**Comparison between the RNN and the LSTM models for text generation**

## Introduction

This paper will discuss the differences between the performance in text generation of a Recurrent Neural Network (RNN) and a Long Short Term Memory (LSTM) based machine learning model. The goal of the model will be to generate readable language using one-ahead predicitions from what the model learned being fed free english text found from an open source online.

The model will be trained using data from the Gutenberg Project. The text used is Frankenstine, by Mary Shelley (Shelley, M 1818). After reading in the data, it was initially quite obvious that the data needed a lot of preprocessing before training a model on it to produce half decent results.

you will rejoice to hear that no disaster has accompanied the commencement of an enterprise which you have regarded with such evil forebodings. i arrived here yesterday, and my first task is to assure my dear sister of my welfare and increasing confidence in the success of my undertaking. i am already far north of london, and as i walk in the streets of petersburgh, i feel a cold northern breeze play upon my cheeks, which braces my nerves and fills me with delight.

## Modelling Strategy

### Preprocessing

The initial read of the text had features in it such as introductions, glossaries, contents and other references, that wouldn’t provide the models any valuable information. So the preprocessing plan followed the steps shown below in table 1.

|  |  |
| --- | --- |
| Step | Description |
| Initial text visualization | Initial look at the data that was read in to recognize potential issues in data usage |
| Find and set text start point to start of real text | Moving the start point of the text from glossaries to the real beginning of the text provides more value to the model. |
| Remove unnecessary special characters | Remove certain text specific special characters and replace them with better values |
| Create character dictionary from cleaned data | Create sorted character dictionary from the data with numerical values to reference |
| Create reversed numerical dictionary of characters to reference | Create a sorted reference list to use to reference the previous dictionary |
| Set model sequence length | Initial sequence length set to 100, but this will be iterated as the model develops |
| Data division | Data will be split into 80/20 segements to use for training and validataion respectively |

Table 1: Data preprocessing plan

### Architecture

The model will use a fairly basic architecture typical for RNN’s. The first layer, the Embedding/Input Layer, this layer will act as the feeding layer that will forward the data to the next layer which will be the learning layer. The last layer is called the output layer, this is where the final predictions are made, my predictions will be between the characters found in the character dictionary that was set in the pre-processing phase. The model will contain a loss function, suggested loss function for neural networks and language processing is called sparse categorical crossentropy. Categorical crossentropy is suited for word prediction (Computing For All 2024), as we are predicting singular values such as numbers, sparce categorical crossentropy is more suited as the loss function (Datamites 2023)

### Modelling plan

The initial run of the model will be done using the hyperparameters defined in Table 1.

Layer connections will be monitored by printing out the model summary as suggested by Brownlee (2017), this will show that the data is flowing forward through the model in a consistent manner.

Inital Epoch size will be set to 20 and be altered as the training goes forward

Training model states will be saved in separate files, that will be used in the prediction model to predict results and generate text with. This should allow for scaled use, and the model to automatically use the best available run from the training data.

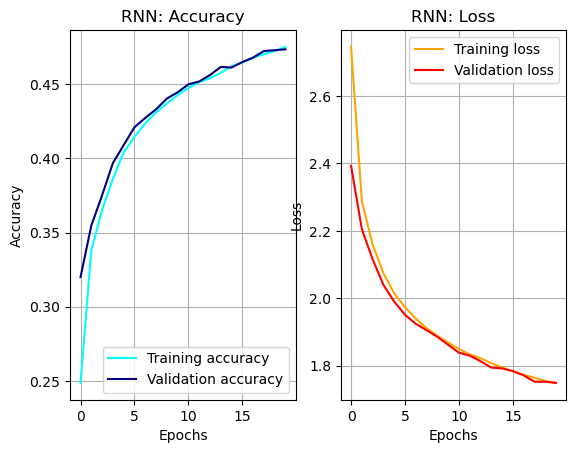
|  |  |
| --- | --- |
| Parameter | Value |
| Sequence length | 100 |
| Epochs | 20 |
| Learning rate | 0.001 |
| Network size/units RNN | 64 |
| Network size/units LSTM | 64 |
| RNN/LSTM Layers | 1 |
| Batch size | 64 |

Table 2: Initial model training parameters

These parameters will be tweaked during training according to the results of the model to find best possible performance for the model. The final model product will be compared visually to the original sample, and then a BLEU-Score will be run against 1-4 ngrams created from both the original text, and the generated texts. BLEU-score is mostly used to look at machine translation precision, but as Kirchoff states in his post the algorithm compares machine translation ngrams against human translated sentences providing a score that represents the success of said translation, so this could potentially be used here as well (2024).

### Initial results

The first metrics fetched from running the models with the parameters from Table 1 can be found from Figures 1 and 2. What can be seen in these, is that, RNN is running very similarly to the LSTM model. The metrics don’t seem to capture a real distincion in model accuracy or loss in this timeframe. While the initial accuracy is faster with RNNlearning vaster, the learning curve seems to plateau on a faster rate than with the LSTM-model.

Figure 1

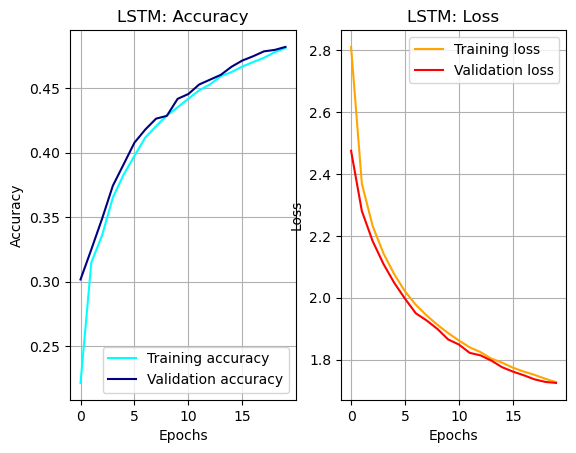


Figure 2

Compute results for the initial paramters were calculated using the “time” magic-function built into Python. There is a diffrerence of about 1 minute between the two models. Texts 1 and 2 represent samples of the two first sentences generated by the models.

|  |  |
| --- | --- |
| Model | Compute time |
| LSTM | Total: 1min 54s |
| RNN | Total: 1min 2s |

Table 3

Text 1: Text 2: LSTM Generated text with first parameters

you will rejoices continues her freed, while this deposted prose to his adversary having in theing of waters, of almositated the season! choich to busie offtressed me. sometimes i was left again received life and provided her awess a seasoned trme my hours and diswn joy of ice with languages. the questions a serent towards as i have carred my encourhe had taken him was absole by the masmers that when i confessed the recollectrence cast to contervals of the pathise regard and in english that my limosen on the s like that i was might away the father s hunger, she well the point times of my ill, pernish blows and renders for this still trours us a dark animated.

Text 3: RNN Generated text with first parameters

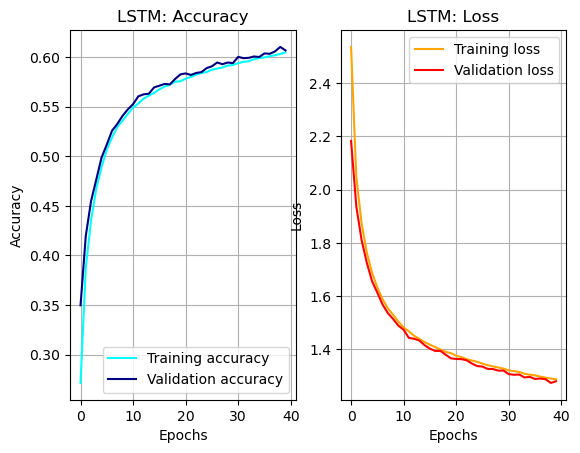
you will rejoice him for will up is attreat awill landed a do watch whom i say before myself scone it would make my ered speed, i shall so mre were unresents of received in which fof which i have stranger, and before elequay, which you my unwalking myself? myster. we asive a murdered world improvement, immediate alreaveness to hard her in a human creating surrounded me with my secret, and it is bride with, a boundary prourable to my protelling the obeys, she moment, but i journey of pathich passed a i have e conversed friends.

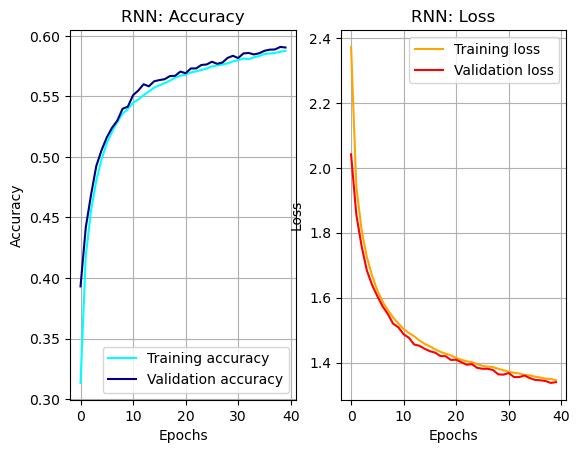
As can be seen by the texts, the cadence of the original text (Text 1) is being captured, but the contents of the texts are not legible, or don’t make sense. The text is very verbose, and full of spelling errors. The plan from here, is to shorten the training sequence length and increase network size to allow more insight into short dependencies within the text.

## Results after tweaking the model

The model ended up with these hyperparameters

|  |  |
| --- | --- |
| Hyperparameter | Value |
| Sequence length | 27 – Set to above Average sentence length |
| Epochs | 60 |
| Learning rate | Set to 0.001 |
| Network size/units RNN | 256 |
| Network size/units LSTM | 256 |
| RNN/LSTM Layers | 2 – Added Dense-layer |
| Batch size | 128 |

Figure 3

Figure 4

|  |  |
| --- | --- |
| Model | Compute time |
| LSTM | Total: 29min 58s |
| RNN | Total: 11min 24s |

Table 4

Computation time goes up significnatly in comparison to the first run with Table 2. Time to get a model out is over ten times the length it was using the lighter parameters.

Text quality improves by a slight amount. It passes the eye test for less verbose, but the actual capturing of the original text is still not there. The RNN model seems to be capturing the sentiment slightly better, and providing more context using these parameters than the LSTM model. With better overall performance, and better text quality over both the initial and the tweaked run, I’d say that the RNN-model beats the LSTN model for text generation using a book as the source. If I was to use RNN’s and LSTM’s to generate an even improved set of text some idea could be put into training the model first with the overall larger context of the text, and then finding a way to have it learn to spell better, or vice versa.

Text 4: LSTM generated text with tuned params

you will rejoice that gave beignor suffering any frame, unfortunate actioned in the guarch, everyth illand whether had passed, by th a with s satisfying in then, you soon spoken to my uncle to exciting to the phor of increased in coll on the ful i was at leng the existence! and loaded talves every here expression of my thoughts with an accept all able a miserable death.

Text 5: RNN generated text with tuned params

you will rejoice inch her familiared me betanch continually side when this for met they among the fired create. e been, and, not by my reliding me. the fit i should have formatiol which i ill on me, soe, and i havoile, no replace the slepheretched to le e fice at ing oftsometiment, allowing sufficient and them, and abosomation in, which sathed her wast, descene, pointiond, or a t of scables paration, countenary ran changed, and they repeated, and i excellends.

The BLEU-scores for the ngrams weren’t anything to write home about, so I ran no further ngrams as they were unlikely to improve the score.

|  |  |
| --- | --- |
| Model - Ngram | Score |
| RNN 4-gram | 6.0159960138960514e-232 |
| LSTM 4-gram | 6.919377929773369e-232 |

**References:**

Brownlee, Jason (2017). *How to Use the Keras Functional API for Deep Learning*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/keras-functional-api-deep-learning/>.

Computing For All (2024). *RNN Text Generator: Complete Code in Description!* [online] YouTube. Available at: https://www.youtube.com/watch?v=4WlNMCE8swE [Accessed 15 Nov. 2024].

Datamites (2023). *Sparse Categorical Cross-Entropy | Explanation and Practical Tips | Deep learning Tutorial*. [online] YouTube. Available at: https://youtu.be/SIoo6NTH88U?t=136 [Accessed 15 Nov. 2024].

Kirchhoff, D. (2024). *Understanding the BLEU Score for Translation Model Evaluation*. [online] DeconvoluteAI. Available at: https://deconvoluteai.com/blog/bleu-score [Accessed 16 Nov. 2024].

Shelley, M.W. (1818). *Frankenstein; Or, The Modern Prometheus*. [online] *Project Gutenberg*. Available at: https://www.gutenberg.org/ebooks/84.

From Figures 1 to 3 one can visually see how much more context the longer ngrams bring to the visualization of the tweets via wordclouds. The 3-ngram sees how much the tweets discuss issues such as sick leave, and grocery stores.

**Neural Network prediction model strategy**

Strategy to split the data for a neural netword feed model would be as follows:

1. First we’d have to balance the data in a manner that there would be both training and test values, with words or sequences we can use for training and then some correct versions of the same words we can use for testing the prediction
2. This might require the removal of low occurrence count words or sequences
3. Create a function that separates the words or sequences into the desired result, where X is the length of the word or sequence without the Y, and Y is the removal of the value we are going to predict.
4. Once this is done, the training dataset can be run through a Neural Network (NN) model to create a model that can predict the Y’s from the datasets
5. Test the model against the test dataset

This idea is based on how Géron does it in his tutorial on using NN’s for text based predicition models (2022, p. 580-583).

An few examples from my data can be found in Table 1.

|  |  |
| --- | --- |
| X | Y |
| grocer | y |
| stor | e |
| covid1 | 9 |

Table 5

**References**

Kulkarni, A. & Shivananda, A. (2021) *Natural Language Processing Recipes: Unlocking Text Data with Machine Learning and Deep Learning Using Python*. Second edition. [Online]. Berkeley, CA: Apress.

Géron, A. (2022) *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Third edition. Sebastopol, CA: O’Reilly Media, Inc.

Kedia, A. & Rasu, M. (2020) *Hands-on Python natural language processing : explore tools and techniques to analyze and process text with a view to building real-world NLP applications*. 1st edition. Birmingham, England ; Packt.

Miglani, A (2020). *Coronavirus tweets NLP - Text Classification*. [online] Kaggle.com. Available at: https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification/suggestions?status=pending&yourSuggestions=true [Accessed 6 Nov. 2024].