CS460G PA 3 RNN Writeup

Tyler Filbert

Steps to run the code:

To run the code, just run the rnn.py script. I have included the dataset within my submission folder, so the data should be automatically pulled in. If there is a problem reading in the data, see line 97 in rnn.py to give a file path to the text file. I have another script to read in the data, but this is imported to the rnn.py file, so no need to open or run that one.

Implementation Details:

I modeled a lot of my code after the given example from in class. I went through that code to understand what was happening, and then I applied batches to it. This meant I read in the data sentence by sentence. I decided to use the batch size in terms of sentences. Therefore, a batch size of 5, which was the batch size I used, is 5 sentences. This means that the actual character batch size was 5 times the length of the longest sentence for each batch. This realistically gets the batch size to be around 300 characters on average. I designed it this way because this is what the given code was like, so I just adapted it in the most straightforward way I could make batches. Subsequently, my hidden layer size was the batch size, however it was dynamic so when I predict values the batch size is 1 because I am only predicting one character.

To prep the data for training I decided to make use of PyTorch's DataLoaders, as well as the provided code to make dictionaries. This helped me easily batch the sentences, as it just creates lists of sentences with the specified batch size. I decided to remove the last set of examples if it was less than the batch size. As I iterated through the batches, I recorded the longest sentence in each batch and then padded the rest of the sentences to have a consistently sized final input tensor. I created one hots example by example, so at the end of iterating through all sentences in a batch I had to do some reshaping of the guessed output and target sequences to correctly calculate the loss. For the loss and optimizer, I used cross entropy and Adam, as modeled in class. I used a value of 5 for the number of epochs to test over. I found this didn't take too long and provided a good loss value.

Results:

Output from running my code:

```
EPOCH NUM: 1  --------> AVG LOSS:  1.5801
EPOCH NUM: 2  --------> AVG LOSS:  1.2863
EPOCH NUM: 3  --------> AVG LOSS:  1.2115
EPOCH NUM: 4  --------> AVG LOSS:  1.1734
EPOCH NUM: 5  --------> AVG LOSS:  1.1478
Completed Training!
QUEEN: LIZABESS: O: the seem the seem of the seem
```

Discussion:

My last epoch loss is usually around this value of 1.1 through all the runs I have had. For the output, my model seemed to learn that a person is followed by a sentence they say. This is good, but I found it interesting that it would repeat the contents of their sentence at the end.