FFNN Result

Description

We utilized all fake, real, mix dataset for training. For each dataset, we implemented a custom encoding and decoding function to process corpus into bio-separated data, where a row includes all the tokens of a bio and its label. We performed padding on all bios to ensure they form a sequence of 865 tokens, which is the length of the longest sequence in training data. We constructed a FFNN model with an embedding layer, multiple MLP+Tanh+Dropout layer, and a sigmoid layer.

1. Hyper-parameters:

batch_size=64, embedding_dim=128, epochs=40, sequence_size = 865, vocab_size =71882, dropout=0.3

The parameters we utilized for each Linear Layer is shown below

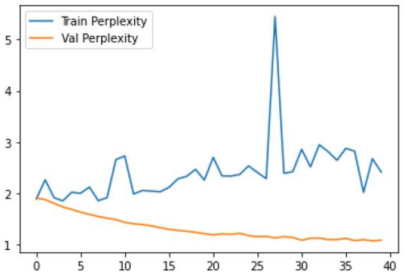
```
self.embeddings = nn.Embedding(vocab_size, embedding_size)
self.layer_1 = nn.Linear(input_size * embedding_size, 1024)
self.layer_2 = nn.Linear(1024, 512)
self.layer_3 = nn.Linear(512, 256)
self.layer_4 = nn.Linear(256, 128)
self.layer_5 = nn.Linear(128,64)
self.layer_6 = nn.Linear(64,16)
self.layer_out = nn.Linear(16, 1)

self.relu = nn.ReLU()
self.tanh = nn.Tanh()
self.dropout = nn.Dropout(p=0.3)
self.batchnorm1 = nn.BatchNorm1d(512)
self.batchnorm2 = nn.BatchNorm1d(128)
self.sigmoid = nn.Sigmoid()
```

Training Time: ~1h (1.5 minute per epoch)

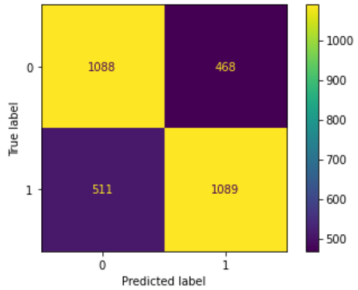
Test Accuracy: 0.69

2. Learning Curves:

For our model, perplexity is calculated as $\exp(BCELoss/data\_num)$



3. Confusion Matrix



4. Limitations

According to learning curves, the model overfits a bit in latter epochs. But we managed to select the best model with best validation performance in earlier epochs to avoid using overfitted model. Also, the number of parameters in our model is massive due to large number of layers, which is a design decision that achieves best performance after several experiments.

LSTM Result

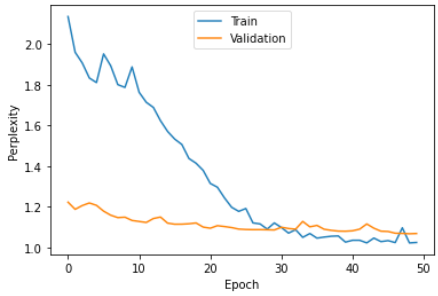
Description:

After several experiments, we noticed that the data preprocessing method we utilized in FFNN models gives poor performance on LSTM models. The explanation might be that sequences are padded too heavily and the vocabulary size is too massive. Therefore, we padded and truncated all sequences to a fixed length of 300; we eliminated all numbers and assigned them with the token <NUM>; we limited the vocabulary to the 30000 top frequent words. We only used real and fake dataset for training. We constructed a model with an embedding layer, a LSTM layer, a Linear Layer, and a sigmoid layer.

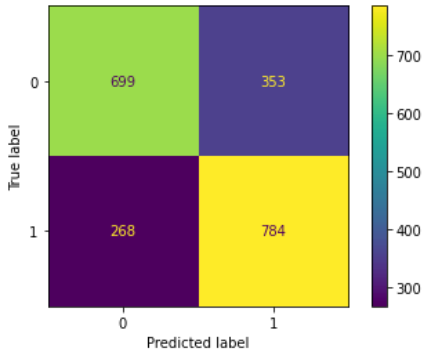
1. Hyper-parameters:

batch_size=64, embedding_dim=256 , epochs=50,sequence_size = 300, vocab_size =30000
Training Time: ~20 mins (0.4 minute each epoch)
Test Accuracy: 0.76188

2. Learning Curves:



3. Confusion matrix



4. Limitations

According to learning curves, the model overfits a bit in latter epochs. In the future, we are considering regularizing the model with methods like dropout.

Final Results

In the end, we selected LSTM as our final model due to its better performance on test dataset. As mentioned in previous part, the limitation of the LSTM model is overfitting. We are experimenting on adding dropout, tuning parameters like lr and so on. In the end, we saved our final model and used to predict the label of the blind dataset.