Final Project Report

By Brant Wesley and Liam Pavlovic

What did you do and why is it interesting?

For our project, we created several models for performing advanced sentiment analysis. Unlike the sentiment analysis tasks that we saw in class, we categorized sentiment into multiple categories beyond positive/negative: sadness, fear, disgust, neutral, surprise, happiness, anger, and love. We used three different models (Naive Bayes, Feedforward Neural Net, and RNN with LSTMs) to assess how performance on our test data varied across different model architectures. Once we determined how our models worked on the training and testing data, we used twitter data scraped using the tweepy API to see how our models performed on real-world data. Using hashtags, we selected groups of tweets that we expected to provoke a certain kind of emotional response, targeting emotions represented in the labels our models recognize.

This project is interesting for several reasons. Firstly, it involves sentiment analysis in a more nuanced context than what we have previously seen in this class.These classifications have more potential applications than their binary counterparts. For example, a PR team trying to gauge the general public's response to a certain event probably wants more information than just a 'positive' or 'negative' label. After all, the tactics for dealing with a saddened consumer base are much different than those for dealing with an angry consumer base. The potential applications are further extended by the inclusion of a 'neutral' label. This label allows our models to differentiate between emotive and neutral sentences in text. So, our model can help editors quickly scan a document to identify problematic sentences in text that is intended to be objective and detached.

This task also demonstrates how different models, especially neural ones, are more suited to certain tasks than others. By classifying texts into one of many different categories, we see how different models adapt to multiclass classification better. Naive Bayes performed significantly worse than both neural models, despite being a popular choice for binary sentiment analysis. RNNs with LSTMs performed better on the test data taken from our datasets, but did not generalize well to the data scraped from twitter. The feedforward model was marginally less accurate than the RNN with LSTMs on the test dataset and generalized to the twitter data significantly better. Furthermore, this project explores how models trained on datasets perform on real-world unlabeled data from Twitter. Though datasets are often very useful for testing the efficacy of a given model, they may not necessarily reflect how the model would perform in data that isn't curated and labeled. This can be seen through LSTMs failure to generalize to the twitter data as well as the feedforward network. Additionally, using Twitter data makes it more likely to come across things like unknown words and misspellings that may not be present in some datasets, so it helps to determine how robust a model actually is.

How do your model(s) work? (explain)

The data we use to train and evaluate our models (links in works cited) proved to be fairly unbalanced and in need of quite a bit of preprocessing in order to make them suitable for use with our models. In particular, the daily_dialog dataset had poor and inconsistent grammar usage. So, we began by standardizing the grammar usage and tokenizing the dataset. After cleaning the datasets, we stemmed the data and added negation tags in order to help improve the performance of our models. Some of our data also contained the emotion labels as numerical encodings, so we also had to convert these numbers to their corresponding emotion. Also, the “joy” label in the kaggle dataset was switched to a “happiness” label to better align with the labels of the daily_dialog dataset.

We also found that while most of our sentences had lengths in the range of 10 - 50 words, there were a few of much greater length, with the greatest being 287 words long. So, we decided to remove all data points with greater than 50 words from the DataFrame in order to make the data padding for our neural models easier and more efficient. After doing all of this preprocessing of the data, we found that the data was heavily unbalanced, with the bulk of the data belonging to the neutral category and the happiness category also having quite a bit more data than the other emotions. So, we removed half of the neutral-labeled texts in order to bring it down to a size much closer to, but still greater than, the size of the next most frequently occurring label (happiness).

Our models take in preprocessed data (see details above) that is originally in the form of a pandas DataFrame. This dataframe was split into training and test sets using sklearns(train_test_split).

The Naive Bayes model takes in the training data as a list of sentences(represented as a list of tokens) with corresponding emotion labels. The model iterates over all the sentences in the training set, counting the number of occurrences for every token under every label. The model uses these counts to calculate the relative probability that a given sequence belongs to a each label and classifies the sequence into the most likely label.

The Feedforward model and the RNN both take in padded sequences. These sequences are first tokenized and encoded using a Keras tokenizer. The tokenizer only recognizes the top 15000 most occurring words in the dataset. This helps accommodate for unknown tokens in the future. Each sentence is padded to the length of the longest sentences (50 characters) using the padding (0) token. The labels for the text are encoded into one hot vectors using Keras’ to_categorical function. Both neural models begin by receiving the padded sequences and passing them into an embedding layer, which stores embeddings for each token number. The embeddings are trained at the same time as the models. This helps ensure that these embeddings are tailored for this specific task.

In the Feedforward model, after the embedding layer, the input is passed to a flatten layer in order to create one long embedding for the input text. The flatten layer feeds into two hidden layers of decreasing size (200 then 100). Both of these hidden layers use ReLu activation. The second hidden layer connects to an output layer which uses softmax.

For the RNN, after the embedding layer the input is passed through a 56-wide LSTM layer to a softmax output layer. The output layer is reached after the RNN recurs through each word within the given sentence.

What are your results/conclusions?

After training and hyperparameter tuning all of our models, we found the RNN (Marge) to be the most accurate model, followed by Feedforward (Bart), and then Naive Bayes (Homie). On our test data, Marge had an accuracy of .764, Bart had an accuracy of .732, and Homie had an accuracy of .577. These are not particularly unexpected results, since multinomial classification in sentiment analysis is fairly nuanced. When tested with tweets (which are of a different domain than the training/testing data), we found Bart to have the best performance. Sentiment analysis often uses bag-of-words, suggesting that the order and proximity of tokens does not matter much for this task. We believe this is why Marge had only marginally better performance than Bart. The gap was likely further closed by the addition of negation tags, which were given to both models and contain most of the relevant information from earlier tokens.

To see how our models perform on data from outside the domain of the training data, we used tweepy to collect tweets based on recently trending hashtags that we thought would provoke specific emotional responses. Since our models were trained on data which larged consisted of dialogs, we did not expect exceptional accuracy since the structure and conventions of tweets are quite different from those of dialogs.

In order to assess the classifiers' ability to detect anger, sadness and surprise, we gathered tweets with the hashtag #DaunteWright, as we expected the murder of Daunte Wright to provoke outrage. Bart performed relatively well on these tweets, though not perfectly, since the majority were tagged as "neutral". The heavy neutral tagging is likely due to the heavy concentration of neutral training data and the presence of news headlines that discuss the subject in an unbiased manner. Despite tagging many as neutral, Bart's next most frequent label for the hashtag was anger, which we expected to be the most common one. The third most common tag was happiness which was alarming, but this was likely due to the unbalanced dataset having a heavy bias toward the happiness and neutral tags. Unlike Bart, Marge performed very poorly on this hashtag with happiness being labeled far more often than anger. That said, Marge, like Bart, predicted neutral for the vast majority of tweets.

Additionally, we used tweets with the hashtag #TommyVlog to gauge responses to a Minecraft YouTuber's vlog. We selected this hashtag because of Minecraft YouTuber fans' tendency to over-emote about relatively mundane occurrences. For these tweets, Bart and Marge had fairly similar performance, as the most common labels were neutral, then happiness, and then anger in a relatively distant third.

Future experiments/work? (What would you want to do next, given more time?)

There are several things that we could do to improve upon our project given more time, For one, our data was very unbalanced before we preprocessed it with a vast majority of data points having the neutral label; so, it would likely benefit our model to find more data for the features with less data, such as love, anger and sadness. Additionally, we could spend more time developing models with more complex architectures and more hidden layers in order to increase the accuracy of our models, although this will present an issue of diminishing returns after a certain point. We did experiment with a more complex feed-forward network, but this mainly just caused the model to overfit the training data. We also could spend more time testing our models' performance on tweets, as we feel this is a good way to gauge the actual efficacy of the models, however we could not explore this area in particularly great depth due to time constraints imposed by the free version of tweepy.

This project could also benefit from the curation of an emotion-label dataset of tweets, as this would allow the models to train on data closer to its intended application. We did not have the resources to curate such a dataset ourselves, but a group with more money, time and employees could likely afford to.

Works Cited

https://python.plainenglish.io/scraping-tweets-with-tweepy-python-59413046e788

https://towardsdatascience.com/a-beginners-guide-on-sentiment-analysis-with-rnn-9e100627c02e

https://stackabuse.com/python-for-nlp-movie-sentiment-analysis-using-deep-learning-in-keras/

Datasets:

https://www.kaggle.com/praveengovi/emotions-dataset-for-nlp

http://yanran.li/dailydialog.html