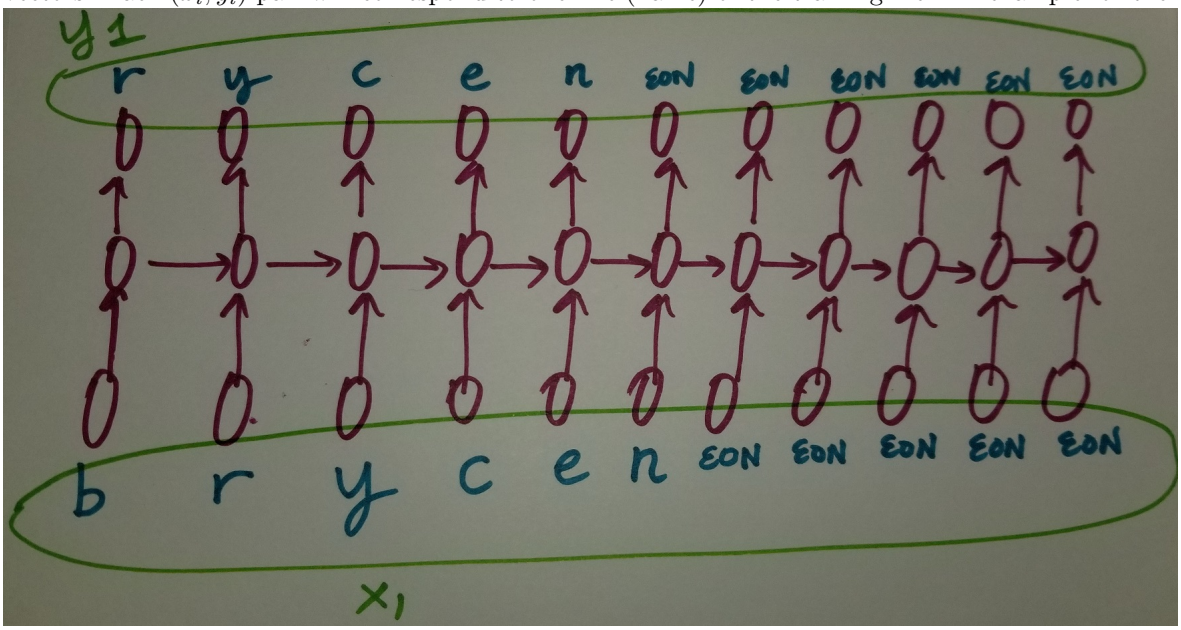
EEE 443/543 Neural Networks, Spring 2025 - Project #7
Due: 05/05/2025, 11pm.

Erdem Koyuncu

The encoding process should be exactly as described in the assignment.

Q1: We will implement a character-level text generation LSTM. Specifically, the goal will be to generate person names. Download names.txt from the Piazza website. Each line is a name of at most 11 characters. There are 2000 lines of names.

1. Preprocess the file so that it is all lower-case letters. Note that there are 26 letters in the English alphabet. We need another letter to indicate End Of Name (EON), making a total of 27 characters that we need to encode. Use the one hot encoding EON$\rightarrow [1\,0\,0\,\cdots\,0] \in \mathbb{R}^{27}$, a$\rightarrow [0\,1\,0\,\cdots\,0]$, b$\rightarrow [0\,0\,1\,\cdots\,0]$, ..., z$\rightarrow [0\,0\,0\,\cdots\,1]$. Generate the training sequence (x_i, y_i), $i = 1, \ldots, 2000$, where each x_i is a length-11 sequence of 27 dimensional vectors to the LSTM, and the y_i is the corresponding desired output, which is again a length-11 sequence of 27 dimensional vectors. Each (x_i, y_i) pair will correspond to one line (name) of the training file. An example for the first line:



   Therefore, we artificially append EONs so that all training elements are the same length. Many neural network libraries do not support unrolling a recurrent neural network a variable number of times during training. Save your training program as train.ext (with the appropriate extension).

2. Train the neural network, name your training code as **0701-IDNumber-LastName.py**. Save the trained model as **0702-IDNumber-LastName.ZZZ**. There are different ways to go here, but I will let you overfit for the sake of simplicity. **Plot the loss value versus epochs and include this graph in your report. Your graph should show the loss value converging. Indicate the mathematical expression for the loss function you have used. As in the previous assignments, you should make sense of the loss values you obtained with respect to the epochs. In particular, discuss the reason why you converged to the particular loss value you got (if it is zero why, if it is non-zero why?).**

3. Now comes the fun part of actual generation of names. Feed an initial letter, say "r," to the LSTM (Of course, you will feed the one-hot encoded version of this). The LSTM will produce a 27-dimensional vector. The most likely next letter will correspond the maximum component. For example, if the LSTM output is $[0.1, 0.9, 0.2, -0.1, \ldots]$, and suppose 0.9 is the maximum component, then we decide that the next letter should

be an "a." Then, feed the sequence "ra" is fed to the LSTM, and get (for example) "x" as the next letter. After some time, you should get "EON" as the output, meaning that the name generation process is complete and you generated "rax" as the name. The problem with this approach is that you can only generate one word per initialized letter, i.e if you start with the letter "r" you will always end up with "rax." That is why, you should soften you algorithm such that you decide on the second "best" letter with a certain probability, third better letter with a certain probability, and so on. I will leave this design choice up to you. Experiment with different choices to see what options give you the most realistic generations. **Include your design choice in your report.**

Your final code should accept an initial letter that I will be able to provide as an input, and generate 20 names starting with that letter. Save this inference part as **0703-IDNumber-LastName.py** (again with the appropriate extension). **Include 20 names starting with the letter "a" and 20 names starting with the letter "x" in your final report generated by your network.** These were the names my network generated: With the letter "a:" adal, andira, andy, analea, arie, andi, ade, anale, anabash, aden, annelis, alac, aleahnd, adree, ariyanastheldal, ary, ariy, aryanaatth, adana, ariy. With the letter "x:": xzoretyn, xalleetto, xznvera, xisiana, xavierheri, xzacey, xzov, xalies, xaldan, xamerya, xive, xinesanyrhhi, xin, xamiralaa, xalve, xzevioeynnao, xev, xzemaeynerothio, xalleero, ximma. I wish I were named "Xzemaeynerothio!" Note that you do not have to constrain your generator to 11 letters like my examples show.

4. Upload **0701-IDNumber-LastName.py**, **0702-IDNumber-LastName.ZZZ**, and **0703-IDNumber-LastName.py** to the codes folder.

5. **Put your report to the reports folder.**