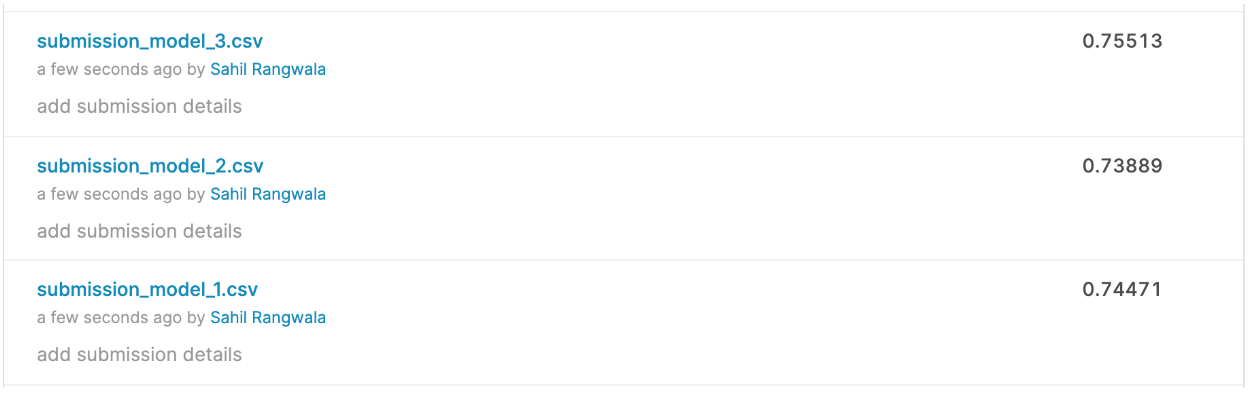
Sahil Rangwala
MSDS_422

Natural Language Processing with Disaster Tweets Write Up

This week we explored how to build a machine learning model that predicts which Tweets are about real disasters and which one's aren't. Twitter is a great resource in news and media for real time disaster events. The hardest component of tweets is that it can range in a wide variety of formats and characters. It is great that a lot of material is merely texts of characters and special characters so it can be easy to clean.

First step is to clean the data and normalize the characters by replacing the unnecessary special characters. By doing this, we can then conduct EDA and other data insights regarding the tweet dataset. I saw that the word 'debris' had the strongest chance of being associated with an actual disaster tweet. Additionally, 'devastation' was the most likely word being improperly associated with disaster tweets. This is understandable, as debris has a negative connotation but is strictly used in situations where debris is actually seen due to the cause of a disaster. Devastation is a synonym of disaster, yet it can be used in many generic situations outside of weather induced disasters. Due to this, it makes sense that devastation does not have many disaster related incidents and could generically be related to other negatively connotated events.

The first model is a simple RNN with 64 nodes with a few layers. The accuracy, validation accuracy, and loss are recorded to showcase the strength of the model. The first model had the least amount of layers, yet the accuracy was still very high (98%). From following the example in the textbook, I added another layer to make a deeper recurrent neural network. From the metrics on the second model, it showcased a slight dip in the accuracy (97%). I could have added all of the necessary layers for this model, however, I am going to explore this idea further by creating another model with an extra replicated layer. This third model showcases a 'deep' RNN by having multiple replicated simpleRNN layers. My assumption was that this would perform the best as it has the most applicable layers, however, it actually performed the worst. The metrics indicated a slightly lower accuracy metric in comparison to model 2. The kaggle test case will provide further insight on the model performance. The scores are below:


submission_model_3.csv 0.75513
a few seconds ago by Sahil Rangwala
add submission details

submission_model_2.csv 0.73889
a few seconds ago by Sahil Rangwala
add submission details

submission_model_1.csv 0.74471
a few seconds ago by Sahil Rangwala
add submission details

The model 2 scored the best, which is great to see as it follows a logical explanation. The model 3 began to overfit due to the additional layer that was added. It is great to see the minute differences are captured within Kaggle.