**Answering A-G Questions on Style Seat Data Science Problem**

**A – Is This a Multi-label Classification Problem:**

The answer is yes. [Multi-label](https://en.wikipedia.org/wiki/Multi-label_classification) classification is a generalization of multi-class classification which is the single-label problem where instances are categories into precisely one of more than two classes. In the multi-label problem, there is no constraint on how many of the classes the instance can be assigned to, it can be one, two or many labels in the output data used for training. In this problem each instance with the given service\_name is assigned to one or more categories separated by comma in service\_category field.

**B – Show and explain any data processing/feature engineering performed.**

Data Clean-up: In lines 6-7 I performed data preprocessing and clean-up by removing punctuations and English stop words from the service\_name column. Also, words with length less than 3 are removed. The same data clean-up is also performed on the testing data.

Preparing training data: In lines 13-21 training input (serive\_name column) is preprocessed first by tokenizing and turning each text into sequence of integers and then by padding and making sure each training input makes up a fixed length array of integer numbers. The chosen length is based on the longest input.

Vectorizing output labels: The service\_cateogry column which is the desired output needs to be converted into vectors with labels appearing to be one for instances for which these labels are assigned and zero for others. Since it is multilabel classification, each instance vector can have multiple 1’s. For this conversion I used CountVectorizer from Keras.

**C - What ML model did you use and why?**

I used stacked two-layer LSTM with an embedding layer on top and a dense layer for classification as the last layer. Embedding layer allows to convert input text that now is tokenized into embedded vectors and learns these word vectors in the same way weights are optimized in neural network jointly with the main task which here is classification. LSTM or in general RNN work well with sequence data like text because it maintains state of information of what is has seen so far, and LSTM is chosen because it does not have vanishing gradient problem of RNN and gives good results with text data. The main reason that two layer LSTM or stacked LSTM is used is to give model more capacity to learn. Finally, the dimension dense layer is the output layer with the number of categories as its number of outputs. For dense layer sigmoid is used as activation function assuming that occurrence of each category does not depend on occurrence of others.

**D - Explain any measures you took against overfitting.**

I used two techniques to overcome overfitting. One using dropout layers and second having a hold-out set called validation set and evaluating model based on that. Two dropout layers with the rate of 0.5 are used one after each LSTM layer that randomly drops a number of output features of the layer during training. Also, 10 percent of training data is randomly selected as validation set, and the validation loss along with the training loss are plotted per epoch. This graph shows that validation loss is less that training loss up till the 12th epoch and after that it surpasses the training loss although it still keeps going down. I also used callback in during training in epochs with patience of 4 epochs to automatically stop training when validation loss does not reduce for 4 consecutive epochs.

**E - How did you assess the efficacy of your model?**

The metric I used is F1-socre to evaluate performance of trained model on the validation set. F1-score is used to avoid the problem of using accuracy or either precision or recall in imbalanced datasets. This F1-score is micro averaged to use it as a metric for multi-class classification and is calculated by counting the value of true positives, false positives, true negatives, and false negatives.

To choose the best threshold value and use it to predict categories in line 29 I run a loop each time with a different threshold ranging from 0.1 to 0.9 where in each iteration only categories that their predicted probabilities are higher than the threshold are predicted for each instance as 1, the rest predicted as zero. Then, f1-score is measured by comparing the categories predicted in each iteration and the original ones in the validation dataset. The best f-1 score that I achieved on the validation set was 0.7244 with precision of 0.7422 and recall of 0.7074 having both higher than 70% with the chosen threshold of 0.3. Increasing or decreasing threshold from 0.3 can increase precision or recall but the overall f1-score goes down.

**F - Run the model against the data in ​testing\_data.csv​ ​and provide the results**

**in ​yourname\_​ ​submission\_problem\_1.csv**

The file is generated with predictions based on trained and persisted model in the same format as the training data. The file name is yahya\_submission\_problem\_1.csv. The trained model that is used to generate predictions is placed in folder multilabel-LSTM-model in Tensorflow pb format.

**G - What further enhancements to the model would you consider if you were given**

**more time?**

Two things I would like to try if I had more time.

One is that instead of considering each service name as a text, looking at it as combination of sub-service names and using them as features because these sub-service names get repeated across different instances. Basically, using these sub-service names as categorical features and turning them into vectors that are non-zero only for service names that they are part of. These features can be used in addition to considering each service name as a text that I used.

Second, I would like to try more models like 1D convolution networks, or more variations of LSTM model like using bidirectional RNN that considers sequences in both directions. Also, it would be interesting to see how the model would perform if instead of training embedded vectors, a pretrained embedding like GloVe would be used.