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**

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Batch size | 64 |
| epochs | 10 |
| Learning rate | 0.05 |
| optimizer | **Adam** |
| K-fold’s | **10** |
| activation | Relu |
| loss | **Binary\_crossentropy** |
| Hidden layer 1 size | 5 |

To set a benchmark, I created baseline models for each of the four architectures being tested. These models were initialized with default hyperparameters, with some adjustments based on intuition and constraints due to time and computational limitations, limiting extensive parameter testing.

The choice of ReLU activation for hidden layers was driven by its ability to capture complex patterns in textual data efficiently. ReLU's non-linearity is crucial for this task, while its computational efficiency, especially with high-dimensional input like 300-dimensional word embedding vectors, is advantageous compared to tanh and sigmoid activations. Sigmoid activation in the output layer constrains outputs to probabilities between 0 and 1, indicating the confidence of text belonging to truthful or non-truthful classes. Binary cross-entropy loss aligns naturally with binary classification, providing mathematical formulation, stability, and interpretability to suit the models' output layer.

During model training, I utilized 10-fold cross-validation to accurately assess error and accuracy on unseen training data. This choice was made due to the relatively small training set (2000 samples), with 10% (200 samples) reserved for validation to ensure sufficient data for validation without compromising training data. Further experimentation will refine hyperparameters and settings for optimal model configuration.

**MLP single flattened word embedding vector.**

The initial implementation of the MLP using single flattened embedding vectors utilizing pre-trained glove embeddings with the baseline hyperparameters revealed a significant issue with its stability during training. Despite this, it demonstrated moderate predictive capability on the test set, achieving an accuracy of around 0.7, indicating some degree of learning. This instability however likely stemmed from a combination of multiple factors, including noisy training data containing numerous stop words and words and punctuation embeddings devoid of semantic meaning causing it hard to train.

To address this instability, I experimented with adjusting the learning rate. Initially set at 0.05, I reduced it incrementally to 0.025 and eventually to 0.001, which successfully mitigated the instability by allowing the learning curve to look more stable because weights were being adjused more slowly. However, this adjustment highlighted another issue: the learning curve of the model's validation data showed only marginal improvement in accuracy after the first epoch in the first epoch from 0.48, 0.68 decreasing in ability to improve to reach a final accuracy of 0.70 and a loss of 27 down to 10. This suggested that the model had reached convergence, but subsequent overfitting occurred, as evidenced by the decreasing nearly to zero after the first epoch however the validation data loss increase after the first epoch consequently from 0.68 – 0.85 over the consecutive 9 epochs.

To address overfitting, I first reduced the model's complexity by decreasing the number of neurons in the hidden layer from 5 to 4. Additionally, I implemented regularization lasso and techniques by decreasing the batch size. This not only improved the model's generalization performance by introducing more variability into the training data but also extended the training time. Originally taking 10 seconds to converge, the model's training time increased to 17 seconds per fold after reducing the batch size to 32. However, this adjustment resulted in an F1 score of 0.812, with only 22 instances classified as false positives out of 406 classified instances.

Regularization further enhanced the performance of the model trained with a batch size of 32 and a learning rate of 0.001, providing better stability and fit for both training and validation data. As an additional measure, I decided to preprocess the text data by removing stop words and lemmatizing words to investigate their impact on model performance.

Additionally, I also experimented with additional pre-processing including lemization, removing stop words and removing punctuation and URLs and http links. I found removing punctuation allowed the model preprocessing to take a little big, shorter when embedding the words but also did not affect the model performance much however when removing the URLs it slightly did affect performance.

Upon incorporating these adjustments and preprocessing steps, the model demonstrated improved stability, reduced overfitting, and enhanced performance on the test set, showcasing the importance of hyperparameter tuning and data preprocessing in optimizing model performance.

A graph of a line

Description automatically generated with medium confidenceA graph of training and validation

Description automatically generated

|  |  |  |
| --- | --- | --- |
| 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 Keras Embedding Layer**

The baseline model utilizing the Keras Embedding Layer exhibited extreme bias towards predicting samples as positive, with every single sample classified as such. This bias indicated potential model complexity issues or noise in the text data, hindering the identification of meaningful patterns. Attempts to address this involved reducing model complexity and applying L1 regularization, which significantly improved the F1 score and accuracy. However, stability issues persisted, likely due to fluctuations in training and validation data. Increasing epochs to 20 and refining the number of neurons led to further performance improvements, reaching an accuracy of 0.826 and an F1 score of 0.843.

A graph of a line graph

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated

In an attempt to deal with stability issues and to try let the model reach convergence in a more stable matter and potentially improve the model performance I also tried reducing the model learning rate to 0.025 then 0.01 then 0.005 with the 0.001 learning rate yielding the best results this allowed the model to attempt to reach global minima with small increments so it would not jump over by accident but also allowed convergence to occur quickly this improved model training stability model f1 score after this was 0.83 with accuracy being 0.83 cross validated and precision being 0.87 meeting client requests to have as little false positives as possible.