I decided the data-set size to be 1000. In each text, the numbers of elements 'a' and 'b' are randomized, but keeps the total text length not more than 20. When the numbers of element 'a' and 'b' are equal, the corresponding text is categorized to 1, which means the string is a member of the language, and vice versa. Same rule applies for the data-set with three-elements-text ('a', 'b', 'c'). The data-set is in the format of a list with 1000 [text, category]. Also, to make the two structures as fair as possible, the only difference is one using nn.RNN and the other using nn.LSTM. We know a lower learning rate slows down the learning and the model converges smoothly. I tried with both 0.001 and 0.0005. I started with 50 epochs, and decided to not change it in order to minimize the number of factors for trying different learning rates.
The following graphs are the results with learning rate = 0.0005, but I will also talk a bit about the other trials I have done without graphs due to page limit. First, let's look at the graphs of loss from each model. Except for the model "RNNs for ab", the training loss for the rest models eventually settled after 15 epochs. In the other trials I have done also with learning rate = 0.0005, the result is the same, but the training loss for "RNNs for ab" is much more stable at the end. In the trial with learning rate = 0.001, the model "RNNs for ab" becomes more or less stable after 40 epochs but its fluctuations are much smaller. Now, let's look at the graph of accuracy over the text length. I thought there is something wrong with the result, since in Figure 2, it seems only three models are plotted, the accuracy of "RNNs for abc" is not visible. However, they might be just highly overlapped, so I plotted separate graphs for RNNs and LSTMs. The separated graphs Figure 3 and Figure 4 proved my speculation, therefore, I plotted a bar graph Figure 5 to make them more visible to compare, and we can see that the red bar and orange bar are almost all at the same height at each text length. In another trial with learning rate = 0.0005, except the same result, the model "RNNs for ab" also highly overlapped with "RNNs for abc" and "LSTMs for abc" when text length is longer than 25. Where the model "LSTMs for abc" almost stabilized around 1.0 instead of stabilized in the beginning then performed the same as "LSTMs for abc".
In conclusion, from both the training loss graphs and accuracy over text length graphs, also with the comparisons of other trials, we can say LSTMs perfroms better then RNNs. More in detail, when the laguange is in the format of ab, LSTMs performs way better than RNNs. However, when the laguage is in the format of abc, the results from both structures do not differ much.

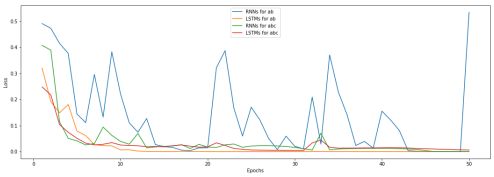

Figure 1: Loss

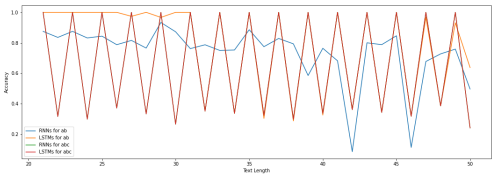

Figure 2: Accuracy

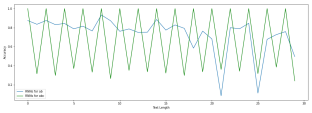

Figure 3: Accuracy for ab and abc while using RNNs

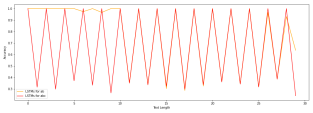

Figure 4: Accuracy for ab and abc while using LSTMs

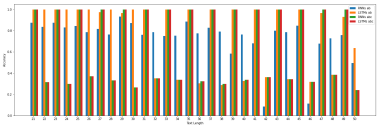

Figure 5: Accuracy comparisons show in bar graph