Anna Pustova

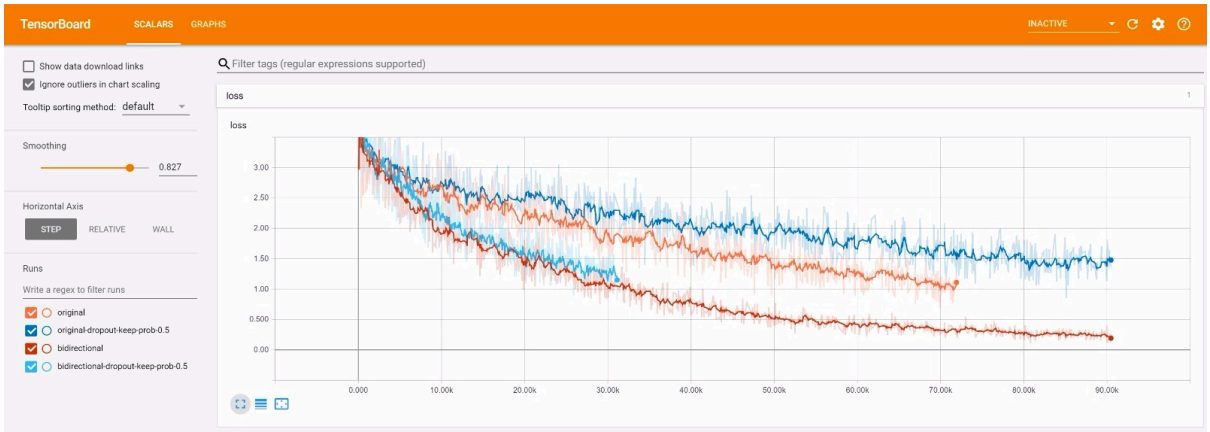
Angelina Pustynskaia

MT: Übung 5

Thema: Encoder-Decoder-Modelle

Repositorium: https://github.com/apusto/daikon

Picture 1.



The first graph (Picture 1. provides the information about our first testing attempts that we needed in order to adjust the final version of the submitted daikon program. The training was done with baseline parameter (bpe 50000, vocabulary size 50000 etc.), however with a significant difference that all 4 preliminary trainings were processed with 1 epoch. *Original* gave us an overview how the program runs overall. Then we implemented a drop-out with a 50% chance to set probability to 0 to the output of the decoder. *Original-dropout-keep-prob-0.5* shows that the result got worse. Then we implemented bidirectional encoding and we spot a significant amelioration in *bidirectional* in Picture 1. The same dropout procedure has been developed for *bidirectional,* as a result we got worse results of the loss in *Tensorboard* as happened to *original*. Therefore, we decided not to implement the dropout to our final submitted version, and keep on bidirectional model with suppressing of unknown words.

Preprocessing

Normalisation:

$ perl mosesdecoder/scripts/tokenizer/normalize-punctuation.perl \ < corpus/corpus.train.de > corpus/corpus.norm.de

Tokenization

$ perl mosesdecoder/scripts/tokenizer/tokenizer.perl -l de -q \ < corpus/corpus.norm.de > corpus/corpus.tok.de

Training of the Truecase model:
$ perl mosesdecoder/scripts/recaser/truecase.perl --model corpus/truecase-model.de < corpus/corpus.train.tok.de > corpus.tc.de
$ perl mosesdecoder/scripts/recaser/truecase.perl --model corpus/truecase-model.en < corpus/corpus.train.tok.de > corpus.tc.en

Training of the BPE model:
$ python subword-nmt/learn_bpe.py -i corpus/corpus.train.tc.de -s 150000 -o bpe.codes.de
$ python subword-nmt/learn_bpe.py -i corpus/corpus.train.tc.en -s 150000 -o bpe.codes.en

Hyperparameters:
HIDDEN_SIZE = 2048 – we increased number of neuron layers, as it helps to improve perplexity score and performance overall.
Vocabulary size was increased to decrease the number of the unknown words:
SOURCE_VOCAB_SIZE = 150000
TARGET_VOCAB_SIZE = 150000
After the decoding step we dealt with the unknown words in the network itself. The second index for the unknown words is set to zero, which means that network never predicts the unknown words. The idea to set the probability of the unknown words to zero. The network will never learn the unknown words.
LEARNING_RATE = 0.001 – we decided to put something in between, as when it's too big or too high it doesn't learn the state normally, and the attempts conducted in the uebung 4 that 0.001 is a good solution as it slightly ameliorated the perplexity score and the training overall was a bit faster.
Number of epoch = 6.

Code is in the repositorium: https://github.com/apusto/daikon
Problem:
We faced a very single problem – GPU 😣. We couldn't get the spot until Saturday evening, despite we've tried literary every hour to get the GPU throughout the week.
The most common traceback was:

ResourceExhaustedError (see above for traceback): OOM when allocating tensor of shape [2048,50000] and type float
[[Node: Optimizer/Logits/fully_connected/weights/Adam/Initializer/zeros = Const[dtype=DT_FLOAT, value=Tensor<type: float shape: [2048,50000] values: [0 0 0]...>, _device="/job:localhost/replica:0/task:0/device:GPU:0"]()]]

We couldn't conduct training even for a single epoch due to the OOM (OutOfMemory) mistake, not saying about 6 or more epoch for the whole week after the daikon code has been changed on Monday.

Post-processing: We followed the steps as in the task 2. We reassembled BPE, then we undid Truecasing, then we detokenized and normalized the text. However, the final result is very surprising, and we didn't manage to solve that 'vocabulary-type' translation.