Brody Wills

Task:

This project examines tweets about six major U.S. airlines and determines the sentiment (positive or negative) of the text using a Recurrent Neural Network and a Convolutional Neural Network. This can be useful for determining popularity of airlines or analyzing what people like and dislike about an airline with further text processing.

Dataset:

The dataset was obtained from Kaggle. The data originally came from Crowdflower's Data for Everyone library. This dataset is a CSV file containing tweets about six major US airlines. The tweets were scraped from twitter in February 2015. Along with the tweets, the sentiment of the tweet is provided as either positive, neutral, or negative. This original data set was reformatted for this project. The unnecessary columns were removed, and the remaining columns were renamed. The neutral tweets were removed from the set due to the low number of these tweets. Finally, the sentiments were converted from text to binary (1 for positive, 0 for negative).

Preprocessing:

For both algorithms, the same preprocessing steps were taken. First the data was loaded into a pandas dataframe object. The tweets were then cleaned in a few steps. First, all text was set to lowercase and usernames were removed from the tweets. Then, additional symbols and punctuation were removed. Using ntlk's stopwords module, the most common stopwords were removed from the tweets. Finally, using ntlk's WordNetLemmatizer, the words in each tweet were lemmatized. After the cleaning process, the data was split into 80% training and 20% test data using sklearn's train_test_split function. Finally, keras' Tokenizer was used to tokenize the tweets based on the hyperparameters max_features and max_length.

RNN:

The Recurrent Neural Network was created using a keras Sequential model. The first layer is an Embedding layer to turn the tweets into feature vectors. Next, a LSTM (Long Short-Term Memory) layer is added with dropout and recurrent_dropout each set to 0.2. Finally, a Dense layer is added with a sigmoid activation. The model is compiled using a binary crossentropy loss function, and the adam optimizer from keras.

CNN:

The Convolutional Neural Network was also created using a keras Sequential Model. The first layer is again an Embedding layer to turn the tweets into feature vectors. Next, a Conv1D (1D convolutional) layer is added with a kernel size of 5 and an elu activation. Then, a GlobalMaxPooling1D layer is added to

downsample the results. Finally, a Dense layer is added with a sigmoid activation as well. The model is compiled the same as the RNN.

Training:

Both models are trained using keras fit function with a 20% validation split. The Convolutional Neural Network was trained with 3 epochs, while the Recurrent Neural Network was trained with 2 epochs. During training, different values of the hyperparameters were changed until the best results were achieved. Any increase or decrease in the number of epochs, number of features, or the max feature length results in either a decrease in accuracy and F1 score or a very similar score.

Results & Observations:

For both algorithms, accuracy, loss, F1 score, and time taken were recorded and examined. The best results are displayed below.

	RNN	CNN
Accuracy	91.29	90.52
Loss	23.78	28.11
F1 Score	75.46	74.27
Time Taken (seconds)	150.29	15.32

As shown in the table above, the accuracy and F1 score of each algorithm is very similar. However, the RNN takes 10 times the amount of time to run, for only a small increase in accuracy and F1 score. Overall, for a very large dataset, the Convolutional Neural Network is the better algorithm for this task as it achieves a similar F1 score but takes much less time.

Challenges & Obstacles:

The first challenge was finding a dataset that included enough samples and each sentiment was balanced. The other main challenge was optimizing the hyperparameters to get the best results for each algorithm. The solution to this problem was to test various parameters and see how the loss, accuracy, and F1 score changed. The issue with this solution is that some algorithms may take a while to run, especially with a higher number of epochs.