Question 1:

Show that the simple RNN (as defined by PyTorch) with sigmoid nonlinearity is a subset of the LSTM (as defined by PyTorch).
Show how to set the weights of the LSTM to mimic the simple RNN. They will be functions of the RNN weight matrices.

Answer:

To show that the simple RNN is a subset of the LSTM, we need to show that the LSTM can be reduced to behave like a simple RNN.

The simple RNN (with sigmoid avtivation) is defined by the following equations:

$$h_t = sigmoid(W_h * h_{t-1} + W_x * x_t + b_h + b_x)$$

where h_t is the hidden state at time t, h_{t-1} is the hidden state at time t-1, x_t is the input at time t, W_h and W_x are the weight matrices for the hidden state and input, and b_h and b_x are the biases for the hidden state and input.

The LSTM is defined by the following equations:

input_gate: $i_t = sigmoid(W_i i * x_t + W_h i * h_{t-1} + b_i i + b_h i)$

forget_gate: $f_t = sigmoid(W_i f * x_t + W_h f * h_{t-1} + b_i f + b_h f)$

cell_input: $g_t = tanh(W_i g * x_t + W_h g * h_{t-1} + b_i g + b_h g)$

cell_state: $c_t = f_t * c_{t-1} + i_t * g_t$

output_gate: $o_t = sigmoid(W_i o * x_t + W_h o * h_{t-1} + b_i o + b_h o)$

hidden_state: $h_t = o_t * tanh(c_t)$

To reduce the LSTM to behave like a simple RNN, we need to make the following modifications:
Set the forget_gate to Zero:

$$W_i f = 0, W_h f = 0, b_i f = -float('inf')$$

We want the cell state c_t not to carry over any information from the previous time step, so we set the forget gate to zero. This means that the cell state c_t will be reset to zero at each time step.

Set the input_gate to One:

$$W_i i = 0, W_h i = 0, b_i i = float('inf')$$

To allow the new input to pass through completely, set the input gate biases to a large positive value to make the input gate always open (i.e., equal to one).

Set the output_gate to One:

$$W_i o = 0, W_h o = 0, b_i o = float('inf')$$

Ensure the entire cell state's activation affects the hidden state by setting the output gate biases to a large positive value.

Adjust the cell_input to match the Simple RNN activation:

$$W_i g = W_x, W_h g = W_h, b_i g = b_x, b_h g = b_h$$

The cell state becomes:

$$c_t = i_t * g_t = g_t$$

To match the simple RNN activation, set the cell input weights and biases to the simple RNN weights and biases.

Adjust the activation functions:
Since the RNN uses a sigmoid activation and the LSTM uses a tanh activation for the cell state, relate them using the identity:

$$\begin{gathered} sigmoid(x) = 1/(1 + exp(-x)) \\ = (exp(x) - exp(-x))/(exp(x) + exp(-x)) \\ = (exp(2x) - 1)/(exp(2x) + 1) \end{gathered}$$

$$= (tanh(x/2) + 1)/2$$

Thus, the LSTM's output becomes a scaled version of the RNN's output:

$$\begin{gathered} h_t(LSTM) = tanh(c_t) = 2 * sigmoid(c_t) - 1 \\ = 2 * sigmoid(g_t) - 1 \\ = 2 * sigmoid(W_ig * x_t + W_hg * h_{t-1} + b_ig + b_hg) - 1 \\ = 2 * sigmoid(W_x * x_t + W_h * h_{t-1} + b_x + b_h) - 1 \\ = 2 * sigmoid(W_x * x_t + W_h * h_{t-1} + b_x + b_h) - 1 \\ = 2 * h_t(RNN) - 1 \end{gathered}$$

With these modifications, the LSTM can be reduced to behave like a simple RNN with sigmoid activation.

Question 2:

Documents contain the words "bad", "good", "not", and "uh". A sentiment score is computed for each document as follows:

$$\sum \text{"good"} + \sum \text{"bad"} - 2 \times \sum \text{"not good"} - 2 \times \sum \text{"not bad"}$$

If the inputs to an LSTM are one-hot-encoded words (using alphabetical order), and the output at each time is the cumulative sentiment score, manually identify specific LSTM sizes and weights that solve this problem.

Answer:

Vocabulary: ["bad", "good", "not", "uh"]

Encoding:

1. "bad": $[1, 0, 0, 0]^T$
2. "good": $[0, 1, 0, 0]^T$
3. "not": $[0, 0, 1, 0]^T$
4. "uh": $[0, 0, 0, 1]^T$

LSTM Architecture

- Input size: 4 (one-hot encoding of the words)
- Hidden state size (h_t) and cell state size (c_t): 2 units
    - $c_t[0]$: Cumulative sentiment score
    - $c_t[1]$: "Not" flag (1 if the previous word was "not", else 0)
- Output at each time step (y_t): The cumulative sentiment score up to time t

LSTM Equations

- Fixing all gates (input gate, forget gate, output gate) to 1.
- Using linear activations (identity function).

1. Cell candidate: $\tilde{c}_t = W_c x_t + U_c h_{t-1} + b_c$
2. Cell state update: $c_t = c_{t-1} + \tilde{c}_t$
3. Hidden state: $h_t = c_t$
4. Output: $y_t = c_t[0]$

Weights and Biases

We need to define W_c, U_c, and b_c such that the LSTM correctly updates both $c_t[0]$ (the cumulative sentiment score) and $c_t[1]$ (the "not" flag).

Weights for Sentiment Score ($c_t[0]$):

Our goal is to update the cumulative sentiment score based on the presence of "good", "bad", and preceding "not".

- When "good" or "bad" is input:
    - If the "not" flag is 0, increase sentiment score by +1.
    - If the "not" flag is 1: decrease sentiment score by -2.
- For other words, no change in sentiment score.
- $W_c[0,:] = [1, 1, 0, 0]$
    - Adds +1 when "good" or "bad" is input.
- $U_c[0,:] = [0, -3]$
    - Adjusts the sentiment score based on the "not" flag ($-3 \times c_{t-1}[1]$).
- $b_c[0] = 0$:

Weights for "Not" Flag ($c_t[1]$)

Our goal is to manage the "not" flag based on the presence of "not", "good", and "bad".

- Set the "not" flag to 1 when "not" is input.
- Reset the "not" flag to 0 after processing "good" or "bad".
- Maintain the "not" flag for other words.
- $W_c[1,:] = [0, 0, 1, 0]$; Sets the "not" flag to +1 when "not" is input.
- $U_c[1,:] = [0, -1]$; Resets the "not" flag ($-1 \times c_{t-1}[1]$) when "good" or "bad" is input.
- $b_c[1] = 0$

Thus, the Final Weight Matrices and Biases:

- $W_c = [[1, 1, 0, 0],$
  $[0, 0, 1, 0]]$
- $U_c = [[0, -3],$
  $[0, -1]]$
- $b_c = [[0],$
  $[0]]$