Ben Prescott
MSDS 458 2021SU
8/8/2021

A.3 Third Research/Programming Assignment

Abstract

This research centers on exploring various neural network architectures for classifying news articles into one of four categories, using a subset of documents from the original AG News corpus, with the goal of comparing different architectures and their impact on text/document classification tasks. The data consists of documents ranging in length, with balanced observations across categories of Business, Sci/Tech, Sports, and World news.

The research started by performing some initial exploratory analysis on the reduced dataset. This included exploring the category balance to ensure we were working with a balanced dataset, the number of vocabulary words across the entire dataset, and the distribution of documents by word count. Three 'core' experiments were then conducted, each having various sub-experiments focused on the overall architecture type, including simple RNN architectures, LSTM-based architectures, and 1-dimensional CNN architectures. The performance of each model was measured, as well as the time taken to train each model.

Literature Review

Many papers and articles exist focusing on using Recurrent Neural Networks for text classification tasks, especially those with LSTM or GRU cells. The same is true for 1D Convolutional Neural Networks and using word embeddings to represent the documents/words being classified. The book *Deep Learning with Python* by Francois Chollet covers these concepts, including more depth around using the Keras functional model and multi-input and multi-output models.

A blog post by David Batista named *Convolutional Neural Networks for Text Classification* has also provided detail into the workings of 1D CNNs in comparison to 2D

CNNs. Understanding how a filter shifts over a 1-dimensional sequence of text provides insight into how to tune the hyperparameters of 1D CNNs, as the filter may be too small to capture meaningful data, or too large capturing too much irrelevant text.

Methods

This research started by loading the ‘AG News subset’ dataset using TensorFlow Datasets. TensorFlow Datasets allows for the ability to interweave training and testing data and their labels. However, I chose to convert the dataset to a DataFrame, providing more intuitive control over the data. I then performed some minor analysis on the data, reviewing the number of documents in the dataset, the classes/categories of each document, and the balance of data between the four classes. Additionally, total corpus word count and document count were determined, as well as the minimum length of a document (3 tokens) and the maximum length (173).

The first set of experiments following analysis were focused on simple Recurrent Neural Networks. I chose to use the TensorFlow ‘SimpleRNN’ layer, as to keep it as simple and standard to an RNN without having to introduce LSTM or GRU layers (yet). This experiment consisted of three sub-experiments, all using a single bidirectional SimpleRNN layer. The changes between sub-experiments were the length of the vocabulary, starting with a vocabulary of 1,000 words, then 2,000 words, and finally 3,000 words. The purpose of this experiment was to determine the impact of changing the length of the vocabulary while keeping all other hyperparameters constant.

The second experiment leveraged the same architecture, only replacing the SimpleRNN layer with one, two, and three bidirectional LSTM layers. This core experiment also stuck with the vocabulary length of 1,000 to reduce the variations of models, as we’ve already seen an example of vocabulary length changes with the SimpleRNN. The first sub-experiment consisted of a single bidirectional LSTM layer with 64 cells, with the second experiment adding another subsequent LSTM layer with half the original cells (64 to 32). The original LSTM layer was then set to return sequences to provide the proper input for the second LSTM layer. The third sub-experiment followed the same pattern, using a third LSTM layer with 16 cells.

The third and final group of experiments used 1-dimensional Convolutional Neural Networks to compare CNN-based architectures with RNN-based architectures. The first sub-experiment used a single convolutional layer with ReLU activation and a global maxpooling layer, using the same encoding and embedding layers as all other models in this research. The second sub-experiment used two convolutional layers with ReLU activation, with the first including a standard maxpooling layer, followed by a convolutional layer with global maxpooling. The third sub-experiment added one additional convolutional/maxpooling layer, bringing the total to three layers. All other hyperparameters were kept consistent to ensure the only changes were to layer counts.

Various metrics were collected throughout the model training and evaluation processes, including vocab length, model training times, and train/validation/test set accuracy and loss for each model. These metrics were then used to create horizontal bar charts showcasing the similarities and differences in model performance.

Results

The results of the experiments attempted to show the difference between SimpleRNN, LSTM-based, and CNN architectures, and their performance using the given dataset. The first experiment focusing on SimpleRNN layers also included tests of different vocabulary sizes –

1,000, 2,000, and 3,000, respectively. It is worth noting that all experiments throughout this research achieved similar test accuracy results, ranging from 85% to 88% accuracy.

The first core experiment focused on using SimpleRNN layers without memory cells. The results of these tests identified a linear increase in model performance when increasing the vocabulary size. What is interesting is that the validation accuracy increased with a larger vocabulary size while the model training times decreased. As these models also used an early stopping callback to ensure the training stops when there was no improvement to the validation accuracy metric, it can be assumed that the larger vocabulary the faster the model reached peak performance. The SimpleRNN layer models were also significantly slower to train, with the fastest model using a vocabulary of 3,000 and a training time of 35 minutes. The slowest used a vocabulary of 1,000 with a training time of 45 minutes.

The second core experiment switched the model to include LSTM cells rather than SimpleRNN layers. The first experiment started with a single bidirectional LSTM layer and with 64 cells, followed by the second experiment with two bidirectional LSTM layers, and the third experiment with three bidirectional LSTM layers. Each additional LSTM layer had its cell count reduced by half of the previous (i.e. – LSTM 1 = 64 cells, LSTM 2 = 32 cells, etc.).

The accuracy and loss on the test set of each model were very similar, with 1 LSTM layer having a test accuracy of 86% and three LSTM layers showing a very similar 85.7%. Each model trained for 10 epochs before triggering the early stopping callback, each showing nearly identical performance. However, the single bidirectional LSTM model had a training time of about 23 minutes, while the two and three layer models were nearly 30 minutes for no improvement.

The final core experiment switched focus to 1-dimensional convolutional neural network models. The models followed suit with the other two core experiments, starting the first sub-experiment with one 1D CNN layer and one pooling layer, the second sub-experiment with two 1D CNN layers and 2 pooling layers, and the third sub-experiment with three 1D CNN layers and three pooling layers. The performance of the 1D CNN models showed very similar results as the SimpleRNN and LSTM models, with test accuracy of ~85% using a vocabulary size of 1,000. However, the training time of these models was significantly faster than the SimpleRNN and LSTM models, with the slowest training time being three 1D CNN layers at just over 3 minutes. For very similar results this shows a drastic improvement in training durations and extended compute requirements.

Conclusions

This research was conducted to explore various RNN, LSTM and 1D CNN models to work with a subset of the AG News dataset, classifying news articles into categories using supervised learning methods. While nearly all hyperparameters remained consistent throughout the entire research, with the exclusion of the core experiment-specific layers (i.e., RNN, LSTM, etc.), the results indicated that 1-dimensional Convolutional Neural Networks can perform well at this test classification task, while reducing the training time and computational requirements drastically.

Another takeaway from this research was how vocabulary length not only made a noticeable impact on model accuracy, but also on the model training time. As the vocabulary increased the training time for each model decreased while the test set accuracy increased. Additional research will need to be conducted to determine if even larger vocabularies would continue to show improvements in training times and test set accuracy, or if a ceiling can be found or potential overfitting with too large of a vocabulary.

References

Batista, D. S. (2018, March 31). Convolutional Neural Networks for Text Classification [web log]. http://www.davidsbatista.net/blog/2018/03/31/SentenceClassificationConvNets/.

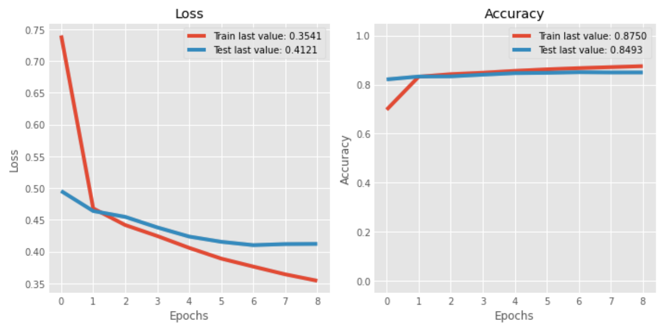
Brownlee, J. (2020, August 27). *How to develop convolutional neural network models for time series forecasting*. Machine Learning Mastery. https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/.

Chollet, F. (2021). *Deep Learning With Python*. S.l.: O'Reilly Media.

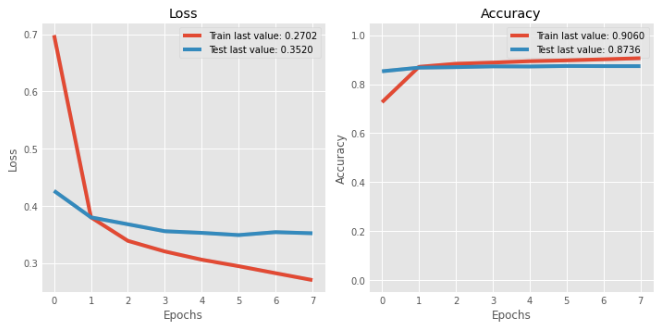
Appendix

Model Performance Plots:

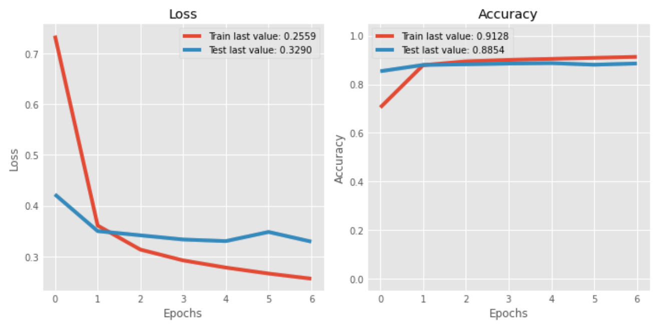
Bidirectional Simple RNN (vocab 1000):



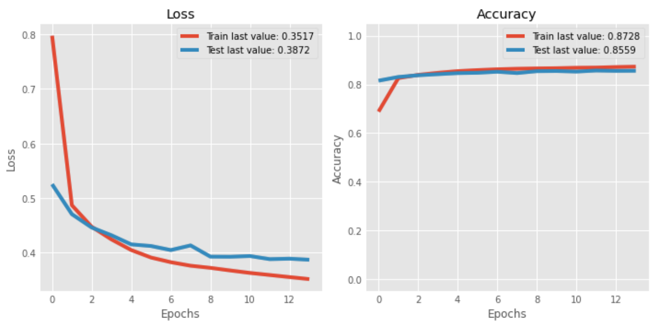
Bidirectional Simple RNN (vocab 2000):



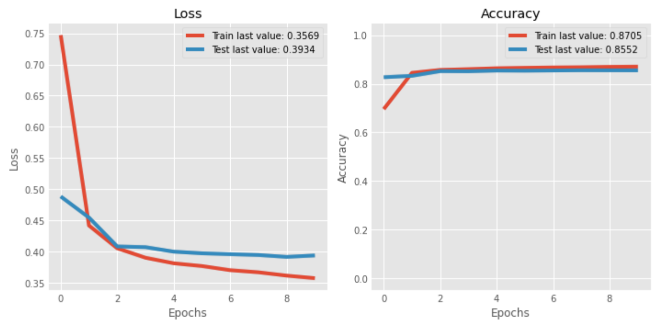
Bidirectional Simple RNN (vocab 3000):



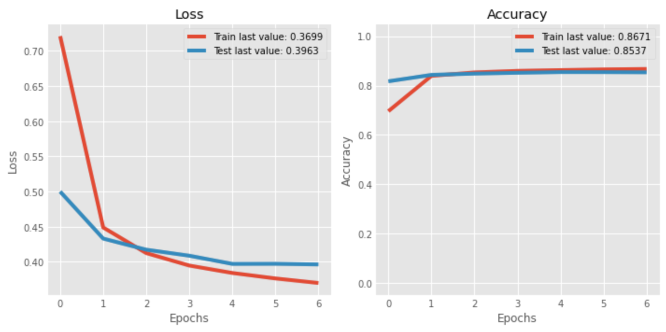
One Bidirectional LSTM Layer:



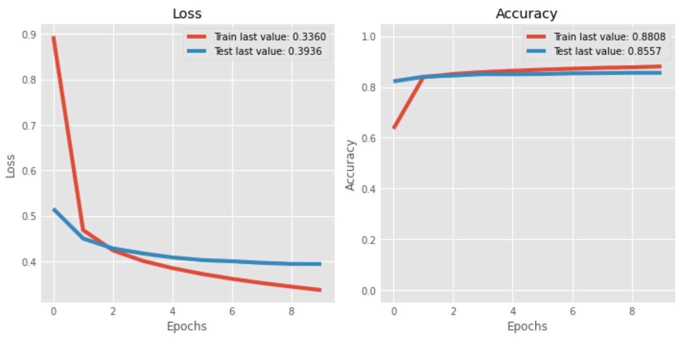
Two Bidirectional LSTM Layers:



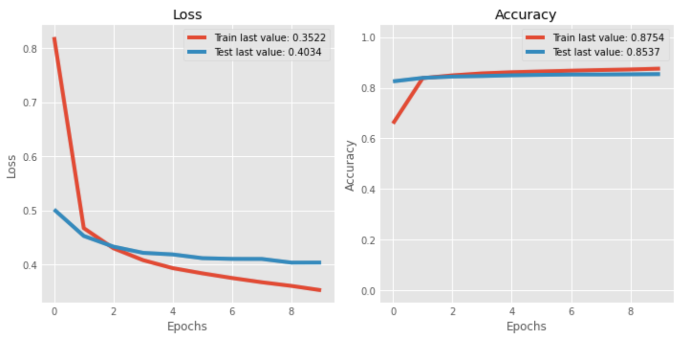
Three Bidirectional LSTM Layers:



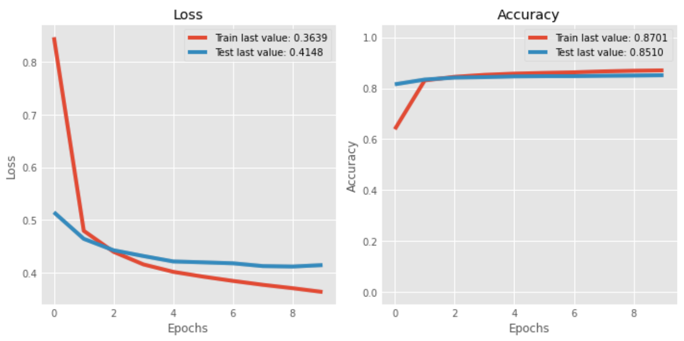
One CNN and Pooling Layer:



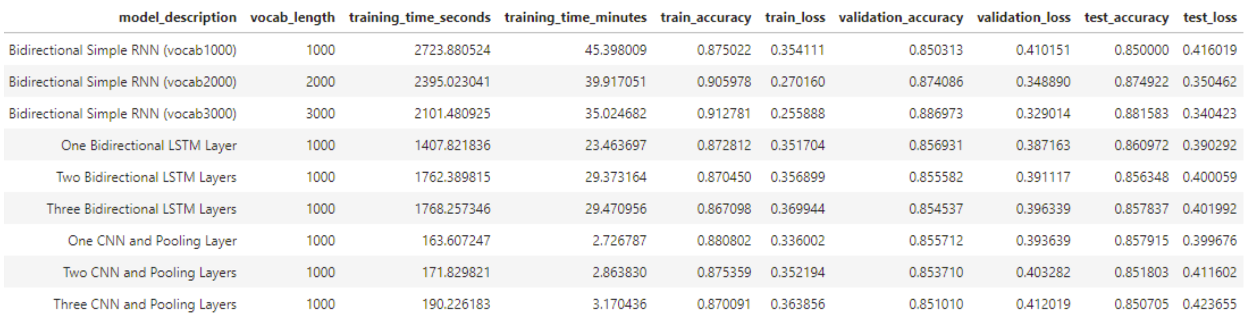
Two CNN and Pooling Layers:



Three CNN and Pooling Layers:

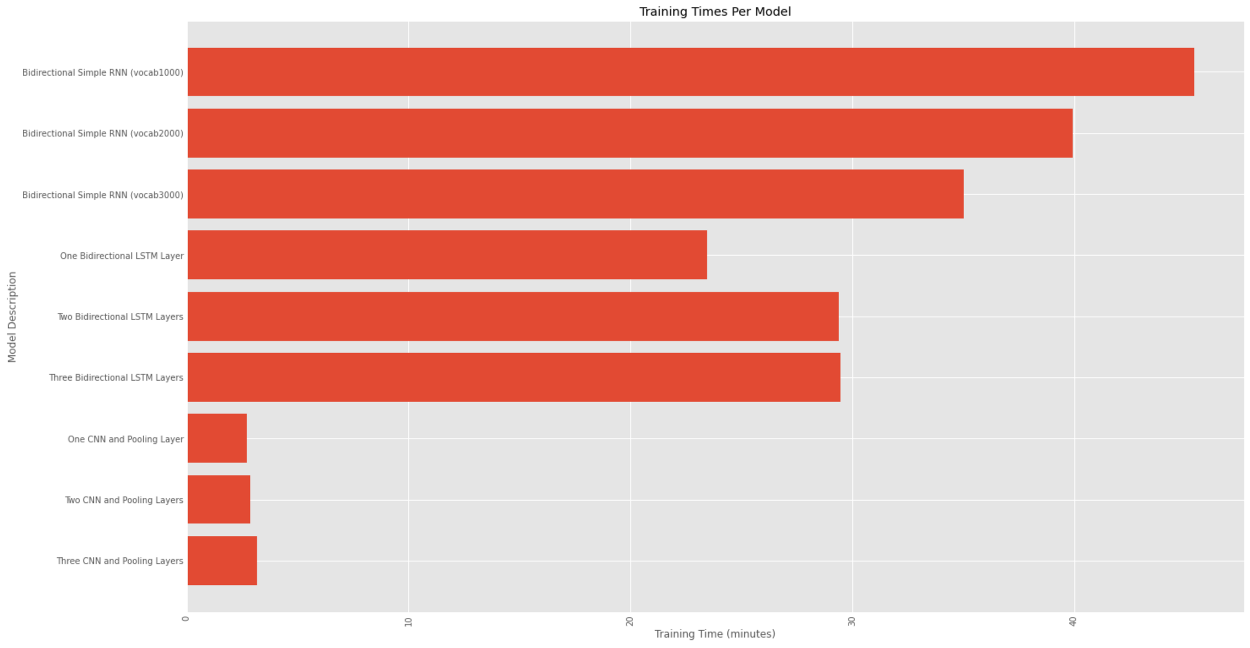


Results Table:

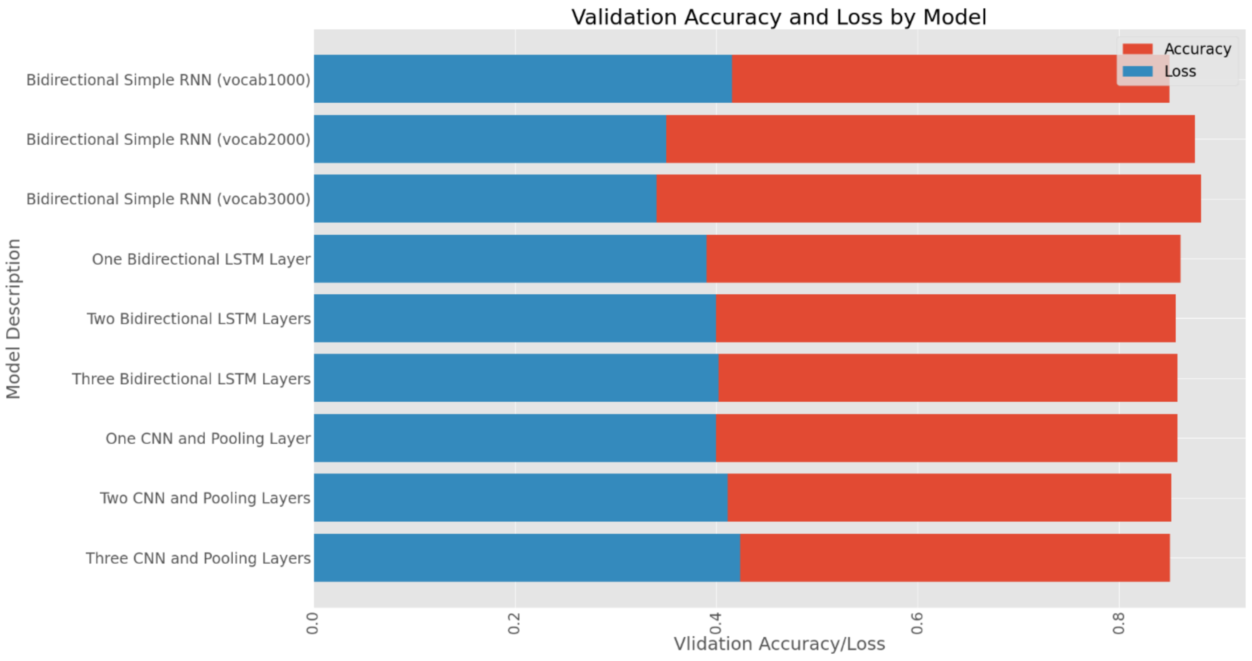


model_description	vocab_length	training_time_seconds	training_time_minutes	train_accuracy	train_loss	validation_accuracy	validation_loss	test_accuracy	test_loss
Bidirectional Simple RNN (vocab1000)	1000	2723.880524	45.398009	0.875022	0.354111	0.850313	0.410151	0.850000	0.416019
Bidirectional Simple RNN (vocab2000)	2000	2395.023041	39.917051	0.905978	0.270160	0.874086	0.348890	0.874922	0.350462
Bidirectional Simple RNN (vocab3000)	3000	2101.480925	35.024682	0.912781	0.255888	0.886973	0.329014	0.881583	0.340423
One Bidirectional LSTM Layer	1000	1407.821836	23.463697	0.872812	0.351704	0.856931	0.387163	0.860972	0.390292
Two Bidirectional LSTM Layers	1000	1762.389815	29.373164	0.870450	0.356899	0.855582	0.391117	0.856348	0.400059
Three Bidirectional LSTM Layers	1000	1768.257346	29.470956	0.867098	0.369944	0.854537	0.396339	0.857837	0.401992
One CNN and Pooling Layer	1000	163.607247	2.726787	0.880802	0.336002	0.855712	0.393639	0.857915	0.399676
Two CNN and Pooling Layers	1000	171.829821	2.863830	0.875359	0.352194	0.853710	0.403282	0.851803	0.411602
Three CNN and Pooling Layers	1000	190.226183	3.170436	0.870091	0.363856	0.851010	0.412019	0.850705	0.423655

Model Training Times:



Validation Accuracy/Loss by Model:


Validation Accuracy and Loss by Model
Accuracy
Loss
Bidirectional Simple RNN (vocab1000)
Bidirectional Simple RNN (vocab2000)
Bidirectional Simple RNN (vocab3000)
One Bidirectional LSTM Layer
Two Bidirectional LSTM Layers
Three Bidirectional LSTM Layers
One CNN and Pooling Layer
Two CNN and Pooling Layers
Three CNN and Pooling Layers
Model Description
Vlidation Accuracy/Loss