Task 3

Evaluation of each regression model

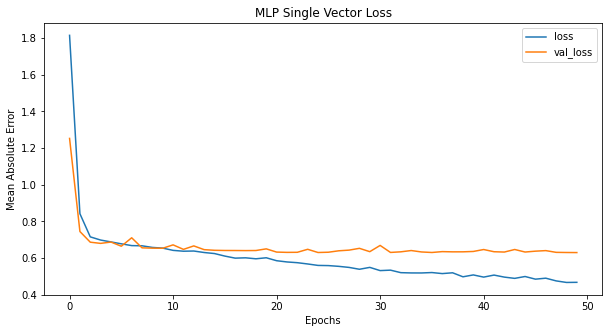
The focus throughout the development of each model in task 1 was to focus on minimizing the loss on the validation data. This proved rather challenging due to the small size of the data set. Each model in turn achieved a very low loss score on the training data compared to a comparatively higher loss when validating. I believe is this is simply due to the training data size causing models to overfit rather quickly rather than generalizing to the problem. To that end I adjusted each model’s hyperparameters in an attempt to improve mean absolute error in each model on validation as detailed below.

MLP Text to Vector

I found that the MLP single vector model trained incredibly quickly with low loss on the training data. However, I believe there isn’t enough training data to accurately describe the problem using a mean based vector. Averaging each input removes any context in a patient’s response. Take these two partial responses as inputs: ‘a previous bad experience’ vs ‘a bad experience’. These inputs contain almost identical vocabulary. However only one response is negative. Creating a numeric representation from these inputs may lead to closely spaced vectors whilst both are separate classes.

In order to improve the model I decided to use large output dense layers in the hope that these create more trainable parameters to adjust at a slower rate. I also experimented with Dropout layers (these randomly set inputs to 0) in an attempt to prevent over-fitting, I also lowered the learning rate to 0.0002. However, the changes to loss across each epoch remained as before, starting with a sharp drop off followed by linear fall in training loss and a plateaued validation loss. It would seem taking the average of a pretrained vector greatly reduces the amount of meaningful data available for model training leading to a very quick but ultimately very non-descriptive models. As a company I would not take this approach further unless the dataset much larger and only if it been preprocessed to ensure there were few spelling mistakes and the responses shared many words in the pretrained vector data to produce meaningful averages.

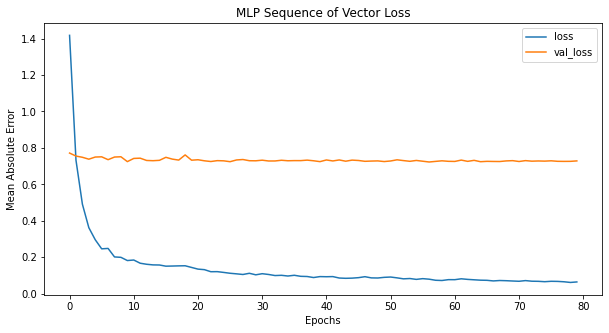
Figure 1



MLP Sequenced Vectors

This approach initially proved more promising. The training curve is much smoother and the final training loss is much lower. Despite this the final validation loss was higher than the previous model (0.73 vs 0.62). This approach could be explored further with a large dataset providing a wider range of vocabulary and a large vocab overlap across both datasets. However, the company should be aware that the training time is much slower with this model than the previous MLP model and it along with the embedding matrix size will grow with as the vocabulary increases (leading to a large increase in RAM usage). I varied the number of neurons in the dense layers along with experimenting with regularisers to keep the weights in layer minimal. However, this did little to improve the validation loss. I also initialized layer weight using he\_uniform. In order to get consistent loss results across multiple runs I has to increase the number of iterations in the model, I increased the epoch count to 80 and reduced the batch size to 32 to provide a higher parameter update frequency. Use of Dropout layers increased final validation loss suggest the model was not in a state of overfitting.

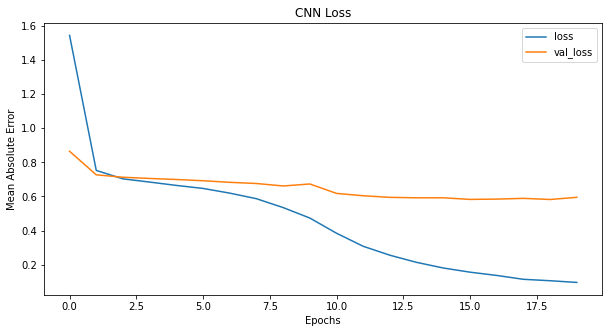
Figure 2



CNN Embedded Model

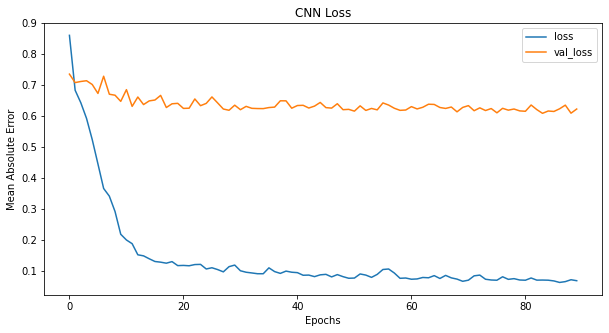
Initially this model looked as equally disappointing as the previous MLP using embedded. The training loss reducing quickly along with the final loss plateauing (Figure 3).

Figure 3



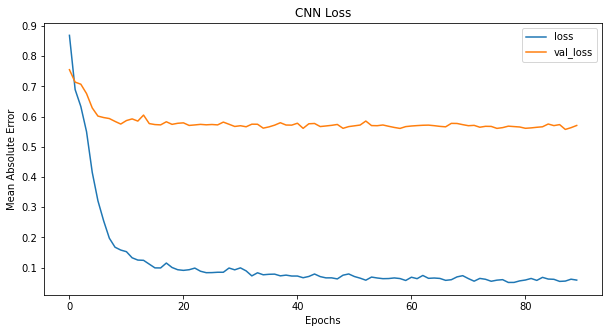
This curve looked as though the CNN was training to quickly overfitting on the training data. In order to improve this, I increased the number of filters in the Convolution layer from 100 to 1024, again to add more trainable parameters. I also attempted adding 2 Conv1D/MaxPooling1D pairs to the sequential model. I then increased the number of epochs to 90 and ran the model again. This did reduce the loss significantly by 0.1 on training, though I think it may have been too reductive as the validation loss was still 0.62

Figure 4



I removed one of the Conv/Pooling pairs reduced the learning rate 0.0005 and the Conv filters to 512 and opted to use global max pooling. The best loss I could achieve for CNN was 0.57 (this was the lowest across all models)

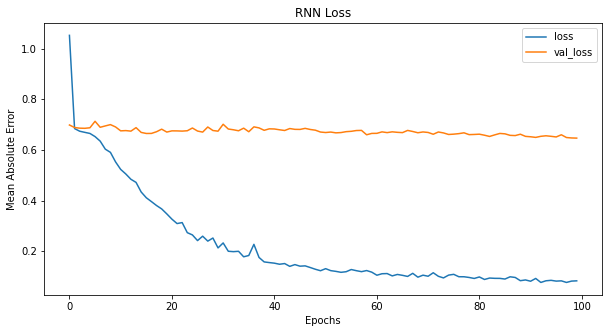
Figure 5



RNN Embedded Model

I attempted to achieve a lower loss than CNN using RNN and thought I would be successful given its ability to ‘remember’ data. RNN achieved a best loss value of 0.64 comparable to the first MLP model. The company could investigate this further again using a larger dataset. RNN also experienced the longest training time to achieve comparable results to the other models (27 mins to train and over 2hrs for larger LSTM output dimensions (512)) (Figure 6) I also trained different weight initialization methods as LSTM is sensitive to different initialization methods.

Figure 6

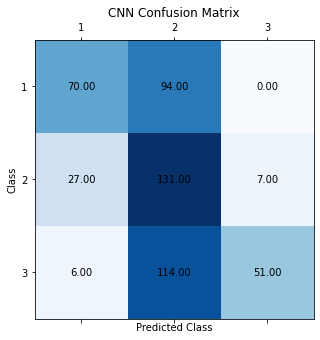


Overall best model

The main metric for evaluating each model was the error calculated from the loss function. CNN achieved the lowest mean absolute error (0.57) and therefore it is my chosen final model. However, had it been comparable to RNN or MLP embedded I still would have opted for CNN. CNN provided the best balance between feature extraction (Conv and Pooling) against training time. RNN also is not feasible at the moment for this company as the dataset is not large enough to be used for training on an LSTM based RNN and at present appears to overfit quickly. Also the company may want to rapidly iterate model development and RNN simply takes too long for to train with large LSTM output units and large datasets.

Figures 7 and 8 show the confusion matrices for the regression and classification models respectively.

Figure 7 Figure 8



These confusion matrices show that a model trained using a softmax activation, 3 output layer (Figure 8) more accurately predicts samples than a single output rounded to the nearest whole number. This makes sense as training on MAE to improve the predictions on a single output reduces the impact of training from the previous layers. Rounding is also less effective than a multiple output layer as the rounding may go one way of another if the sample is less discriminative. The softmax confusion matrix shows that TP samples are identified more often and the FP and FN samples are identified similarly but the model would require more training data to generalize more effectively to reduce these.

Conclusion

Overall I believe the project has been successful with the CNN model providing the most positive results.