LSTM – Content for Report

[LSTM Neural Networks](https://en.wikipedia.org/wiki/Long_short-term_memory)(**L**ong **S**hort-**T**erm **M**emory) are a particular type of recurrent neural networks, that manage to keep contextual information of inputs.

LSTM networks have some internal **contextual state cells** that act as long-term or short-term memory cells. The output of the LSTM network is **modulated** by the state of these cells. This is a very important property when we need the prediction of the neural network to depend on the **historical context** of inputs, rather than only on the very last input.

Not been able to reach the expected accuracy with conventional ML models as SVM, NB, Ensembles, we decided to use LSTM to solve this Automated Ticket Classification problem. The architecture used is as below –

tf.keras.models.Sequential([

tf.keras.layers.Embedding(vocabulary\_size, output\_dim=100, input\_length=maxlen),

tf.keras.layers.LSTM(64, return\_sequences=True,dropout=0.2),

tf.keras.layers.BatchNormalization(),

tf.keras.layers.TimeDistributed(Dense(32, activation='relu')),

tf.keras.layers.Flatten()tf.keras.layers.Dense(24, activation='softmax')])

There were total 74 categories in the data but the data was imbalanced across different categories with some categories having as high as 3900 records while around 49 categories contained less than 50 records. When executed with all 74 categories LSTM model reported a low accuracy ~53%. Thus a threshold for Automation criteria was setup. All categories with records less than 50 records were merged into single category and this category was not considered for automation or model training. The tickets in these categories should be considered for manual triaging. 24 categories with ticket count more than or equal to 50 were used for LSTM model training. Model was trained with following setting of optimizers.

1. SGD
2. RMSP
3. Adam
4. Nadam
5. Adagrad

And different learning\_rates [0.001, 0.01,0.1]. The model was trained(with validation split of 0.2) for varying values of batch [100,200,250,500]. The model weights for each execution was saved and was reloaded for next execution for respective optimizer/learning rates.

Of the above parameters Adam with learning rate 0.001 and batch of 200, we have been able to attain a Training Accuracy of 90% and Test Accuracy of 88%.