Word/Character Level Language Modeling With RNN

Lefteris Soulas
Computer Science Department
New York University
es3431@nyu.edu

1. Question Answers

Q1. The lua script that executes the code is included separately.

Q2. The $i, prev\_h$ variables are used in a function inside the LSTM cell and returned as a linear combination. This function maps to $T_{2n,4n}\binom{h_t^{l-1}}{h_{t-1}^l}$, in the paper. The $prev\_c$ maps to c_{t-1}^l, which is the memory cells from the previous time step.

Q3. The create_network() function, returns a single instance of the network which is later cloned many times in order to form the various time steps. The network at this point is folded.

Q4. $model.s$, is a vector which contains the hidden state of the model. $model.ds$, is a vector which contains the gradients from every hidden state and is updated in the backprop. $model.start\_s$, is a helper variable that initializes the very first hidden state of the model ($s[0]$). $model.start\_s$, is reset to 0 in the beginning of the program and in the very end, if the last batch contains less inputs than $batch\_size$, in which case we discard the last mini-batch and retrain on the beginning of the sequence as it was the very end.

Q5. In order to deal with the vanishing/exploding gradients, we perform gradient clipping above a certain threshold (max_grad_norm). If the norm of the gradients is above this threshold, the gradients are normalized so they have maximum value equal to max_grad_norm.

Q6. The optimization method used, is mini batch gradient descent. The batch length is determined by $batch\_size$. This algorithm doesn't use momentum, but uses a decay factor which decreases the learning rate after a number of epochs.

Q7. We have to zero out the extra output in the back propagation step in order not to affect the gradients.

2. Word level language model

For this part of the assignment we created the function *query_sentences.lua* which takes as input a number and some words and tries to complete the sentence in the best possible way. We didn't train a very elaborate model for this part, but we rather tested the functionality on a model similar to the baseline. In order to allow variations in our sequences, we didn't simply select the word with the highest probability, but used a multinomial sample generator to pick the predicted word.

3. Character level language model

For this part of the assignment we experimented with the LSTM RNNs in order to model a character based language model. In particular, our implementation is based on Wojciech Zaremba implementation [3].
As a first step we changed the code so it can predict characters instead of words and added an extra output, which is the log probabilities of each word given a certain input. In the beginning we trained the baseline model with no change at all and noted validation perplexity of 430. Following the assignment's instructions we increased the sequence length to 50 and retrained the model. There new validation perplexity was 350.

In order to improve the model even more we tried to tune the optimization parameters. An interesting observation we noticed is that the model's perplexity doesn't decrease if we increase the RNN's size. On the contrary, we tried three different models, with rnn_size $=\{1000, 1500, 2000\}$ and all of them were very slow and very difficult to converge. Eventually we gave up the effort as the perplexity was extremely large even after the 10^{th} epoch. On the other hand, we noticed that relatively smaller RNN sizes (around 400-500) perform much better. That's the range we kept for the rnn_size in the rest of our experiments.

During the hyper parameter tuning, we noticed that the initial learning rate of 1 can be too harsh in most cases and the mode's perplexity starts to decrease only after reaching the *max_epoch*, when the learning rate is divided by *decay*.

Thus, we kept an initial learning of 0.5 during the rest of our experiments.

The network has as a first layer a LookupTable module, which is responsible for learning the word/character embeddings. As we know very well, the initialization of the weight parameters is really important for the convergence of the model. Instead of assigning them arbitrarily to some value, we thought of using the vector representations from GloVe[1] or Word2Vec[2], in order to initialize each 200-D parameter vector for each of the 10000 words that exist in our vocabulary. While this could be a meaningful warm start for the word language modeling, there is little that we can do when it comes to character model, as our class space is limited (only 50 characters). Therefore we intuitively initialize the weights to 0.02, which is the uniform probability of each of the 50 words.

In our continuous effort to optimize the hyper parameters we run a few more models, trying to mimic the simple modification that the increase of the sequence length brought to the perplexity of the model. We tried to test the hypothesis that by increasing the sequence length, we are lowering the perplexity (something which intuitively makes sense as the RNN will use more ”state” in order to distinguish different sequences of characters).

The best model we found till now achieves training perplexity 163.713 and validation perplexity of 262. We used a sequence length of 100, with batch size of 50 and rnn size of 500. The model was trained for 20 epochs with a decay factor of 2. As you can see in Figure 1.1, the model already started to overfit after 20 epochs. That’s logical as we used a dropout rate of 0.

In order to mitigate this phenomenon and be able to scale, we decided to commit and train another model that will take into account all the above lessons we have learned so far. More specifically it will have a relatively small rnn size (we picked 400), a large sequence length (we picked 400) and a dropout rate of 0.6 in order to under-perform on the training set but generalize better. The use of a dropout layer is proven to increase generalization [4].

We trained the above network for 200 epochs, but because we wanted to have absolute control on the learning process, we controlled manually the learning rate after the 100th epoch. The model achieved the perplexity of 187.84 in the validation set and 444.5 in the training set 2. The discrepancy between the two, shows that we were able to learn a well generalized model and achieve a very good perplexity comparing to our other efforts. The total time of tuning the model was 2068 minutes.

The most bizarre/disappointing realization, is that when we run the a4_grading.py script, the perplexity gets to be close to 300. When we use an interactive session to load the model and train with the run_valid() function from the initial code, the perplexity is 187. We couldn’t figure out why this bug appears. You can find the core_network for this model inside the folder *forThe187PerplexityModel*, as well as a script that loads it into memory and sets up the model so you can test the validation function yourself.

Moreover, we tried changing the lstm cell functions as the ones in the code weren’t exactly the same as the ones presented in [5], but the accuracy of the model didn’t improve.

References

[1] Pennington et al, *GloVe: Global Vectors for Word Representation*

[2] Mikolov et al, *Efficient Estimation of Word Representations in Vector Space*

[3] Wojciech Zaremba, *Long Short Term Memory Units Code*

[4] Wojciech et al, *Recurrent Neural Network Regularization*

[5] Alex Graves, *Generating Sequences With Recurrent Neural Networks*

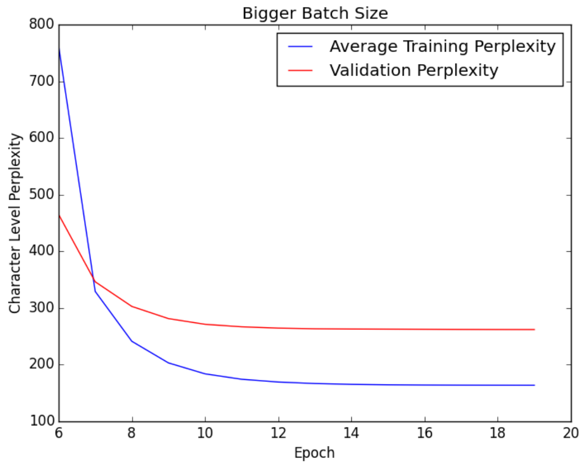
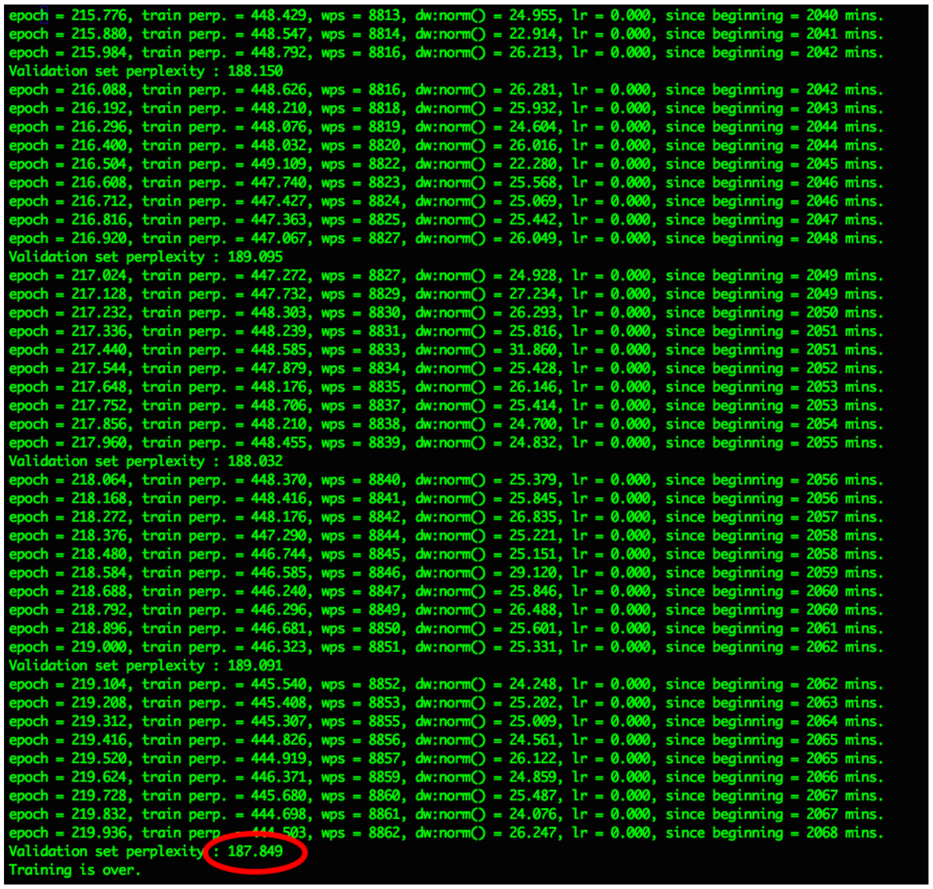


Figure 1. Perplexity of qualifying model



```
epoch = 215.776, train perp. = 448.429, wps = 8813, dw:norm() = 24.955, lr = 0.000, since beginning = 2040 mins.
epoch = 215.880, train perp. = 448.547, wps = 8814, dw:norm() = 22.914, lr = 0.000, since beginning = 2041 mins.
epoch = 215.984, train perp. = 448.792, wps = 8816, dw:norm() = 26.213, lr = 0.000, since beginning = 2042 mins.
Validation set perplexity : 188.150
epoch = 216.088, train perp. = 448.626, wps = 8816, dw:norm() = 26.281, lr = 0.000, since beginning = 2042 mins.
epoch = 216.192, train perp. = 448.210, wps = 8818, dw:norm() = 25.932, lr = 0.000, since beginning = 2043 mins.
epoch = 216.296, train perp. = 448.076, wps = 8819, dw:norm() = 24.604, lr = 0.000, since beginning = 2044 mins.
epoch = 216.400, train perp. = 448.032, wps = 8820, dw:norm() = 26.016, lr = 0.000, since beginning = 2044 mins.
epoch = 216.504, train perp. = 449.109, wps = 8822, dw:norm() = 22.280, lr = 0.000, since beginning = 2045 mins.
epoch = 216.608, train perp. = 447.740, wps = 8823, dw:norm() = 25.568, lr = 0.000, since beginning = 2046 mins.
epoch = 216.712, train perp. = 447.427, wps = 8824, dw:norm() = 25.069, lr = 0.000, since beginning = 2046 mins.
epoch = 216.816, train perp. = 447.363, wps = 8825, dw:norm() = 25.442, lr = 0.000, since beginning = 2047 mins.
epoch = 216.920, train perp. = 447.067, wps = 8827, dw:norm() = 26.049, lr = 0.000, since beginning = 2048 mins.
Validation set perplexity : 189.095
epoch = 217.024, train perp. = 447.272, wps = 8827, dw:norm() = 24.928, lr = 0.000, since beginning = 2049 mins.
epoch = 217.128, train perp. = 447.732, wps = 8829, dw:norm() = 27.234, lr = 0.000, since beginning = 2049 mins.
epoch = 217.232, train perp. = 448.303, wps = 8830, dw:norm() = 26.293, lr = 0.000, since beginning = 2050 mins.
epoch = 217.336, train perp. = 448.239, wps = 8831, dw:norm() = 25.816, lr = 0.000, since beginning = 2051 mins.
epoch = 217.440, train perp. = 448.585, wps = 8833, dw:norm() = 31.860, lr = 0.000, since beginning = 2051 mins.
epoch = 217.544, train perp. = 447.879, wps = 8834, dw:norm() = 25.428, lr = 0.000, since beginning = 2052 mins.
epoch = 217.648, train perp. = 448.176, wps = 8835, dw:norm() = 26.146, lr = 0.000, since beginning = 2053 mins.
epoch = 217.752, train perp. = 448.706, wps = 8837, dw:norm() = 25.414, lr = 0.000, since beginning = 2053 mins.
epoch = 217.856, train perp. = 448.210, wps = 8838, dw:norm() = 24.700, lr = 0.000, since beginning = 2054 mins.
epoch = 217.960, train perp. = 448.455, wps = 8839, dw:norm() = 24.832, lr = 0.000, since beginning = 2055 mins.
Validation set perplexity : 188.032
epoch = 218.064, train perp. = 448.370, wps = 8840, dw:norm() = 25.379, lr = 0.000, since beginning = 2056 mins.
epoch = 218.168, train perp. = 448.416, wps = 8841, dw:norm() = 25.845, lr = 0.000, since beginning = 2056 mins.
epoch = 218.272, train perp. = 448.176, wps = 8842, dw:norm() = 26.835, lr = 0.000, since beginning = 2057 mins.
epoch = 218.376, train perp. = 447.290, wps = 8844, dw:norm() = 25.221, lr = 0.000, since beginning = 2058 mins.
epoch = 218.480, train perp. = 446.744, wps = 8845, dw:norm() = 25.151, lr = 0.000, since beginning = 2058 mins.
epoch = 218.584, train perp. = 446.585, wps = 8846, dw:norm() = 29.120, lr = 0.000, since beginning = 2059 mins.
epoch = 218.688, train perp. = 446.240, wps = 8847, dw:norm() = 25.846, lr = 0.000, since beginning = 2060 mins.
epoch = 218.792, train perp. = 446.296, wps = 8849, dw:norm() = 26.488, lr = 0.000, since beginning = 2060 mins.
epoch = 218.896, train perp. = 446.681, wps = 8850, dw:norm() = 25.601, lr = 0.000, since beginning = 2061 mins.
epoch = 219.000, train perp. = 446.323, wps = 8851, dw:norm() = 25.331, lr = 0.000, since beginning = 2062 mins.
Validation set perplexity : 189.091
epoch = 219.104, train perp. = 445.540, wps = 8852, dw:norm() = 24.248, lr = 0.000, since beginning = 2062 mins.
epoch = 219.208, train perp. = 445.408, wps = 8853, dw:norm() = 25.202, lr = 0.000, since beginning = 2063 mins.
epoch = 219.312, train perp. = 445.307, wps = 8855, dw:norm() = 25.009, lr = 0.000, since beginning = 2064 mins.
epoch = 219.416, train perp. = 444.826, wps = 8856, dw:norm() = 24.561, lr = 0.000, since beginning = 2065 mins.
epoch = 219.520, train perp. = 444.919, wps = 8857, dw:norm() = 26.122, lr = 0.000, since beginning = 2065 mins.
epoch = 219.624, train perp. = 446.371, wps = 8859, dw:norm() = 24.859, lr = 0.000, since beginning = 2066 mins.
epoch = 219.728, train perp. = 445.680, wps = 8860, dw:norm() = 25.487, lr = 0.000, since beginning = 2067 mins.
epoch = 219.832, train perp. = 444.698, wps = 8861, dw:norm() = 24.076, lr = 0.000, since beginning = 2067 mins.
epoch = 219.936, train perp. = 444.503, wps = 8862, dw:norm() = 26.247, lr = 0.000, since beginning = 2068 mins.
Validation set perplexity : 187.849
Training is over.
```

Figure 2. Output from the 187 Perplexity model