Scott Lam-McGonnell
40227918
Coursework 02

Coursework 02

To keep comparisons fair, all models were trained on the same number of epochs (50), as well as having shuffle = true for all models to ensure an even spread of classes for each batch passed to the models.

MLP Vector

The base structure for this model is as follows:

Model = Sequential()
Model.add(Dense(64))
Model.add(Dense(32))
Model.add(Dense(1))
Optimiser = Adam
Loss = mean squared error

The numbers 64 and 32 for the dense layers were chose through trial and error, as these number of nodes seemed to give me the best results in terms of training loss and validation loss.

Embedding Method	Regularizer Used	Dropout	Eval MSE	Eval MAE
Word2Vec	N/A	N/A	0.643	0.641
GloVe	N/A	N/A	0.699	0.666
Word2Vec	N/A	0.5	0.605	0.664
GloVe	N/A	0.5	0.601	0.651
Word2Vec	L1	0.5	0.628	0.671
GloVe	L1	0.5	0.630	0.662
Word2Vec	**L1**	**N/A**	**0.569**	**0.629**
GloVe	L1	N/A	0.636	0.672

The dropout layers were added in between the first and second, and the second and third dense layers. Dropout and the L1 kernel regulaisation helped prevent the model overfitting on the training data, as the without them the model seemed to have a slight increase in validation loss as the epochs increased.

Sequence MLP

The base model architecture for this MLP was as follows:

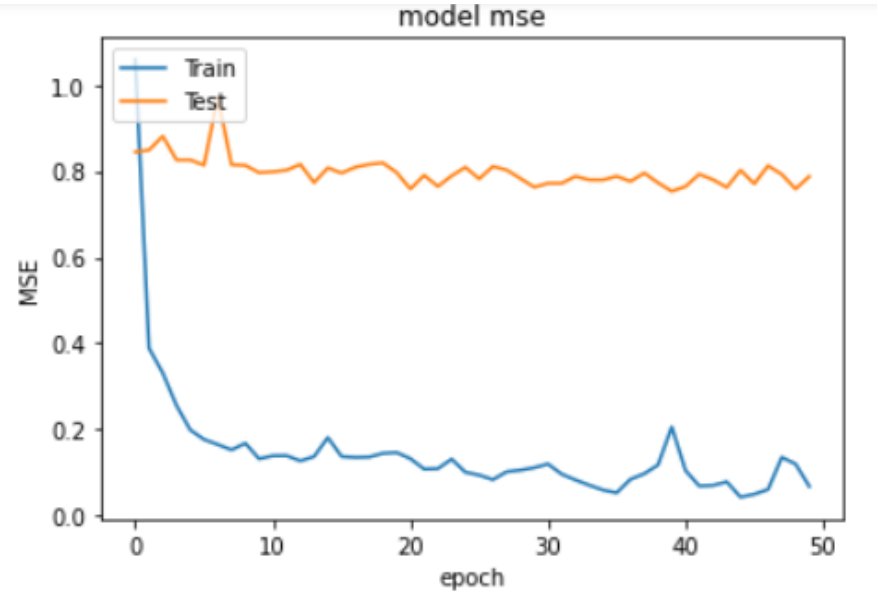
Model=Sequential()
Model.add(Flatten())
Model.add(Dense(64))
Model.add(Dense(32))
Model.add(Dense(1))

For this model, a very similar architecture to the one above was used, with the added addition of the embedding layer and no dropout layers. For the embedding layer, trainable was set to false for this model and all other subsequent models. This is because we are using pre-trained word vectors obtained from Google's Word2Vec file and the GloVe file, due to this the weights of the model shouldn't be updated.

Scott Lam-McGonnell
40227918
Coursework 02

Embedding Method	Regularizer Used	Dropout	Eval MSE	Eval MAE
Word2Vec	N/A	N/A	0.771	0.739
GloVe	N/A	N/A	0.775	0.739
Word2Vec	N/A	0.5	0.672	0.688
GloVe	N/A	0.5	0.853	0.799
Word2Vec	L1	0.5	0.685	0.714
GloVe	**L1**	**0.5**	**0.673**	**0.705**
Word2Vec	L1	N/A	0.788	0.749
GloVe	L1	N/A	0.817	0.753

The use of a regularizer helped somewhat to improve the models final loss results.



This is a graph showing the overfitting achieved by the model without the use of dropout layers, showing the importance of using them in task like this. With the use of dropout layers, the validation line fits much closer to the training line and decreases much more over time.

CNN

The base model for this CNN was as follows:

```
Model = Sequential()
Model.add(Conv1d(128, 5))
Model.add(Dense(32))
Model.add(Dense(10))
Model.add(Flatten())
Model.add(Dense(1))
```

These numbers again were found by trial and error, a higher or lower number of nodes in the conv1d layer and the dense layer were giving slightly worse outputs, this could be due to insufficient training epochs but to compare the models fairly I kept this number consistent throughout.

Scott Lam-McGonnell
40227918
Coursework 02

Embedding Method	Regularizer Used	Dropout	Eval MSE	Eval MAE
Word2Vec	N/A	N/A	0.496	0.597
GloVe	N/A	N/A	0.546	0.617
Word2Vec	N/A	0.5	0.505	0.585
GloVe	N/A	0.5	0.593	0.534
Word2Vec	**L1**	**0.5**	**0.463**	**0.569**
GloVe	L1	0.5	0.564	0.628
Word2Vec	L1	N/A	0.508	0.583
GloVe	L1	N/A	0.549	0.602

CNN's for text analysis can become very prone to overfitting, even though this model was tested without regularizers and dropout that was just for the sake of comparison. The regularizers and dropout layer are important to help prevent the model overfitting too much on the training data, which is why they give improvements to the evaluation errors.

RNN

The base model for this RNN was as follows:

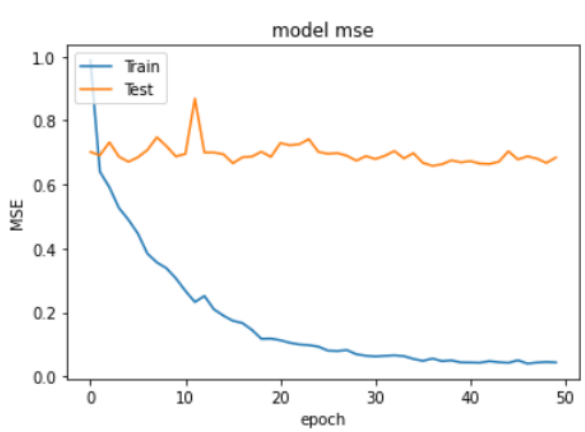
```
Model.add(LSTM(64))
Model.add(Dense(128))
Model.add(Dense(64))
Model.add(Dense(32))
Model.add(Dense(1))
```

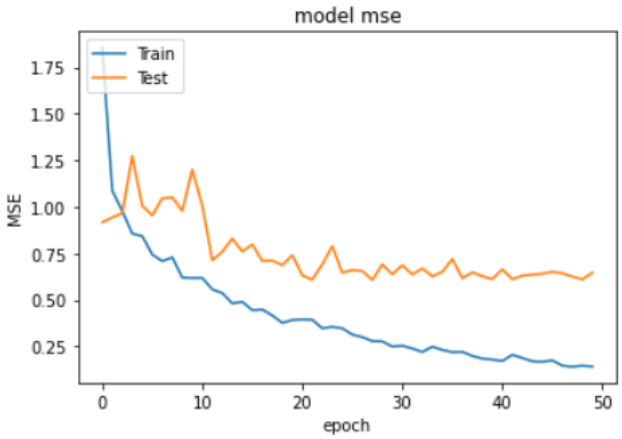
Embedding Method	Regularizer Used	Dropout	Eval MSE	Eval MAE
Word2Vec	N/A	N/A	0.689	0.649
GloVe	N/A	N/A	0.731	0.684
Word2Vec	N/A	0.5	0.669	0.660
GloVe	N/A	0.5	0.647	0.670
Word2Vec	**L1**	**0.5**	**0.570**	**0.615**
GloVe	L1	0.5	0.647	0.667
Word2Vec	L1	N/A	0.607	0.612
GloVe	L1	N/A	0.685	0.670

Dropout and regularizers were added to help reduce potential overfitting on the training data. This model performed best when using dropout layers, but not kernel regularizers.

Within the LSTM layer, dropout and recurrent dropout were both set to 0.2. 64 LSTM nodes were chosen for this layer as it provided good results, without the heavy computational load that comes with a larger number of nodes, however performance across the board may have benefitted from choosing a much larger number.

Scott Lam-McGonnell
40227918
Coursework 02





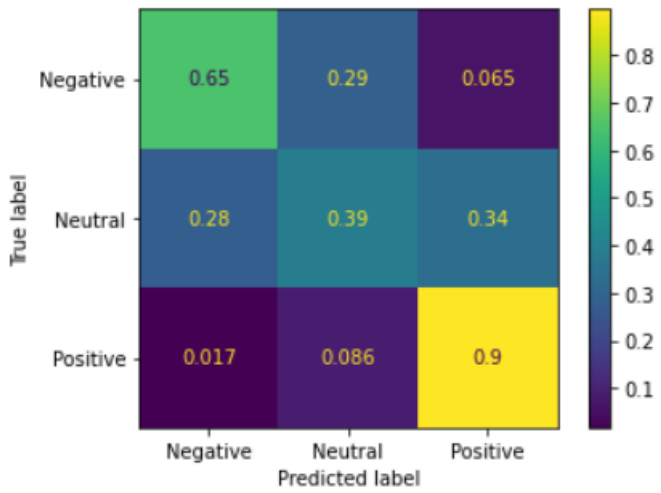
The above graph on the left was achieved using GloVe, and no dropout. Here we can clearly see that the model is overfitting to the training dataset, while making little improvement on the validation set, showing the importance of using dropout with this particular model. Similar graphs for the CNN and MLP models were achieved without using dropout.

Comparatively, the graph on the right with dropout included shows a much better validation fit, closer to that of the training, again this was a common theme with the CNN and MLP models

Chosen Model

The model I chose to implement for task 2 was a CNN model, using word2vec, l1 regularizers and dropout layers. This was chosen as it gave me the best values for both mean squared error and mean absolute error when performing the regression task.

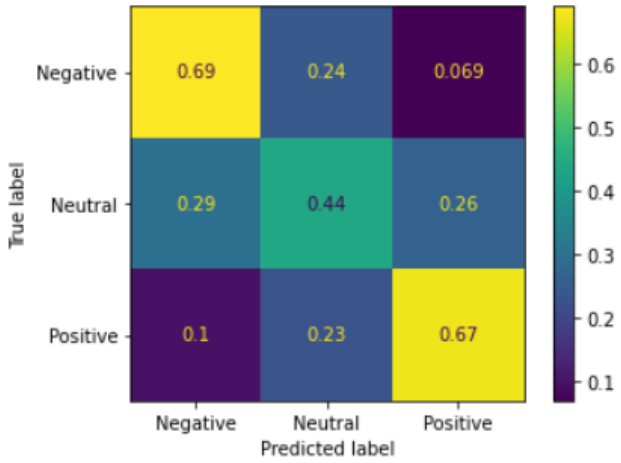
Results



The results above are shown from rounding the output from the regression model to the nearest integer, and comparing the results obtained from the predictions of the model with the true labels.

Here we can see that it correctly predicts the negative labels 65% of the time, the neutral labels 39% of the time and the positive labels 90% of the time. The overall accuracy of the model was 50.2%.

From this we can deduce that the majority of the time, the output of the regression model was 1 – 1.5 and 2.5-3. It appears the model has a difficult time deducing whether a review is neutral, as it appears to predict negative and positive for neutral reviews just as much. This is due to the fact that it Is hard to understand if a review is neutral, and the model is not understanding the context around certain words that may be contained within those reviews that would be applicable to a negative/positive sentiment, such as ‘good’ or ‘bad’. These words would be used heavily in their respective negative and positive reviews, so the model may associate most uses of these words to a negative or positive statement.



These are the results achieved from converting the CNN regression model into one of classification, done by changing the number of output neurons to 3 and the final activation function to softmax.

We can see here that the model still struggles with classifying neutral text sentiments comparatively to the other categories.

Improving the Model

It is clear that the model still suffers from overfitting issues, as the accuracy on the training dataset is much higher than that of the validation set. To help improve this, it would be appropriate to add more layers, and more dropout. More convolution layers could be added as well as training for mant more epochs, however this would be at the expense of computational time. Other regularizers could also be investigated to see if they would benefit and improve the current model. Other areas to improve upon that would help achieve better results would be finding larger training and testing datasets, so the model has more data to work with.

The success of this project is not too high, as the final accuracy of the classification model leaves a lot to be desired and a lot of room for improvement.