

Introduction

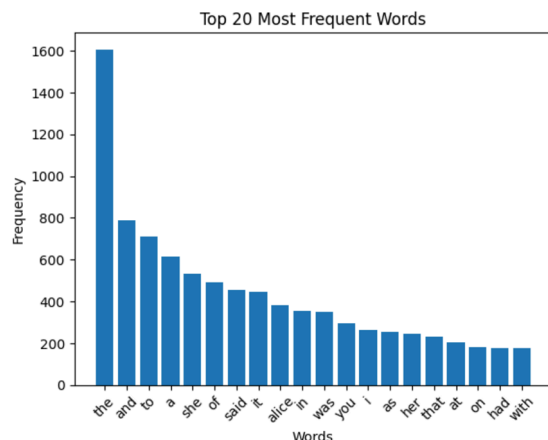
This assignment explores sequential models, specifically LSTMs, GRUs, and SimpleRNNs in my case. The goal is to generate new lines of text based on patterns learned from a chosen book, *The Three Little Kittens* by an anonymous author. Additionally, to build it from scratch using Keras/tensorflow without using any pre-trained models or transfer learning. The book was sourced from Project Gutenberg as a raw text file. The text will be preprocessed and converted to tokens to be inputted in each sequential model I mentioned. It will then be trained for a reasonable number of epochs and compared between the three types of models and hyperparameters for the most accurate. Finally, the best model used to generate new text sequences that reflect the style and content of the inputted book.

Analysis

The raw data was first loaded and cleaned using a series of preprocessing steps. Unwanted characters such as "--" were replaced with spaces, and punctuation was removed to ensure a clean tokenization process. The text was then split into individual tokens(words), and any non-alphabetic tokens were filtered out, leaving only meaningful words for analysis. To ensure consistency, all tokens were converted to lowercase. The additional text in the text file that was not a part of the book was removed. The total number of tokens in the dataset was calculated, and the number of unique tokens was determined by converting the list of tokens into a set. The results are given in the following table:

Total Tokens	Unique Tokens	Total Sequences
651	250	600

To prepare the data for sequential mode, sequences of length 51 were generated using a sliding window approach, where each sequence consists of 50 input tokens and the following token as the target. This generated a large number of sequences, forming the basis for training the model. The vocabulary size was found by fitting a tokenizer on the text, which mapped each unique token to a unique index. Additionally, a histogram of the 20 most frequent words in the dataset was found to give deeper understanding to potential outputs of the generated sequences from the model.



Methods

The first model I built was the LSTM model. I started with a blank sequential model. I added a LSTM layer with 128 nodes, input shape of the sequences, l2 kernel regularizer with a strength value of 0.01, and tanh activation function. An additional regularization of 0.3 dropout followed. Next, another LSTM layer with 64 nodes that didn't return the sequences. Finally a fully connected dense layer with softmax activation. The model was compiled with categorical cross entropy loss function and Adam optimizer with a learning rate of 0.001 using accuracy as the metric. The summary of the model is:

Layer (type)	Output Shape	Param #
lstm_27 (LSTM)	(None, 50, 128)	66,560
dropout_16 (Dropout)	(None, 50, 128)	0
lstm_28 (LSTM)	(None, 64)	49,408
dense_16 (Dense)	(None, 2418)	157,170
Total params: 273,138 (1.04 MB)		
Trainable params: 273,138 (1.04 MB)		
Non-trainable params: 0 (0.00 B)		

The model was fit on the dataset with a validation split of 0.2 and batch size of 128 over 20 epochs.

The next model I built was a basic RNN model. I started with a blank sequential model. I created an embedding layer with 100 dimensions per word. Next, I added a simpleRNN layer with 128 nodes and an elastic net kernel regularizer with 0.01 strength values for both l1 and l2. Another simpleRNN layer with 128 nodes. Then a 0.2 dropout regularization. 2 more simpleRNN layers with 64 nodes. All simpleRNN layers had tanh activation functions. A final fully connected dense layer with softmax activation was added. The model was compiled using categorical cross entropy loss function and adam optimizer with 0.001 learning rate using accuracy as the metric. The summary of the model is:

Layer (type)	Output Shape	Param #
embedding_12 (Embedding)	(None, 50, 100)	241,800
simple_rnn_16 (SimpleRNN)	(None, 50, 128)	29,312
simple_rnn_17 (SimpleRNN)	(None, 50, 128)	32,896
dropout_18 (Dropout)	(None, 50, 128)	0
simple_rnn_18 (SimpleRNN)	(None, 50, 64)	12,352
simple_rnn_19 (SimpleRNN)	(None, 64)	8,256
dense_18 (Dense)	(None, 2418)	157,170

A learning rate schedule that monitored the validation accuracy for plateau and would reduce the learning rate by a factor of 0.2. Additionally an early stopping was added that monitored validation accuracy that restored the best weights, and had a patience of 15 epochs. The model was fit with a validation split of 0.2 and batch size of 128 over 50 epochs(stopped at 19).

Results

The LSTM model obtained a final peak validation accuracy of 10.89% and training accuracy of 8.14%. The simpleRNN model performed worse with a final validation accuracy of 9.48% and training accuracy of 7.6%. The outputted sequences for the LSTM model was:

[illegible]

The outputted sequences for the simpleRNN model was:

[illegible]

Reflection

I learned a lot from this assignment, especially about the hundreds of ways to optimize a model. I spent over 40 hours trying hundreds of different combinations of hyperparameters, models, layers, regularization techniques, tokenizations, loss functions, activation functions, batch sizes, epochs, callbacks, and books. So I was really able to learn how everything works with each other and what scenarios to use which combination and hyperparameter values. In the end, I gave up trying to get higher validation accuracy and just decided to let it ruin my grade, as I now have to turn it in 5 days late with no results to show for it. However, I gained a lot of knowledge from it which is what really matters.