Homework 3 of Deep Learning

Wang Yikai, 2017310740

May 29, 2018

Guidance

1. Train: run **python main.py**, the loss and perplexity will be printed in the screen during training. After training, run **tensorboard –logdir=logs** to see the curves of training loss and training perplexity. The model will be saved in models/.

2. Test: Put the test file named 'test.txt' into the director data/ptb/, and run **python main.py –testing True** to test a well-trained model saved in the path saved_model/.

Details

The network designed in lstm.py has 2 layers of LSTM. Each LSTM layer is wrapped with a dropout layer. Each word is converted to an embedding with 200 dimensions, as well as the hidden dimensions of the LSTM. The embeddings also need to pass a dropout layer.

The memory state of the network is initialized with zero. In order to ovoid gradient vanish, make the learning process tractable, I use an unrolled version of network which contains a fixed length(step_size) of words to feed in LSTM's inputs and targets.

The loss function of the network is:

$$loss = -\sum_{i=1}^{T} \ln P(x_i | x_1, \cdots, x_{i-1})$$

which is called negative log probability loss, where T is the step_size of the network. Perplexity equals to **average negative log probability loss** square e, which is:

$$perplexity = \exp\left(\frac{loss}{T}\right)$$

I use a decaying learning rate with 0.7 decay rate during the training, and train for 20 epochs. After finishing an epoch, I cross validate the current model using data valid.txt.

Novelty

There are some extra techniques I find in other materials to further improve the model.

I use dropout both in LSTM cells and in embeddings, although there still remains a little bit over-fitting in the results.

I try to use attention, but it does not improve the performance. This because attention is to differentiate the weight of each input word, which is suitable for sequence to sequence, but may be not suitable for single word prediction.

During training, an exponential decay of learning rate during training is used to speed up training at the beginning and avoid oscillation in the end.

I use a truncated back propagation by limiting the maximum gradient during training.

Results

After 45k step, the training loss is around 90, the training perplexity is around 70, and validation perplexity is around 102.

The training curves in tensorboard is:

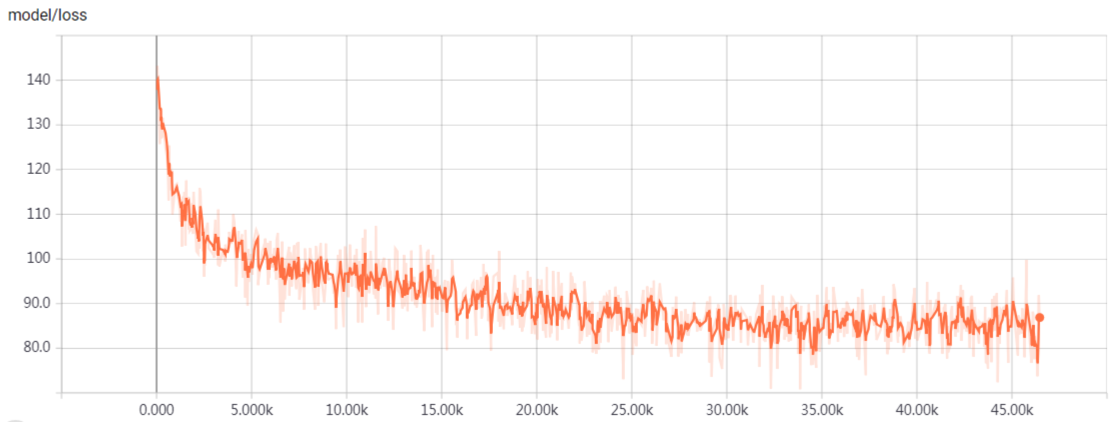

Figure 1: Training loss w.r.t. step

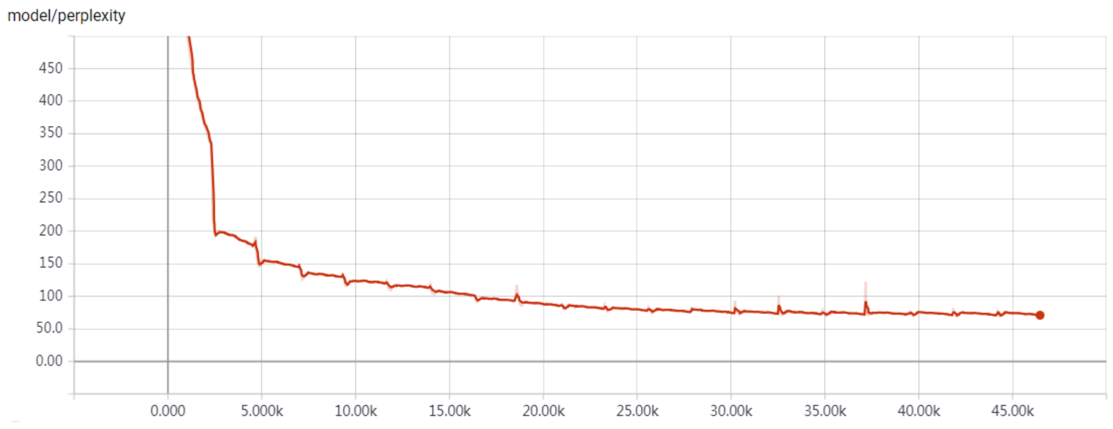

Figure 2: Training perplexity w.r.t. step

The perplexity during training and validation w.r.t. epoch is:

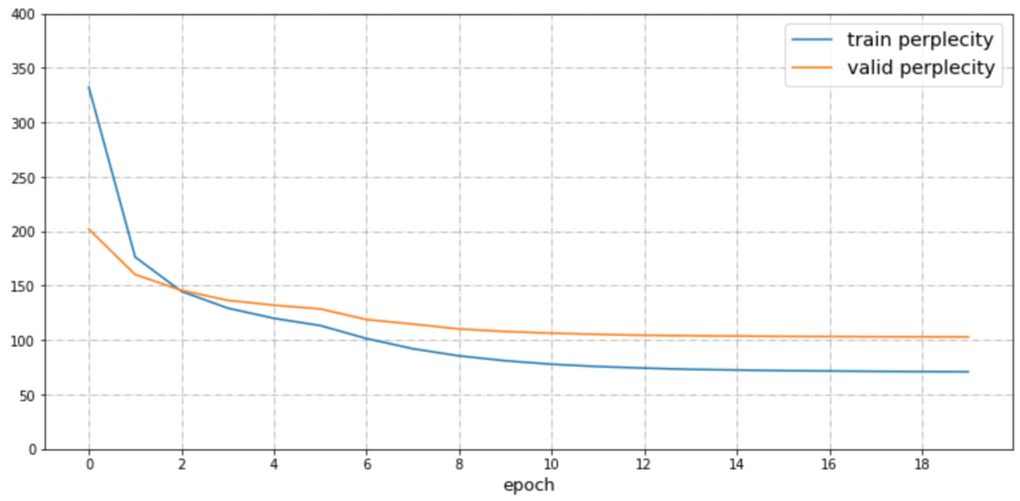

Figure 3: Perplexity w.r.t. epoch

If you want to test my model, please follow the guidance described at the beginning of this paper. The testing logs on the screen will look like:

-Restoring model and start testing:

-Progress = 100.0%

-Test perplexity = 102.963