2/4/2024

Student Name: Ryan Mckee

Student number: 40294886

Assignment 2

CSC3066 Deep Learning (Fake News Detection)

**Model Analysis And Reporting Results From Task 1**

Initially, hyperparameters were set as follows due to time and computational constraints:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Batch size: 64 | Epochs: 10 | Learning rate: 0.05 | Optimizer: Adam | K-folds: 10 | Loss: binary cross entropy | ReLu hidden layer size 5 |

ReLU activation was chosen for its effectiveness in capturing complex textual patterns efficiently. Its non-linearity is crucial for this task, especially with high-dimensional 300-dimensional word embedding vectors produced by GloVe, making it advantageous over tanh and sigmoid activations.

Sigmoid activation in the output layer constrains outputs to a probability between 0 and 1, ideal for binary classification tasks like identifying fake information, hence employing cross-entropy as the natural loss function.

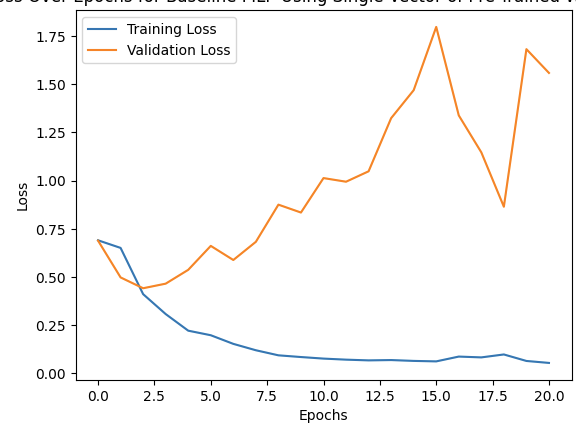
10-fold cross-validation was utilized during model training to accurately assess errors and accuracy on unseen training data. This approach was chosen due to the relatively small training set (2000 samples), with 10% reserved for validation to ensure sufficient data for validation without compromising training.

Further experimentation will refine these parameters and settings for an optimal model configuration. Other baseline parameters will also be explored throughout the project to achieve the best possible solution.

**MLP single flattened word embedding vector.**

Experimentation aimed at enhancing the performance of MLP models for text classification tasks, using flattened word embedding vectors involved a series of adjustments and techniques. Initial tests exhibited promising results, with an accuracy of 0.78 on the testing set. However, these experimental models encountered stability issues, notably observed in the learning curve, attributed to a mix of potentially too high learning rate and noisy training data containing stop words and redundant phrases, leading to overfitting. To address this, the learning rate was incrementally lowered from 0.05 to 0.001, resulting in improved stability and accuracy. Additionally, increasing the number of epochs facilitated convergence.

**A graph of a line

Description automatically generated with medium confidence**

Further exploration included architectural modifications by reducing the number of hidden layer neurons from 5 to 4 to mitigate overfitting. This adjustment not only enhanced accuracy and precision but also decreased false positives, fostering better learning fit. Regularization Lasso positively affected training performance leading increased learning curve stability and a 0.1 increase in convergence accuracies by increasing weight sparsity, Decreasing the batch size also tackled overfitting well consequently enhancing generalization performance giving training and validation a closer together fit, albeit with increased training time from 7.5 seconds originally to 8.4. Preprocessing steps such as removing stop words and lemmatizing were also employed to reduce data dimensionality and enhance training efficiency by retaining only relevant information for classification these data cleaning techniques combined with the other successful techniques resulted in a noteworthy accuracy increase to 0.81 from the original 0.67 that I began with and reduced training time from 8.4 seconds per fold to 6.4 seconds.

**A graph of a line

Description automatically generatedA graph of a graph

Description automatically generated**Incorporating these adjustments and preprocessing steps led to significant enhancements in stability, reduced overfitting, and improved performance on the test set making this MLP model viable for use by the client for classification of false information.

|  |  |  |
| --- | --- | --- |
| Final Model Testing stats | Predicted negative | Predicted positive |
| True negative | 203 | 22 |
| True Positive | 72 | 203 |
| Accuracy | 0.812 |  |
| Precision | 0.902 |  |
| Recall | 0.738 |  |
| F1 Score | 0.812 |  |

**MLP with Kera’s Embedding Layer**

A graph of a line

Description automatically generated with medium confidenceA graph of a graph with blue and orange lines

Description automatically generatedThis model builds upon the initial experimentation with the first MLP model which used flattened vector embeddings, The experimentation of this model revealed similarities in optimal hyperparameters. However, notable differences emerged. Utilizing the optimized hyperparameters, it became apparent that compared to the previous model, there was some overfitting. This can be attributed to the nature of the Keras embedding layer, which inherently seeks more meaningful sentence representations, capturing semantic relationships and contextual information more effectively within smaller dimensions. Consequently, these smaller dimension inputs contributed to overfitting in the current model hyperparameters due to the model being too complex for the embedding’s vectors input into it causing it to be an overfit model.

A graph of a graph

Description automatically generatedA graph of a graph with blue and orange lines

Description automatically generatedTo address the overfitting, the model’s neurons were reduced from 4 to 3 which returned similar results to the initial MLP model.

|  |  |  |
| --- | --- | --- |
| Final Model Testing stats | Predicted negative | Predicted positive |
| True negative | 196 | 29 |
| True Positive | 77 | 198 |
| Accuracy | 0.788 |  |
| Precision | 0.872 |  |
| Recall | 0.72 |  |
| F1 Score | 0.788 |  |

In conclusion, this model exhibits comparable performance metrics to the previous MLP. Notably, training time has increased from 6.4 seconds per fold in the previous model to 10.3 seconds in this iteration. However, this increase is justified as it simplifies usage for the client, eliminating the need for manual embedding of input data. Additionally, leveraging Keras embedding holds promise for further performance enhancements, particularly with larger training datasets.

**CNN with glove embeddings**

CNN builds upon the performance of the previous MLP’s although both MLP’s fit for purpose I believe that CNN will improve on the text classification’s utilizing it’s 1d convolution on the flattened embeddings and global polling to capture spatial dependencies within sentences potentially leading to improved performance in this task. CNN experimentation that this model benefited also from the previous text-cleaning in pre-processing. However experimentation needed to be performed on this models architecture as this model needed it’s own custom architecture parameters, initially I set the number of convolutional layer filters to 16 and convolutional window size to 5 with the dense layer’s having the same architecture as previous model with 4 hidden layer neurons however significant overfitting occurred. With

A graph with a line and a line

Description automatically generatedA graph of a line

Description automatically generated with medium confidence

|  |  |  |
| --- | --- | --- |
| Final Model Testing stats | Predicted negative | Predicted positive |
| True negative | 180 | 45 |
| True Positive | 31 | 244 |
| Accuracy | 0.848 |  |
| Precision | 0.844 |  |
| Recall | 0.887 |  |
| F1 Score | 0.865 |  |

Although significant overfitting on this model. This was the best performing model so far.

To fix this overfitting however I did several things I reduced number of hidden layer neurons to 3 I also experimented with different numbers of filters being used in the convolutional layer initally was 32 4x4 filters in order to further convolve the image’s I updates the number of filters to 64 with 5x5 which improved both accuracy and error this model performed best with 5 neurons instead of 4 I assume because of reasons.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

A close-up of a graph

Description automatically generated

Finally RNN effectively model sequence data, coupled with their capabity to capture long-range dependencies and contextual information, it makes them well-suited for text classification tasks therefore I expected this to be the best model of all models in terms of performance. At the end however after all my epxeimentation with hyperparameters and such I discovered this modelp performed best with with a non complex LSMS recurrent model with 3 units and a signmoid activation function at the end.