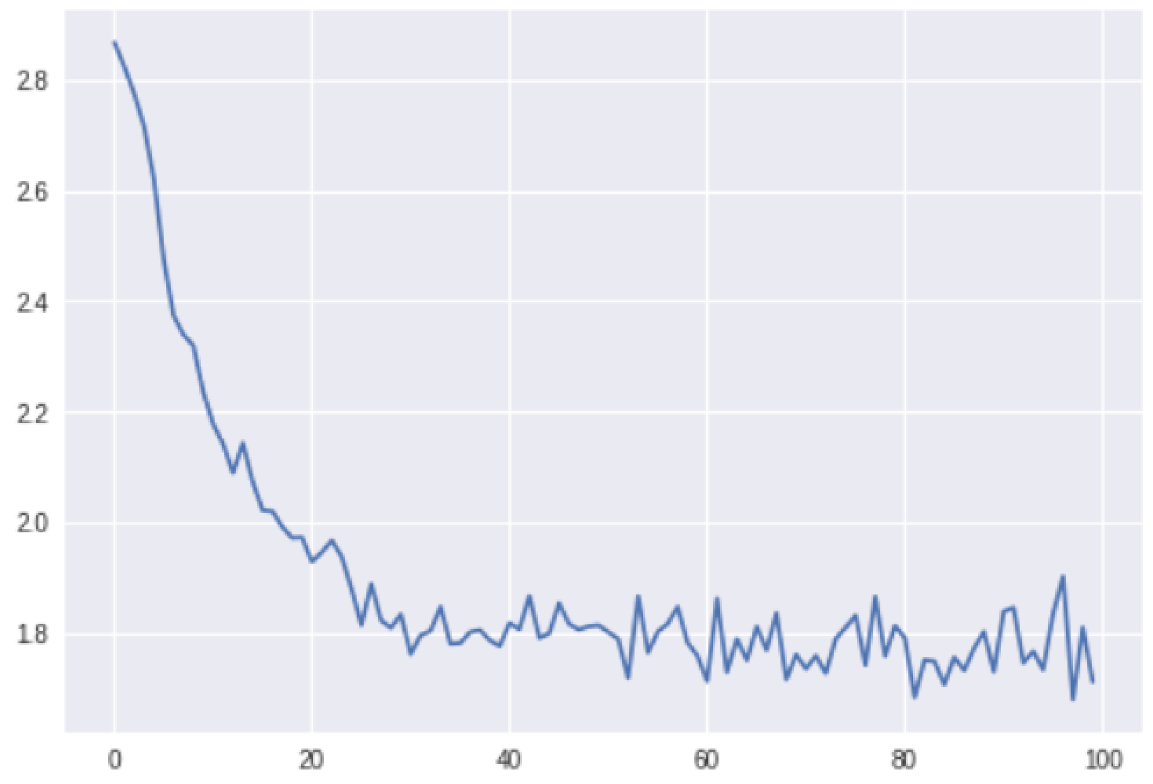
Note all graphs have the y-axis as the negative log likelihood loss and x-axis as the number of thousands of iterations.

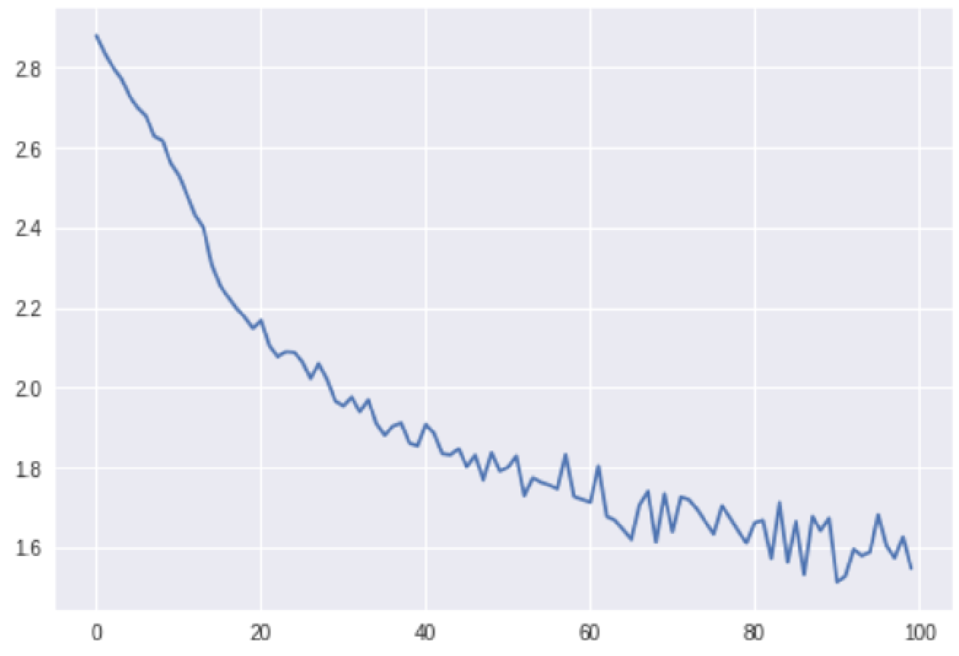
Linear

See char_rnn_class_tutorialNormal.py for code



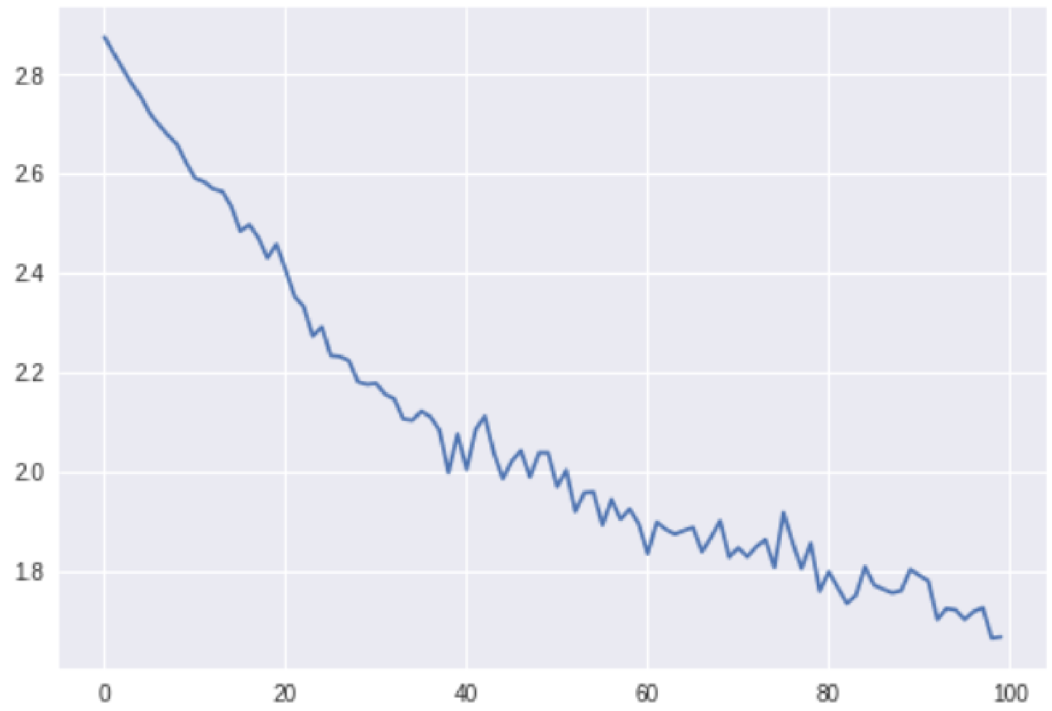
LSTM

See char_rnn_class_tutorialLSTM.py for code



GRU

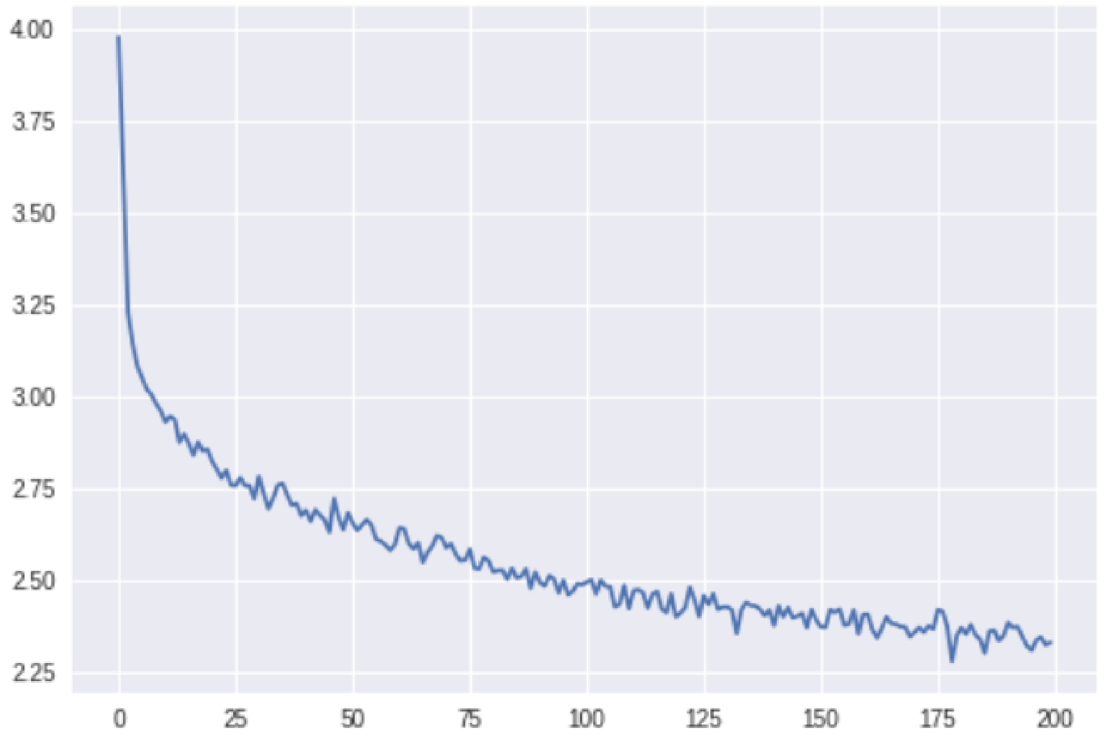
See char_rnn_class_tutorialGRU.py for code



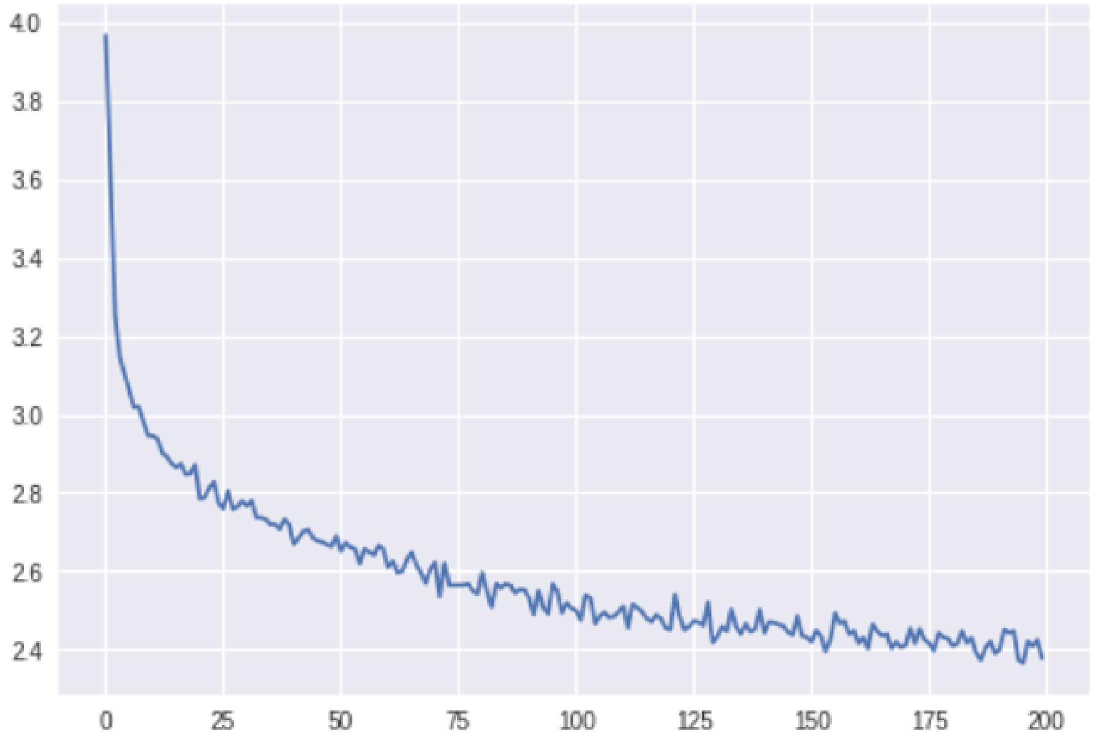
So the LSTM hidden units performed the best, then the GRU units, and finally linear units. Linear did the worst, since GRU and LSTM both utilize ways of gating information to prevent vanishing gradients. This leads to better results for GRU and LSTM. Due to the fact that LSTM has a memory unit and with the large amount of data used to train the model, LSTM is far more expressive than GRU and led to the better results shown above.

2b)

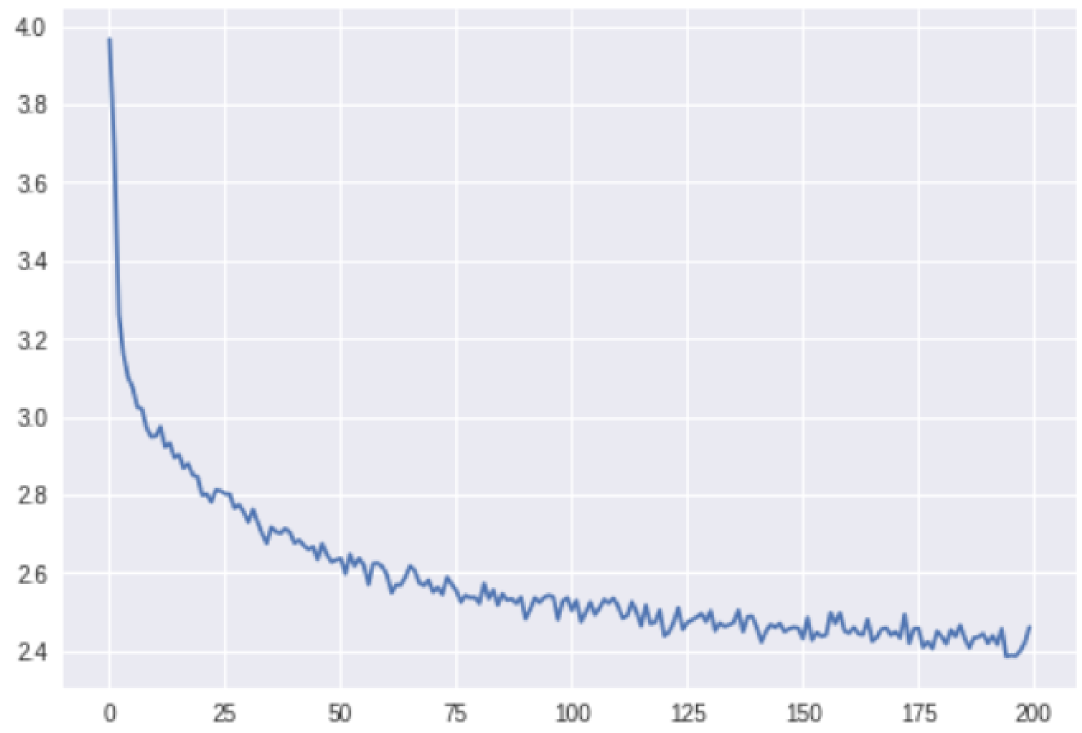
i) See char_rnn_generation_tutorial(i).py for code



ii) See char_rnn_generation_tutorial(ii).py for code

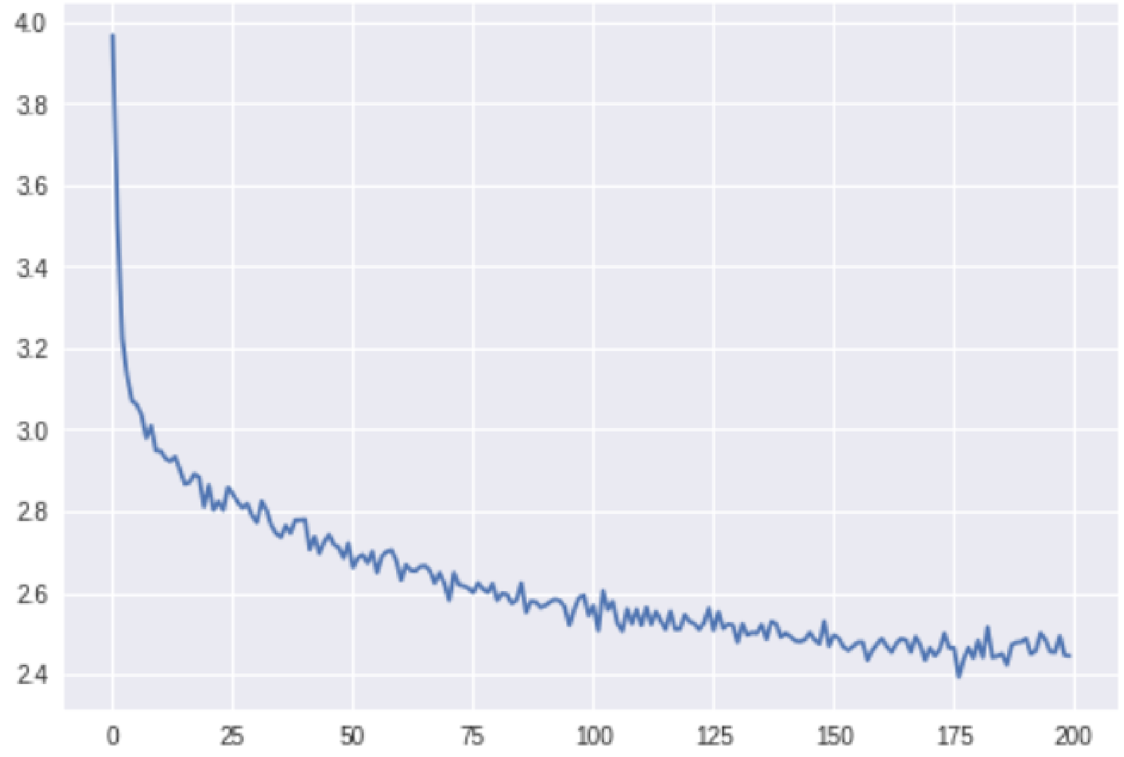


iii) `2.4613005558858063,` See char_rnn_generation_tutorial(iii).py for code



iv) See char_rnn_generation_tutorial(iv).py for code

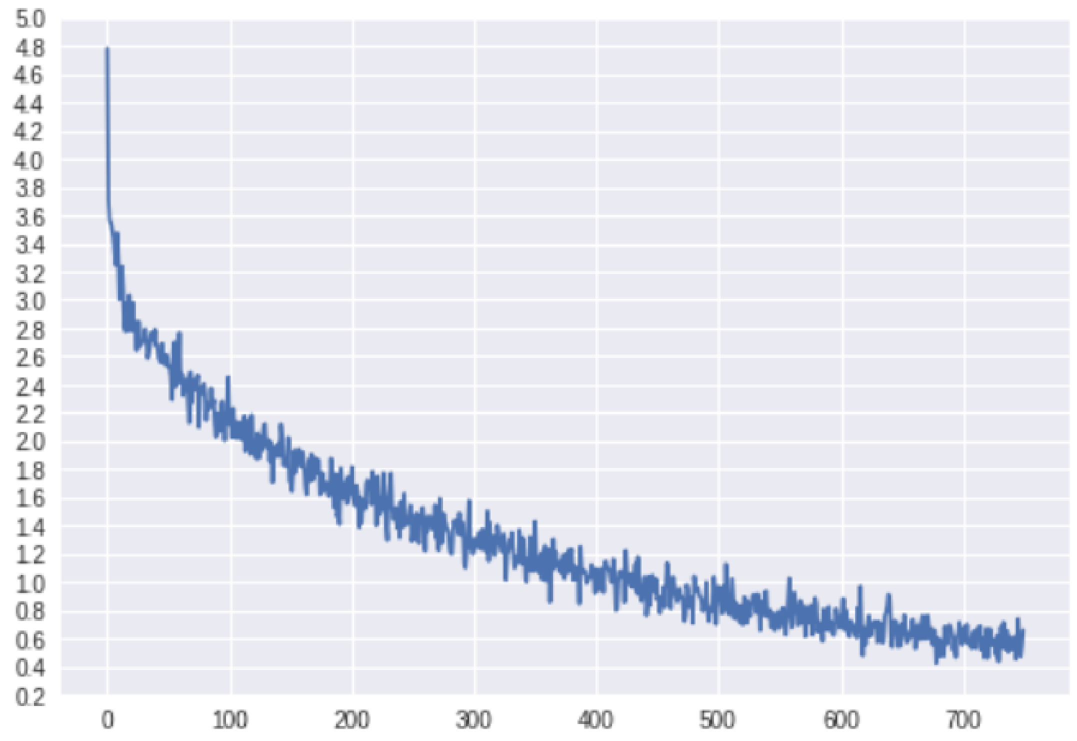
`2.476129837776916`



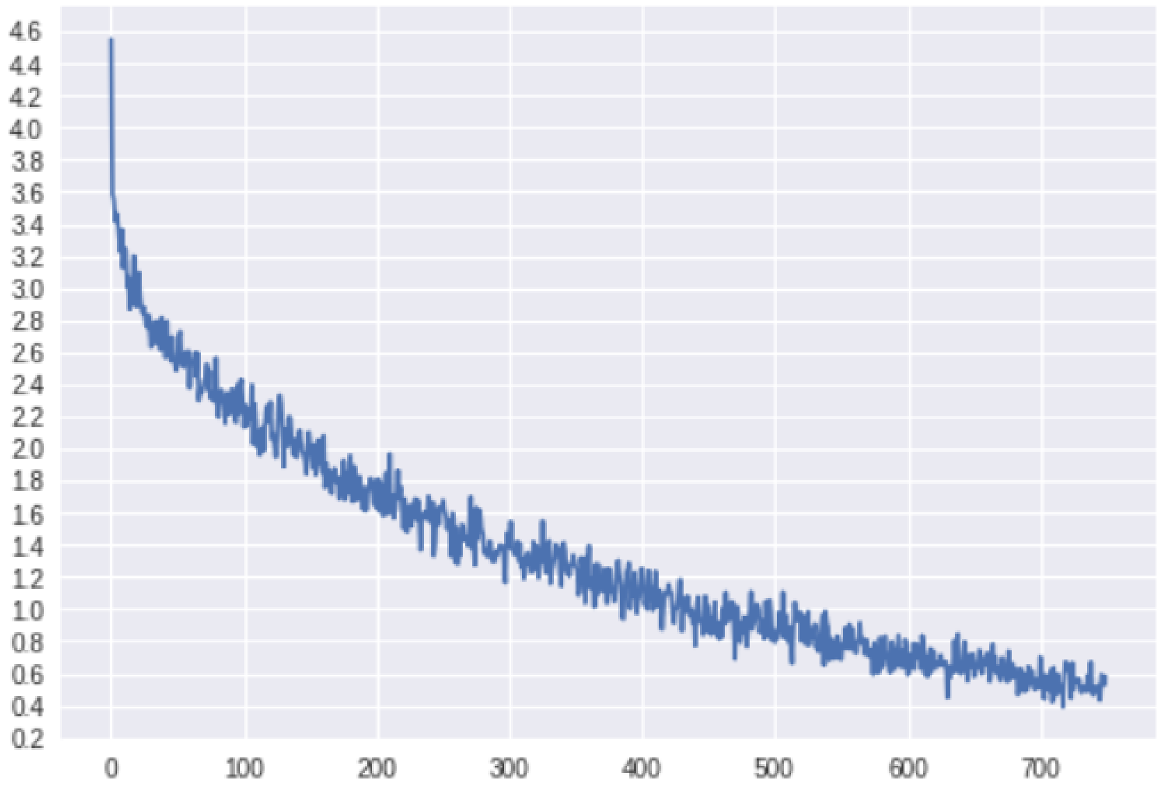
The best model was when the previous hidden unit, previous character, and category is passed to the input. This is fairly intuitive, since with more information passed to the RNN, the more context the model has to predict. That also means model (iv) would do the worst, since only the hidden network is passed as input. Then between models (ii) and (iii), model (ii) performed the best, by about 0.1 loss value. So, previous character information was more valuable than the category info. That is most likely because category information is still at least passed as the initial hidden state for (ii), whereas previous character information is never given for model (iii).

Question 3

Without attention, find code in seq2seq_translation_tutorialWithoutAttention.py. Last loss: 0.5820



With attention, find code at seq2seq_translation_tutorial.py. Last loss 0.5805079



The model with attention mechanisms outperformed the one without. This is intuitive, since attention solves a major problem of forcing input sequences to be encoded to a fixed-length internal vector. With a fixed-length internal vector, this could lead to information loss, since sentences are a variable length. Attention is computed with context of the sentence, allowing a more accurate translation.