Aadharsh Pannirselvam and Jei Ho
Professor Trimble
Introduction to Machine Learning
12 May 2023

Project Report: Cross Dataset Tweet Sentiment Analysis

Introduction to the Dataset

For our training dataset, we used a dataset made by Muhammad Tariq from Kaggle that had 1 million tweets from September 2020 to October 2022.[1] The dataset has three columns which consist of tweets, languages ("en", "fr", etc), and labels ("positive", "negative", etc). For testing, we used a completely separate dataset from Crowdflower filtered by Kaggle user Kritanjali Jain which contains 1630 tweets relating to Apple products from the year 2016 which was when products like the iPhone 6S/7 were released. The dataset has two columns which, like the training dataset, consist of tweets and labels (1: "positive", 0: "neutral", -1: "negative").[2] Crucially, the Apple dataset has a strong class imbalance towards negative tweets while the train dataset is relatively balanced, which impacts classification. By virtue of the second dataset's context being much more specific than the first, classification isn't fully cross-applicable despite both datasets consisting of tweets, which is an issue that we will further explore.

For instance, the apple dataset has some very sarcastic negative tweets like "@apple#ipad #irig for the price to connect my guitar I could buy a real amp loud enough to perform! #ripoff" and "@steviebuckley It's doing strange things to @dropbox files as well - create a new folder and it disappears etc. @apple #yosemite #bugs" which, when tokenized could only feasibly be determined to be negative with words like ripoff, strange, and bugs. While these words might be common in negative tweets about Apple, they're much less common in general tweets, and so such tweets tend to slip through classification techniques trained on generalized data. This often manifests itself in the form of tweets with negative sentiment being labeled as positive, which we will refer to throughout this report as false positive results.

Research by Others

One study, *Sentimental Analysis on Apple Tweets* by Dupindar Kaur performed sentiment analysis on a dataset containing 16000 tweets relating to Apple products.[3] Kaur had a similar preprocessing method as ours in that he removed non-alphabetical characters, stopwords, URLs, etc and lemmatized the remaining words. Afterwards, he divided the dataset in two where he ran a Naive Bayes model on the training set. Another research *Apple Sentiment Analysis* by Harrison Jansma performed sentiment analysis on the same Apple dataset as ours.[4] He employed a different preprocessing technique where he used TFIDF to convert the tweets into numerical data. For the model, he used classifying methods like Logistic Regression and SVM. One big difference between our approach and both studies mentioned above is that they used the same Apple dataset to both directly train and test their models, so their accuracies on the Apple tweets are higher at the expense of being biased for general use.

Exploratory Data Analysis

We preprocessed our training/test data through filtering to include tweets that were only in English. We also got rid of rows that were not labeled "positive" or "negative" as they would indicate an ambiguous sentiment which added excessive noise in our classifying methods. We also replaced the labels with binary numbers, so 1 represents positive and 0 represents negative. For tweets, we cleaned them by using different NLTK libraries. Some of our cleaning methods included getting rid of stop words ("the", "and", etc.) and lemmatizing words so

[1] M, Tariq. *Sentiment Dataset with 1 Million Tweets*, https://www.kaggle.com/datasets/tariqsays/sentiment-dataset-with-1-million-tweets. Kaggle, 2016.
[2] K, Jain. *Twitter Sentiment Analysis LSTM*, https://www.kaggle.com/code/kritanjalijain/twitter-sentiment-analysis-lstm/input. Kaggle, 2022.
[3] D, Kaur. *Sentimental Analysis on Apple Tweets with Machine Learning Technique*, https://ijcset.net/docs/Volumes/volume7issue9/ijcset2017070901.pdf. IJCSET, 2017. ISSN: 2231-0711, Volume 7 Issue 9, 76-78.
[4] H, Jansma. *Apple Sentiment Analysis*, https://harrisonjansma.com/apple. June 20, 2018.

that words are reduced to their base form. We also made sure to remove URLs and non-alphabetic characters within the tweets. After preprocessing, our generalized training dataset had a size of 394,129 tweets with 198,812 positive tweets and 195,317 negative tweets. The Apple dataset had a size of 829 tweets with 686 negative tweets and 143 positive tweets, which is a high degree of class imbalance which will become relevant later. We decided to divide the training set so that 80% would be for training purposes and 20% for validation within the scope of the train set before we tested using the Apple Tweets dataset.

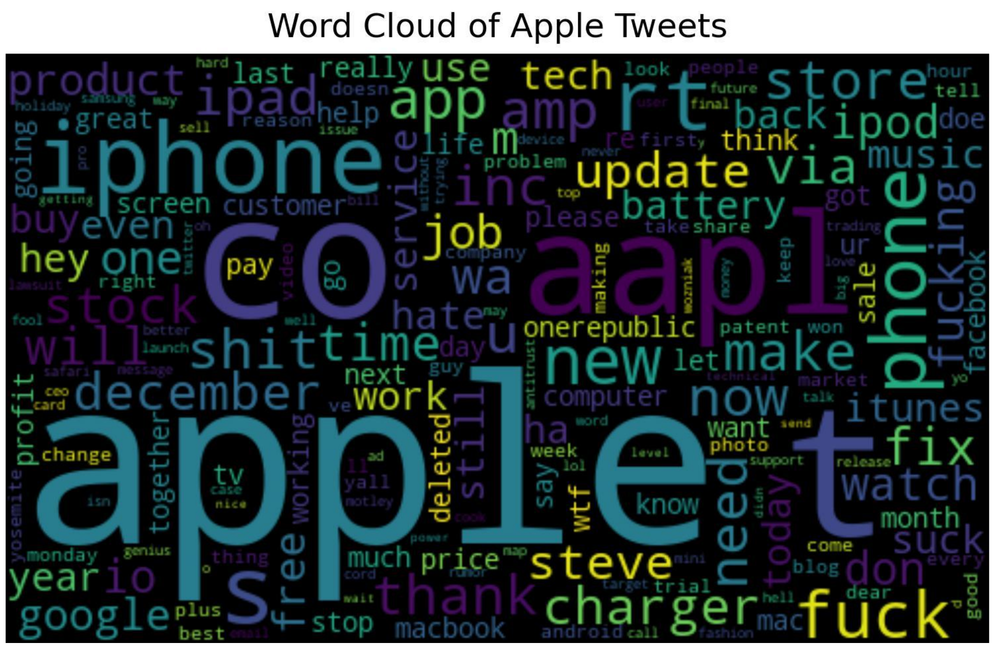

Figure 1: Word Cloud of Apple Tweets. After preprocessing the Apple dataset, we created a word cloud visual to show what words commonly appeared relative to other words. We see that unique words like Apple, app, iphone, ipod, and AAPL (stock name of Apple) appeared the most. We can also start to tell that there will be a large portion of negative sentiments given the many instances of profane words in the word cloud.

<u>Data Modeling Technique 1: Naive Bayesian Classifier Scoring Per Word</u>

As a starting point, we used the simple Naive Bayesian model used with the Spam Dataset to classify tweets as positive/negative. The classifying process is largely credited to a scoring function which takes an individual tweet as input and scores it from a score ranging from -10 (most negative sentiment) to +10 (most positive sentiment) based on the frequency of the tokens in the tweet relative to the frequency of tokens in the two label subsets, with the overall score being divided by the number of clean tokens to derive a score per word so as to not bias the classifier based on the length of each tweet. The threshold for classification was set at 0, so if a tweet had a score that was less than 0, then its associated sentiment is 0; otherwise, its associated sentiment is 1. We then plotted a confusion matrix of the model on the validation set, which we note is fairly balanced.

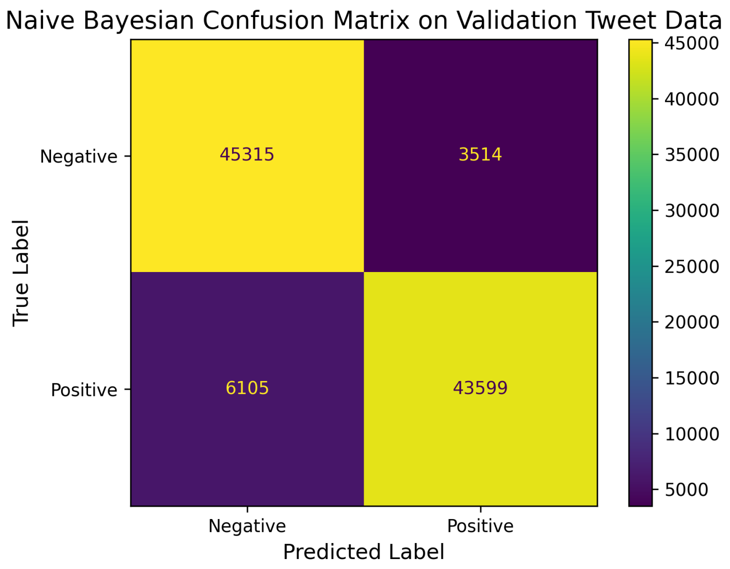

Figure 2: Naive Bayesian Confusion Matrix on Validation Tweet Data. The confusion matrix reveals the counts of true positives, true negatives, false positives, and false negatives where positive indicates that the tweet had a positive sentiment and negative indicates that the tweet had a negative sentiment. There were 43599 tweets that were true positives, 45315 tweets that were true negatives, 6105 tweets that were false negatives, and 3514 tweets that were false positives. Thus, the accuracy of our Naive Bayesian model on the validation set was (43599 + 45315) / (43599 + 45315 + 6105 + 3514) = 90.24%

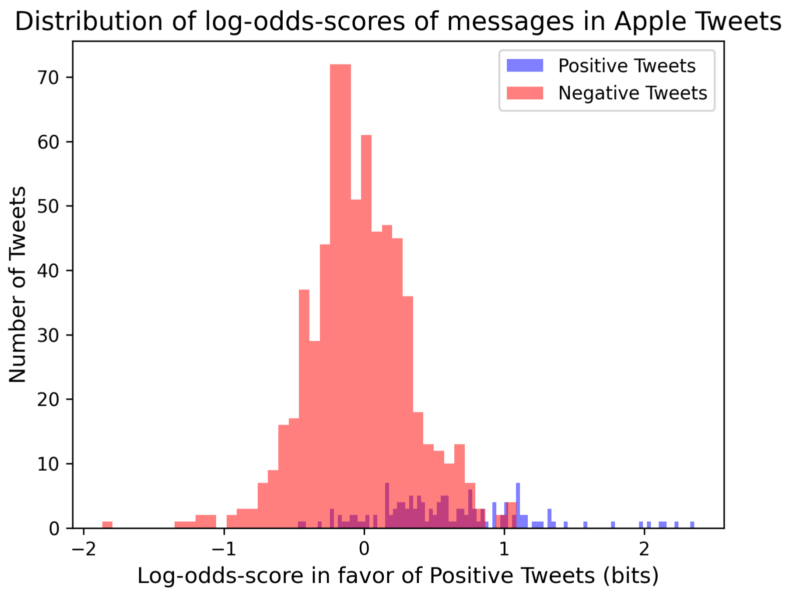

Figure 3: The distribution of log-odds-scores of messages in Apple Tweets. The log-odd score measures the likelihood that a given tweet belongs to a positive or negative sentiment. So, a more positive score indicates that the tweet is more likely to belong to the positive sentiment category whereas a negative score indicates that the tweet is more likely to belong to the negative sentiment category. The distributions above show that there are many more negative tweets than positive tweets. Further, it shows that there is substantial overlap between positive and negative tweets, with the negative tweet distribution being bell shaped, and centered barely to the left of 0, indicating lots of false positives. The positive tweet distribution is mostly to the right of 0, indicating that we will not see too many false negatives.

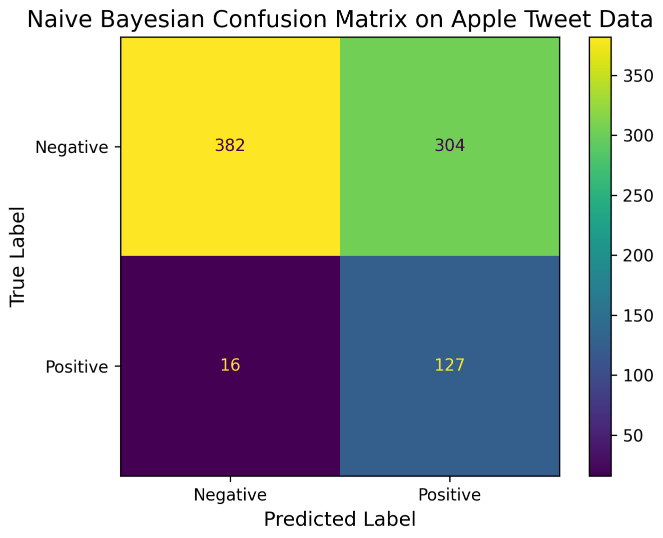

Figure 4: Naive Bayesian Confusion Matrix on Apple Tweet Data. The confusion matrix from testing the Naive Bayes model on the Apple testing data resulted in 127 true positives, 382 true negatives, 16 false negatives, and 304 false positives. As such, the model had a testing accuracy of (127 + 382) / (127 + 382 + 16 + 304) = 61.40%.

The high amount of false positives in Figure 4 indicate that the model still lacks understanding of specific tweets with negative sentiments. We think this is because, as aforementioned, certain negative Apple related tweets incorporate idiosyncratic speeches like slang, sarcasm, and words with multiple meanings. so the model would not be able to detect such negative sentiments well. The drop in accuracy in Figure 4 relative to Figure 2 can be attributed to the entirely different context and dataset that the model was trained on relative to the Apple tweets. While we did not generate confusion matrices like Figure 2 using the validation data with our other classification methods, they all exhibited similarly higher accuracies compared to their Apple tweet counterparts.

Data Modelling Technique 2a: Bidirectional LSTM

Given that the purely mathematical Naive Bayesian Classifier didn't look at the order of words in the Apple Tweets, we decided to use a Long Short-Term Memory (LSTM) RNN for our other major classification technique, as they are aware of the order and relation of words, and thus able to capture long-term trends in the data, remembering or forgetting information based on the context. To begin with, we made our own 128-dimensional embedding using the train data with a 10,000 word vocabulary, which yielded the following results.

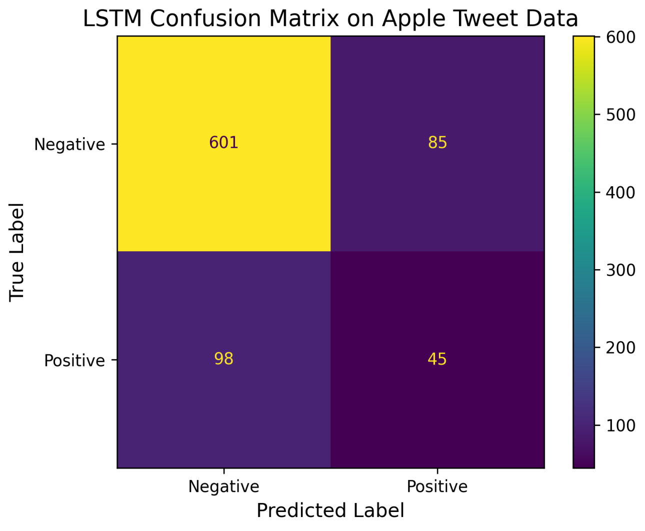

Figure 5: LSTM Confusion Matrix on Apple Tweet Data. The confusion matrix from testing the LSTM model with embeddings from the train data on the Apple testing data had 45 true positives, 601 true negatives, 98 false negatives, and 85 false positives. As such, the model had a testing accuracy of (45 + 601) / (45 + 601 + 85 + 98) = 77.93%.

Relative to the Naive Bayesian classifier, this technique is much better at evaluating true negatives and avoiding false positives, though it trades this improvement off for a significantly higher rate of false negatives, to the point where more of the positive tweets in the Apple data are labeled by the model as negative than positive, which while not ideal, is likely better than deeming negative tweets as positive in a real world implementation. This being said, this model does still have quite a few negative tweets from the Apple dataset that are falsely labeled as positive as well.

A natural progression from this classifier was to use a pre-trained word vector as a drop-in replacement for our trained embedding, since the train data has previously proven itself to not mesh with the Apple tweets. We opted to use the Global Vectors for Word Representation (GloVe) pre-trained word vector, specifically the 100 dimensional variant that draws data from Twitter, with 2B tweets and 27B tokens with the same LSTM model, giving us the following results.

Data Modelling Technique 2b: Bidirectional LSTM with GloVe (Transfer Learning)

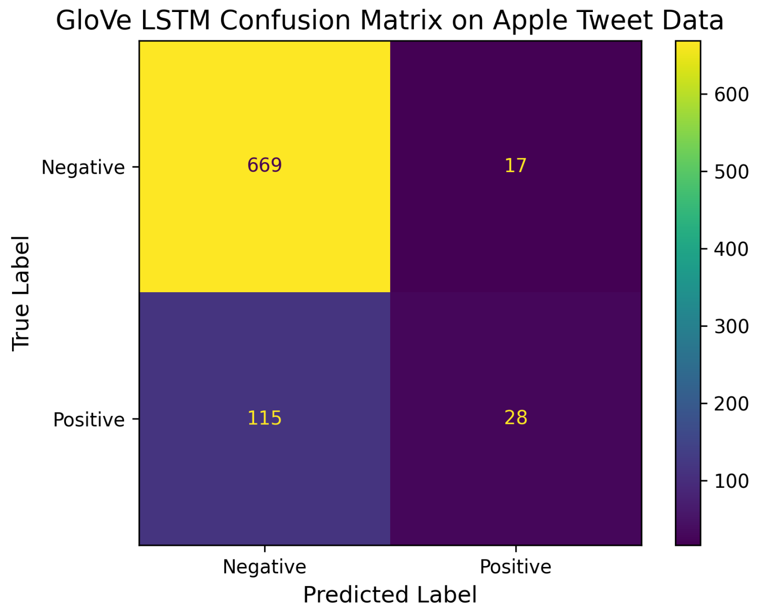

Figure 6: GloVe LSTM Confusion Matrix on Apple Tweet Data. The confusion matrix from testing the LSTM model with GloVe embeddings on the Apple testing data had 28 true positives, 669 true negatives, 115 false negatives, and 17 false positives. As such, the model had a testing accuracy of (28 + 669) / (28 + 669 + 115 + 17) = 84.08%.

Relative to the previous LSTM model, the GloVe model has a higher accuracy, but has an even more aggressive tradeoff relative to the Naive Bayesian classifier, with even fewer false positives than the previous model in exchange for even more false negatives, though the negative predictive value is still around 85%. Once again, in a real world use case, this would likely be the best of our models thus far, since it has the lowest rate of false positives, even if it comes at the cost of relatively high false negatives.

Following these techniques with increasing complexity and persistently weak classification metrics, we thought about ways in which they could be made stronger, without further adding complexity and further overfitting to the mixed tweet dataset.

Improvement Technique 1: Ensemble Consensus Between Methods

Given that the Naive Bayesian classifier proved to have many true positives and false positives, but the LSTM methods had many true negatives and false negatives, we naturally looked at running all three of our models and assigning predicted labels based on an unweighted consensus, resulting in the following confusion matrix.

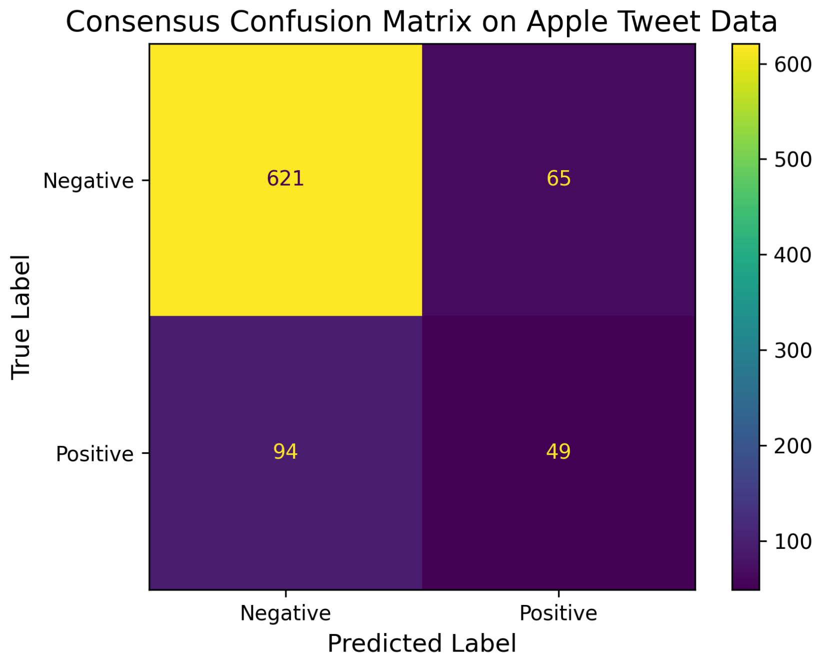

Figure 7: Consensus Confusion Matrix on Apple Tweet Data. The confusion matrix from testing a equally weighted consensus of the three prior models on the Apple testing data had 49 true positives, 621 true negatives, 94 false negatives, and 65 false positives. As such, the model had a testing accuracy of (49 + 621) / (49 + 621 + 65 + 94) = 80.82%.

As expected by ⅔ of the techniques in the consensus being LSTM models with broadly similar characteristics, the consensus method yielded many false negatives somewhere in between the two LSTMs. While this method is fundamentally more computationally intensive than running a single classifier, it’s likely that with three different methods with distinct confusion matrices, we could obtain a more balanced confusion matrix. Another set of directions that we could take with this approach is applying a sequence of techniques or weighting them within the consensus, either by cross-validation or based on their individual accuracy measures. The issue there of course, is that we can only do so using our train data or a holdout set, which in this problem is different from the Apple tweets.

Improvement Technique 2: Simulating Live Data: Occam's Razor

Given our initial observations of the sarcasm and word choices used in the Apple tweets being fundamentally different from the training datasets, we repartitioned the Apple Tweets into 20% train and 80% test, adding the ~120 Apple Tweets to our two LSTM models and testing on the holdout 80%. This was meant to simulate a running backfill of training data in a real world algorithm, with the addition of tweets being fairly minuscule relative to the training data for our LSTM and GloVe LSTM models. This yielded the following results, to be compared with the LSTM methods above.

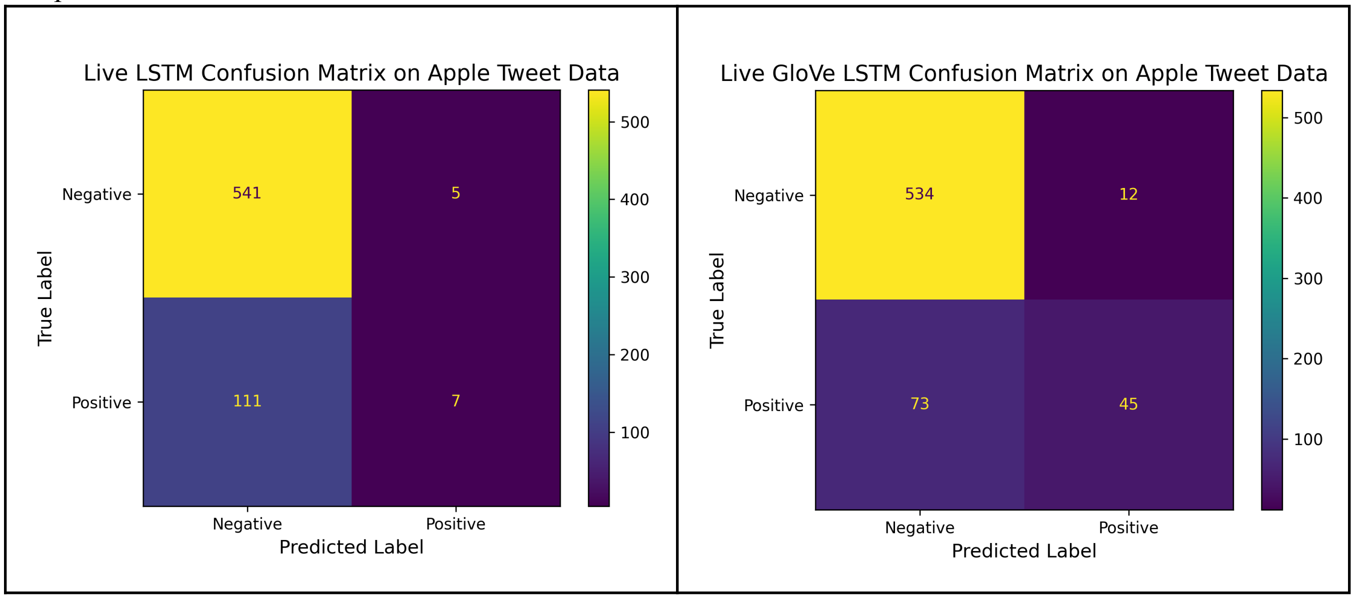

Figure 8a and 8b: Live Confusion Matrices on Apple Tweet Data. The confusion matrix from the non-GloVe LSTM with Live Data on the Apple testing data had 7 true positives, 541 true negatives, 111 false negatives, and 5 false positives. As such, the model had a testing accuracy of (7 + 541) / (7 + 541 + 111 + 5) = 82.53%. Meanwhile, the confusion matrix from the GloVe LSTM with Live Data on the Apple testing data had 45 true positives, 534 true negatives, 73 false negatives, and 12 false positives. As such, the model had a testing accuracy of (45 + 534) / (45 + 534 + 73 + 12) = 87.20%.

Exposing the model to a small portion of the Apple Tweets relative to the original training data led to a 4.6% improvement in accuracy when evaluating the holdout 80% of the Apple data on the LSTM model using our embedding, and a 3.1% improvement in accuracy when doing the same on the LSTM model using GloVe embeddings. Both LSTM models still exhibited high rates of false negatives, with a slightly worsened negative predictive value for the LSTM model using the embedding derived from the train data, and a better negative predictive value for the GloVe model using the live data. Interestingly, the previous trend of the GloVe embeddings reducing false positives at the cost of increasing the number of false negatives relative to the plain LSTM is reversed when adding in live data, which might be due to how the pre-trained embedding responds to additional data compared to the embedding built on our training data.

Conclusion / Future Research

On balance, when comparing our different modeling and improvement techniques on the cross-dataset problem, we get a few key insights. First, we note that in cross-dataset problems, a purely mathematical model has a distinct pattern of faulty classifications in this problem compared to RNNs, which could be leveraged in a consensus-based approach to classifying text across contexts if we can use weighting or other types of models with distinct result profiles. We see that pre-trained embeddings slightly improve the issue of jumping across contexts, but that there is a ceiling to raw accuracy in the case of jumping across contexts, which in our case was around 85%. Next, we see that class imbalance in testing data can make model evaluation and tuning less straightforward, though this is not as large an issue as class imbalance in training data. Lastly, it seems that the simplest solution is the best one in terms of the cross-dataset relevancy problem, as even a small addition of relevant data in training improves results immensely on a holdout test set.

As for future research, we would suggest testing the model on other slightly more related datasets, with the understanding that breadth doesn't necessarily make for good training data. For example, we could consider testing Apple tweet corpuses from different time periods to gain insights on how customers' attitude towards Apple products change over the years. We could also consider testing on datasets involving analogous customer sentiment-based contexts like Samsung, Zoom, etc since sentiment analysis is an important and universal technique that businesses use in general to better comprehend the market and make informed strategic decisions. Another angle that we would explore in the future given more time and compute is a wider range of individual models and their hyperparameters to get a more thorough understanding of the potential for consensus-based methods or sequential methods like boosting and bagging to outscore individual models.

Group Statement

We both split the work fairly equally, with Aadharsh focusing on tweaking the embedding for the non-GloVe LSTM model, a CNN classifier that we didn't end up using, the improvement techniques, and workflow of the models and Jei focusing on the dataset preprocessing, data interpretation, Naive Bayesian classifier scoring, and some logarithmic regression classification and decision tree techniques that we didn't end up using.

Bibliography

K, Jain. *Twitter Sentiment Analysis LSTM*, ‘apple_twitter_sentiment_texts’ Version 1, https://www.kaggle.com/code/kritanjalijain/twitter-sentiment-analysis-lstm/input. Kaggle, 2022.
M, Tariq. *Sentiment Dataset with 1 Million Tweets*, https://www.kaggle.com/datasets/tariqsays/sentiment-dataset-with-1-million-tweets, ‘dataset.csv’ Version 1, Kaggle, 2016.
H, Jansma. *Apple Sentiment Analysis*, https://harrisonjansma.com/apple. June 20, 2018.
D, Kaur. *Sentimental Analysis on Apple Tweets with Machine Learning Technique*, https://ijcset.net/docs/Volumes/volume7issue9/ijcset2017070901.pdf. IJCSET, 2017. ISSN: 2231-0711, Volume 7 Issue 9, 76-78.
Jeffrey Pennington, Richard Socher, and Christopher D. Manning, *GloVe: Global Vectors for Word Representation*, https://nlp.stanford.edu/projects/glove/ 2014.