Automated Emotional Analysis

Adam Sayre
adamsayre@berkeley.edu

Erin Werner
etwerner@berkeley.edu

Abstract

Social media has allowed for unprecedented access to individuals' thoughts and opinions. In this paper, we studied Twitter data in order to understand the underlying emotion behind a tweet. This expands on the task of learning the binary positive or negative attitude by including more emotions in the classification. Our dataset provided the emotion-labeled uncleaned tweets, enabling us to design several of our own custom text cleaning methods, each with different goals in mind. We then experimented with multiple models, including logistic regression, perceptron, convolutional neural networks (CNNs), and BERT language modeling. As a result from our models, the original uncleaned dataset outperformed each cleaning method based on the F1 Score. This then suggests that deep learning models are sensitive to data cleaning, so aggressive changes to the original raw text can actually reduce model performance. We achieved the best results with BERT on the original uncleaned data, but maintain that some amount of data cleaning is necessary and should be undertaken for tweet classification.

1. Introduction

Social media has provided the public with a platform for personal expression. This new outlet has given businesses the ability to easily assess the opinion of the consumer. By automatically detecting the overall feeling behind social media discourse, businesses can make more accurate decisions. The purpose of this project is to determine the nuanced feeling behind a tweet, which goes even further than just learning the binary positive or negative attitude of the user. More specifically, we want to attempt to detect the actual emotion that is expressed from the text. This is important because it will provide an even better understanding of what the user is trying to communicate. In this paper, we will examine several different data cleaning methods as well as various machine learning models in order to determine what combination of techniques best classifies the emotion of a tweet.

2. Background

Our Twitter data consists of natural language text from 916,575 tweets. The data comes from a dataset that was found on Kaggle (2020) [5]. Each tweet has been labeled with its corresponding emotion: happy, angry, or disappointed. As tweets are often shorter in length and consist of informal language, we will need to employ text preprocessing and deep learning models to determine the feelings behind the text. We chose this data because it includes a cleaned dataset (stripped of retweets, user-tags, and emojis) as well as the original raw, uncleaned tweets. We are interested in applying our own cleaning methods, such as the techniques for handling emojis discussed in Singh (2019) [4].

Next, after the data has been preprocessed, we will build multiple models by implementing several different NLP techniques, with the goal of classifying the tweet's emotion. First, we will create a simple logistic regression classifier as our baseline model. We will also construct a single-layer perceptron classifier, like in Donicke (2019) [1]. Then, we will apply more advanced techniques, such as constructing both shallow and deep convolutional neural networks, similar to Cai (2018) [3], as well utilizing Google's BERT language modeling architecture. To compare all of our results, we will use the F1 Score as our primary accuracy metric. The F1 Score is given by the formula:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

As we will use multiple techniques for both text preprocessing and the emotional classification, we plan to experiment with different combinations of these methods in order to determine which performs the best.

3. Methods

In order to determine the nuanced emotion of a tweet, we will employ multiple strategies for both data cleaning as well as classification modeling. Each technique will be developed to address Twitter-specific text data, which may have different results compared to text with a more formal language structure.

3.1 Data Cleaning

As our data consists of text from tweets, we are dealing with short, informal language patterns that include more casual phrases as well as emojis, user tags, and reference links. Due to these irregular forms of text, an important part of our analysis will include our approach to cleaning the data before we build our machine learning models. This preprocessing can impact how the model determines the emotional classification of a given tweet.

Initially, our dataset includes the raw tweet text as well as a simply cleaned version of each tweet. We want to expand this dataset with our own custom preprocessing methods. These methods will be designed to try and optimize the classification potential of the text. So, in order to understand how to best accomplish this and create our own methods, we first need to explore the raw tweet data.

Emotion	Content
angry	300990
disappointed	313714
happy	301871

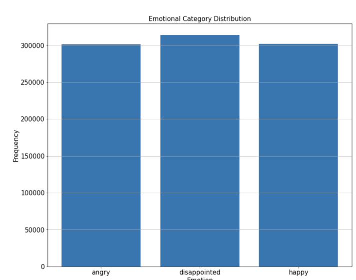

Figure 3.1 Distribution of Text Classification Labels.

From the histogram in Figure 3.1, we can see that each of the classes are relatively balanced, which means that they are each represented equally in the dataset. This is ideal for building our models as there will not be any significant bias towards one group or another.

We also want to know some of the frequent terms that appear in each of the tweets. These terms could be indicative of the sentiment behind the tweet, meaning that certain words could be particularly influential in our model. In order to get a general idea of what some of these key phrases are, we produced a word cloud that displays the top 100 most common words and phrases from the text in the overall tweet data set.

Figure 3.2 General Word Cloud.

From the word cloud in Figure 3.2, we know that larger words occur more frequently in the data set. So, it is apparent that verbs and adjectives related to a user's emotion are actually very common in the Twitter text. We can also see that pairs of emojis are a very popular way for users to express themselves in tweets. As emojis are clearly a very expressive part of the text, we want to take a closer look into the emojis that are most commonly used.

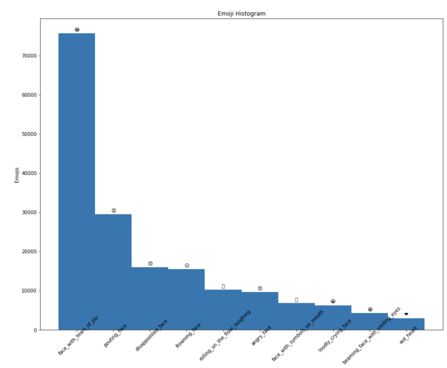

Figure 3.3 Histogram of Emojis.

From Figure 3.3, we can see that the 'crying laughing face' is by far the most common emoji, followed by the 'pouting' and 'disappointed face' emojis. Although these might be more indicative of a certain emotional class, it is important to keep in mind that users may use certain emojis to express different things, such as sarcastic remarks or general exaggeration.

With the exploration of the data in mind, we designed two different custom text preprocessing techniques that might help to improve the emotional classification of each tweet.

First, we will apply a tweet-specific cleaning technique that makes several changes to the original content. To start, we are going to clean the text of special characters, remove stopwords, and lower the casing of the words. Then, we are going to replace user tags and website instances with the tokens 'usertaginstance' and 'websiteinstance' respectively, rather than just removing them. This is because there might be an influence in sentiment related to these Twitter interactions that can be useful in our model. These replacements will allow us to generalize these actions similar to how numbers would be replaced in other NLP tasks. Last, we will split up the emoji name descriptions into individual tokens. This is because each emoji name contains phrases that might be more influential as individual tokens compared to as a single token. Therefore, this cleaning approach should have different results than the original data.

Then, we will also construct a more general cleaning method that focuses on just removing special characters from the text. We did this because there are so many special characters in tweet data, such as '@' or ':emoij:'. However, we did not remove any other text, like stopwords. This is because we thought that removing words from the data might also remove meaning that the algorithms could capture. Additionally, emoji names were left as a unique string (i.e.. ":grinning_similing_face:" was turned into "grinningsmilingface") as it could be a key contributor for the emotional classification, provided that the algorithms recognize it. This cleaning method leaves most words in the text, allowing the model to rely on smoothing and other techniques in order to gather meaning from the data.

3.2 Classification Modeling

Once the data has been preprocessed, we will need to build different machine learning models that will then be able to classify the emotion that the tweet actually conveys. As our goal is essentially a topic classification problem, there are several different approaches that we can take in order to accomplish this.

3.2.1 Logistic Regression

To start, we will build a logistic regression model to serve as our baseline. Logistic regression is used for binary outcome data, where Y = 0 or Y = 1. Yet, in a one-vs-all approach, a binary classification problem is fit for each of our three emotion-labels. Logistic regression is defined by the following probability mass function:

$$p(y=k|x) = \frac{\exp(\theta_k^\top x)}{\sum_{i=1}^K \exp(\theta_i^\top x)}$$

This discriminative regression model allows us to build a simple, yet effective classifier that can determine the multi-class emotion of a tweet based on the vectorized text input.

3.2.2 Single-Layer Perceptron

We also built a single-layer perceptron model. The perceptron algorithm classifies patterns and groups by finding the linear separation between different objects and patterns that are received through its input. Using the GloVE vectorization method, the model structure is very simple. It uses the following formula to back-propagate the weights from the output back to the input, which is represented by:

$$p_j(t) = \sum_i o_i(t) w_{ij}$$

This classifier, similar to our logistic regression model, uses a one-vs-all approach to convert a binary classification problem to a multi-class output for our three emotion-labels. Since the perceptron is a more complex algorithm, we would expect the perceptron to outperform the logistic regression baseline.

3.2.3 Convolutional Neural Networks

Neural networks are computational networks which were vaguely inspired by the neural networks in the human brain. The CNN takes a feature vector, generated from the vectorized text data, as an input and then passes it through hidden layers, concluding with an output layer that then makes the classification. The formula from one layer to the next is represented by:

$$o_j = f\left(\sum_i w_{i,j} a_i + b_i\right)$$

As there can be many different structures for a CNN, we will experiment with both shallow (single-layer) and deep (multi-layer) networks in order to emotionally classify our Twitter text data.

3.2.4 BERT & Sequence Classification

BERT, which stands for Bidirectional Encoder Representations from Transformers, pretrains representations of unlabeled text by learning from both the left and right directions of an input. This allows BERT to produce embeddings that are useful for many different NLP tasks, such as language modeling or text classification. The BERT model captures more complex language understanding, including sentence semantics, which then helps it to make more accurate text classifications. So, the combination of BERT embeddings with sequence classification forms a model that will produce multi-class predictions, which allows us to determine the given emotion of a tweet.

4. Results & Discussion

The purpose of this paper is to determine the best combination of a text preprocessing method as well as a machine learning model that will be able to accurately classify the specific emotion of a given tweet. In order to do this, we will run four different text data sets in each model. These data sets include the original raw tweet text, the original cleaned text, our tweet-specific custom cleaning method, as well as our more general custom cleaning method. In order to train and test each model, we did a 70%-30% train-test split.

First, we will train our logistic regression model on the vectorized text data and evaluate how each data set performs based on its resulting F1 Score.

	Cleaning Method	Accuracy	F1 Score
0	Orig. Uncleaned	0.900045	0.900447
1	Orig. Cleaned	0.894895	0.895385
2	Custom Cleaned #1	0.894044	0.894481
3	Custom Cleaned #2	0.852389	0.853353

Table 4.1 Results from Logistic Regression.

From Table 4.1, we can see that our baseline logistic regression model performs quite well in classifying the emotion of each tweet with a maximum F1 Score of 0.90. With regards to the various cleaning methods, the original raw tweet text actually performs slightly higher than any of the cleaned text data.

Next, we will train our single-layer perceptron model in the same fashion. The perceptron model takes the tokenized text as an input and consists of a simple flattened embedding and dense ReLU layer.

	Cleaning Method	F1 Score
0	Orig. Uncleaned	0.342268
1	Orig. Cleaned	0.342268
2	Custom Cleaned #1	0.342268
3	Custom Cleaned #2	0.342268

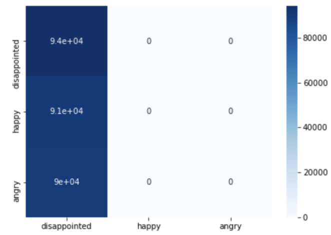

Table 4.2 Results & Confusion Matrix from Single-Layer Perceptron.

The perceptron model does not perform very well, receiving an average F1 Score of 0.34. This is because the model actually always predicts the same 'disappointed' class label for every single tweet, shown in the confusion matrix in Table 4.2. The confusion matrices were the same for all four models. These results make sense because single-layer perceptrons can only separate classes if they are linearly separable. Therefore, the simple structure of the model means it cannot generate enough complexity to fully model this problem.

Then, we will construct both a shallow and deep CNN. The shallow CNN consists of three dense ReLU layers followed by a sigmoid output layer.

	Cleaning Method	Training F1 Score	Training Loss	Validation F1 Score	Validation Loss
0	Orig. Uncleaned	0.449804	-2851716.25	0.406066	-1979731.625
1	Orig. Cleaned	0.444784	-2462558.00	0.411303	-1756000.250
2	Custom Cleaned #1	0.443693	-2360606.25	0.395592	-1868989.500
3	Custom Cleaned #2	0.456569	-1676605.75	0.390574	-1237937.250

Table 4.3 Results from Shallow CNN **on small data**.

In Table 4.3, we can see that …

So, we will also train a deep neural network to see how more layers influence the emotional classification. The deep CNN has three convolutional layers, each with dropout, max pooling and flattening. These are then followed by a dense ReLU layer and a sigmoid output layer that produces the predicted classification.

	Cleaning Method	F1 Score	Loss
0	Orig. Uncleaned	0.329369	-7.370033e+04
1	Orig. Cleaned	0.329260	-1.626010e+02
2	Custom Cleaned #1	0.329806	-9.349309e+04
3	Custom Cleaned #2	0.336461	-4.222868e+07

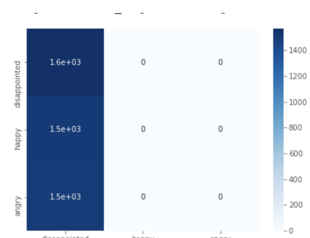

Table 4.4 Results & Confusion Matrix from Deep CNN **on small data**.

As a result, shown in Table 4.4, the deep CNN receives a lower F1 Score.
The deep CNN performs best on ...

Last, we will apply BERT embeddings and sequence classification modeling. We used the 'bert-base-uncased', which limits the embedding to 500 tokens. The sequence classification model then takes the BERT embeddings as inputs while employing the AdamW adaptive optimizer in order to generate emotion oriented topic classifications.

	Cleaning Method	F1 Score	Training Loss	Validation Loss
0	Orig. Uncleaned	0.924618	0.295197	0.275127
1	Orig. Cleaned	0.920712	0.310757	0.289297
2	Custom Cleaned #1	0.915477	0.316773	0.296010
3	Custom Cleaned #2	0.915809	0.309894	0.289413

Table 4.5 Results from BERT.

The best F1 Score that the BERT model receives is 0.925, which is on the original uncleaned text data. The different cleaning strategies likely removed a number of important features that BERT could have used to add context as BERT is able to understand sentence semantics. The inclusion of useless or nonexistent words were of less consequence than the removal of other words. BERT itself also adds a large increase in model performance over the other models on the same cleaning method. As a result, the most accurate emotional tweet classification is a result of the BERT model trained on the raw text data.

5. Conclusion

Include conclusions here …
*losing context in cleaning/reducing features, then stuffing into one embedding vector -> why original raw tweet does better
Bad to clean out stop words with BERT, they mark key semantic relationships.

Appendix - References

- [1] T. Donicke, F. Lux, and M. Damaschk. 2019; “Multiclass Text Classification on Unbalanced, Sparse and Noisy Data”. University of Stuttgart, Institute for Natural Language Processing. https://www.aclweb.org/anthology/W19-6207.pdf

- [2] E. Jonsson and J. Stolee. 2016; “An Evaluation of Topic Modelling Techniques for Twitter”. University of Toronto, Department of Computer Science. https://www.cs.toronto.edu/~jstolee/projects/topic.pdf

- [3] M. Cai. 2018; “Sentiment Analysis of Tweets using Deep Neural Architectures”. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15786247.pdf

- [4] A. Singh, E. Blanco, and W. Jin. 2019; “Incorporating Emoji Descriptions Improves Tweet Classification” https://www.aclweb.org/anthology/N19-1214.pdf

- [5] Emotion Dataset from Kaggle, 2020 - https://www.kaggle.com/kosweet/cleaned-emotion-extraction-dataset-from-twitter
- [6] J. Devlin, M. Chang, K. Lee, K. Toutanova. 2019. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, https://arxiv.org/pdf/1810.04805.pdf