How Are We Feeling?

A text analysis of Twitter Sentiment

Matthew Chung, Quang Duong, Zach Gendreau

Dr. Lucy Lu Wang
INFO 498 Section B Autumn 2024
December 10, 2024

I. Introduction and motivation

The core natural language processing concepts used in this project are sentiment analysis and named entity recognition (NER). Sentiment analysis is the process of analyzing digital text to determine its emotional tone (e.g., negative, neutral, and positive), and NER is the process of identifying named entities in text (e.g., people, places, and organizations). Specifically, this project explores how sentiment analysis can be used on a dataset of user-generated Twitter posts to extract patterns that can help predict whether a user is expressing positive or negative feelings in their tweet. Combining this with NER, the project also examines how the users in these posts feel about different named entities mentioned within the dataset, and the popularity of different named entities.

This analysis is extremely relevant in the age of information, as many corporations are actively analyzing social media data to understand how users are feeling about specific people and products. Using sentiment analysis and NER, large corporations can incorporate the information they extract into their marketing strategies. This information can be applied to create things like personalized, targeted ads that aim to sell users exactly what they want, or to make changes to existing products that users have expressed frustration or dissatisfaction with.

II. Corpus

Dataset Description: Twitter Sentiment140
The Sentiment140 dataset is a dataset of Tweets that was collected as part of a research paper which can be found at this link:
(https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf).

The original use of the dataset was to test how machine learning models would perform when classifying the sentiment of Tweets when trained on data including emoticons like ":)". Rather than hand-labeling data, they used the emoticons as a noisy label for the sentiment of each tweet. Tweets with emoticons like ":)" were labeled as containing positive sentiment, while tweets with emoticons like ":(" were labeled as negative. Due to this labeling system, there are no human-annotated ground truth labels and as such we will be referring to the labels that originally came with the corpus as “corpus labels. ” The Tweets were gathered through various queries using Twitter's API, and tweets containing both positive and negative emoticons were removed from the dataset. Each tweet is also recorded with data like when it was created, the user who created the tweet, the sentiment label given to the tweet, and the query used to find the tweet, if any.

There are 1,600,000 Tweets in the dataset, and there are tweets from 659,775 different users. The tweets are taken from the time period of April 17th, 2009 to May 27th, 2009. The distribution of sentiments is even, with 800,000 positive tweets and 800,000 negative tweets. The average length of tweets in the corpus is 14.85 words.

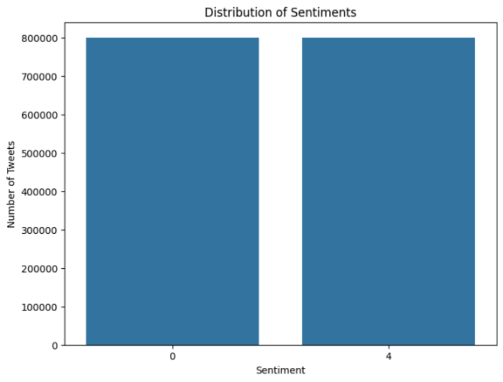


Figure 1: looking at the positive and negative sentiment count

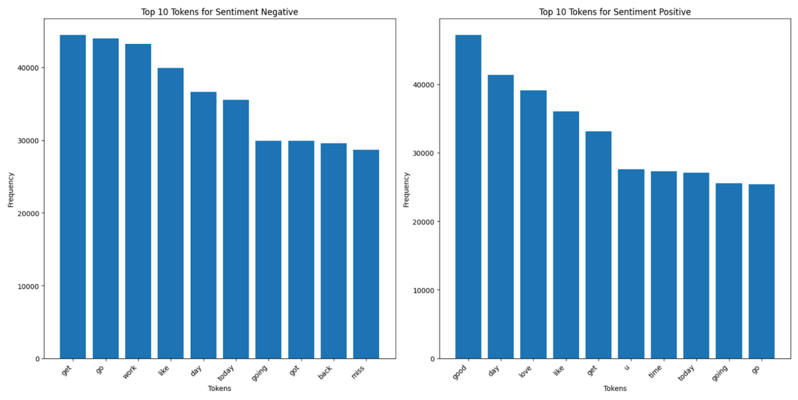


Figure 2: Looking at the positive and negative tokens

III. Modeling

Training data

Dataset Description: Twitter Airline Sentiment
The Twitter Airline Sentiment dataset originally came from Crowdflower's "Data for Everyone" library. It was originally created for a sentiment analysis, specifically in an attempt to find problems that customers noticed with each airline in the dataset. Contributors were asked to classify the tweets as positive, neutral, or negative, and also asked to explain what issue was present in a tweet if it was negative (ex: "delayed flight"). Each tweet is also recorded with data like when it was created, the user who created the tweet, the sentiment label given to the tweet, and the airline the tweet is associated with.

There are 14,640 tweets total in the dataset, and the tweets were scraped from the time period of February 16th, 2015 to February 24th, 2015. The average tweet length in the dataset is 19.12 words.

Tweet distribution per airline: United: 3822 || US Airways: 2913 || American: 2759 || Southwest: 2420 || Delta: 2222 || Virgin America: 504

Tweet distribution per sentiment: Negative: 9178 || Neutral: 3099 || Positive: 2363

Dataset Description: large-twitter-tweets-sentiment
Pulled from Huggingface (link: https://huggingface.co/datasets/gxb912/large-twitter-tweets-sentiment).

There are 224,994 tweets in it total, with 179,995 tweets in the training split and 44,999 tweets in the testing split. The dataset didn't include any data aside from the text and sentiment label of each tweet. There was no other information on the source of the dataset aside from the fact that it was annotated specifically for sentiment analysis.

In the training split, there are 104,125 positive tweets, and 75,860 negative tweets. In the test split, there are 26,032 positive tweets, and 18,967 negative tweets. This is an uneven spread with more positive tweets, but nothing too extreme. The average length of tweets in both the training and testing splits is about ~15 words.

Pre-trained Model Description: spaCy
We used the spaCy model trained on “en_core_web_sm.” This is a model trained on written web text like blogs, news, and comments. According to the official documentation for the en_core_web_sm spaCy model, the training data for the model is composed of OntoNotes 5, WordNet 3.0, and ClearNLP Constituent-to-Dependency Conversion (Spacy Model, n.d.). OntoNotes 5 is a product of a collaboration between BBN Technologies and several universities. The goal of the collaboration was to annotate a large corpus of various genres of text with

structural information and shallow semantics (OntoNotes Release 5.0 n.d.). WordNet 3.0 is a large English lexical database created by Princeton University. It contains nouns, verbs, adjectives, and adverbs grouped into related groups that express distinct concepts (The Trustees of Princeton University, n.d.). Finally, there's the ClearNLP Constituent-to-Dependency Conversion that enables users to generate "rich dependency labels similar to the Stanford dependency labels," (Clir, n.d.).

OntoNotes Release 5.0 is the final release of the OntoNotes project, a collaborative effort between BBN Technologies, the University of Colorado, the University of Pennsylvania and the University of Southern California's Information Sciences Institute. The goal of the project was to annotate a large corpus comprising various genres of text (news, conversational telephone speech, weblogs, usenet newsgroups, broadcast, talk shows) in three languages (English, Chinese, and Arabic) with structural information (syntax and predicate argument structure) and shallow semantics (word sense linked to an ontology and coreference).

Additional note on using the spaCy model trained on "en_core_web_sm":
Following our presentation, Dr. Wang noted that our assumption that though en_core_web_sm is composed of web text, the writing in the articles and web pages that are included in that web text is very different from the language and writing style that can be seen in Twitter posts. Thus, the spaCy model we used for NER on our corpus was not the best choice due to the difference in domains (web text vs social media).

Model architecture

Model Architecture: Logistic Regression (Quang)

Logistic regression is one of the models we used for the task of generating sentiment labels for the tweets in the Sentiment140 Corpus. It takes an input vector of features (some representation of a word), and generates a weight vector with a weight corresponding to each feature in that input vector. Then those weights and a bias value are passed into a sigmoid function, and then the model outputs a predicted class for that input vector, in this 0 (negative) or 1 (positive).

In this case, we trained the logistic regression models with Term Frequency-Inverse Document Frequency (TF-IDF) vectors, giving more weight to words that appear in less tweets and less weight to words that appear in more tweets. If a word appears in a few documents in a corpus → higher TF-IDF score. It's important to remember that TF-IDF only accounts for the frequency of words in a tweet, but does not take into account the context in which words appear in.

Training Considerations for Logistic Regression

The biggest decision we made while training the logistic regression model for sentiment classification was what word representation technique to use as input for the model. We originally tested with word2vec representations, as we figured that the “context window” technique word2vec uses would capture more information than the isolated words TF-IDF uses and would thus yield better sentiment classification results. What we found was actually the opposite! TF-IDF performed slightly better (0.78 macro-F1, 0.79 micro-F1) than word2vec (0.74 for both macro and micro-F1) on predicting labels for the test split of the training dataset. Taking that into account, and that TF-IDF was a bit faster and easier to train, we opted to use TF-IDF vectors.

Model Architecture: kNN (Matthew)

The second model we used to explore sentiment analysis was a k-Nearest Neighbors model (kNN). This model uses the idea of proximity to classify inputted data, it looks at the data points “neighbors” to the inputted value and determines from there based on the majority, what the value should be classified as. Before testing the model on the Twitter corpus, we trained the model on the hugging face twitter training data, and inputted this data as a TF-IDF, term frequency Inverse Document Frequency, which gives different weights to different words based on their prevalence within the corpus.

The other parameters within the kNN model that we set were the neighbors that the model would use to determine each value, this was n = 5. The standard practice is to choose 3 as the n value, but with the size of the corpus we believed that a value of 5 would allow for less variance within the data and overall give better predictions within our model. The next parameter that we defined within our model is the metric that would be used to determine the distance between the points to find their neighbors. The standard metric is a euclidean distance but for the purpose of our corpus we went with cosine similarity. The cosine metric we saw was the best metric to use, since the data was high dimensional and sparse, the angular distance used within a cosine metric would be more optimal for our training model.

Training Considerations for kNN Model:

One of the considerations for training the kNN model was to make sure that the data was cleaned properly when looking at the training corpus. The training corpus had tweets that had @’s in the beginning, to ensure that the data was usable we got rid of these @s and tokenized the tweets using the nltk Tweet Tokenizer. After tokenizing all of the tweets we joined them to make sure they were one chunk of text rather than a list of each tokenized tweet. Then as stated above we continued forward using a TF-IDF to create the features for each tweet within the training corpus.

Model Architecture: Fine-tuned BERT (Zach)

The third model we implemented for sentiment classification was a pre-trained fine-tuned BERT model. More specifically, we chose to train a DistilBERT model due to its reduced parameter count and increased efficiency. This choice was made after several attempts at training and classification with a standard BERT model, as the volume of data and comprehensive nature of the model simply took a long time and continuously crashed the kernel. In the future, we have an opportunity to return to the original BERT model or pre-train on a domain-specific corpus, rather than 'distilbert-base-uncased'. Another action we took to reduce runtime was to take a random sample of the test data. We did this in a manner that retained the evenly split class distribution.

After training the model on this 80 percent split, the inputs to the model were test X values (individual tweet contents), with an output of predicted y values (either 1 for positive sentiment or 0 for negative sentiment). A confusion matrix of these results may be seen below. While the model achieved a macro-F1 score of 0.73, this approach could be improved with a more domain-specific BERT model.

Training Considerations for Fine-tuned BERT:
As discussed above, a decision was made to train on a subset of the data due to parameter constraints and runtime. Around 10% of the original dataset was used, maintaining the 50% split of binary classes. We also used a weights and balances account through GitHub to visualize the runtime and troubleshoot any crashes on personal devices.

Pre-trained Model Architecture: spaCy

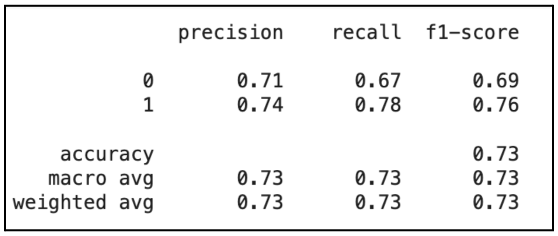
The specific components of the spaCy model include tok2vec, tagger, parser, senter, ner, attribute_ruler, and lemmatizer. Tok2vec is used for creating word representations from input text. Then, tagger and parser work together to assign different part-of-speech tags to words in the text and establish the relationships between those tagged words. Next, the parser defines sentence boundaries within the text, and the most important part for our use case, the named entity recognizer identifies named entities within the text. Finally, the attribute_ruler can be used to add attributes to tokens, and the lemmatizer can be used to reduce words to their base form (https://spacy.io/usage/linguistic-features).

We used the spaCy model because of its ability to conduct NER as a pre-trained model. w We specifically used the model trained on “en_core_web_sm.” Since we’re conducting analysis on a corpus of Twitter data, we thought it best to use this specific spaCy model, as it’s a pipeline trained on written web text like blogs, news, and comments. It took hours to run NER on the corpus even when using en_core_web_sm, so we opted not to use the spaCy model trained on en_core_web_lg (https://spacy.io/models/en).

IV. Describing/visualizing results

Note: for BERT and kNN, performance metrics for sentiment prediction are based on predicting a sample (10%) of the corpus

(Distil)BERT Performance:



```
              precision    recall  f1-score

           0       0.71      0.67      0.69
           1       0.74      0.78      0.76

    accuracy                           0.73
   macro avg       0.73      0.73      0.73
weighted avg       0.73      0.73      0.73
```

Figure 3: the scores for BERT model

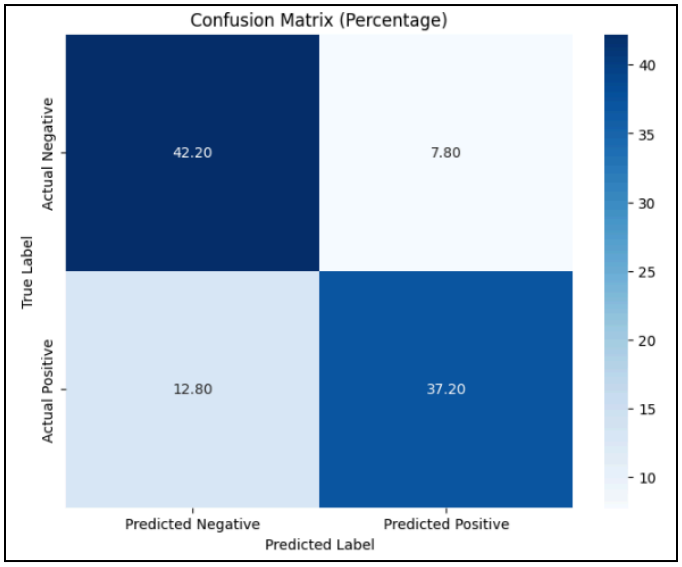
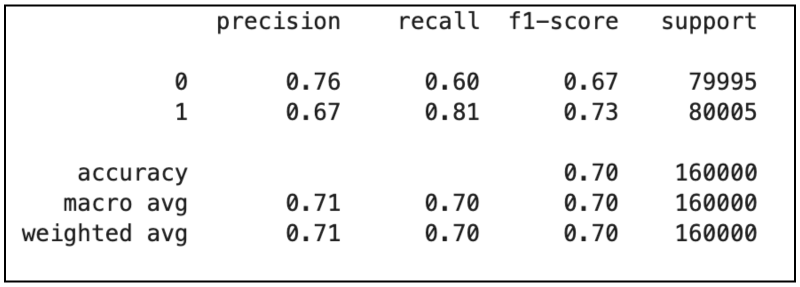


Figure 4: the confusion matrix for BERT model

kNN Performance



```
              precision    recall  f1-score   support

           0       0.76      0.60      0.67     79995
           1       0.67      0.81      0.73     80005

    accuracy                           0.70    160000
   macro avg       0.71      0.70      0.70    160000
weighted avg       0.71      0.70      0.70    160000
```

Figure 5: the scores for kNN model

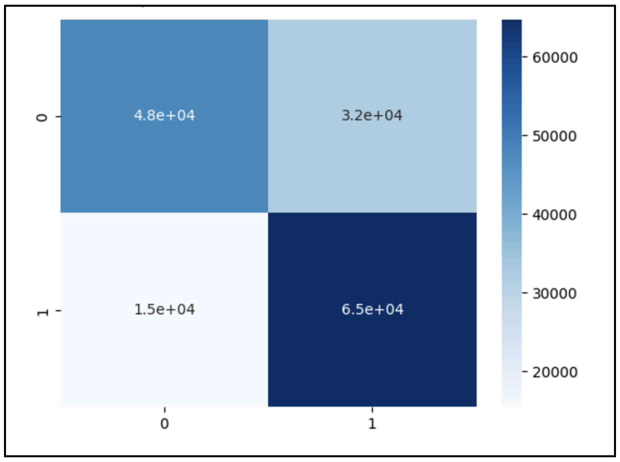
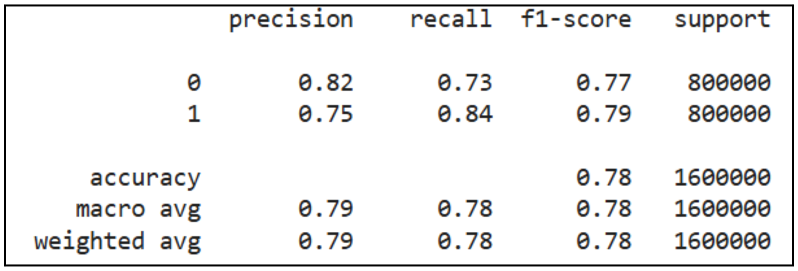


Figure 6: the confusion matrix for kNN model

Logistic Regression Performance



```
              precision    recall  f1-score   support

           0       0.82      0.73      0.77    800000
           1       0.75      0.84      0.79    800000

    accuracy                           0.78   1600000
   macro avg       0.79      0.78      0.78   1600000
weighted avg       0.79      0.78      0.78   1600000
```

Figure 7: the scores for Logistic Regression model

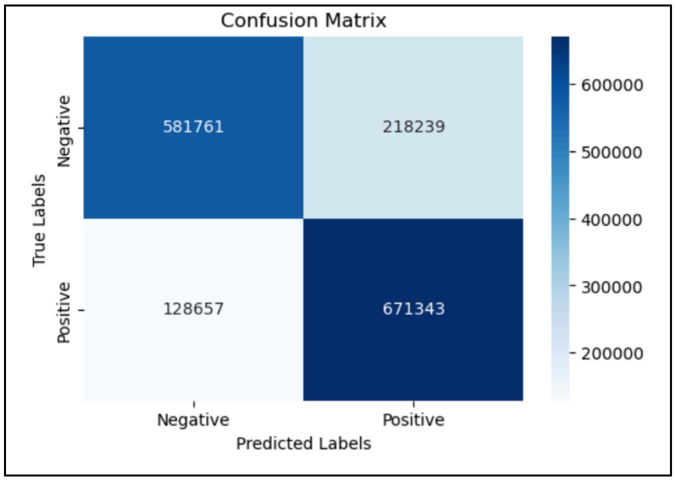


Figure 8: the confusion matrix for Logistic Regression model

V. Discussion

Between the 3 models, it seems that they had pretty similar performance when predicting sentiment labels of tweets in the corpus. KNN (figure 5) did the worst with macro and micro-average F1 scores of 0.70, BERT (figure 3) 2nd best with micro and macro-average F1 scores of 0.73, and logistic regression (figure 7) did the best with micro and macro-F1 scores of 0.78. Based on these results, with the spaCy model we continued forward using the Logistic Regression Model when comparing our predictions of sentiment labels and the corpus labels. Not only taking in performance but both the BERT model and the kNN model were trained on 10% of the data because of the performance speed, concluding forward that the logistic model should be used when looking at sentiment labels compared to NER.

The first question that we explored was if there was a relationship between the average sentiment and the frequency of that sentiment. After mapping the top ten entities (figure 9), and examining the results there seemed to be no relationship between average sentiment and the frequency of the entity. This did not match our expectations, as we thought there would be some sort of relationship, positive or negative. This could result from the size of the corpus and the small time frame, Twitter is such an open forum that there is just varying sentiments throughout the corpus.

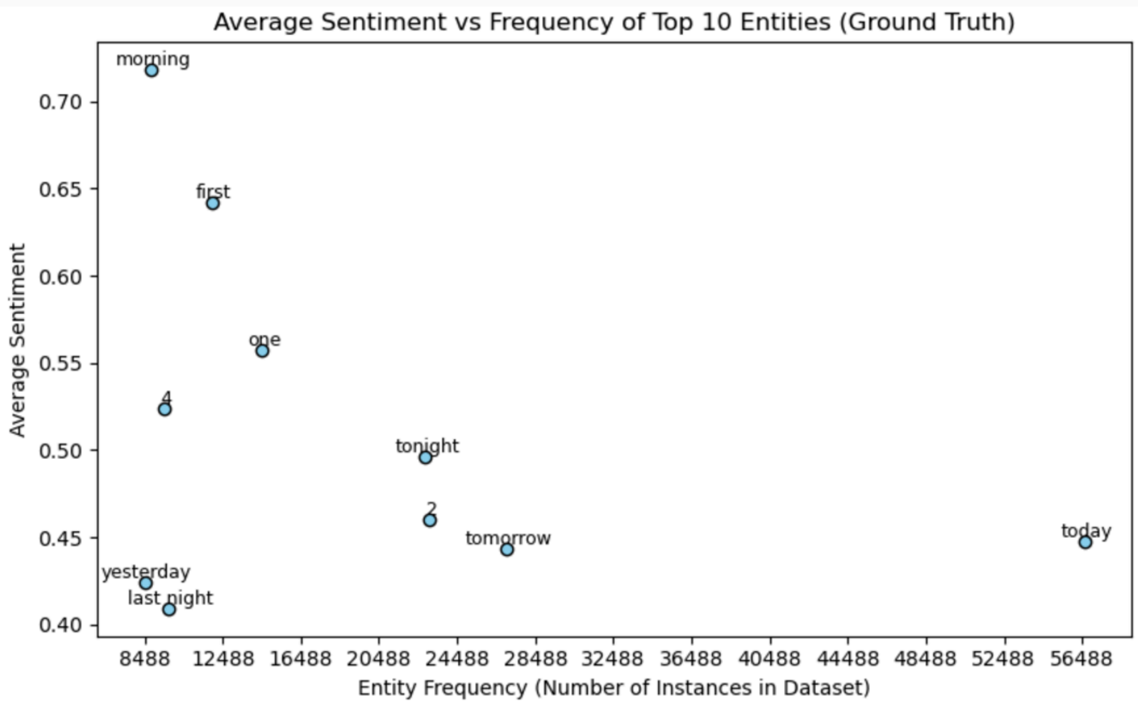


Figure 9: graphing the average sentiment and the frequency for top ten entities.

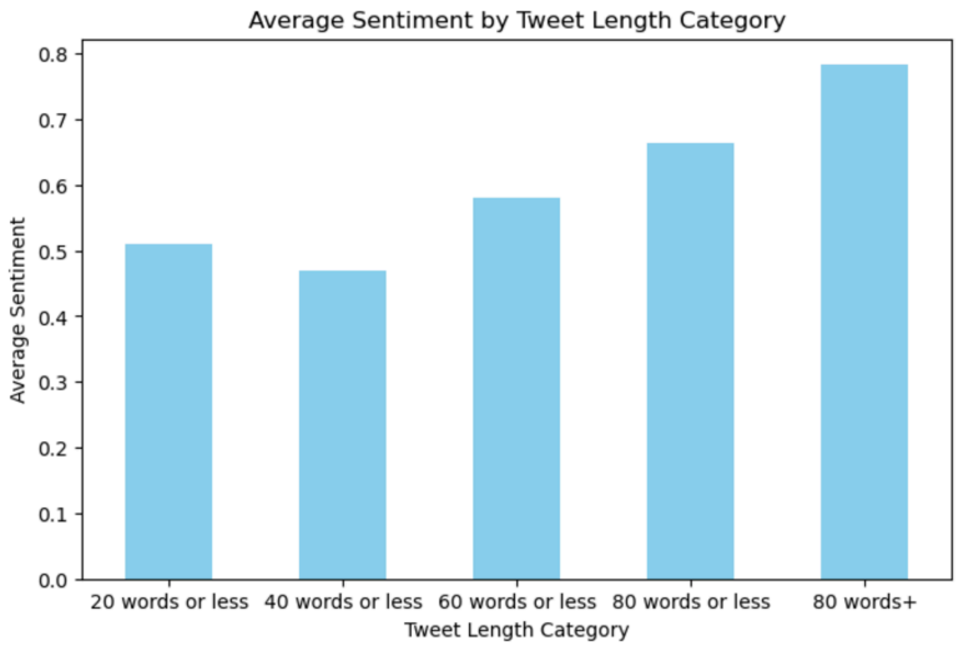


Figure 10: graphing the average sentiment and length of tweets based on corpus labels

To continue forward with our findings, we took the results of the spans/entities and looked at the ones that were most positive and that were most negative. First, we looked at the differences between the results that logistic regression produced and the corpus labels provided by the dataset and found the results were similar. We used these results to explore which entities had the most positive and the most negative sentiment. We found that the span/ entity “sweet dreams” and “#followfriday” had an average sentiment value of .99, being the most positive, and"ugh ","ouch " and"a sad day” had an average of .01, being the most negative sentiment. The negative sentiment aligned with typical vernacular as these are words with negative connotations, but the positive sentiment seemed to alluded to the time period, such as “#followfriday” since the data was from 2009, this could have been a trend that was popular at the time.

When looking at the different characteristics of Tweets, we explored how often different entity labels appeared in tweets, between the two sentiments, and how lengths of tweets correlated with the sentiment. When looking at the types of entities in tweets, the entity labels were similar between positive and negative tweets but the last entity, which was “work_of_art” for positive tweets and “money” for negative sentiment tweets. This correlates to what we understand with the culture behind these two entity labels, as more people are going to talk about art in a positive tone while money typically has a negative connotation. Finally, when looking at both our predicted labels and the corpus labels, the average sentiment sentiment score for tweets seems to get higher (more positive) as the length of the tweets increases.

VI. References

Afeworki, M. (2021, September 9). *A comparison of machine learning algorithms: KNN vs decision trees*. Medium. https://milena-pa.medium.com/a-comparison-of-machine-learning-algorithms-knn-vs-decision-trees-d6110e08bfea#:~:text=*%20Decision%20trees%20can%20be%20faster,generalize%20the%20data%20in%20advance

Clir. (n.d.). *CLEARNLP-guidelines/*. GitHub. https://github.com/clir/clearnlp-guidelines/blob/master/md/components/dependency_conversion.md

Distilbert. DistilBERT. (n.d.). https://huggingface.co/docs/transformers/model_doc/distilbert

Home. W&B. (n.d.). https://wandb.ai/home

OntoNotes release 5.0. Linguistic Data Consortium. (n.d.). https://catalog.ldc.upenn.edu/LDC2013T19

Spacy Model. English. (n.d.). https://spacy.io/models/en

The Trustees of Princeton University. (n.d.). *WordNet*. Princeton University. https://wordnet.princeton.edu/