8d_Prob-0.75bSize-24_20180330-131415/summaries/test | 0.7778 | 0.8333 | 4.800k | Fri Mar 30, 14:44:11 | 1h 29m 52s |
| nLST-256lr-0.0001n_hid-8d_Prob-0.75bSize-24_20180330-131415/summaries/train | 0.9583 | 1.000 | 4.800k | Fri Mar 30, 14:44:11 | 1h 29m 52s |
| nLST-512lr-0.0001n_hid-2d_Prob-0.75bSize-24_20180330-145101/summaries/test | 0.6976 | 0.6250 | 4.800k | Fri Mar 30, 16:06:25 | 1h 15m 22s |
| nLST-512lr-0.0001n_hid-2d_Prob-0.75bSize-24_20180330-145101/summaries/train | 0.9738 | 0.9167 | 4.800k | Fri Mar 30, 16:06:25 | 1h 15m 22s |
| nLST-72lr-0.001n_hid-2d_Prob-0.25bSize-48_20180326-152241/summaries/test | 0.7708 | 0.7917 | 4.800k | Mon Mar 26, 15:36:36 | 13m 54s |
| nLST-72lr-0.001n_hid-2d_Prob-0.25bSize-48_20180326-152241/summaries/train | 1.000 | 1.000 | 4.800k | Mon Mar 26, 15:36:36 | 13m 54s |

Figure 4. Plot showing the BCLSTM overfitting at a variety of hyper-parameter combinations

wass, therefore, unable to properly classify unseen testing data.

The accuracies shown in Figure 4 all relate to classifying tweets as containing anger or not. The highest accuracy achieved by a BoW model on the anger emotion, as previously shown in Table 5 was the Logistic Regression model at an accuracy of .808. This was a trend that persisted throughout the binary classification, with the BCLSTM model always coming in at an accuracy slightly less than the Logistic Regression model, no matter which emotion was being classified.

**4.3.4. Multi-label Classification GloVe Models.** All of the multi-label GloVe models were implemented with Keras. Four types of neural networks were deployed: Feed Forward, Bi-directional, Convolutional, and LSTM (the term Keras-LSTM will be used to avoid confusion with the BCLSTM). As previously mentioned, the CNN performed quite poorly and the Bi-directional network performed even worse. So poorly, in fact, that it was outperformed by two dummy classifiers. The first dummy classifier chose labels randomly and the second always predicted the class with the most labels. The Jaccard Indexes of both dummy classifiers were well under .001, and since the Bi-directional network performed even worse, it is not mentioned in the descriptions in Section 4.3.2.

As with the binary classification network, many hyper-parameter configurations were implemented on a trial and error basis in order to find the combination that produced the highest Jaccard Index. While these hyper-parameters are similar to those used for the BCLSTM, they are listed below none the less:

- Twitter pre-processing: no, yes
- GloVe embedding dimensions: 25, 50, 100
- Number of LSTM neurons per layer: 6, 12, 24, 48, 64, 72, 128, 256, 512
- Number of hidden layers: 1, 2, 3, 4, 6, 12
- Dropout layer keep probability: 0.2, 0.4, 0.5, 0.6, 0.75
- Activation functions: relu, tanh
- Optimizer: adam

The majority of networks trained for the multi-label task performed terribly, with Jaccard Index scores less than .05. Of interesting note is that the ideal hyper-parameter combinations for both the Feed Forward and Keras-LSTM neural network were extremely similar. Table 8 shows the hyper-parameters both networks had in common.

TABLE 8. IDEAL HYPER-PARAMETERS SHARED BY FF AND KERAS-LSTM NETWORKS

| Hyper-parameter | Setting |
|---|---|
| learning rate | .0005 |
| loss function | Jaccard Distance |
| activation | relu |
| optimizer | adam |

The Keras-LSTMs performed best with two hidden layers. Adding a third or fourth hidden layer had little impact on the results but adding more layers beyond that resulted in Jaccard Index scores that were much worse. The Feed Forward networks performed best with three hidden layers, a max pooling layer followed by two regular fully connected layers. Adding more layers to the Feed Forward network

seemed to have little impact, so for efficiency sakes, the tested models were left at three hidden layers.

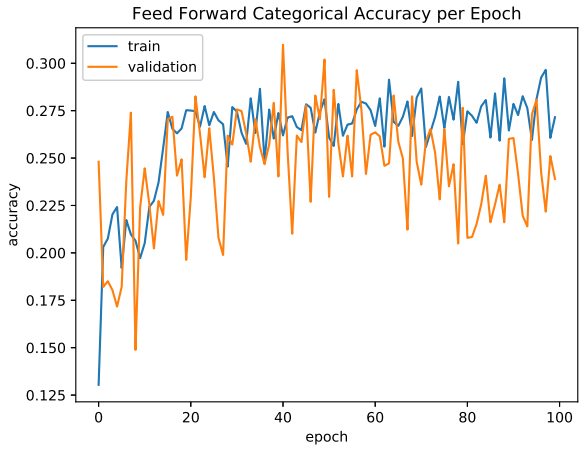



Figure 5. First layer: 24 neurons, Second layer: 12 neurons

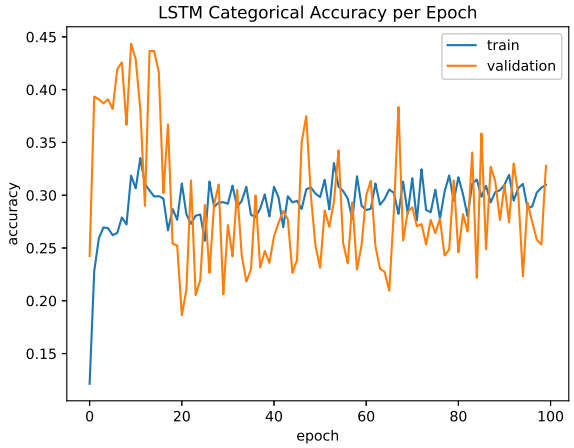



Figure 6. First layer: 24 neurons, Second layer: 12 neurons

The number of neuron cells per layer ended up being the biggest difference maker for both network types. The first time either network showed any signs of success was with 24 cells in the first layer and 12 in the second (the max pooling layer was always set to the input size and, therefore, is not included in the layer numbers here). The training and validation curves for the Feed Forward and LSTM networks with these settings are shown in Figures 5 and 6 respectively. Notice that the JI accuracies for both networks were similar ( .240) and both have very high variance. The shapes of their curves are somewhat distinct from one another, however. The Feed Forward network's training and validation curves trend in a similar, somewhat positive direction throughout the training process. The Keras-LSTM network's validation and training curves, however, vary greatly, with the validation curve prone to massive amounts of variance.

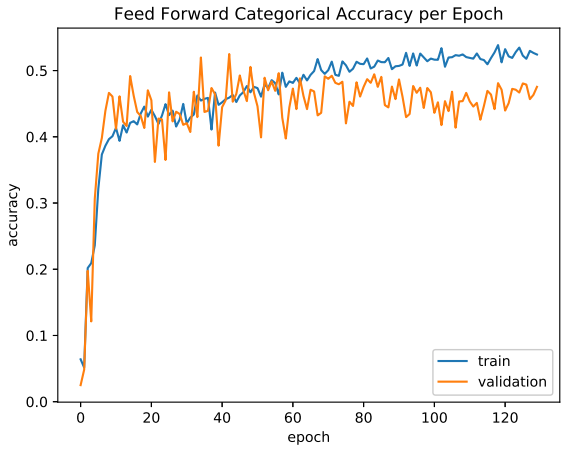



Figure 7. First layer: 128 neurons, Second layer: 6 neurons

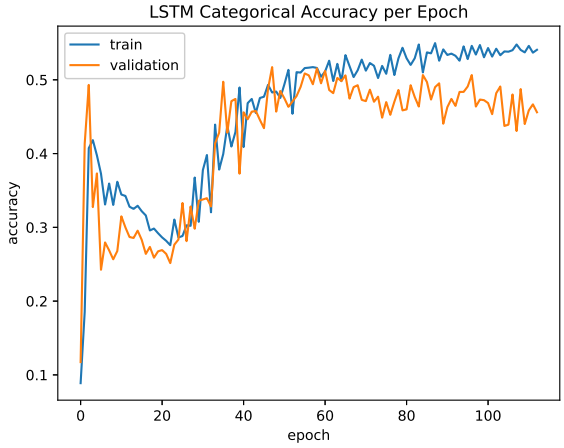



Figure 8. First layer: 128 neurons, Second layer: 6 neurons

Figures 7 and 8 show that both classifiers achieved their greatest success with 128 neurons in hidden layer one and 6 neurons in hidden layer two. Both learned their (close to) optimum weights almost immediately, although the Keras-LSTM network takes quite a dip between ten and twenty epochs before recuperating and trending positive again. Both networks also seem to be overfitting at roughly the same amount of epochs as well.

To ensure the overfitting was not just a temporary blip, both networks were trained over a longer period of time. The results for the Keras-LSTM network are shown in Figure 9 and it is clear that the overfitting was not a blip, as the overfitting continues and in fact worsens over time.

Table 9 shows the best Jaccard Index, and micro and macro F1 scores achieved by both neural networks. In addition, the competition's top overall scores for the E-c subtask have also been included for comparison. The scores achieved in this paper would have come in 8th place in the E-c competition. It should be pointed out, however, that only

TABLE 9. GLOVE NEURAL NETWORK FINAL MULTI-LABEL ACCURACIES

| Model | Jaccard Index | Micro F1 | Macro F1 |
|---|---|---|---|
| Feed Forward | .464 | .586 | .435 |
| Keras-LSTM | .467 | .600 | .453 |
| Winner | .595 | .709 | .542 |

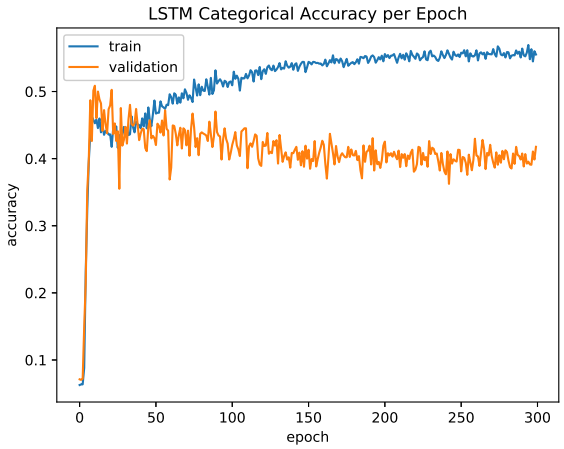



Figure 9. Same as Figure 8 but with 300 epochs

twelve competitors uploaded their scores.

## 5. Conclusion

Multi-label classification is an inherently difficult task. Performing that classification on short sequences of text data is even more difficult. With that in mind, the Jaccard Index accuracy scores achieved by the Feed Forward and Keras-LSTM networks (.464 and .467 respectively) were (somewhat) satisfying results, but still quite a distance from the top accuracy achieved in the competition (.595).

The best accuracy achieved on an individual emotion (defined by the greatest increase in accuracy over a max frequency classifier) was the Logistic Regression model when classifying whether tweets did or did not contain the *joy* emotion. It scored an accuracy of .819 compared with a max frequency accuracy of .638.

Of interesting note was how similar both the ideal hyper-parameters, and results were for both the Feed Forward and Keras-LSTM neural networks. Some future work will need to be done on this point to determine whether this was a reflection of the data itself or if it stemmed from poor implementation of the networks.

### 5.1. Lessons Learned

This project proved to be an excellent first step toward a hopeful career in machine learning research. Much was learned about the specifics of language, the math and theory behind numerous algorithms, the reasons for starting small and building up from there, and the profound importance of careful note taking during the research and data exploration stages.

For anyone new to machine learning research who stumbles across this paper, the following suggestions may be of use (time lines are based off of a four month semester):

If given a dataset to work with, spend at least ten days exploring the data while researching related work. Build the bibliography at the same time and save both it and the summaries and visualizations of the data in the same format that will be used for the final paper. Then, let that related work research be a guide toward which models and techniques are chosen for the project.

After the related work research and data exploration, start with Sci-kit Learn. Take advantage of its fantastic documentation and easy implementation. Carefully record all results and document the decisions made and reasons for them.

If / when designing a custom neural network, start with Keras. With Keras, Tensorflow neural networks can be built and tested in minutes. Try a bunch of different models, then switch to using Tensorflow directly if further customization is necessary.

Don't get frustrated. Take a step back from time to time to realize just how much has been accomplished and don't let the work of other's with far more experience be intimidating.

Write the paper throughout the semester. Don't just take notes. Ideally, by two weeks before the deadline, all that's necessary is removing the less important information.

Once the paper is written, don't touch it for at least three days. Then, take one day to proofread it and make changes. Submit it the next morning, go out for lunch, and order a bottle of wine.

## 6. Future Work

As this project was done as a directed course in machine learning, there was a rigid four month time window in which it had to be completed. Over the course of the semester and especially at the end while writing this paper, several opportunities for improvement were presented.

For starters, the script for the Twitter Pre-processing (Section 4.1) could be greatly expanded. Emojis are used a great deal in tweets and the better the algorithms are able to recognize them, the better success they're likely to have at correctly classifying those tweets.

In addition, character level embeddings were not explored at all in this project. Tweets are short by design so there is no reason not to frame up a quick Keras model that allows for them and compare the baseline results to those mentioned throughout this paper.

Access to Compute Canada's super computers was made available for this project but never taken advantage of. After implementing the changes listed above, and given some further time to explore and learn the nuances of submitting jobs to clusters, their processing power and storage capabilities could certainly take this project to another level.

Finally, Tensorflow Hub was launched near the end of this semester. Hub is Tensorflow's new platform of pre-built and pre-trained modules. Each module is a Tensorflow graph that's been pre-trained on specific training data. The idea would be to implement their module (trained on an enormous amount of text data) and then continue its training (with a low learning rate) on the data provided for this project.

## References


[1] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: Understanding microblogging usage and communities," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, ser. WebKDD/SNA-KDD '07. New York, NY, USA: ACM, 2007, pp. 56–65. [Online]. Available: http://doi.acm.org/10.1145/1348549.1348556

[2] unknown. (2017) Twitter statistics and facts. [Online]. Available: https://www.statista.com/topics/737/twitter/

[3] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith, "From tweets to polls: Linking text sentiment to public opinion time series," pp. 26–33, 01 2010.

[4] U. Rao Hodeghatta, "Sentiment analysis of hollywood movies on twitter," pp. 1401–1404, 08 2013.

[5] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: Sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, ser. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 79–86. [Online]. Available: https://doi.org/10.3115/1118693.1118704

[6] J. M. Soler, F. Cuartero, and M. Roblizo, "Twitter as a tool for predicting elections results," in *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, ser. ASONAM '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1194–1200. [Online]. Available: http://dx.doi.org/10.1109/ASONAM.2012.206

[7] S. M. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko, "Semeval-2018 Task 1: Affect in tweets," in *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA, 2018.

[8] P. Ekman, *An Argument for Basic Emotions*, 1992, vol. 6.

[9] J. Prinz, *Which Emotions are Basic?*, 04 2004.

[10] R. Plutchik and H. Kellerman, *Emotion: Theory, research and experience*, 1st ed. New York: Academic Press, 1980.

[11] S. M. Mohammad and S. Kiritchenko. (2017) Best-worst scaling vs. rating scale annotation. [Online]. Available: http://saifmohammad.com/WebPages/bwsVrs.html#SHR

[12] M. Hasan, E. Rundensteiner, and E. Agu, "E.: Emotex: Detecting emotions in twitter messages," in *Academy of Science and Engineering (ASE*, 2014.

[13] M. Bouazizi and T. Ohtsuki, "A pattern-based approach for multi-class sentiment analysis in twitter," *IEEE Access*, 8 2017.

[14] B. Y. Pratama and R. Sarno, "Personality classification based on twitter text using naive bayes, knn and svm," in *2015 International Conference on Data and Software Engineering (ICoDSE)*, Nov 2015, pp. 170–174.

[15] J. Read and F. Pérez-Cruz, "Deep learning for multi-label classification," *CoRR*, vol. abs/1502.05988, 2015. [Online]. Available: http://arxiv.org/abs/1502.05988

[16] N. Colneri and J. Demsar, "Emotion recognition on twitter: Comparative study and training a unison model," *IEEE Transactions on Affective Computing*, pp. 1–1, 2018.

[17] Z. S. Harris, "Distributional structure," pp. 146–162. [Online]. Available: https://doi.org/10.1080/00437956.1954.11659520

[18] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[19] U. Pavalanathan and J. Eisenstein, "Emoticons vs. emojis on twitter: A causal inference approach," *CoRR*, vol. abs/1510.08480, 2015. [Online]. Available: http://arxiv.org/abs/1510.08480

[20] R. P. J Pennington, M Wu. preprocess-twitter.py. [Online]. Available: https://gist.github.com/tokestermw/cb87a97113da12acb388

[21] H. Gouk, B. Pfahringer, and M. J. Cree, "Learning distance metrics for multi-label classification," in *ACML*, 2016.

[22] L. C. Jain and L. R. Medsker, *Recurrent Neural Networks: Design and Applications*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1999.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, nov 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735