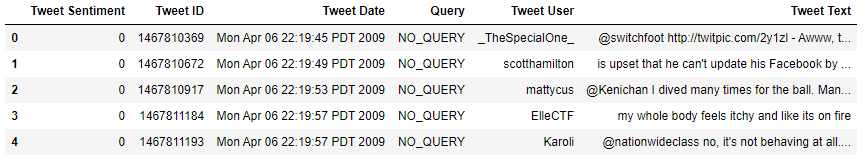
**Sentiment Classification of Twitter Tweets - Springboard Capstone 2 Henry Sue**

**Abstract:** Keeping up in today’s online ecosystem is becoming tougher and tougher as traffic continues to increase over the internet. It is crucial for a business to have a presence online, as well as managing their reputation to attract and retain customers. With the advent of social media, it is a significant goal for a business to monitor how consumers feel and react to their activities. As more data and posts accumulate on social media, companies require a way to parse this data effectively and efficiently in order to generate useful insights about consumer attitudes.

**Proposal:** In order to fill this business need in the modern day, we can employ Natural Language Processing, a subset of Machine Learning that focuses on extracting value from language. The proposed API to be developed is a combination of a social media (twitter) scraper to extract, transform, and load real time social media data, and a sentiment analysis model that will rate how consumers feel about a company. In addition to generating these insights, an interactive dashboard will be created in order to effectively navigate and visualize the insights. The API should be modular and able to be applied to potentially any company, requiring only a keyword or phrase representing the company to search.

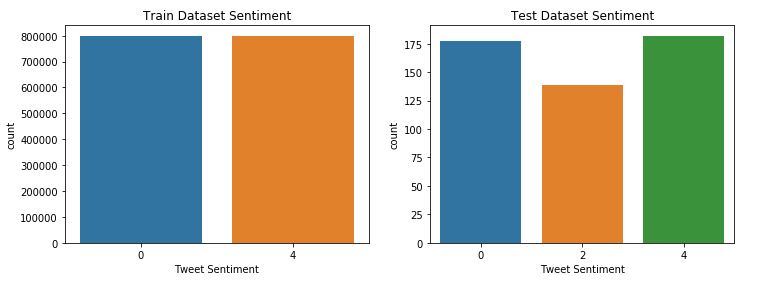
**Data**

In order to build a robust model, we require an appropriate dataset. Because the goal of this project is to build a model to analyze tweets from twitter, an ideal dataset is Sentiment 140. Sentiment 140 started as a class project for Stanford’s CS224 series, which focused on Natural Language Processing. The code base is not open source, but the dataset is free for academics. The dataset contains 16 million unique data points, or “tweets”, gathered from twitter. The dataset features training and test data with the following parameters: tweet sentiment, tweet ID, tweet date, query used, tweet user, and tweet text.



For our use case, the only relevant features are the tweet text and the tweet sentiment, as user and query level information is not correlated with tweet sentiment, and tweet ID and tweet Date are only identifiers for the tweet.

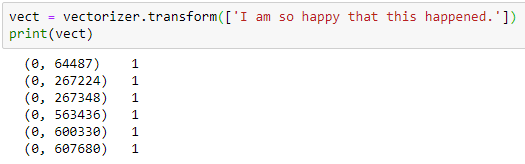
The dataset was created by automatically scraping twitter for tweets and classifying the tweet’s sentiment based on the presence of a happy smiley “ :) “ or a sad smiley “ :( “. This approach to automatically creating a sentiment classification dataset is catalogued in their paper “Twitter Sentiment Classification using Distant Supervision” (Go, Bhayani, and Huang 2009). The training data consists of a balanced 50/50 split of 16 million data points. The polarity of the tweet is marked “0” for negative, and “4” for positive. The test set contains 175 positive and negative tweets, and 130 neutral tweets.



For simplicity, we drop the tweets with neutral sentiment in the test set. Then, to allow for our model to output positive or negative sentiments, we change the labels to “0” = Negative sentiment, and “1” = Positive sentiment. This binary classification makes it easier to distinguish whether

**Classification**

The first step to establish a baseline is to create a dummy classifier. In order to process text, we have to format it in a way that our code can recognize. By substituting each word in a sentence or paragraph with a token, we can represent the sentence or paragraph with a matrix of vectors. For example, we can tokenize the sentence “I am so happy this happened” using scikit-learn’s CountVectorizer:



In this example, our vector matrix indicates a single count for each token, similarly, if we have repeated words, the count for each term-count pair will increase. Applying a dummy classifier to our dataset allows us to roughly estimate the accuracy of a model that randomly guesses a label. Our result is a model accuracy of 46.8%, a little less than just purely guessing 50%.



To establish a standard for our model, we evaluate the use of a Bernoulli Naive Bayes classifier on our dataset. A Bernoulli Naive Bayes Classifier assumes features as independent booleans that describe inputs. In our case, the presence of each token represents one feature. Let us try training the Bernoulli Naive Bayes Classifier on our dataset and testing it on our test dataset:

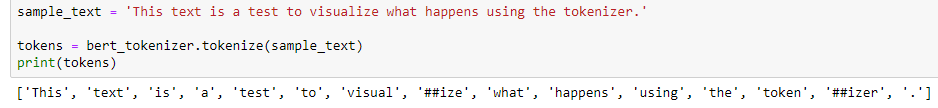


Our Naive Bayes classifier performs significantly better than our dummy classifier, indicating that our model has some degree of success.

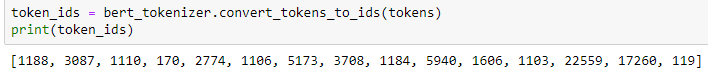
**BERT Model**

BERT - or Bidirectional Encoder Representations from Transformers is a state-of-the-art model that is trained on the Transformers network architecture (Vaswani et al 2017), reading all the tokens at once to remove directionality. Additionally, BERT is trained by masking 15% of training text and asking the model to guess what words are masked. In our project, we use the pretrained base model from the Hugging Face library “Transformers”. We will use the case-sensitive version “Bert-Base-Cased” in order to distinguish between words and letters that have been capitalized.

In order to use our pretrained BERT model, we must first create a tokenizer in order for our model to process our data. In our project, we use the pretrained BERT tokenizer. Here we can see how the tokenizer breaks down a sentence into specific words and modifiers for use in the tokenizer:

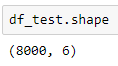


Then, the tokenizer changes the tokens into their corresponding token IDs in order for the model to recognize the token for each corresponding word:

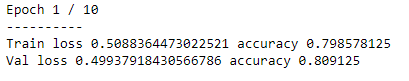


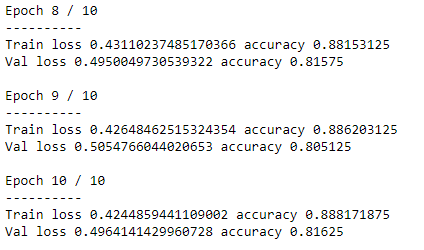
We include special tokens to denote separators, classification, unknown, and padding tokens to allow our model to adequately process our data. We also set the max length of our token matrix to 140, as twitter tweets are limited to 140 characters, therefore the maximum number of tokens is also 140.

In order to transfer train the pretrained BERT model on our tweet dataset, we must split our dataset into a smaller subset to allow iteration through epochs in a reasonable training time. We split the training dataset into a much smaller subset of 64,000 tweets for our training data, and 8000 tweets for our validation and test datasets.

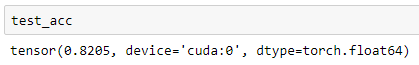


Then, the BERT model was trained for 3 days on an Nvidia RTX 2060 for 10 epochs, with a batch size of 8, using Pytorch. We can see that BERT’s pretrained model is already fairly accurate before transfer learning in the first epoch:



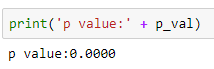
The model begins to show diminishing returns at the last few epochs:  


After training, we test our transferred model on our test dataset:



We see that our transferred model is 82.05% accurate on the test data of approximately 8000 tweets. Let us compare to our Bernoulli Naive Bayes classifier on our test dataset:

  
If we perform a t-test to determine statistical significance, we find that our truncated p-value is much less than 0.01, indicating that our model performs better than the Bernoulli Naive Bayes with a confidence level of greater than 99%. Additionally, we find that the BERT model performs better than the dummy classifier with a confidence level of greater than 99%.



**Bert Training Accuracy on Hand - Labeled Training Data**

**Conclusion and Next Steps**

As we have seen in our project, our transfer-learning trained BERT model is significantly better than a Bernoulli Naive Bayes classifier on a similar dataset, while also being trained on a smaller dataset (64000 points vs 16 million). Perhaps to improve accuracy, we can use transfer learning on the full dataset, which would take much longer than the 3 days training time. Additionally, we can train for several more Epochs (at the risk of overfitting) to further increase accuracy of our model.

**Works Cited**

1. Vaswani, Ashish, et al. “Attention Is All You Need.” ArXiv:1706.03762 [Cs], Dec. 2017. arXiv.org, <http://arxiv.org/abs/1706.03762>
2. Go, Alec; Bhayani, Richa; Huang, Lei; 2009 “Twitter Sentiment Classification using Distant Supervision”. <https://cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf>
3. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT:

Pre-training of Deep Bidirectional Transformers for Language Understanding.

(2018). https://doi.org/arXiv:1811.03600v2 arXiv:1810.04805