CptS475 Final Project Report

Twitter Sentiment Analysis

December 13th, 2022

Project Authors:
Dawson Whipple
Evan Kimmerlein

Course Professor:
Dr. Assefaw Gebremedhin

Table of Contents

Abstract

The goal of this project was to create a program that can effectively and accurately analyze and predict the sentiment of the author of a string of text based on the contents of the message. To create a program that can analyze and predict information from strings of characters, a machine learning (ML) algorithm was trained on a data set of 1.6 million tweets that already had a sentiment label. The beginning data set was split into two separate sections, 80% of the entries were used for training the ML algorithm and 20% being used as testing and approval data after training. The algorithm is capable of giving one of three possible scores to a text string, positive, negative, or neutral. An average accuracy of 78.4% was achieved by the end of training, with obvious statements such as '#happy' and '#sad' having major positive and negative ratings respectively.

Introduction

Understanding the public perception of any given topic can be very valuable information but also difficult to calculate. For this project, in order to find accurate and dynamic public sentiments, Tweets from Twitter that are filtered by keyword were used. By performing sentiment analysis on Tweets the results are more accurate to the broader public opinion rather than select corporate opinions that may be found using other sources such as news feeds. Results of this analysis can have many practical applications such as being able to generalize public reception of a product launch, political campaign, or social movement and can be quite valuable for beneficiaries of the previously listed examples.

Using the Twitter API, any recent Tweets containing a desired keyword or keyphrase can be sorted, cleaned, and stored. The ML algorithm is then able to perform analysis on the stored Tweet information and produce a sentiment for each individual Tweet as well as an overall average sentiment of all gathered Tweets. Preliminary tests have shown desirable results with minimal error and reasonable loss.

Definition of Problem

How does one understand and interpret what the public opinion of something is, and what is the benefit of knowing? By having the knowledge of the average sentiment among a large sample size of individuals it is possible to make informed decisions that wouldn't be possible otherwise.

Corporate brands are able to use this technology to gather and understand data regarding their products and their performance. When a new product is launched, during a busy holiday season, or when a product gets discontinued, the given brand is able to analyze what the general opinion is. If a product launch has an overall negative response, changes can be made such as software updates, price decreases, or product overhauls in order to rectify the negative opinion. This is especially useful because previously the primary indicator of a poorly received product would be poor sales, resulting in revenue loss. It is now possible to see a negative reaction of the public even before a product is available for purchase, increasing potential sales and profits.

Political campaigns can use the results of this work to gain insight into what techniques and strategies are working and continue to implement as well as what strategies are not working and modify them. Effectively using sentiment analysis can assist in winning votes and reforming policies.

Even on a smaller scale the results of this project are very useful. Consumers can use sentiment analysis to make informed purchasing decisions for products they may be interested in. if the general opinion of a new product release is negative, while informing the associated brand that changes need to be made, the consumer would also be informed to not buy the product. While on a significantly smaller scale than brand management or political campaigns, information is still gained, used, and valued by individuals.

Models/Algorithms/Measures

There are three main steps to our solution. The first step is the recent tweet collection. After being given a hashtag or keyword, our application needs to fetch a reasonably large number of relevant recent tweets from twitter. Initially we planned to do this through creating our own web scraping api. However, there are a couple potential problems with this. First our api would likely be treated like a human viewer, and would receive biased tweets due to the twitter algorithm giving our bot preferential results due its to past searches and location. Second, for mass fetching thousands of tweets, the api would fetch data at a rate underneath our wishes. Fortunately we discovered twitter's official api had enough features to meet our needs. This would be far less biased, and far more time efficient than a custom made api. However, this also came with the limitation of only being able to scrape half a million tweets per month. Although likely unacceptable for real world application, for the purpose of our project and demonstration, we found this to be acceptable.

After the fetching process, our application needs to analyze the sentiment of each tweet. The most sensible approach for creating an algorithm that would fit our needs was to utilize machine learning. In our case, we found that it would be most convenient to use a simple Sequence model from the Python library Keras. Our plan was to be able to feed a cleaned body portion of a tweet, then output a confidence value between zero and one, one being positive and zero being negative sentiment.

The first layer of our sequence model is an embedding layer. By starting with an embedding we immediately convert text into a series of numerical vectors. This allows the rest of our model to efficiently process our inputs. To generate this embedding layer we utilized an unsupervised word to vector model from the Gensim library. This model would be trained off the same tweet data we would use to train the sequence model to ensure its learned vectors are appropriate.

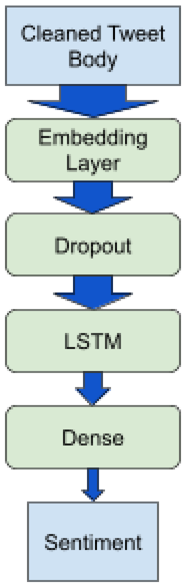


Fig 1.0 Basic structure of the sequence model

The second layer added is a dropout layer. The intent of this layer is to help prevent overfitting of our model. This ultimately did not turn out to be an issue due to our large amount of data we acquired, and likely could have been toned down.

The third layer utilized LSTM. The intent of this layer is to identify and focus on relevant details from the training data, and ignore the others. It then passes this narrower and more relevant data to our final layer.

Our final layer of our model is a Dense layer. This layer is used to change the dimensionality of our data down from one hundred to one. Its activation function is set to sigmoid such that we know the result will be between one and zero. This final output is the sentiment score for our input tweet.

Now that we have analyzed the sentiment of each tweet, the final step is to display the information to the user. We found that the easiest and cleanest way to show the overall sentiment was simply to take the average sentiment value of all the analyzed tweets.

Implementation/Analysis

One of the most critical decisions in this process was the selection of a dataset. We ended up settling on a free pre-labeled tweet dataset from Kaggle titled “Sentiment140 dataset with 1.6 million tweets.” As the title suggests it contains 1.6 million tweets, each labeled 0 for negative, 2 for neutral and 4 for positive.

Although this was the best dataset we found based on labeling and sheer number of tweets, there were a few notable flaws. The tweets were all from around the year 2009. This may cause a potential bias towards sentiments that may have been true in 2009, but are no longer relevant today. However, due to our LSTM layer prioritizing values that are significant, the model will still likely be influenced the most by common positive and negative terms that are likely still in use today. Another flaw was the classification type labeling. As we wanted to value between zero and one for how positive the tweet is, it would be far better to have our training data that was rated on that same continuous scale, as opposed to 0, 2, and 4.

In addition, despite labeling being defined for neutral tweets in the dataset's description, after analysis of the data we discovered that no neutral tweets were present in the actual dataset. 800,000 tweets were labeled positive, and 800,000 were labeled negative.

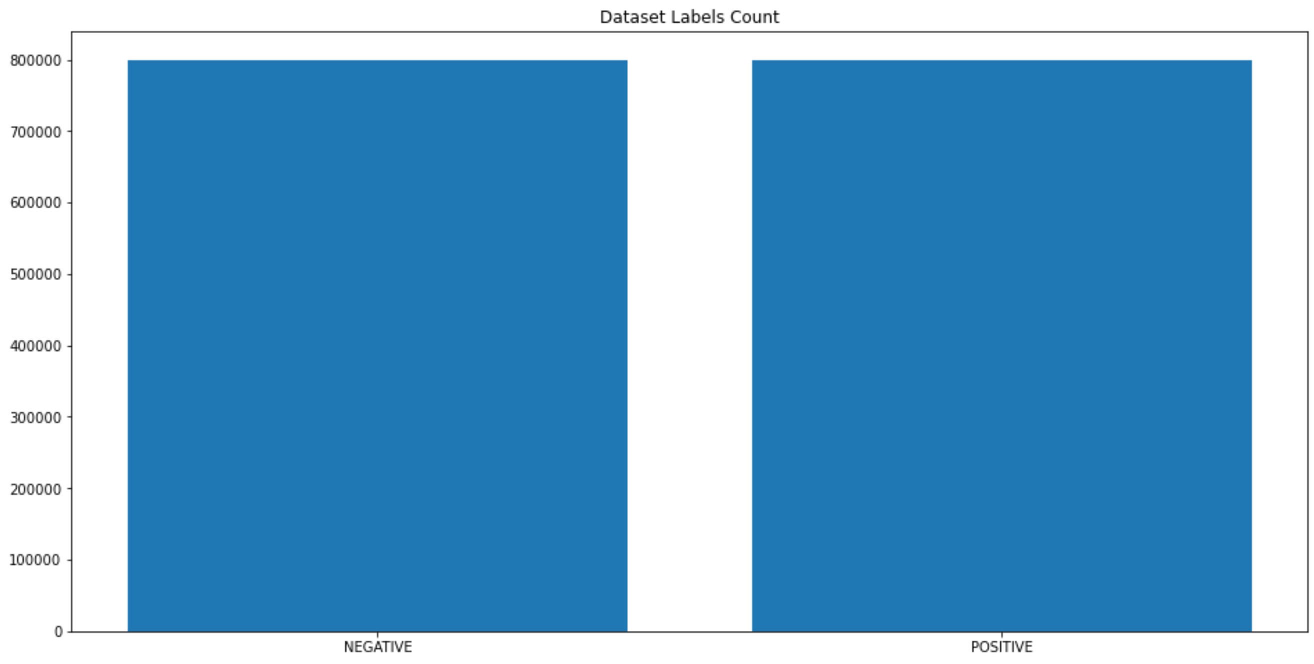


Fig 2.0 Frequency of negative and positive tweets within our dataset

As these flaws are likely to have a small but measurable impact on the accuracy of our model, it would be ideal for us to collect our own large dataset, and label them appropriately to better fit our needs. However, for the sake of this project, it was not realistic for us to manually collect and label data to an extent where it would be more effective than this Sentiment140 dataset.

After settling on this dataset, the data was cleaned such that all special characters, links, and emojis were removed, leaving only words. The data was then split into separate training and testing groups, with the training utilizing 1,280,000 tweets and the testing group using the other 320,000 tweets. Now that our data was cleaned and sorted, it was ready to be fed to the models. First the Word2Vec model had to be trained

As the Word2Vec model utilizes unsupervised machine learning it only needed the various tweets bodies, and not their connected sentiment value. For this reason there really is no external data we can test it against. There is no accuracy rating, or loss value. However, even if we can't quantitatively evaluate its direct performance, we can qualitatively do it, by finding the top similar words to other common words, and checking to see if they have similar meanings and tones. Ideally, the top results for each word should be primarily filled by synonyms. The accuracy will also be indirectly measured by the performance of the overall sequence model, as it plays an vital role as the embedding layer.

The overall sequence model will be much easier to test. As it is a supervised model, the model actively calculates its accuracy and loss as it trains. In addition, as mentioned previously, 320,000 entries are being withheld from training so that we can check its accuracy after.

Once both models are trained, and connected to the twitter API, in theory, a Hashtag result such as #Happy should return overwhelmingly positive results, and #Sad should return overwhelmingly negative results.

Results and Discussion

The model that was required to be trained was the Word2Vector model. The training was quick, and showed promising results. To help verify Word2Vec had effectively chosen vectors that would properly represent the given words, we checked to see what words were closely related to each other given their vectors. The following tables are some of the terms we examined.

Similar words to 'Love'

Word	Similarity
ador	0.7033
loves	0.6674
luv	0.6504
loooove	0.6163
loved	0.6044
looove	0.5984

Fig 3.0 Similar Words to 'Love' According to word to vector model

Similar words to 'OK'

Word	Similarity
okay	0.8114
alright	0.7531
well	0.6528
fine	0.5856
alrite	0.5409
lol	0.5093

Fig 3.1 Similar Words to 'OK' According to word to vector model

Similar words to 'Hate'

Word	Similarity
hates	0.6791
sucks	0.6510
suck	0.6482
dislike	0.6339
stupid	0.6266
despise	0.6015

Fig 3.2 Similar Words to 'OK' According to word to vector model

The results above are promising that our word to vector model is acceptable. Not only do all the top six similar terms displayed show words with the same tone, but most of them are synonyms, if not alternate spellings. This is exactly what we wanted to see out of the model for our embedding layer.

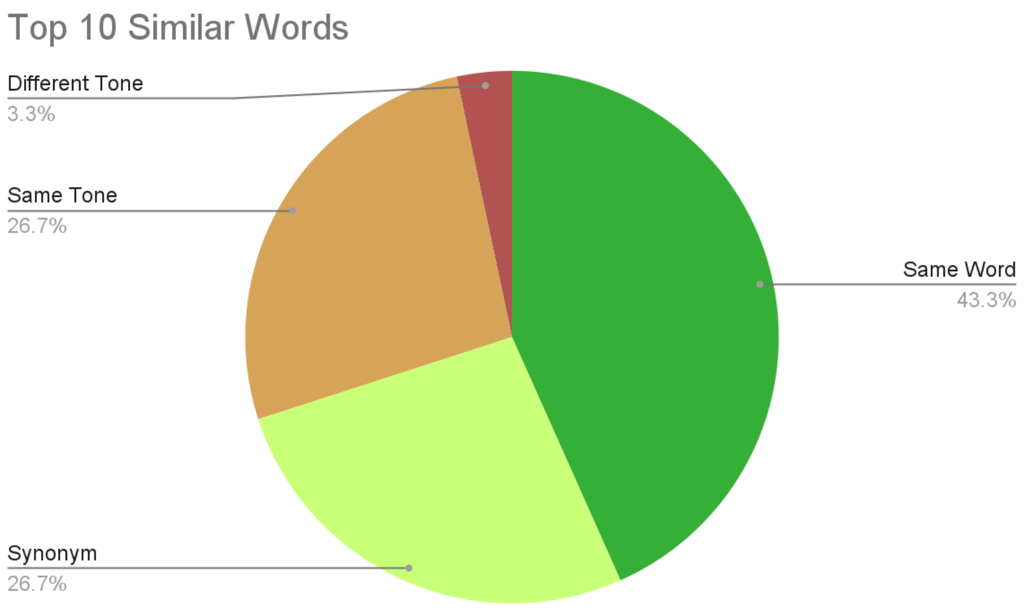


Fig 4.0 Relations of words found in the Top 10 similar words according to our model. Note: we did not exhaustively test all words in the dataset, this chart is only an approximation and not a perfect representation of accuracy of the model.

Further analysis only confirmed this as we found that out of the top 10 most similar words to the samples we evaluated, only 3.3% of the similar words had a different tone, and 70% of the words were either Alternate spellings or Synonyms. With this information we are confident that our embedding layer is effective enough for our purposes.

Our Keras Sequence model took a substantially longer time to train, requiring over two hours. Being a supervised learning model, it was substantially more straightforward to test. For the sake of training and testing against the classification system of the dataset we evaluated any sentiment value over 0.5 as positive, and any value under 0.5 as negative. During the model training, both its accuracy and loss were recorded:

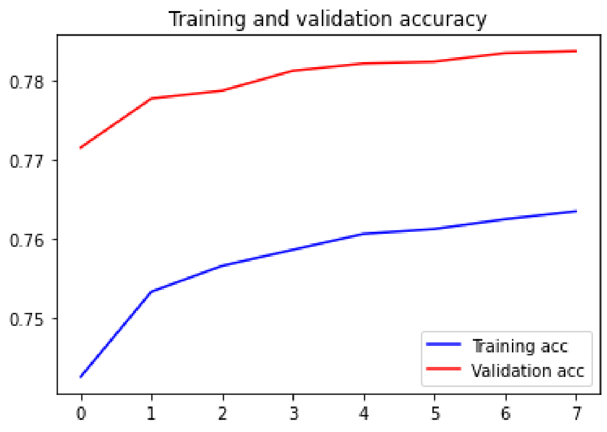


Fig 5.0 Accuracy over time for sentiment analysis sequential model

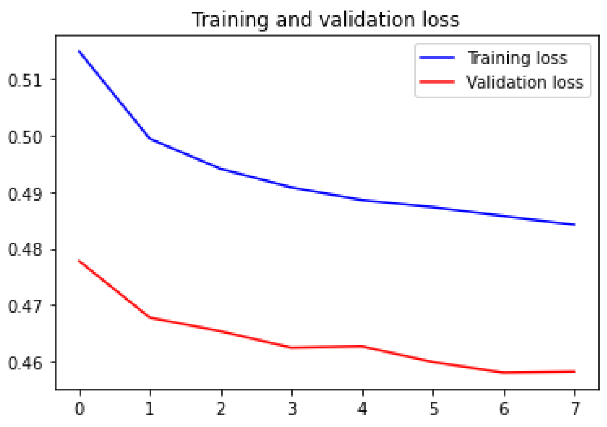


Fig 5.1 Loss over time for sentiment analysis sequential model

The results showed a reasonably high test accuracy of 78.6% and a reasonably low test loss of 0.46. Nearing 80% accuracy is an acceptable rate for our purposes. Further analysis of our model and its bias can be done through a confusion matrix:

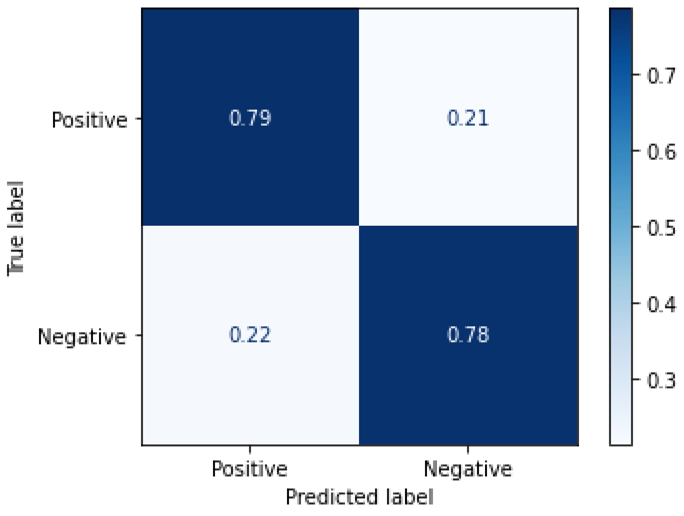
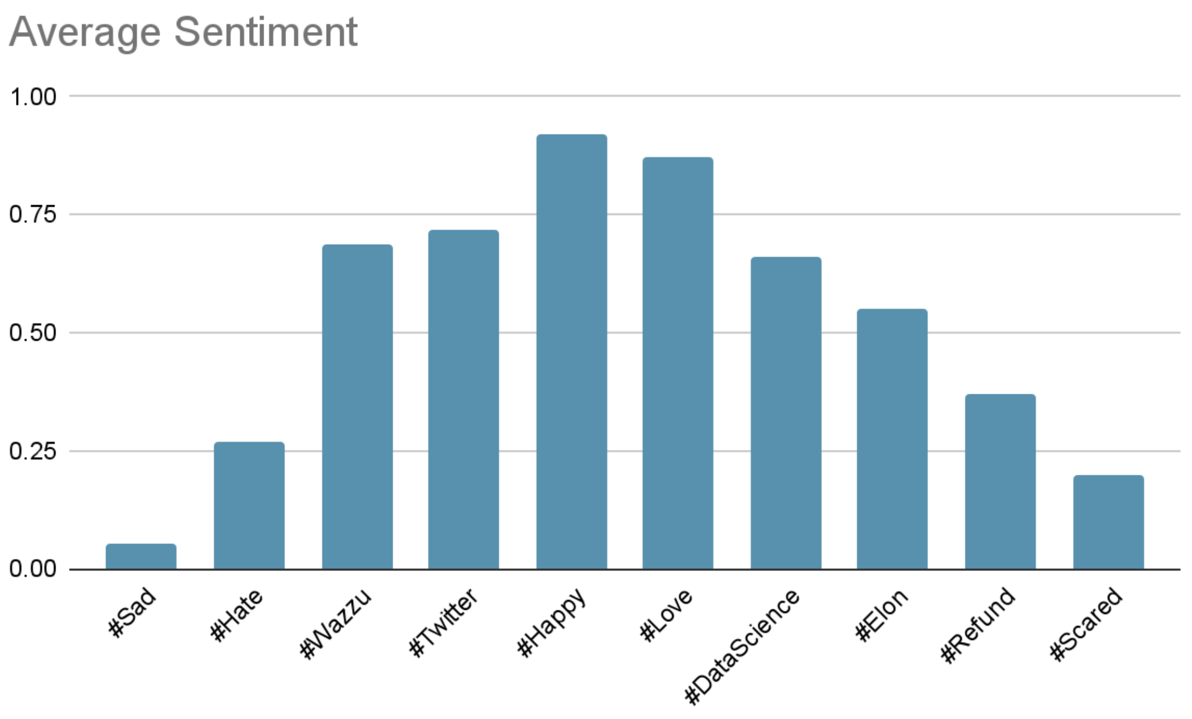


Fig 6.0 Confusion matrix for sentiment analysis sequence model based off of 10,000 samples from our test dataset

As seen above our model has very low bias, favoring negative predictions by just around 1%. While in an ideal world we would have higher accuracy, low bias will lead to more balanced wrong incorrect predictions and will be less likely to drastically skew the overall sentiment of a hashtag. All that is left is to evaluate the model with new tweets fetched by the twitter API.

Evaluation of models utilizing newly fetched tweets is significantly more difficult, as the tweets are not pre-labeled with their sentiment. We also ran into issues with the limitations of the twitter API. When fetching new tweets, twitter would occasionally give us tweets in languages other than English. As our word-to-vector model, and sequence model were both trained on a dataset of English-only tweets, tweets in other languages cause the accuracy to fall drastically. While this can be fixed with extra filtering, this requires increased tweet fetching which is limited on a standard twitter dev account. Another issue is spam tweets. As spam tweets are not representative of public opinion, skewing the overall sentiment analysis. Despite these limitations, the application still behaved as expected. Analyzing positive hashtags consistently returned average values nearing 1, while analyzing negative hashtags consistently returned average values near to zero. More complex hashtags tended to have less extreme values, but still often leaning towards negative or positive, giving you a general understanding of twitter's user base current feelings towards the topic.



Conclusions

There is a lot of room for expansion and improvement with regards to this sentiment analysis project. The biggest and most obvious improvement is to increase the accuracy of the ML algorithm developed through other ML strategies. Future work would focus primarily on completing this task as it has the largest effect on the functionality and reliability of the project. Other desired future improvements include adding an easy to use, intuitive UI containing the data as well as graphs of the data to allow for a broader audience to be able to effectively use the program. This would particularly assist with users who may not be software literate, as currently, results are printed to terminal. There's also improvement possibilities in the cleaning of Tweet data gathered from Twitter. Features such as filtering by language, filtering out spam, or even creating a whitelist/blacklist to see only results from desired Twitter users.

Bibliography

M. M. KazAnova, “Sentiment140 dataset with 1.6 million tweets,” *Kaggle*, 2009. [Online]. Available: https://www.kaggle.com/datasets/kazanova/sentiment140. [Accessed: 2022].

Paoloripamonti, “Twitter sentiment analysis,” *Kaggle*, 02-Jan-2019. [Online]. Available: https://www.kaggle.com/code/paoloripamonti/twitter-sentiment-analysis. [Accessed: 2022].

“Save and Load Keras models :   Tensorflow Core,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/guide/keras/save_and_serialize. [Accessed: 2022].

“Sklearn.metrics.confusion_matrix,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html. [Accessed: 2022].

“Tweepy documentation,” *Tweepy Documentation - tweepy 4.12.1 documentation*. [Online]. Available: https://docs.tweepy.org/en/stable/. [Accessed: 2022].

“Twitter API documentation | docs | twitter developer platform,” *Twitter*. [Online]. Available: https://developer.twitter.com/en/docs/twitter-api. [Accessed: 2022].