Duong Hoang – CS 460G Spring 2022

CS460G MACHINE LEARNING
Project 4 – Text Generation with RNN Writeup

Part 1: Create a char-RNN to Generate Shakespearean Text

Implementation Choices:

- For data preprocessing, I read the entire file using *read(),* get unique characters using Set(), then created 2 dictionaries to convert from character to integer and vice versa.
- I created mini batches to train with each batch containing *BATCH_SIZE* (80) number of sequences, each sequence is of *SEQUENCE_LEN* (80) characters, by cutting off the characters that do not form a complete mini batch then split the sequences that would form complete mini batches into arrays of the size *BATCH_SIZE* x *SEQUENCE_LEN*. After that, each of the batches that got split will be shifted to create input and desire output arrays.
- As I used PyTorch, my input and desired output would be offset by 1. My input sequence contained values up to the last character, so for a sequence of length n, the input should only contain n-1 values (the last one is cut off). My desired output values should not contain a first value. This offset made the input and desired output line up with the RNN structure. The first input would be paired with the first desired output (which would technically be the second character in the sequence), and so on and so forth.
- Then the encoded input sequence (characters to integers) will be converted to one-hot vectors, which is a column vector where only it's corresponding integer index will have the value of 1 and the rest of the vector will be filled with 0's.
- To build the char-RNN, I used PyTorch library's LSTM model with a dropout layer (to avoid overfitting). The model had the following hyperparameters:

Input Size	Output Size	Hidden Nodes/Layer	Hidden Layers	Dropout Rate
Vocab Size = 65		800	2	0.5

- For loss function, I used PyTorch's *CrossEntropyLoss()* and for optimizer, I used PyTorch's *Adam* with a learning rate of 0.005.
- I did not use GPU for running the RNN, so the number of epochs was pretty small (10) due to time limitations caused by running on CPU.
- I used the *top_k* parameter in the process of predicting and generating text sample to make the generated text seems more reasonable as *top_k* means the model would consider choosing from the K most probable characters instead of some uncommon ones. In this model, K would be 7, so the model would choose from the top 7 probable next characters result from the current characters instead of considering all the possible next characters.
- I used *softmax* function to calculate the probability distribution of next characters.
- During training, I used PyTorch's *clip_grad_norm_* with *clip* = 5 to prevent the exploding of gradient problem.

Duong Hoang – CS 460G Spring 2022

Training:

- Training regimen:

Text File	Epochs	Batch Size	Sequence Length	Learning Rate {ALPHA}	TOP_K	Sample File
tiny-shakespeare.txt	10	80	80	0.005	7	generated_char.txt

- Training Loss:

Epoch 0: Loss = 3.1767...	Epoch 5: Loss = 1.6684...
Epoch 1: Loss = 2.3710...	Epoch 6: Loss = 1.6636...
Epoch 2: Loss = 2.0462...	Epoch 7: Loss = 1.5807...
Epoch 3: Loss = 1.8602...	Epoch 8: Loss = 1.5241...
Epoch 4: Loss = 1.7514...	Epoch 9: Loss = 1.4869...

- Final training loss after 10 epochs of training with PyTorch's CrossEntropyLoss() is about 1.4869
- Sample with prime, a string, **"QUEEN"** (the full 1000-character-long sample can be found in text file *generated_char.txt*):

QUEEN:

I twenty he therefore.

LEONTES:

And masters: the good true-pollow. They love you the love,

And mistrusting the morrow of thy life

Is be subjected, the stany of any traitors take the predent.

LUCENTIO:

I thank you;

I did: that the weary well, that thy head.

Wongeances, he's him and here bears the must,

The mind and there one and brave tell that brothers

And made him, or to malise't fair word,

As hear those as they tell her by her father:

I have and so warm with me; which, then worthied

Would thou have mine own subjection are my since, thy badd help to drink you.

Bonus: Word-Based RNN for Shakespeare

Implementation Choices:

- I also use PyTorch's LSTM model to generate word-based text. The only changes from the char-RNN model are how data were preprocessed, hyperparameters, and the prime for sampling.
- For data preprocessing, I read the entire file using *readlines()* to get a list of lines, then I split each line using split() and added each word along with a space to a list of words. At the end of each line, I added a new line character to that list. I created the list in this way so that I can still keep the original structure of the text data after the words' whitespace were stripped off. Then, in the same method as char-RNN, I created 2 dictionaries to convert from word to integer and vice versa.
- To build the word-RNN, I again used PyTorch library's LSTM model with a dropout layer (to avoid overfitting). The model had the following hyperparameters:

Input Size	Output Size	Hidden Nodes/Layer	Hidden Layers	Dropout Rate
Vocab Size = 25672		300	2	0.5

- For loss function, I also used PyTorch's *CrossEntropyLoss()* and for optimizer, PyTorch's *Adam* with a learning rate of 0.005.
- The prime for this RNN is a list of string instead of a string (a list of characters). So, the prime is *["QUEEN"]* instead of *"QUEEN"*

Training:

- Training regimen:

Text File	Epochs	Batch Size	Sequence Length	Learning Rate {ALPHA}	TOP_K	Sample File
tiny-shakespeare.txt	10	30	10	0.001	7	generated_word.txt

- Training Loss:

Epoch 0: Loss = 3.9584...

Epoch 1: Loss = 3.7906...

Epoch 2: Loss = 3.8522...

Epoch 3: Loss = 3.7357...

Epoch 4: Loss = 3.6224...

Epoch 5: Loss = 3.5287...

Epoch 6: Loss = 3.4225...

Epoch 7: Loss = 3.3848...

Epoch 8: Loss = 3.4086...

Epoch 9: Loss = 3.4047...

- Final training loss after 10 epochs of training with PyTorch's CrossEntropyLoss() is about 3.4047
- Sample with prime, a list of strings, **["QUEEN"]** (only the first 95 words are included here, the full sample can be found in text file *generated_word.txt*):

QUEEN people and

have not not been and the king and be

Duong Hoang – CS 460G Spring 2022

To see me to a own mother;

But I am no king to the king

Than I shall see you is the state,

I have the king and be

To make the day to tell

The noble beams of her own

And many that have the man of his death,

But I will not see my brother is

That they are in our own air,

To not not to be a beams and that have

The army of his body with a crown?