All the Feels: A Multi-layer Approach to Sentiment Classification

Priya Gupta
University of California, Berkeley
priyagupta@berkeley.edu

Melanie Costello
University of California, Berkeley
melanie.costello@berkeley.edu

Abstract

One of the limitations of many existing implementations of sentiment classifiers is that they are restricted to predicting two classes - positive and negative. But there is much more depth to human emotion that is not reflected in existing language models.

This project takes a two-layered approach to sentiment classification using an existing data set of short text segments gathered from Twitter. We first predict polarity using an SVM. Then we used a LSTM model to sub-categorize tweets labeled negative as either surprise, fear, sadness, disgust or anger. Tweets labeled positive were also run through the same classifier algorithm to determine the probability of the subcategory being joy.

Our results indicate our two-layered approach delivers utilizing LSTM deliver a 44% f-score on test data, compared with 22% f-score when using only a classifier that blindly chooses the most common emotion. In this paper, we will further discuss how our baseline model utilizing an SVM + Logistic regressions in scikit delivered a higher f-score (55%) than our LSTM model, as well as our attempts to improve LSTM results.

1 Introduction

The days of talking on the phone and having meetings in person are slowly fading as the progression of technology transforms communication. It started with texting through cell phones and quickly transformed into online messaging, which spawned tweeting – a word specifically used to describe short status text messages on Twitter. Though the length and depth of personal communication may be decreasing, the need to communicate complex emotion remains the same.

Think back to the rapid adoption of emoticons or bitmoji, an image depicting an action or mood. They offered a way to see a person through messaging! Emoticons started with a simple :) to indicate happiness and :(for sadness, but now emojis have evolved to incorporate a multitude of emotions - disappointment, joy, embarrassment, irritation and more. However when it comes to decoding these short messages through natural language processing, we are still in the era of :) and :(.

Much of the work around sentiment analysis focuses on simply labeling messages as positive or negative in tone. But we know human communication is not always just positive or negative. As short messaging becomes more popular, it is vital that emotional complexity be retained to better improve communication and alleviate misunderstandings between individuals and to improve AI technologies.

2 Background

Snapchat, Facebook, LinkedIn, WhatsApp. All of these these successful applications utilize short text as a way of interaction and communication amongst users. Even Venmo, a payment transaction application requires users to write a brief message when sending payments (**?**). These businesses also have a vested interest in understanding the emotions embedded in these messages. Are people frustrated and complaining about a bug in the app? Are they amused by a fun new feature?

Accurately identifying these more granular levels of sentiment proves to be one of the bigger challenge in NLP. How do you teach a machine to understand sarcasm? Or the difference between disappointment and sadness? Progress has been made in recent years in this area of research. Ghazi (2010) used a hierarchical approach to emotion classification. Their work culminated in a threelevel approach that first assessed whether a piece of text was emotional or neutral, then positive or negative, and then sub-categorized the positive and negative examples into one of six basic emotions as defined by Paul Ekman (1992).

We will also take a hierarchical approach to classification but will make some improvements upon Ghazis work. First, we will use more data. Ghazis data set contained fewer than 5,000 text samples. The data set weve chosen holds more than 20,000. Additionally, Ghazi leaned most heavily on SVM classifiers. While we also intend to use SVM, our work also explores the use of logistic regression and deep learning methods. Noguiera dos Santos (2014) achieved state of the art results against the Stanford Sentiment Treebank corpus using a convolutional neural network. We will utilize a TensorFlow implementation of a recurrent neural network with LSTM cell.

3 Methods

3.1 Data Exploration

We utilized a twitter dataset "Tweets clean" from the National Research Council which contained 21,051 labelled tweets. On average, there were 16 words in each tweet. Please find the labels in table 1. This dataset can be further aggregated into two classes of polarity – negative or positive. There is only one positive emotion (joy) while the rest fall into the negative polarity bucket as you can see in table 2. Tables 3-4 show the top words ranked by most frequent, with and without stop words.

Sentiment	Count
Surprise	3,849
Fear	2,816
Joy	8,240
Sadness	3,830
Disgust	761
Anger	1,555

Table 1: Example counts by emotion.

Polarity	Count
Negative	12,811
Positive	8,240

Table 2: Example counts by polarity.

Word	Count
the	9,098
i	8,867
to	8,101
a	6,230
my	5,487

Table 3: Most frequent words (including stop words).

Word	Count
im	1,730
love	1,301
today	1,264
day	1,153
dont	1,093

Table 4: Most frequent words (without stop words).

3.2 Data Cleansing

Because of the use of hashtags and handles, tweets are full of excess punctuation. Additionally, users are inconsistent with capitalization. We wrote a data cleansing function to standardize case to lowercase, remove punctuation and strip excess whitespace to reduce the number of distinct words in our vocabulary. Doing this cut the number of unique words in our vocabulary from 57,028 to 33,813. Making tokens like MAD!! and mad the same reduces the sparsity in our input data and increase the term frequency of words that will be strongly associated with the predicted class (in this case, negative) and sub-emotion (in this case, anger), so that they arent removed during the vectorization process.

4 Models

4.1 Blind Model

To have a very basic point of comparison, we developed a blind model that predicts the most common emotion in the dataset in every case. In this case, if every result was predicted as joy, we get an F1-score of 22%. Although recall for joy is 100%, precision (which incorporates the false negatives) is 38%. This tells us that while joy is the dominant class, it does not overwhelm the other classes in a way that will make it difficult to improve the model's performance. Table 5 summarizes the aggregated results across all six emotions.

Label	Precision	Recall	F1 Score
Total	0.15	0.39	0.22

Table 5: Results when predicting the most common class

4.2 Baseline Model

We created two baseline models - a single-layer classifier to predict sub-emotion (without using polarity as a feature) and a two-layer classifier that included polarity as a feature.

In the first model, we used TfidfVectorizer to transform the text (eliminating words that appeared 2 or more times). The vectorizers use of term frequency (TF) over inverse document frequency (IDF) is useful in text classification because the IDF portion essentially penalizes words in a document that also appear frequently throughout the corpus. So the appearance of the word what in two tweets has less influence on their similarity score than the appearance of furious because what is common in many contexts, whereas furious is not. After vectorization, we fed the features into an SVM classifier. The results can be found in Table 6.

To demonstrate the two-layer classifier approach, we created a second baseline in scikit using two classifiers, the first for polarity and the second for emotion. Similar to our first baseline, our first classifier used TfidfVectorizer to transform the words into features to predict polarity using a SVM. Then the predicted polarity was added to the TfidfVectorizer features to classify the test data into sub-emotions using logistic regression.

We tested all combinations of SVM and logistic regression for both polarity and sub-emotion, expecting an SVM with a linear kernel would perform slightly better than logistic regression since SVMs can perform well on small datasets. In testing both, the two models produced similar results, but we moved forward with the SVM as the polarity classifier to allow for better comparison to previously published work cited earlier in this paper. The results of our first multi-classifier implementation can be found in table 7.

Incorporating the polarity classification as a feature resulted in a 1% increase in precision but no change in f-score overall. The multi-layer model also produced some changes in the precision and recall for individual emotions. There were noticeable improvements in precision for disgust and

Label	Precision	Recall	F1-Score
Anger	0.55	0.29	0.38
Disgust	0.44	0.08	0.13
Fear	0.73	0.50	0.60
Joy	0.59	0.84	0.69
Sadness	0.43	0.39	0.41
Surprise	0.57	0.46	0.51
Total	**0.57**	**0.58**	**0.55**

Table 6: Results of single classifier baseline.

joy, so when we predict disgust or joy - we are getting it right more often. However, the tradeoff was reduced recall (our ratio of true positives to true positives plus false negatives), meaning when the emotion is disgust or joy we are confusing it with something else. Consequently, we see only minimal changes to f-score (the harmonic mean of precision and recall).

5 Incorporating LSTM

In an effort to improve the performance of our classifier, we incorporated the use of a TensorFlow implementation of a recurrent neural network (RNN) using LSTM cells. Since LSTMs are suited to capturing long-term dependencies, we hypothesized this approach would more effectively distinguish the nuance that separates anger from disgust, for example. Other options like bi-gram or tri-gram models might miss some of the context needed to correctly identify the emotion.

5.1 Word Vector Embeddings

Due to the small size of our data set, we expected our test data would include many unseen words, so we elected to use pre-trained word vectors. We chose Stanfords Global Vectors for Word Representation (GloVe) because it is manageable in size and was trained on tweets, making it as close to representative of our data as possible (5).

We spent some time exploring the GloVe vectors and discovered something that could affect our results. Emojis and text-speak like LOL are space-saving and emotion-dense, so they are used frequently on Twitter. We found GloVe has vectors for a few acronyms like LOL, which implies laughter, and WTF, which could imply surprise, anger or disgust - depending on the context. However, GloVe does not have vectors for a handful of other acronyms, like ROFL or LMFAO, nor does it have vectors for the UTF-8 representation of emo-

Model	Type	Polarity Feature	F1-Score
Blind Classifier	Predict same class (joy) for all	N	0.22
Baseline 1	SVM with linear kernel	N	0.55
Baseline 2	SVM + Logistic Regression	Y	0.55
Iteration 1	1 LSTM	N	0.42
Iteration 2	SVM + 1 LSTM (polarity in embedding layer)	Y	0.46
Iteration 3	SVM + 2 LSTM (polarity in embedding layer)	Y	0.48
Iteration 4	SVM + 2 LSTM (polarity appended to raw data)	Y	0.44

Table 7: Results of all model iterations.

jis. In a tweet like Watching Stephen Colbert on The Late Show ROFL, the acronym is the best indicator of polarity and emotion. Accurately predicting polarity without it could be difficult.

5.2 Two-layer Approach

We developed multiple iterations of our two-layer, SVM + LSTM plan to meet several challenges we will later discuss. Ultimately, we made two key findings. First, the incorporation of the polarity prediction was more effective at improving performance for the SVM + LSTM models than it was for the baseline SVM + logistic regression. Despite improved performance using polarity predictions as features, we could not get any SVM + LSTM iteration to outperform the SVM + logistic regression baseline. Performance metrics for all models can be found in Table 7.

In Iteration 1, we simply used an LSTM to predict the sub-emotions without using polarity as a feature. To achieve the desired two-layer model for Iteration 2-4, we appended the predictions from our SVM polarity classifier to the arrays containing the word vector representations of our tweets. Determining the appropriate way to add in the polarity results as features was a challenge. We tested two approaches. For Iteration 2 and 3, we added polarity as a numeric feature in the embedding layer. This resulted in improvement to our model performance overall, but that was driven by strong gains for a couple of emotions and decreases in performance for a handful of others. As a wildcard for Iteration 4, we appended the word positive or negative to the front of each tweet and recreated the word vector embeddings, hypothesizing that the vector representation of positive would perhaps be similar enough to the vector representation for joy to deliver improved results. Results of this iteration can be found in Table 8.

Although this iteration was not the highest scoring LSTM implementation, adding in polarity as a word gave us more balanced improvement across all emotions and results that were closer to our SVM baseline across all categories, rather than just a handful. Ideally with more time we would have also liked to feed the polarity results into the LSTM right before the softmax layer.

After evaluating all our attempts, we selected Baseline 2 as our best model (using precision as a tie-breaker, since Baseline 1 2 have the same F-Score). The lackluster performance of the LSTM is likely due to our small dataset which was further divided into each class. It is evident that the challenges we describe in the next section more strongly affect the LSTM model than the SVM implementation.

Label	Precision	Recall	F1-Score
Anger	0.25	0.13	0.17
Disgust	0.14	0.03	0.05
Fear	0.47	0.46	0.46
Joy	0.54	0.77	0.63
Sadness	0.29	0.25	0.27
Surprise	0.48	0.26	0.34
Total	**0.44**	**0.47**	**0.44**

Table 8: Results of Iteration 4.

5.3 Challenges and Improvements

One of the most noticeable shortfalls of the SVM-LSTM implementation was a reduction in accuracy when predicting disgust. The National Research Council Twitter data set has fewer examples of disgust than any other emotion. We surmised this shortage could be contributing to the problem and also discovered that the bias term for disgust was more strongly negative than any other class (see Table 9).

We elected to try oversampling disgust in the batch selection of the training process. We adjusted the batch selection process to include more samples of disgust and iterated through a variety of thresholds. For example, with a batch size of 100, we tried ensuring disgust had equal representation (16 examples, since 16 is roughly 1/6 of 100). Ultimately, setting a threshold of at least 5 results delivered the most improvement, we saw a blended 10% increase in f-score for disgust emotion specifically.

Label	**Bias**
Anger	-0.1009
Disgust	**-0.3189**
Fear	-0.1300
Joy	0.2073
Sadness	0.0833
Surprise	-0.0327

Table 9: Bias terms.

While disgust stood out as the most obvious problem in our classification report, it was not the most common misclassification in the SVM-LSTM implementation. Our confusion matrix revealed that the most common problem was a prediction of joy when the true class was sadness. Joy is the most common class, so the prediction of joy in the face of relative uncertainty makes sense. In looking at specific examples that had been misclassified we noticed two things. First, a small handful of examples included keywords for both joy and sadness and our model didnt correctly interpret the contrasting emotions, as was the case with this tweet:

> “Last day at work and the sadness in those amazing Chinese eyes of all of my staff is killing me I had an awesome team”

Second, most of the tweets lacked emotionally charged words and the connection to sadness wasnt immediately clear, as with this tweet:

> “Feel like I havent talked to flexxbeatz in forever”

If we had included neutral as a label in our polarity classifier, this may have helped weed out tweets like the second example and would have allowed us to predict sadness for those that were only clearly negative.

Other parameters that affected our LSTM model was batch sizing. Batch sizing ensured a random pull of cross sampling from our data set. This empowered our model to 1) sample randomly to account for variance within our data set and 2) to ensure that the model accounted for emotions that had lower counts than other emotions. Through brute force testing starting at 50 and incrementing our model by 10, we determined that the ideal batch size for our model was 150. This is likely because we set a minimum threshold in each batch for disgust and anger (threshold of 5 records in each case) because of the small representation of each of these emotions. 150 led to a more equal distribution between emotions and also ensured that records in a singly batch would not be chosen more than once.

6 Conclusion and Future Work

Future work should focus on testing this approach against a larger volume of data. The LSTM model we hypothesized that would perform the best is likely overfitting our small dataset and falling to the noise which is evident in the examples from the confusion matrix above. The baseline combination of SVM and Logistic regression works well with this dataset becauses its based on word frequency and creates parameters for each class to reflect common words while the LSTM searches for dependencies. With more data, we would expect to see the most significant improvements in the neural network portion of our classifier. However, one tradeoff of a larger volume of data could be increased processing time for the SVM portion.

Additional pre-processing to reduce the data sets vocabulary size could also be explored. Options to consider include stemming words, removal of all hashtags (not just the) and removal of all Twitter handles (i.e., @Google). We suspect hashtags and handles repeat across the data set fairly infrequently. Though we can tune hyper parameters to ignore words used below a set rate, explicit removal of hashtags and handles could allow us to fine tune that hyper parameter, knowing we are focusing on the inclusion and exclusion of legitimate words.

Acknowledgments

We acknowledge sentiment analysis tutorials from O’Reilly publishing and marcobonzanini.com were used as code references for this project, as

was material from UC Berkeley's W266: Natural Language Processing & Deep Learning.

References

Paul Ekman. 1992. *An Argument for Basic Emotions*. *Cognition and Emotion*, 6(3/4):169–200.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2010. Hierarchical Approach to Emotion Recognition and Classification in Texts. *Farzindar A., Keelj V. (eds) Advances in Artificial Intelligence. AI 2010. Lecture Notes in Computer Science, vol 6085.*. Springer, Berlin, Heidelberg.

Cicero Nogueira dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. *25th International Conference on Computational Linguistics: Technical Papers*, 69–78. COLING 2014, Dublin.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. Stanford, Palo Alto, CA.

Sam Slaughter. 2015. *Text Me? Ping Me? Communications Overload in the Digital Age*. The New York Times Company, New York, NY.