2/4/2024

Student Name: Ryan Mckee

Student number: 40294886

Assignment 2

CSC3066 Deep Learning (Fake News Detection)

***Introduction***

In the era of rapid social media information dissemination, our task is to develop machine learning models to automatically detect fake news on Twitter. As data scientists, our aim is to provide our client with a scalable and accurate solution to combat misinformation risks. Leveraging artificial neural networks (ANN) and word embedding models, we seek to create a robust system capable of classifying tweets as genuine or false. This report documents our exploration of various ANN architectures, including Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) models. We analyse techniques and configurations to enhance model performance, offering insights to counter the spread of fake news effectively.

**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, we 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, 10-fold cross-validation was used 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.

\*show table with results for each model baseline\*

Based on the results of our initial baseline model experiments, it became evident that all models struggled to learn effectively. This difficulty likely stemmed from the models attempting to learn in high-dimensional spaces and identify patterns and relationships within raw data. Although ReLU activation function may aid in this process to some extent, without proper preprocessing of the data, noise in the dataset significantly hindered model performance.

Upon inspection of the dataset, it was clear that various forms of noise, such as punctuation, special characters, stop words, and URLs, were present. These elements do not contribute to the semantic meaning of the text and can confuse the model. Additionally, the presence of multiple occurrences of the same word or variations of singular and plural forms further challenged the model's ability to differentiate between relevant and irrelevant information. While class imbalance was initially a concern, further examination of the training data alleviated this issue.

The models also exhibited signs of overfitting, as demonstrated by the diagram [insert figure number] depicting a slight increase in training data accuracy and validation loss over epochs. This suggests that the models were learning from irrelevant patterns in the training data rather than generalizable representations.

To address these challenges, several preprocessing steps were explored:

* Lowercasing words and removing special characters, including URLs and punctuation.
* Lemmatization of tokenized words to reduce variations and standardize word forms, potentially improving text classification.
* Another pre-processing technique I enquired about was removal of numbers however after analysis of text data I determined it would be best not to remove these as numbers are common in text and provide semantic meaning to the models.
* Implementation of regularization techniques, such as L1 regularization, to mitigate overfitting and enhance model generalization. Notably, L1 regularization showed better results when it was able to converge effectively

\*show some diagrams and tables of results after to prove hypothesis \*

Model hyperparameter tunings.

Talk about model configuration for each of the models here

Model training performance with baseline settings.

MLP flattened embeddings:

A blue squares with white text

Description automatically generatedA graph of a graph

Description automatically generatedA graph of a line graph

Description automatically generated

Figure 1: Confusion matrix MLP flattened embeddings confusion matrix

Figure 2: MLP flattened embeddings baseline model accuracy

Figure 3: MLP flattened embeddings baseline model loss

Accuracy: 0.536

Precision: 0.938

Recall: 0.167

F1 Score: 0.284

Technique explored 1: Based on results from the initial baseline model expereiments it was obvious to see that all the models were struggling to learn well likely because the original models are trying to learn from high dimension spaces and pick up on patterns and relationships which exist on raw data. Without pre-processing there is a lot of noise in the dataset including punctuation, special characters and stop words which do not contribute semantic meanings to the text additionally the model will not realise that there is multiple occurrences of the same word or variations of that word singular or plural therefore making the model struggle to differentiate between relevant and irrelevant information. Other reasons could be class imbalance of the training data were negative samples are more represented than positive in the training data therefore making the model bore biased toward being negative. And without proper preprocessing it is obvious the models are prone to overfitting in the model’s shown above this diagram is semi-represetnative of the pattens seen in all models where training data validation and accuracy increases slightlys while the loss for validation increases over epochs therefore the model is learning from irrelevant patterns in the training data rather than learning generalizable representations solutions to this include reguralisation.

To deal with the noise in the data sets I experimented with several pre-processing steps to see their combined effects on the training and test data:

* Lower casing words and removing special characters including URL’s and punctuation.
* Performed lemmatization on each of the tokenized words to reduce words to their base root forms (lemmas) to standardise the different variations of the same words to so the same words are treated as the same token during the text analysis which will hopefully improve the text classification of the model.
* Regularization: To aid in the overfitting issue I also implemented regularisation to try and aid in the models generalization abilities. L1 reguralisation was better less extreme went it was able to converge well enough without.
* Edited the number of hidden layers neurons from 5 to 3 for MLP single vector embeddings representation as model may have been too complex and picking learning too much from training data variables that don’t matter as much.

A graph of a graph

Description automatically generated

A graph with blue and orange lines

Description automatically generated

A diagram of a confusion matrix

Description automatically generated

Accuracy: 0.724

Precision: 0.720

Recall: 0.815

F1 Score: 0.765

MLP Keras Embedding

A graph of training and validation

Description automatically generated

A graph with blue lines and orange lines

Description automatically generated

A comparison of blue squares

Description automatically generated

Accuracy: 0.5

Precision: 0.55

Recall: 1.0

F1 score: 0.710

Further discussion:

Final Outcome of the Project

Which model was the best candinate for the task, with justification?

Provid assessment of projects success, address any challenges or issues encountered during the project delivery process discuss the limitations of the developed model if any and their implications for future use

Given more computing power like access to cloud computing services to test which hyperparameters would be best it would be best to perform an automated testing algorithm like grid search to iterate through different params, like different numbers of hidden layers, neurons, learning rates batch sizes and so on, however due to the computational expensive nature already of this ai model we were trying to produce I ran into issues with assigning memory to run these grid search algorithms even after attempting to have tensorflow keras run using GPU therefore for the most ideal models it would be best to utilizie a high computing stack.

Discuss strategies strategies for enhancing the performance of the ANN model in this specific task.

In a case where the model is not ready to be used as a fully automated solution for fake content detection, propose stratiegies for using the model as a supportive tool for manual checkers