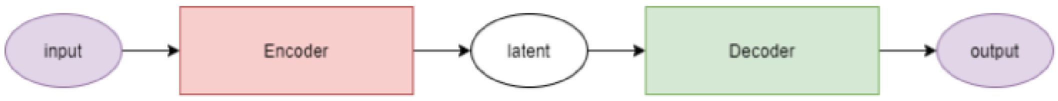
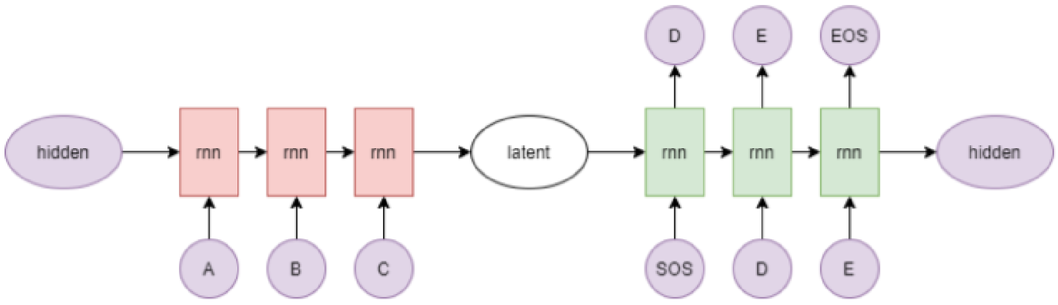
Lab 3: Sequence-to-sequence Recurrent Network

309833023 簡廷羽

1. Intro

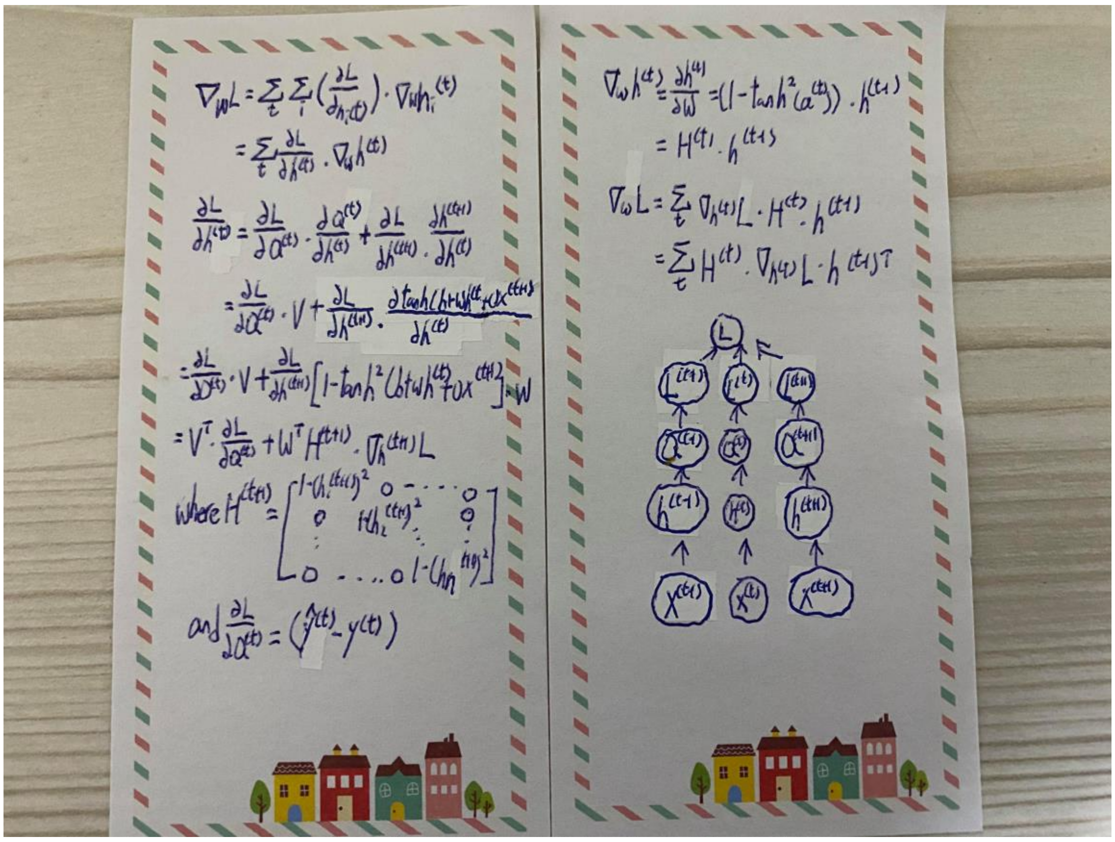
   Our purpose is to implement a sequence to sequence network in order to correct the spelling of English vocabularies.





   Above is the diagram of how encoder-decoder architecture works. Each input is spitted to character and then being embedded into vectors. After lstm prediction, decoder will decode the vectors then output the predicted word.

2. Derivation of BPTT



$$\nabla_W L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \cdot \nabla_W h_i^{(t)}$$
$$= \sum_t \frac{\partial L}{\partial h^{(t)}} \cdot \nabla_W h^{(t)}$$

$$\frac{\partial L}{\partial h^{(t)}} = \frac{\partial L}{\partial a^{(t)}} \cdot \frac{\partial a^{(t)}}{\partial h^{(t)}} + \frac{\partial L}{\partial h^{(t+1)}} \cdot \frac{\partial h^{(t+1)}}{\partial h^{(t)}}$$
$$= \frac{\partial L}{\partial a^{(t)}} \cdot V + \frac{\partial L}{\partial h^{(t+1)}} \cdot \frac{\partial \tanh(b+Wh^{(t)}+Ux^{(t+1)})}{\partial h^{(t)}}$$
$$= \frac{\partial L}{\partial a^{(t)}} \cdot V + \frac{\partial L}{\partial h^{(t+1)}} \left[1-\tanh^2(b+Wh^{(t)}+Ux^{(t+1)})\right] \cdot W$$
$$= V^T \cdot \frac{\partial L}{\partial a^{(t)}} + W^T H^{(t+1)} \cdot \nabla_{h^{(t+1)}} L$$

where $H^{(t+1)} = \begin{bmatrix} 1-(h_1^{(t+1)})^2 & 0 & \cdots & 0 \\ 0 & 1-(h_2^{(t+1)})^2 & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1-(h_n^{(t+1)})^2 \end{bmatrix}$

and $\frac{\partial L}{\partial a^{(t)}} = (\hat{y}^{(t)} - y^{(t)})$

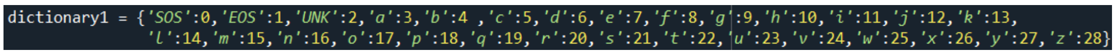
$$\nabla_W h^{(t)} = \frac{\partial h^{(t)}}{\partial W} = (1-\tanh^2(a^{(t)})) \cdot h^{(t-1)}$$
$$= H^{(t)} \cdot h^{(t-1)}$$

$$\nabla_W L = \sum_t \nabla_{h^{(t)}} L \cdot H^{(t)} \cdot h^{(t-1)}$$
$$= \sum_t H^{(t)} \cdot \nabla_{h^{(t)}} L \cdot h^{(t-1)T}$$

$$\begin{tikzcd}
 & L & \\
L^{(t-1)} \arrow[ur] & L^{(t)} \arrow[u] & L^{(t+1)} \arrow[ul] \\
a^{(t-1)} \arrow[u] & a^{(t)} \arrow[u] & a^{(t+1)} \arrow[u] \\
h^{(t-1)} & h^{(t)} & h^{(t+1)} \\
x^{(t-1)} \arrow[u] & x^{(t)} \arrow[u] & x^{(t+1)} \arrow[u]
\end{tikzcd}$$

3. Implementation details.
   A. Describe how you encode or embed words into vectors

      First I built a dictionary, creating each characters into consistence words.

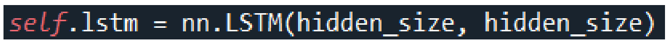


```
dictionary1 = {'SOS':0,'EOS':1,'UNK':2,'a':3,'b':4 ,'c':5,'d':6,'e':7,'f':8,'g':9,'h':10,'i':11,'j':12,'k':13,
               'l':14,'m':15,'n':16,'o':17,'p':18,'q':19,'r':20,'s':21,'t':22,'u':23,'v':24,'w':25,'x':26,'y':27,'z':28}
```

      Then reading input alphabets form string separately, calling functions to make them changing to numeric type.

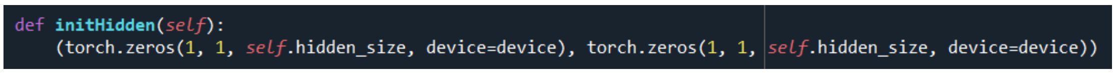
      Finally combine those numbers as a vector.
   B. Describe how you implement your LSTM model.

      I just modify the sample code of ”nn.GRU” part to “nn.lstm”.



```
self.lstm = nn.LSTM(hidden_size, hidden_size)
```

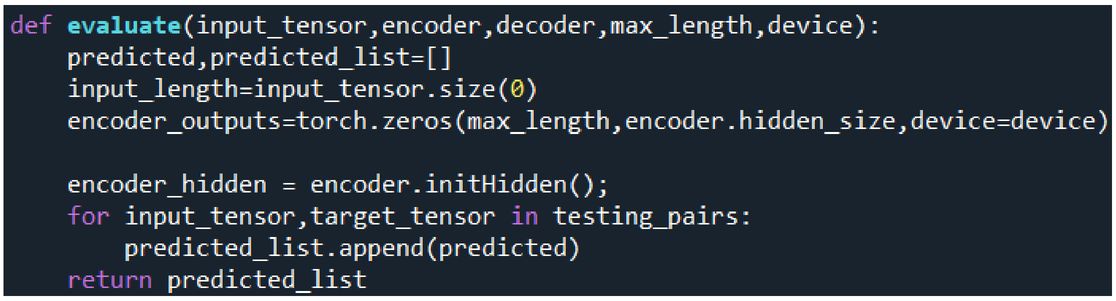
      And in “initHidden”, initialize the cell and hidden state instead of one.



```
def initHidden(self):
    (torch.zeros(1, 1, self.hidden_size, device=device), torch.zeros(1, 1, self.hidden_size, device=device))
```

   C. You must screen shot the code of evaluation part to prove that you do not use ground truth while testing.

      I try to build my evaluate function by following the logic of code I found in Internet and try to make some modify, but it seems that this part failed to work.



```
def evaluate(input_tensor,encoder,decoder,max_length,device):
    predicted,predicted_list=[]
    input_length=input_tensor.size(0)
    encoder_outputs=torch.zeros(max_length,encoder.hidden_size,device=device)

    encoder_hidden = encoder.initHidden();
    for input_tensor,target_tensor in testing_pairs:
        predicted_list.append(predicted)
    return predicted_list
```

4. Results and discussion
   A. Show your training of spelling correction and plot the training loss curve and evaluation curve during training.

      I didn’t complete the whole model, so there doesn’t exist result and plot can show.
   B. Model performance: BLUE-4 score (see below for performance scoring details)

      Due to the reason elaborate above, I didn’t output the BLUE-4 score.
   C. Discussion of the results

      I have not finish this lab due to inability of writing train and evaluate functions. It’s not difficult to realize and build lstm (also rely on Pytorch supply this function).

      When writing this lab I face some cumbers, one is I could not connect the words I cut fed into training pairs, others are errors that appears in trainIter and evaluate functions. I thought I'm not familiar with using the tensor is the main reason that I failed to complete the work. Also, I had not writing python code before, so the fact that I'm not good at calling some libraries in python may increase the difficulty. I thought I still need make some effort to getting over this problem.