A Comparison of Multilayer Perceptrons and Recurrent Neural Networks as Applied to Sentiment Classification

Maeve Hutchinson

I. Brief Description and Motivation of the Problem

Classifying text based on its sentiment is a deep-learning task with several applications and a fast-growing demand in the commercial world. Specifically, determining whether online customer reviews have a positive or negative sentiment is incredibly useful to businesses, allowing them to gain an overview of public opinion, thus being able to market to consumers more effectively.

This paper explores a selection of reviews from Amazon, IMDb, and Yelp, classifying them based on whether they have a positive or negative sentiment. This task is used to evaluate two deep learning models: a Multilayer Perceptron (MLP) and a Recurrent Neural Network (RNN).

II. Description of the Dataset

The dataset used is a compilation of reviews from Amazon, IMDB, and Yelp, as obtained from the UCI Machine Learning Repository [1]. In total, it has 2,748 reviews, each labelled with either a 1 or a 0, indicating a positive or a negative review respectively. In total there are 1,386 positive reviews and 1,362 negative ones. The difference in number between the two classes is marginal and thus does not need to be addressed.

Mean	13.1
Standard Deviation	34.2
Maximum	1319
Minimum	2

TABLE I: Statistical Summary of Review Lengths

The text is preprocessed by removing all punctuation, numbers, and symbols, and put into lowercase. It is then tokenised into individual words for inputting into the models. The reviews, by nature, are of variable length, so after the processing, this was analysed with a statistical summary shown in table 1. There is a wide range of review lengths, however, this will be dealt with in the networks themselves.

III. Brief Summary of Multilayer Perceptrons and Recurrent Neural Networks

A. Multilayer Perceptron

An MLP is a type of feedforward neural network, meaning that information moves only forward through the layers of the network. An MLP consists of an input layer, at least one hidden layer, and an output layer. The nodes between each layer are fully connected to the next layer. Each node in the hidden layers and output layers uses a non-linear activation function, most often a ReLU function [2]

MLPs train using backpropagation: the training data is fed through the network to produce an output, which is used to calculate the loss, which is then propagated back through the network to update the weight and bias of each node in the network [2, 3].

The avdantages of MLP over simpler models, such as Linear Neural Networks, are that they can learn non-linear models due to the non-linear activation function between each layer. Broadly, they are just complex enough to tackle interesting problems, whilst remaining simple to implement. They are also very flexible in terms of the data that they can classify, meaning that MLPs can be applied to a broad range of problems.

However, MLPs do have many drawbacks, notably having many hyperparameters that are very sensitive, thus MLPs often take a considerable amount of time to tune. In the case of text classification, a significant drawback is that they cannot interpret inputs that are sequential.

B. Recurrent Neural Network

RNNs have a similar structure to MLPs, consisting of an input layer, at least one hidden layer, and an output layer. The main difference lies in the hidden layers — in contrast to MLPs, RNNs are not feedforward networks, meaning that the hidden layers receive information both from the current time step and the previous time step, forming a recurrent loop in the network [2].

This difference in structure leads to one of the key advantages of RNN: they can process inputs that are sequences of variable length [3]. This is a particular advantage when processing text data.

However, like MLPs, a key drawback is the sensitivity to hyperparameters. Also, due to the more complex structure, there are even more available parameters, so RNNs will likely take longer than MLPs to train and tune on the same task.

IV. Hypothesis Statement

Given previous results on the same dataset using, it is expected that both methods will perform quite well, with accuracies much better than a random guess [4]. It is predicted that RNN will outperform MLP due to its ability to take textual sequences of variable length as an input, which is key to this task. However, it is also expected that RNN will take considerably longer both to train and to optimise than MLP, as it is a more complex model.

V. Description of Choice of Training and Evaluation Methodology

The models have a very similar structure to the basic structure described above, but the input layer for both models is an embedding layer. This layer allows each input to be transformed to a vector the size of the entire vocabulary of the dataset. This gives significant advantages in allowing both models to take inputs of variable length, but may increase the training time due to increased dimensionality.

The MLP then has 2 fully connected layers of the same size, each with a ReLU activation function. The RNN has a tunable number of recurrent layers, followed by two fully connected layers before the output, also with ReLU activation functions.

The data are split into training, validation, and test sets in the proportions 80%, 10%, and 10% respectively, giving sizes of 2198, 275 and 275. The evaluation method used is to hold out the validation set and use that to evaluate the model after each epoch. Finally, both models will be tested using the remaining test data.

The loss function used is Binary Cross Entropy loss, and both models are trained through backpropagation. The optimiser used is an Adam algorithm.

VI. Choice of Parameters and Experimental Results

Several parameters for each model were optimised using a TPE algorithm for sampling over a manually defined space. For each model, the selected parameters can be seen in table 2 and table 3. These are the parameters that are used to train the models.

Parameter	Value
Learning Rate	0.0039
Embedding Layer Size	195
Hidden Layers Size	150

TABLE II: Tuned Hyperparameters for MLP.

Parameter	Value
Learning Rate	0.0026
Embedding Layer Size	230
Recursive Layers Size	90
Fully Connected Layers Size	150
Number of Recursive Layers	2

TABLE III: Tuned Hyperparameters for RNN.

The accuracy and loss curves for training and validation for both models can be seen in figure 1 and figure 2. The results of running the optimised models on the test set can be seen in figure 3. The key metrics of training, validation, and testing are summarised in table 4.

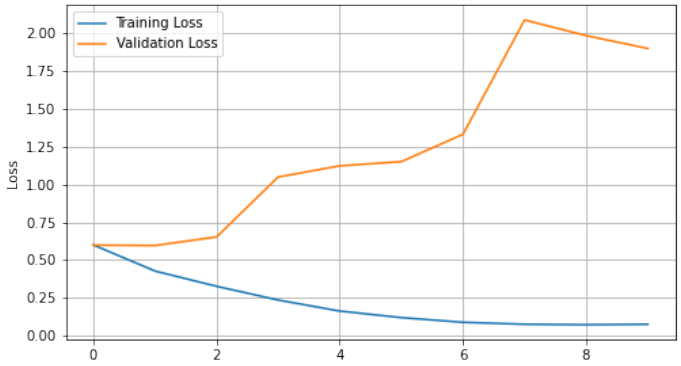

Fig. 1: Loss and Accuracy curves for MLP.

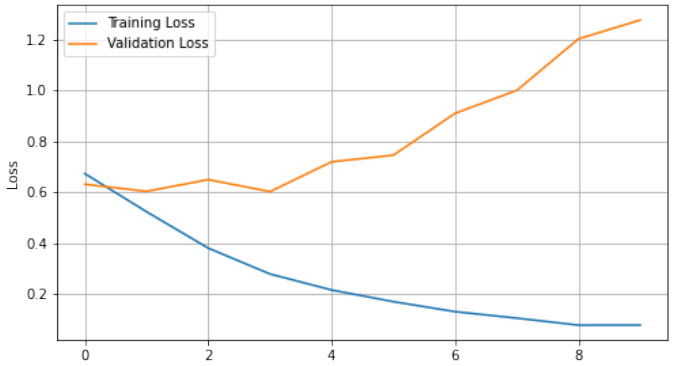

Fig. 2: Loss and Accuracy curves for RNN.

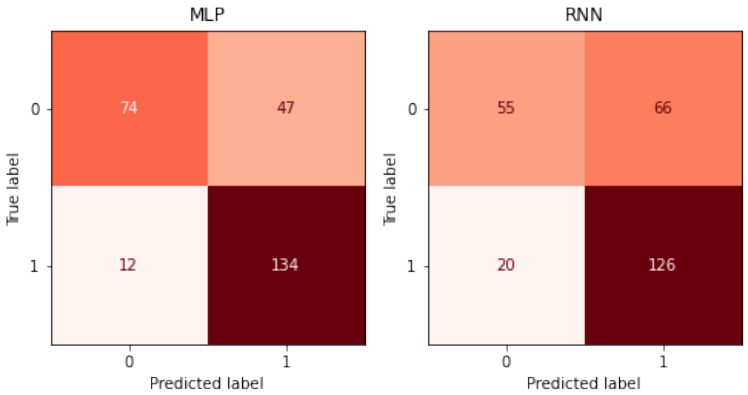

Fig. 3: Testing confusion matrices for both models.

	MLP	**RNN**
Training Accuracy	97.7%	97.1%
Validation Accuracy	78.0%	75.1%
Testing Accuracy	76.1%	65.2%
Training Loss	0.074	0.077
Validation Loss	1.901	1.277
Testing Loss	0.514	0.590
Precision	74.0%	65.6%
Recall	91.8%	86.3%
F1 Score	82.0%	74.6%
Training Time	5.6s	21.7s

TABLE IV: Evaluation Metrics for both MLP and RNN

VII. Analysis and Critical Evaluation of Results

From the loss it is clear that both models are severely overfitting on the training data – whilst the training loss continues to decrease with every epoch, the validation loss is increasing. This is also reflected in the accuracy, where both models had a training accuracy of over 97%, but only a testing accuracy of 76.1% and 65.2% for MLP and RNN respectively. This also demonstrates that the overfitting is worse for the RNN than the MLP.

Surprisingly, the MLP performed better than the RNN, which is the opposite of what was hypothesised. This could be due to the fact that both models had an embedding layer, allowing them to process data of variable lengths, so the model structure between the two was not drastically different. It is also interesting to compare F1 scores, where both models have a better value than for accuracy. This metric demonstrates that the models are not performing extremely badly, but it is still hard to compare them effectively because overfitting dominates the results.

The training time does align with the hypothesis — the RNN took nearly 4 times as long to train than the MLP. In this small dataset, this does not have a huge impact as the time is still in seconds, but this is important to note when thinking about scaling the task to a larger dataset.

In order to fix the overfitting issue there are a few things that could be done. Firstly, the validation evaluation method could be changed to use cross-validation, which gives a better estimate of the generalisation of the model. This would reduce overfitting because the hyperparameter tuning is based upon maximising the validation accuracy, so the better the validation accuracy estimates the training accuracy, the better the hyperparameters selected allow the model to generalise.

The structure of the models could also be modified to include a dropout layer. This is a regularisation technique that works by randomly removing neurons from layers of the network when training [2]. This creates an equivalent effect to training on different networks each epoch, overall reducing overfitting.

VIII. Conclusions, Lessons Learned, and Future Work

Overall, MLP performed better on the task than RNN, and took less time to train. However, it is hard to draw any broad conclusions about the efficacy of the models as applied to sentiment analysis because both models were heavily overfit on the training data. This gives the key lesson learned which is to apply techniques to reduce overfitting, such as cross validation and dropout layers.

Future work of interest would be to develop the RNN model further, by making it bidirectional, or exploring other RNN models such as LSTM, a model which is commonly used in sentiment analysis [5]. It would also be interesting to scale this task to a much larger dataset with a larger vocabulary to see how the models perform.

References

[1] “UCI Machine Learning Repository: Sentiment Labelled Sentences Data Set.” https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences (accessed May 08, 2022).

[2] S. Raschka, Y. Liu, V. Mirjalili, and D. Dzhulgakov, Machine learning with PyTorch and Scikit-Learn: develop machine learning and deep learning models with Python. Birmingham: Packt Publishing, 2022.

[3] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, “State-of-the-art in artificial neural network applications: A survey,” Heliyon, vol. 4, no. 11, p. e00938, Nov. 2018, doi: 10.1016/j.heliyon.2018.e00938.

[4] D. Kotzias, M. Denil, N. de Freitas, and P. Smyth, “From Group to Individual Labels Using Deep Features,” in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney NSW Australia, Aug. 2015, pp. 597–606. doi: 10.1145/2783258.2783380.

[5] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, “Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling,” 2016, doi: 10.48550/ARXIV.1611.06639.

Appendix A
Glossary

Input Layer - the first layer in the network that recieves the data
Hidden Layer - layer in the network that does not either recieve data or output results, unseen
Output Layer - layer that outputs the results
Optimiser - function used to minimise the loss Batch - set of data put into input layer, comprising of several instances from the dataset
Epoch - a single round of training

Appendix B
Implementation Details

Both models were implemented using PyTorch. The hyperparameter tuning was done using optuna.

Before hyperparameter tuning, the validation accuracy was only slightly less for each of the models, meaning it did not make that much of a difference. This is likely due to the dominating effect of overfitting.