Aesop Fables Predictive Text using LSTM and BiLSTM Model

By: Daniele Petrillo, Roberto Formicola and Daniel Fynn

For: Professor Giuseppe Longo, Astroinformatics 2022

Aim

The aim of this project was to implement a normal LSTM and bidirectional LSTM on a chosen unique data set, in this case Aesop's Fables were chosen. The aim of using the LSTM variants is to train a model that has the ability to predict the word following an input sequence. An attempt was also made to show how different lengths of predictive sequences can affect the ability of the model. The model was then utilised to apply prediction to a variety of sentences to determine its effectiveness.

Literature

Normal LSTMs

LSTMs (Long Short-Term Memory) neural networks are an update of Recurrent Neural Networks, trying to correct the aforementioned issues. These have been invented by *Hochreiter* & *Schmidhuber* in 1997 in their paper Long Short-Term Memory, a groundbreaking paper discussing the structure of the NN but especially the computational efficiency with respect to the RNNs.

Bidirectional LSTMs

Bidirectional LSTMs are an additional evolution of unidirectional ones. They consist in two LSTMs, one going forward end the other backwards; the main reason behind this choice is that it supposedly helps with the prediction and the understanding of the relations intra- and inter-sequences.

Method

Preprocessing

After the selection of the Aesop Fable text to perform predictive analysis on, it was imperative to create a demonstration of the data that is readable by a neural network.

The preprocessing consisted of first importing the .txt file, then stripping and cleaning the data using simple string operations in python, returning a list of all the words in the text in lower case without grammar. The unique values were then taken from the list, and a dictionary was generated containing the unique words as key and a unique number as value from 1 to the length of unique values. The text was then dissected into fragmented sentences of the desired length, plus one value for the response being the next word of the sentence, translated with the dictionary into a list of unique values, creating the model training set and response for training.

For example, having a selected length of 8 words for predicting the ninth:

"*A cock was once strutting up and down*" is equivalent to

[1,2,3,4,5,6,7,8] for the model, each word a unique token

And the predicted word is "**the**" [CORRECT].

The sequence is shifted forward one word at a time resulting in a test set having length of: all words minus the length of the words in the sequence. A second dictionary is also generated, switching the key and value pair for transferring the output of the model, back into non tokenized form.

These sequences are then one hot encoded, so each token is treated as its own variable, making a sparse matrix with one TRUE value each row.

Modeling

Normal LSTM

A callback method was used for each of the two models: normal LSTM and bidirectional LSTM.

A simple sequential keras model was developed having two LSTM layers, the first with 64 and second with 128 plus and a softmax activation later. The optimiser used was a Root Mean Square Propagation (RMSProp), an extension of gradient descent and the AdaGrad version of gradient descent with a learning rate of 0.01. The loss metric was categorical_crossentropy, and the metric assessed was accuracy. The LSTM required 1,005,961 parameters that required training. The model was then fitted with a batch size of 128 for 50 epochs.

Bidirectional LSTM

The bidirectional model instead uses 2 bidirectional layers with 64 and 128 nodes respectively and again a softmax activation layer. RMSprop optimiser is used with a learning rate of 0.01. Again the categorical cross entropy is used and the accuracy for the model is outputted. There are a total of 2,075,145 parameters that are required for training. The same batch size and epoch are used as before.

Results

Ideal Sequence Length

The first test performed was comparing the different sequence length of the two models as outlined above. The loss function for each can be seen in the two figures following.

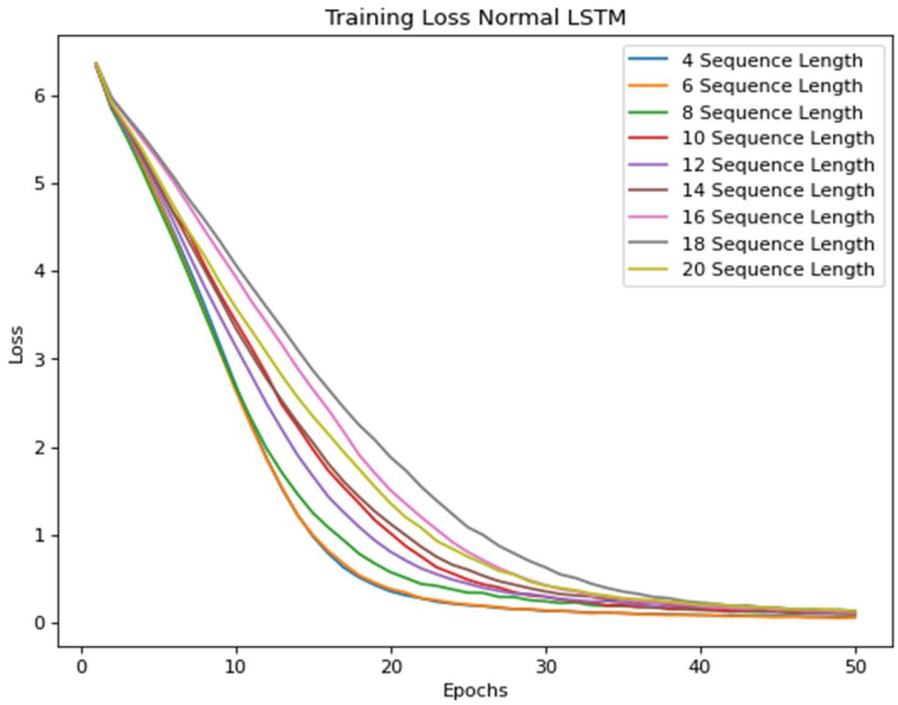

Figure 1 - Training Loss for the Normal LSTM for various sequence lengths of the predictor.

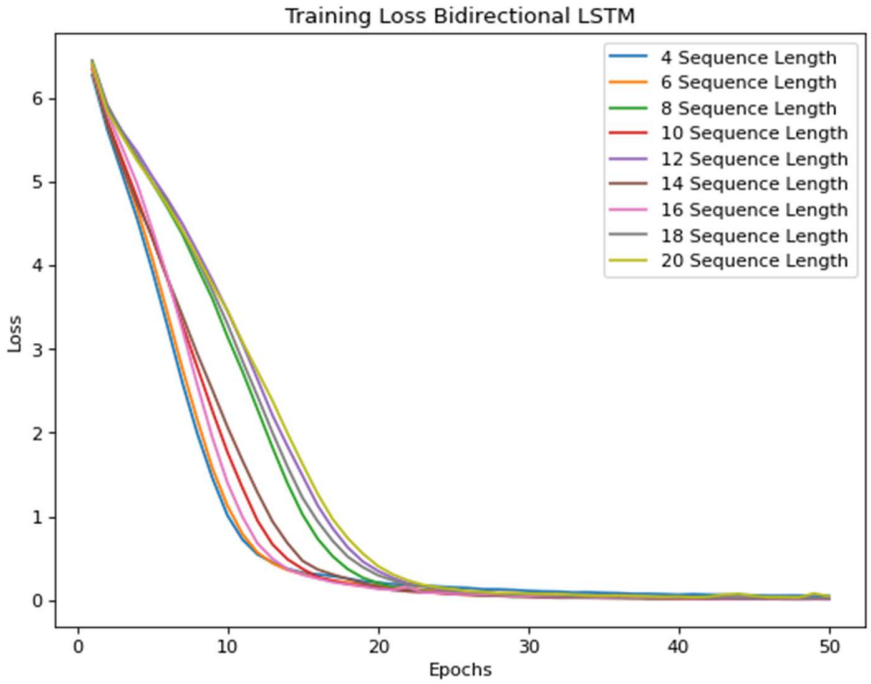

Figure 2 - Training Loss for the bidirectional model for varying sequence length

As can be seen from the overall trends is that the bidirectional model loss function decreases much more rapidly than the LSTM model.

Also can be seen is that this loss is affected by sequence length. For both models the quickest decrease in loss has been made by the shortest sequences 6 and 4. Having a general pattern of reducing the rate of loss as sequence increases. But this is not consistent for all langeths with for instance a length of 20 in the LSTM model having a faster decrease in loss than that of 18 and 16.

To compare the two models a sequence length of 10 was chosen, as amongst all the models it had a similarly placed line (as would be expected by a midway length) and also is still quite long.

Comparing LSTM and Bidirectional Results

With the chosen sequence length of the model, this was compared for both Normal and Bidirectional LSTM. The results can be easily compared in the figures below.

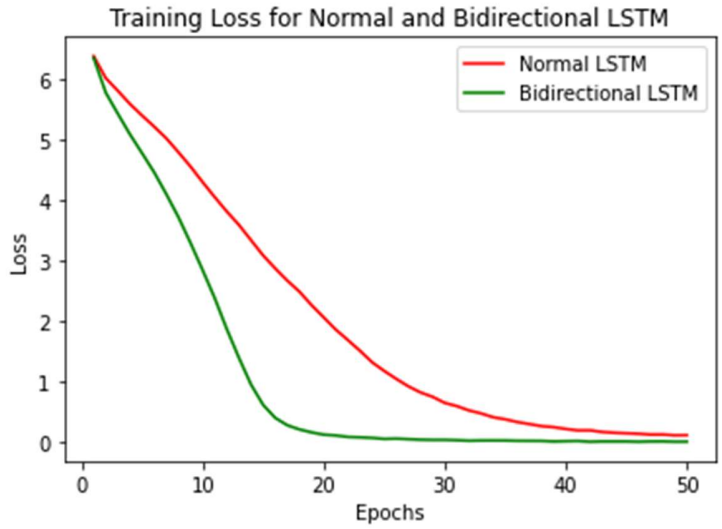

Figure 3 - Training loss for normal and bidirectional LSTM with sequence length 10.

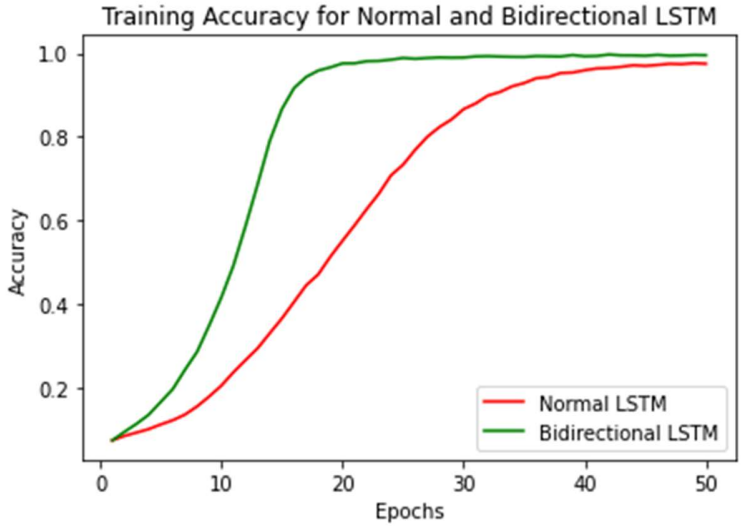

Figure 4 - Training accuracy for normal and bidirectional LSTM with sequence length 10.

Similar to the graphs seen earlier, the graph above just show a higher accuracy with fewer epochs. This could be attributed to how the model works, as it would learn the representation quicker as it has more parameters with the combination of two directional LSTM models.

Prediction

Attempts were made at evaluating the bidirectional model with sequence length 10 by applying a prediction function. This required an input of a sentence of any length, though those attempted were around 10 words or less. The predicted output was then generated for comparison by the user to see how it compares.

The predicted sentence was similarly stripped of grammar and minimized for consistency with the symbols. Please note each word/symbol used in the prediction samples must be contained in the overall word list, otherwise there will be an error. This was the most important past of the choosing predictors, and the reason a larger text was unable to be selected.

The first example was one taken from the text:

"but surely I was right in trying to revenge him " with the predicted word to be *“now”*. This was successful.

Next shuffling of this sample sentence and leaving the predicted word was attempted. This had 1 word from 10 correct, so the whole structure of the sentence is important

Next we tried fixing the last word and shuffling the rest, expecting that the last word would play a high importance in predicting the next word. But again this gave a poor result, of normally around 1 and two.

Then a new sentence from the text was chosen that had a more unique word in the final position. The same test as above was made, and again a poor result was given, for with complete shuffling having no positive responses, and with the last word help, giving a correct response.

Lastly, attempts at applying general sentences to the model, and seeing if the model outputs the predicted value that matches the users intent were matched. Again this was unsuccessful. The model hasn’t learned to apply a general syntax to unseen data.

Conclusion

This was a good learning experience, learning how to better implement the LSTM and bidirectional LSTM model for word prediction. The length of the predictors were compared, and they do play a part in the speed of convergence and improvement of accuracy.
The bidirectional and LSTM model were compared, and it was found that the bidirectional model converged and learned faster than its LSTM counterpart.
The models were then used for prediction, and within the data set, proved effective, but once applied in a more general case, it didn't perform as well.

Possible Future Ideas:

- Could be an option to keep the grammar or some of the grammar as symbols for the model
- Apply more texts to increase the domain of studied symbols
- Find a more appropriate test set

References

- Project Gutenburg, Aesop's Fables, 2022, https://www.gutenberg.org/ebooks/21
- Aesop's Fables an ancient text
- Keras, 2022 <https://keras.io/examples/>
- Matplotlib, <https://matplotlib.org/>