Data Science Student Championship

First of all, I want to thank MachineHack and Praxis Tech School for hosting such an amazing competition. The last 1 month was quite exhausting for me, with a steep learning curve and tons of ideas I wanted to try out. In the following, I will try to summarize the main points of my solution.

Solution Approach:

The following is a brief summary of my solution.

* Preprocessing of Dataset
* CNN model
* LSTM - RNN model
* Blending strategy
* What didn't work for me

Preprocessing of Dataset

* Thanks to Debmalya Ray for providing a public preprocessing notebook

(preprocessing1.ipynb)

* Using this, the Dataset was cleaned by removing unwanted words like stopwords, HTML tags, numbers, links, etc
* The dataset was further preprocessed using stemming and lemmatization techniques and removing frequent words

CNN model (LB 0.5674):

* 1D Convolution: To extract local patterns from the sequence using a sliding window (filter

size 3) and ReLU activation for non-linearity.

* Global Max Pooling: To Capture most important feature in each channel for robustness.
* Softmax Activation: Produces class probabilities for multi-class classification tasks.
* 4 epochs training: Less epochs to prevent overfitting

LSTM-RNN model:

* I replaced the Convolution layer with the LSTM layer with “return\_sequence=True” to learn long-term dependencies within the sequence.
* The other layers remain the same as for the CNN model.

Blending strategy:

Both CNN and RNN model has their own strengths; CNN captures local patterns using 1D convolutions, while the RNN(LSTM) excels at capturing long-range temporal dependencies across the entire sequence.

Both the models are merged using concatenation. This combines the strengths of both models. These concatenated features are then fed into a dense layer (Dense(9)) with softmax activation.

This joint learning process in the dense layer resulted in more accurate classifications compared to using a single model CNN or RNN **(LB 0.5674 -> 0.5722)**

What didn't work for me

I tried a variety of different model structures, some of them are Multinomial Naive Bayes, XGBoost classification, Support Vector Machine, etc, but none of them were able to cross an Accuracy of around 0.53. So, I had to shift to neural networks to tackle this.

In neural networks, I had to keep the model simple, as increasing epochs (more than 3) or adding multiple layers didn’t help improve the accuracy; instead, it started to overfit on the training data.