**Introduction**

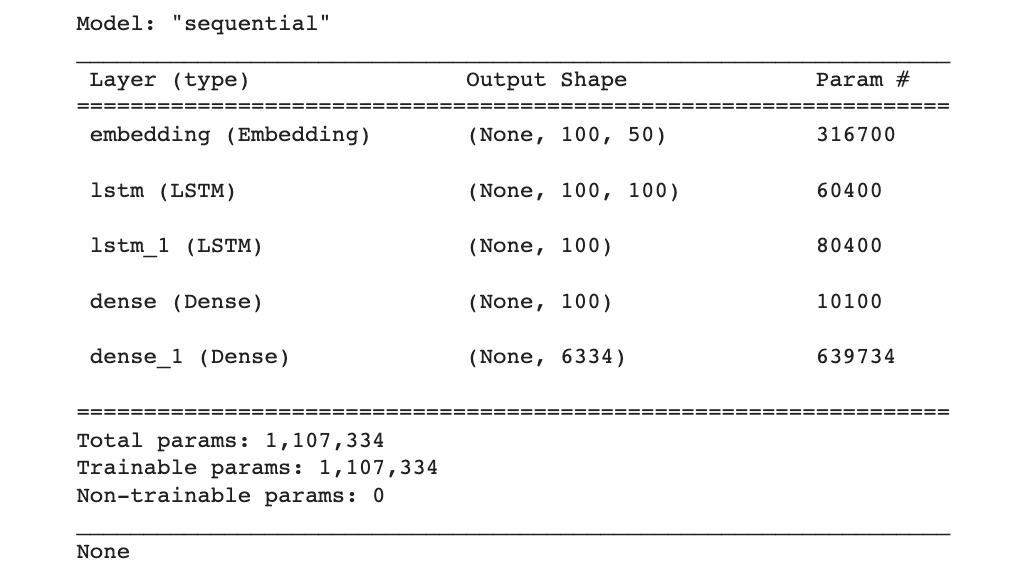
For this project, we were tasked with building a sequential model that can generate new lines or sequences based on the training data, which was a copy of *The Enchanted April,* by Elizabeth Von Armin (via Project Gutenberg). These types of neural networks are growing at a rapid pace in today’s industry, with lots of big tech companies using AI generative models to enhance their products. This project was a good introduction to working with generative models and gave me a good base model that I can adjust to work with different projects.

**Analysis**

To begin this project, I scrolled through Project Gutenberg’s online library for the online version of one of the included books, which I chose to be *The Enchanted April.* I copied the everything included in the downloaded UTF file, and pasted it into a .txt file, which I uploaded to my Google Collab notebook. After loading in the data, I printed out the first couple snippets from the book to see if the data uploaded properly, which it did. I then began preprocessing the text to make it usable for the generative model. This included processing the text, removing punctuation, non-alphabetic tokens, and converting all characters to lowercase. Using that cleaned text, I generated tokens while calculating the total number of tokens and unique tokens in the dataset. Then, I grouped together the tokens into sequences of a specified length, which I set to 100 – these sequences were saved to a new file called ‘simple\_seq.txt’.

**Methods**

*Model 1*

**

Starting the model building part of the assignment, I separated the text data into sequences of tokens. The sequences were the features in the model used to predict the next word in the sequence. The data was then tokenized into numerical vectors and ready to be used in the neural network. I then constructed the architecture of the first neural network, which included: one input layer, one output layer and three hidden layers. The first input layer was an embedding layer, and the hidden layers consisted of two LSTM layers. Long Short-Term Memory layers are used to capture long-term dependencies in the input data, while the original information stays preserved and then compared to the long-term dependencies at the end. Lastly, the neural network contained a dense layer with ReLu activation, and a final dense layer with Softmax activation, which transforms the output into a probability distribution over the vocabulary.

After defining the model’s architecture, I compiled the model using a categorical cross entropy loss function, which I deemed appropriate for this model’s goal: predicting the next word from the probability distribution. I used the Adam optimizer for updating the model’s weights during training, to minimize the loss function.

*Model 2*

*Table

Description automatically generated*

After building the first neural network, I decided to make significant improvements to the initial model by incorporating Bidirectional LSTM layers and adding Dropout for some sort of regularization. These improvements were made to enhance the model’s ability to learn complex relationships in the data and lower the loss score. The second model contained an input layer, and output layer and multiple hidden layers. For the second model, each LSTM layer was wrapped with a Bidirectional layer, which allowed the model to process the input sequence both forward and backward directions. As mentioned before, I added Dropout layers between the Bidirectional LSTM layers and after the dense layer with ReLu activation. Dropout is a regularization technique that helps models with overfitting by randomly setting a selected proportion of inputs to zero during training. I set my dropout rate to 0.2, meaning 20% of nodes in the layer were set to 0. I left the last two layers the same as the initial model and compiled and fit the model with the same loss function, optimizer and number of epochs.

**Results**

*Model 1*

Table

Description automatically generatedBased on the outputted results, my first model achieved a loss of 2.6993 and an accuracy of 40% in the last epoch. Though the model did not reach very high levels of, it was still able to learn patterns from the data that can be used for text generation. Since the goal of the model is to generate new text sequences, the results of the training data suggests that the model will produce somewhat coherent sentences, but no accurate or fluent text.

This is exactly what the model outputted, as it generated a text sequence of 50 words, but did not really make much sense. This is what the first model outputted in the first generated sequence:

‘talk was not talked so much important if it was looking at and had informed him she was not shut her trouble to the other of it was not a blank been taken out of her stick and the way and accompanied her that severe her opinions and the lighthouse’

Though the sequence does not make much sense, it was cool to see the neural network spit out a sequence of text (that tried to make sense) on it’s own.

*Model 2*

Table

Description automatically generated

After compiling and fitting the second model, I observed that it reached a loss of 3.2446 and accuracy score of 28.05%. In comparison to the first model, the second model performed significantly worse, as it produced an accuracy score of 40.01%. Though the second model increased complexity, the performance was way below expected and the generated text showed that.

Seed text: intelligent and very competent she had at once discovered that it was he who really ran the house who really did everything and his manners were definitely delightful and he undoubtedly was a charming person it was only that she did so much long to be let alone if only only she could be left quite quiet for this one month she felt that she might perhaps make something of herself after all she kept her eyes shut because then he would think she wanted to sleep and would go away romantic italian soul melted within him at the sight for

Generated text: it was as if she had been a little nervy her mother was a little peninsula it was in the other of the same breath and peach trees and a little boring she had been a little nervy her mother was a little boring she had been able to be robbed and she had been a little nervy her mother was the stove was the other of the same breath and was a little boring she had been a little nervy her skirts sweeping up to be able to see her a man and the first time and therefore had been

**Reflection**

Throughout the process of this homework assignment, I learned about the challenges and intricacies of building a generative text model. In addition, I gained more insight into Natural Language Processing, and the steps it takes to make text data usable for machine learning. One key takeaway from this assignment, is choosing the right model architecture. I saw that even though the second model was more complex, it did not outperform the first model. This shows that a more complex model does not always guarantee better results, as proven by my past homework assignments, and can sometimes even lead to overfitting.

When approaching a similar project in the future, I will be certain to keep a lot of these things in mind. For another project, I will start with a simple model architecture and gradually increase complexity and monitoring performance. It was harder to do so with this experiment, as the computing time for each model was very expensive and took a lot of time. With more resources, I can gradually adjust my model and take everything into consideration to try and create the most accurate generative model. In conclusion, the assignment gave me some really valuable hands-on experience working with generative models, and gave me a really good baseline on what to do for my final project.