CS 7180 Special Topics in AI: Deep Learning
Northeastern University, Spring 2019

Assignment 2

Name: Tyler Brown UID: 001684955

1 Problem 1

Assignment 1 requested that we use PyTorch to build a RNN to generate poetry based on 154 sonnets from William Shakespeare. A sonnet is thought to be amenable to generative modeling because they follow a specific format. The approach I chose is based on a PyTorch tutorial [1] on generating Shakespeare using a character-level RNN. Improvements to this model were based on Deep-speare by Lau et. al. (2018) [2]. I review the pre-processing, RNN, and RNN improvements I took during this assignment. One of my key take-aways is that feature engineering is still an important part of model performance for this application of deep learning.

1.1 Pre-processing

The pre-processing steps I took for the RNN are similar to those taken in the PyTorch tutorial [1]. The string module provides a list of all unique characters. Shakespeare sonnets are read in from the text file. The tutorial made a point of using the 'unidecode' package to handle any issues with unicode. I added regular expressions to remove digits because we know poetry does not involve numbers. I also removed multiple newline and space characters because I considered this to be redundant information which add noise instead of signal for the RNN. Model input was essentially a string of text which would then be sliced into random segments of 40 characters for the RNN to process as a batch.

My final pre-processing included words as well as characters. Each training batch for my improved RNN consisted of a character-based batch concatenated with a word-based batch. This approach was taken as part of the Deep-speare [2] modeling process. Words were tokenized on spaces only. I realize that other good tokenizations are available but tried to make a simplifying assumption to fit within the scope of this assignment. The pre-processing for words involved finding all unique words. I also assigned the first word of each line in a sonnet to a hash table where each hash value mapped to a list of associated sonnet lines. I converted all the hash keys to a list, randomly chose a hash key, randomly chose one of the sonnet lines associated with that key, then returned those words as training batch input. I didn't want to randomly select words during preprocessing because the rhyme scheme between words sharing a line is not random whatsoever. I then concatenated these selected words with randomly chosen characters as to create my improved training batch.

1.2 RNN

The RNN model, SimplePoet, I created was heavily based on the PyTorch Shakespeare tutorial [1]. It consisted of the following layers in Figure 1.

Embedding Layer (encoder) → LSTM Layer → Linear Layer (decoder)

Figure 1: RNN Model Architecture

The Embedding layer in Figure 1 is used to store character embeddings and retrieve them using indices. The input to the Embedding module in PyTorch is a list of indices, and the output is the corresponding character embeddings. The LSTM layer applies a multi-layer long short-term memory (LSTM) RNN to an input sequence. Finally, the Linear layer applies a linear transformation to the incoming data in the form $y = xA^T + b$.

Tunable parameters included the learning rate, hidden size, number of epochs, number of characters depending on pre-processing steps, output size, and temperature. Previously, I've found learning rate to be one of the most important hyperparameters. In this assignment, I spent most of my time on pre-processing and understanding the temperature parameter.

The LSTM, SimplePoet, does appear to pick up on sentence structure and/or sonnet structure in Figure 4. The amount of runtime/amount of training data is fairly quick for SimplePoet. However, qualitatively, the output is also fairly bad. I do not think increased training time would necessarily fix some fundamental issues with SimplePoet such as lack of context-awareness. I did read several examples which requested increased training data for better performance. Figure 2 shows that the RNN SimplePoet model is converging after about 200 epochs.

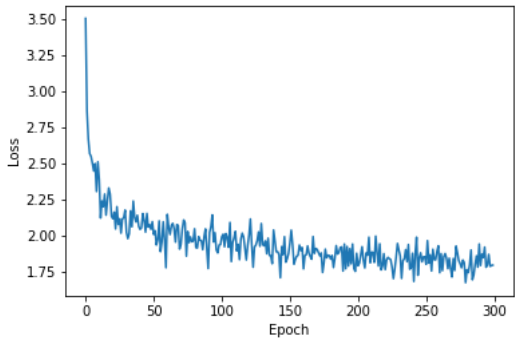

Figure 2: RNN: SimplePoet Epoch vs. Loss

The most sensible temperature appears to be 0.75 in Figure 4b although output is quite a ways from being mistaken for a human. The most noticeable thing about temperature is the amount of words that appear to be made up. When temperature is too low or too high, the poem structure will appear to be in place but the words seem made up.

1.3 Improving RNNs

Improvements to the SimplePoet RNN were based on Deep-speare [2] model. There appeared to be several good ideas in this paper which could have been pursued in this assignment. I added a fairly straight-forward feature from Deep-speare which included word

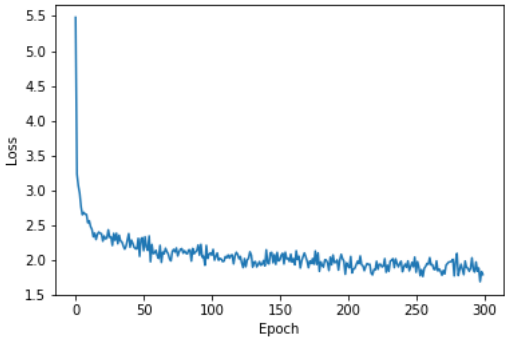

Figure 3: Improved RNN SimplePoet++ Epoch vs. Loss

embeddings concatenated to character embeddings. The model took longer to train but the example output, see Figure 4d appears much more sensible than a strictly character based model. Quantitatively, this model achieved a similar loss function to SimplePoet.

1.4 Discussion

This assignment was helpful for getting more familiar with RNN models. I was able to get qualitative improvements with SimplePoet++ using word embeddings in addition to character embeddings.Given more time, I would have liked to explore the Deep-speare [2] use of encoder/decoder models like the Seq2Seq paper for generating Shakespeare Sonnets.

References

[1] spro. practical-pytorch/char-rnn-generation.ipynb at master · spro/practical-pytorch. `https://github.com/spro/practical-pytorch/blob/master/char-rnn-generation/char-rnn-generation.ipynb`, Jul 2017. (Accessed on 02/24/2019).

[2] Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. Deep-speare: A joint neural model of poetic language, meter and rhyme, 2018.

1.5 Appendix

```
print(evaluate_SimplePoet('Shall I compare thee to a summers day?\n', 300, temperature=1.5))
```

```
Shall I compare thee to a summers day?
Lof p'aye loven of ret (fincee
Nectorm imy:prl2s;
Abudnoelay
A but rest whert dith in?
Withep:Be shoul, co giBlingle?
Let onp id
Wi?:
Chlok nicks eeenwangquiquess viell, ner thous eekphalps, paty, kees it-not rrummeftcops thy:
riwnlay huse bload eyes love I fronguir.
Wie frowrrite iTen perfantieyumi
```

(a) SimplePoet, temperature=1.5

```
print(evaluate_SimplePoet('Shall i compare thee to a summers day?\n', 300, temperature=0.75))
```

```
Shall i compare thee to a summers day?
Which not of it if you thee,
But deeps it me self love reet aftainter thou rue is the so crove thee,
And eyes see,
With thou the which of thee, byath contin fuly for wherome as thee after fick thee, with so thee, fee,
And thiss aftreing be to in my his impite it the bough him love's it it be,
All po
```

(b) SimplePoet, temperature=0.75

```
print(evaluate_SimplePoet('Shall i compare thee to a summers day?\n', 300, temperature=0.25))
```

```
Shall i compare thee to a summers day?
When the sun thee the the sun the sund the sun the self the beauty the summer the sun the so thee,
And the summer the plove the still the the sing the sund with the sun thee,
And the sun thee,
And for the see,
And the shing the summer thee the sun this the sun thee the sund beauty the sunding of in
```

(c) SimplePoet, temperature=0.25

```
print(evaluate_SimplePoet('Shall I compare thee to a summers day?\n', 300, temperature=0.75))
```

```
Shall I compare thee to a summers day?
Mie self wee in when thy flow.
The hed beault were the deates, thou deauty.
Whom thy of what the wouth state weed some this chat whil mast woMt the and thy (diunkingiver,
Forld is doth be with with sweect and hould wil with fore the se.
Som thou with but vients wele plepence sould and thou lies,
And
```

(d) SimplePoet, added word embeddings, temperature=0.75

Figure 4: Poetry Output of RNN (SimplePoet) and Improved RNN (SimplePoet++)