**Deep learning HW3**

Submitted by: Liron McLey 200307791, Matan Hamra 300280310 & Ariel Weigler 300564267

## Model Architecture

Five layers network built according to the following stricture:

1. Encoder – (10,000 🡪 220)
2. LSTM(220🡪 220)
3. Dropout(0.3)
4. LSTM(220🡪220)
5. Decoder- Linear( 4\*220🡪 10,000)

Number of parameters: 2,987,920

The model based on tutorial 8 with two LSTM layers and dropout layer in between them. The vocabulary size and the hidden size chose to be the same size

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model parameters** | | | | | | | | | |
| Hidden layer size | Vocabulary size | # layers | # epochs | # samples | Batch size | Sequence  Length | dropout | Learning rate |
| **220** | **220** | **2** | **10** | **30** | **20** | **30** | **0.3** | **0.005** |

Loss criterion: CrossEntropyLoss.

Optimizer: RMSprop

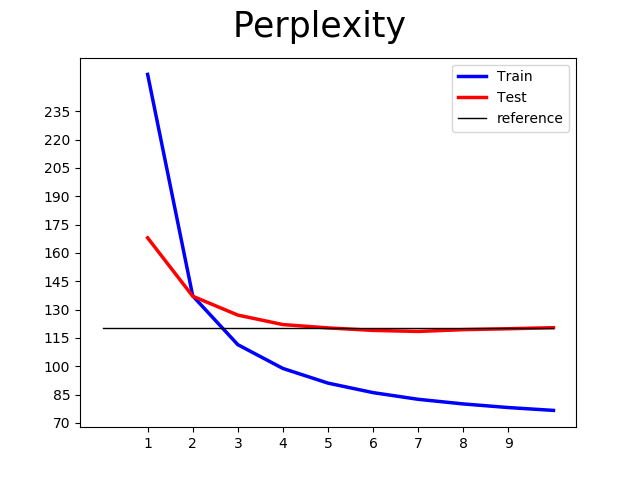
## Training procedure

1. Input the entire training set into batches of 30, were each batch:
   1. Forward propagation into the network
   2. Loss and perplexity calculation according to the selected loss criterion.
   3. Backward propagation through the network to calculate gradient weights using RMSprop as our optimizer.
2. Calculate loss and perplexity values for network predictions on validations and test sets respectively.

## Results

Are results are as follows, where the x axis is the epoch number:





The horizontal black line in the perplexity graph represents a value of 120. Our best training epoch was epoch number 7, where **the perplexity reached 118.43.**

## Discussion & Summary

At first we tried a number of hyper parameter combinations with our network model to get us to a reasonable result, but it required to many trainable parameters. To reach the desired result of test set perplexity < 120 at the 3M parameter limit we decided to use the inverse of the embedding layer as the final linear decoding layer. This reduced the number of trainable parameters considerably without compromising on performance. We then used the maximum size of embedded/hidden vector possible to get us as close to the allowed 3M parameters as possible.

Additionaly, we …

## **Inference**

In this part of the exercise we used the network to generate N-word sentence continuations to a given sentence beginning. To do that, we ran each word through the model to calibrate the state of the system to the given sentence beginning. From there on, we used the output given by the model to generate additional words for our paragraph: we divided the output by a variable called temperature and raise the result by an exponent. This was then considered as the probability to determine the next word in the sentence until the code reached the predefined word limit.

The sentence we fed our network was: ""Buy low, sell high is the…", and we sampled 30-word sentences for 3 different temperature values:

Temperature = 1 output:

buy low sell high is the <unk> of the effects of directors <eos> in contrast to the cyclical noble system is the first from the west <eos> washington-based genetic apple <eos> it has been more neither

Temperature = 3 output:

buy low sell high is the workstations questions forecasting white occurred than at impact wants whitten walters raised pressed va. decline machinery and wertheim cost-cutting clean cycles cleanup happens wears microsoft commercial arrow coverage page until

Temperature = 20 output:

buy low sell high is the mega-issues when-issued valid based en insisted resource area suspend wildly miles drain accessories prestigious talked resulting pipelines restated streamlining speeds realist twist professors associations seniority side pc idle across blockbuster

One can see the output for temperature = 1 is more strict, the chance of new sentence beginnings is higher and the structure sounds more like what one would expect from simple sentences, with relatively reasonable sentence structure. Temperture = 3 is much less reasonable. We got one long sentence without much sense. However, the low level noun + verb structure repeats quite often with the noun and verb having some correlation to one another. Temperature = 20 is again one long sentence, yet this time the words seems to have been drawn quite randomly. Hence it seems the function of temperature is to allow for a richer sentence by manipulating the probability which the model outputs. The higher the temperature, the flatter the probability distribution over the dictionary, thus the pool of probable words is larger.

Another way to perform a generating process: