Project 4 - RNN and LSTM

*

Bharath Renjith
Department of ECE
Raleigh, NC
bchenna@ncsu.edu

Shashank Shekhar
Department of CS
Raleigh, NC
sshekha4@ncsu.edu

Amit Cudykier
Department of BME
Raleigh, NC
acudyki@ncsu.edu

***Abstract*—In this report,we discuss the implementation of Recurrent Neural Network (RNN),specifically Long-Short Term Memory Units (LSTM) for counting digits in a sequence and language modelling .**

***Index Terms*—RNN ,LSTM, PTB, Keras**

I. INTRODUCTION

The goal of this project is to understand and get familiarized with a wildly used RNN unit, Long-term Short Memory (LSTM). This project is divided into two parts. In the first part, we study the behaviour of one LSTM unit for the use case of counting digits in a sequence. For the second part , we utilize our understanding of LSTM on a real world scenario, namely language modelling. The following of the report is structured as follows. Section II explains our study of LSTM unit for counting sequences. Section III briefs our implementation of LSTM for text prediction on PTB Dataset.

II. PART - 1

As mentioned before, we consider one LSTM unit and analyze the behaviour of different gates and internal states of the unit for different scenarios of counting digit 0 in a sequence. We use a slightly modified LSTM structure for the analysis purpose. The LSTM structure is as follows:

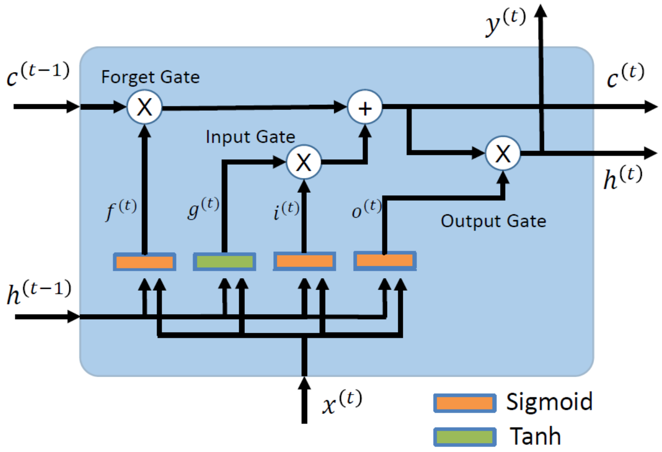

Fig. 1. Slightly Modified LSTM unit

where,

$$
\begin{aligned}
g^{(t)} &= tanh(W_{xg}^T.x^{(t)} + W_{hg}^T.h^{(t-1)} + b_g)\\
i^{(t)} &= \sigma(W_{xi}^T.x^{(t)} + W_{hi}^T.h^{(t-1)} + b_i)\\
f^{(t)} &= \sigma(W_{xf}^T.x^{(t)} + W_{hf}^T.h^{(t-1)} + b_f)\\
o^{(t)} &= \sigma(W_{xo}^T.x^{(t)} + W_{ho}^T.h^{(t-1)} + b_o)\\
c^{(t)} &= f^{(t)}\bigotimes c^{(t-1)} + i^{(t)}\bigotimes g^{(t)}\\
y^{(t)} &= h^{(t)} = o^{(t)}\bigotimes c^{(t)}
\end{aligned}
$$

We then manually tune the weights and biases of each gate to to make the LSTM unit count the digit 0 for given conditions.

A. Parameters

We have to increase the output dimension of internal state or output to 2 so as to include the extra conditions beyond counting zeros.

1) Task2: We update the input gate values to update a flag when first 2 has occurred as well as to clear g(t) until that happens.

$$
\begin{aligned}
W_{xg}[0] &= [100.\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\\
W_{hg}[0] &= [0\ 0]\\
b_g[0] &= 0\\
W_{xg}[1] &= [0\ 0\ 100.\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\\
W_{hg}[1] &= [0\ 0]\\
b_g[1] &= 0
\end{aligned}
$$

$$
\begin{aligned}
W_{xi}[0] &= [0.\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\\
W_{hi}[0] &= [0\ 200]\\
b_i[0] &= 100\\
W_{xi}[1] &= [0\ 0\ 100.\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\\
W_{hi}[1] &= [0\ 0]\\
b_i[1] &= -100
\end{aligned}
$$

$$\begin{aligned} W_{xf}[0] &= [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{hf}[0] &= [0\ 0] \\ b_f[0] &= 100 \\ W_{xf}[1] &= [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{hf}[1] &= [0\ 0] \\ b_f[1] &= 100 \end{aligned}$$

$$\begin{aligned} W_{xo}[0] &= [0.\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{ho}[0] &= [0\ 200] \\ b_o[0] &= 100 \\ W_{xo}[1] &= [0\ 0\ 100.\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{ho}[1] &= [0\ 0] \\ b_o[1] &= -100 \end{aligned}$$

2) Task3: We modify the forget gate values from task2 to incorporate the forget condition.

$$\begin{aligned} W_{xg}[0] &= [100.\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{hg}[0] &= [0\ 0] \\ b_g[0] &= 0 \\ W_{xg}[1] &= [0\ 0\ 100.\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{hg}[1] &= [0\ 0] \\ b_g[1] &= 0 \end{aligned}$$

$$\begin{aligned} W_{xi}[0] &= [0.\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{hi}[0] &= [0\ 200] \\ b_i[0] &= 100 \\ W_{xi}[1] &= [0\ 0\ 100.\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{hi}[1] &= [0\ 0] \\ b_i[1] &= -100 \end{aligned}$$

$$\begin{aligned} W_{xf}[0] &= [0\ 0\ 0\ \ -200\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{hf}[0] &= [0\ 0] \\ b_f[0] &= 100 \\ W_{xf}[1] &= [0\ 0\ 0\ \ -200\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{hf}[1] &= [0\ 0] \\ b_f[1] &= 100 \end{aligned}$$

$$\begin{aligned} W_{xo}[0] &= [0.\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{ho}[0] &= [0\ 200] \\ b_o[0] &= 100 \\ W_{xo}[1] &= [0\ 0\ 100.\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ W_{ho}[1] &= [0\ 0] \\ b_o[1] &= -100 \end{aligned}$$

3) Plots: With the above values for weights and biases we were able to count digit 0 in a sequences under given conditions. We plotted the internal states and gate values at each instance of sequence to analyze how the LSTM works.The internal state has two dimensions, one that keeps the count of 0 and other a flag for first occurance of 2, (which is nothing but a count of 2 in our implementation).So we plot the LSTM state for each dimension separately.

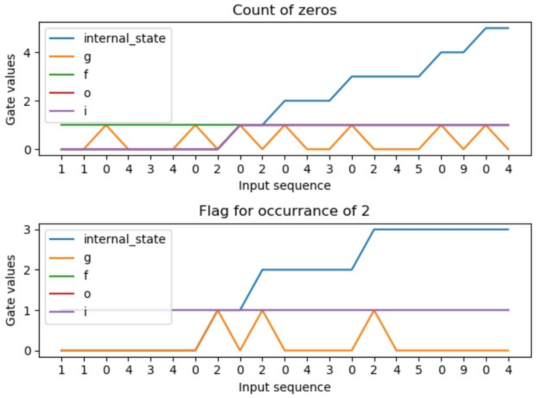

Fig. 2. Plot of LSTM structure states at each instance of sequence for Task 2. We can observe LSTM refraining from counting zeros until a 2 has occured

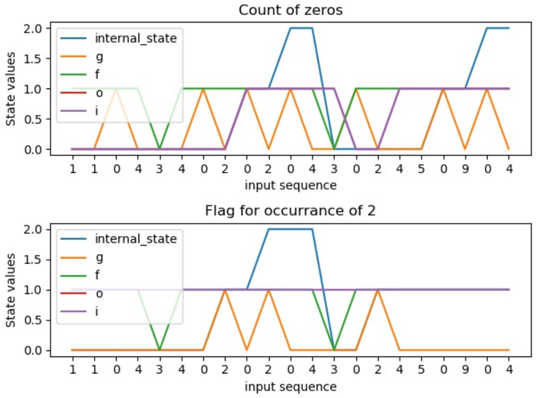

Fig. 3. Plot of LSTM structure states at each instance of sequence for Task 3. We can observe that whenever a 3 comes both the dimensions of internal states are reset to 0

III. PART II

The goal of this part of the project is language modeling using RNN with LSTM units. The model is trained on PTB dataset. We perform word and character level training. The raw data from the file is tokenized and passed to the model. The trained model takes a sequence of words to generate text predictions.

We tuned the word and character model separately for the following hyperparameter values:

Hyperparameter	Word Model	Character Model
Epochs	10	10
Dropout Rate	0.2, 0.4, 0.6, 0.8	0.2, 0.4, 0.6, 0.8
LSTM Layers	10, 15, 25	10, 15, 25
Activation	Softmax	Softmax
Learning Rate	0.001	0.001
Dimensions	25, 50, 100	25, 50, 100

In the optimization process we plotted the number of epochs vs perplexity for the training and validation set. This helped us to assess how well our model is preforming.

Below is the plot obtained for the word model trained with 100 dimensions, 15 lstm layers and 0.6 dropout rate.

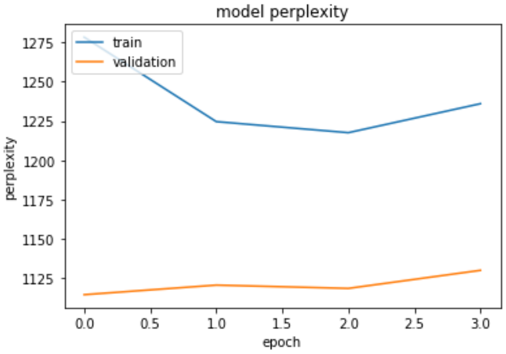

Fig. 4. Word-level Plot

Below is the plot obtained for the character model trained with 50 dimensions, 25 lstm layers and 0.4 dropout rate.

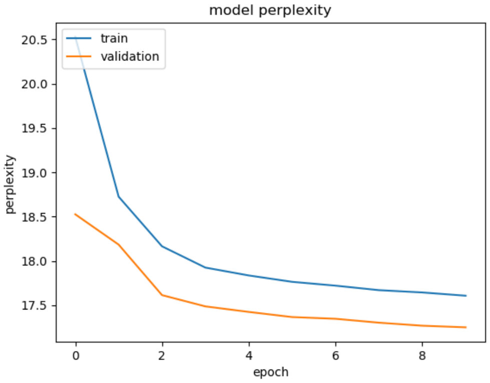

Fig. 5. Character level Plot

To test the final output we generate a 100 words sequence using out models:

Word level model output: these stocks unk

Character level model output:
p o r t s _ o f _ c a l l _ i n c . _ r e a c h e n