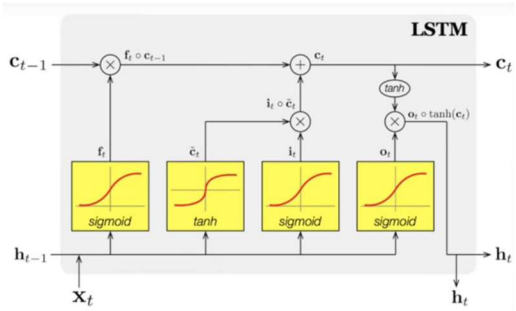
Deep Learning 097200 - HW #3 : Recurrent Networks

Alon Naor, ID 304818735

Daniel Engelsman, ID 300546173



- Input Data specifications (from *Penn Treebank tagset*) :
  - Training dataset : 929,580 [words]
  - Validation dataset : 73,760 [words]
  - Testing dataset : 82,420 [words]
  - Vocabulary size : 10,000 [words]

- Model architectures description :
  - **Total parameters –** 2,903,040
  - Embed size - 215
  - **LSTM** - 2 layers
  - Hidden size - 215

- Hyper Parameters :
  - Sequence length - 20
  - Batch size - 20
  - Learning rate - 20 → 5
  - Tie word embedding and softmax weights
  - Epochs - 20
  - Gradient clipping – 0.25
  - Dropout - 0.2

- Optimization details :
  - Criterion: CrossEntropyLoss
  - Optimizer: SGD / Gradient Descent

- Performance :

Last training results (epoch #20)	:	Train loss 4.12	Train ppl 61.37
Last validation results (epoch #20)	:	Valid loss 4.54	Valid ppl 94.08
Test results	:	Test loss 4.50	Test ppl 90.14 (< 120 V)

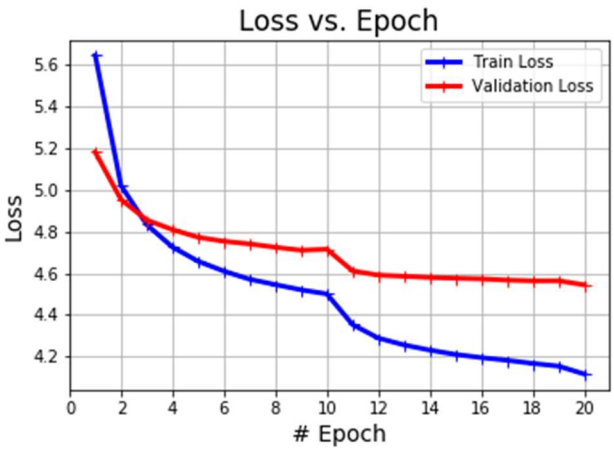
- Short summary of the working process :

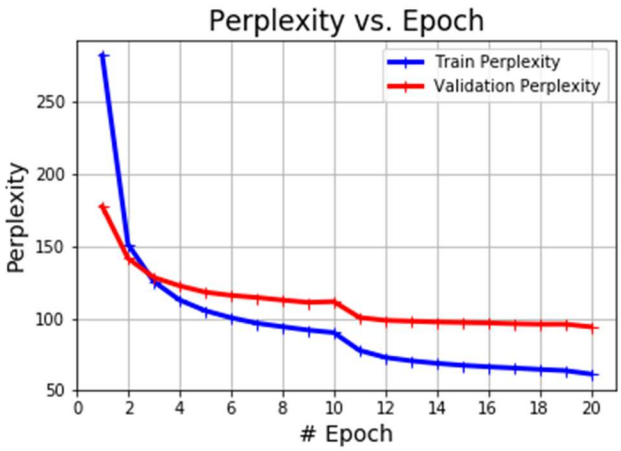
Different attempts have been sampled along the working process. At first, a one *layer* architecture was tried for its short-time epochs, that enabled large number of iterations in reasonable time. The obtained convergence was quick but final perplexity was not sufficient (> 120). After few changes in configurations (*hidden* size, *embedded* size, *dropout* and *sequence length*) it seemed like the "holy grail" has been found, and the above model was fixed and ready to go.

Additionaly, for the sake of variance, both of us have approached the problem independently such that the different solutions gave us a wider point of view. One model was using *Adam optimzer* while this one used *SGD*. The other model has shown an impressive initial *test perplexity* (~34.58%), but unfortunately instead of decreasing along time, it diverged and thus pointed on some malfunction.

- Graph presentation :

The below graphs exhibit the loss and perplexity vs. the corresponding epochs. As can be seen, the training starts at higher loss, but after two epochs improves its convergence rate. Note that both figures perform rather similar behavior as perplexity is expressed as : $f(loss) = e^{loss}$, such that the main is difference is just the graphs' amplitude.



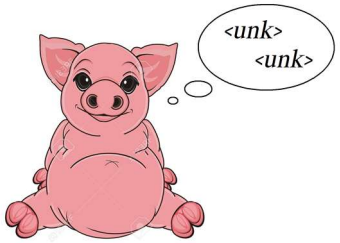


At every epoch's validation, our algorithm was conditioned to ensure that the average loss decreases along time. When performances seem to stagnate and no improvement shown, the algorithm commanded decreasing of the *learning rate* by 4.0 (20.0 → 5.0), which is seen as "bump" at epoch #10.

```
-----------------------------------------------------------------------------------------
| end of epoch   9 | time: 1018.37s | valid loss  4.71 | valid ppl   111.05
-----------------------------------------------------------------------------------------
train ntokens= 10000
| epoch  10 |   200/ 2323 batches | lr 20.00 | ms/batch 429.65 | loss  4.46 | ppl    86.60
| epoch  10 |   400/ 2323 batches | lr 20.00 | ms/batch 445.83 | loss  4.59 | ppl    98.98
| epoch  10 |   600/ 2323 batches | lr 20.00 | ms/batch 426.13 | loss  4.50 | ppl    89.86
| epoch  10 |   800/ 2323 batches | lr 20.00 | ms/batch 441.66 | loss  4.50 | ppl    90.32
| epoch  10 |  1000/ 2323 batches | lr 20.00 | ms/batch 418.74 | loss  4.52 | ppl    91.86
| epoch  10 |  1200/ 2323 batches | lr 20.00 | ms/batch 423.17 | loss  4.54 | ppl    93.94
| epoch  10 |  1400/ 2323 batches | lr 20.00 | ms/batch 454.35 | loss  4.44 | ppl    84.86
| epoch  10 |  1600/ 2323 batches | lr 20.00 | ms/batch 430.59 | loss  4.52 | ppl    92.03
| epoch  10 |  1800/ 2323 batches | lr 20.00 | ms/batch 450.54 | loss  4.59 | ppl    98.71
| epoch  10 |  2000/ 2323 batches | lr 20.00 | ms/batch 426.24 | loss  4.42 | ppl    83.07
| epoch  10 |  2200/ 2323 batches | lr 20.00 | ms/batch 450.79 | loss  4.38 | ppl    80.12
-----------------------------------------------------------------------------------------
| end of epoch  10 | time: 1064.39s | valid loss  4.71 | valid ppl   111.57
-----------------------------------------------------------------------------------------
train ntokens= 10000
| epoch  11 |   200/ 2323 batches | lr 5.00 | ms/batch 441.80 | loss  4.43 | ppl    84.21
| epoch  11 |   400/ 2323 batches | lr 5.00 | ms/batch 420.69 | loss  4.53 | ppl    92.68
| epoch  11 |   600/ 2323 batches | lr 5.00 | ms/batch 443.13 | loss  4.41 | ppl    82.35
| epoch  11 |   800/ 2323 batches | lr 5.00 | ms/batch 463.16 | loss  4.39 | ppl    80.80
| epoch  11 |  1000/ 2323 batches | lr 5.00 | ms/batch 450.78 | loss  4.40 | ppl    81.64
| epoch  11 |  1200/ 2323 batches | lr 5.00 | ms/batch 428.55 | loss  4.39 | ppl    81.01
| epoch  11 |  1400/ 2323 batches | lr 5.00 | ms/batch 443.47 | loss  4.28 | ppl    72.09
| epoch  11 |  1600/ 2323 batches | lr 5.00 | ms/batch 591.07 | loss  4.35 | ppl    77.30
| epoch  11 |  1800/ 2323 batches | lr 5.00 | ms/batch 490.49 | loss  4.40 | ppl    81.05
| epoch  11 |  2000/ 2323 batches | lr 5.00 | ms/batch 458.77 | loss  4.21 | ppl    67.04
| epoch  11 |  2200/ 2323 batches | lr 5.00 | ms/batch 448.21 | loss  4.14 | ppl    62.91
```

- Inference :

Finally we were requested to test the model's performance de-facto, namely, predicting and the most logical combination w.r.t to the initial input : *"Buy low, sell high is the …"*



At this stage we must admit that it seemed like a little child learns to speak. Many of the outputs were found rather ridiculously amusing. Hereby are three outputs with a varying temperature (short explained) :

>> Temperature=0.6 (Low) : Super conservative, mainly a template without real content:

"buy low sell high is the <unk> of the <unk> and the <unk> of the official <eos> the <unk> <unk> of the <unk> <unk> resort has been <unk> from the <unk> and <unk> <eos> the <unk>"

>> Temperture 1.0 (Medium-Low) - more conservative but logical with some context:

"Buy low sell high is the value of the stock market into the public and lagging the firm to keep out new cash <eos>
whoever notes that the investor has a mind of <unk> <unk> two"

>> Temperature 5.0 (High) – longer sentences without sense and logical context between words :

"Buy low sell high is the ind. option drain expenditures pfizer undertaken answer fastest talked deferring conflicts examined largest theaters hispanics feelings calculate later eagle predictions laws greater eurodollars disease noted anc american seats lower enthusiastic"

- But what is this *temperature* thing ??

It is a hyperparameter of the LSTM that's used to control the *randomness* of predictions by scaling the logits before applying *softmax* $\left(\frac{logit}{temp.}\right)$ When (temp. = 1.0), we simply compute the softmax directly on the logits. But when calculating (temp. = 0.6) the model computes a larger softmax result. Growing probabilistic values improve the model's confidence since less input is needed. But the output's variance is more conservative, ending up in poor output sentences. Contrarily, higher values (temp. = 5.0) enhanced the prediction's diversity but context level decreased dramatically, exhibiting a random selection of words from the vocabulary.

- Suggest a different way of performing the generating process :

(i) Raffling temperature value out of a predefined subset of values. Then sampling the multinomial distribution of the words' weights and scaling them.

(ii) Sampling various temperature values every epoch and make a "sense test" between them.

(iii) Picking the maximal word's weight instead of the classic multinomial distribution.