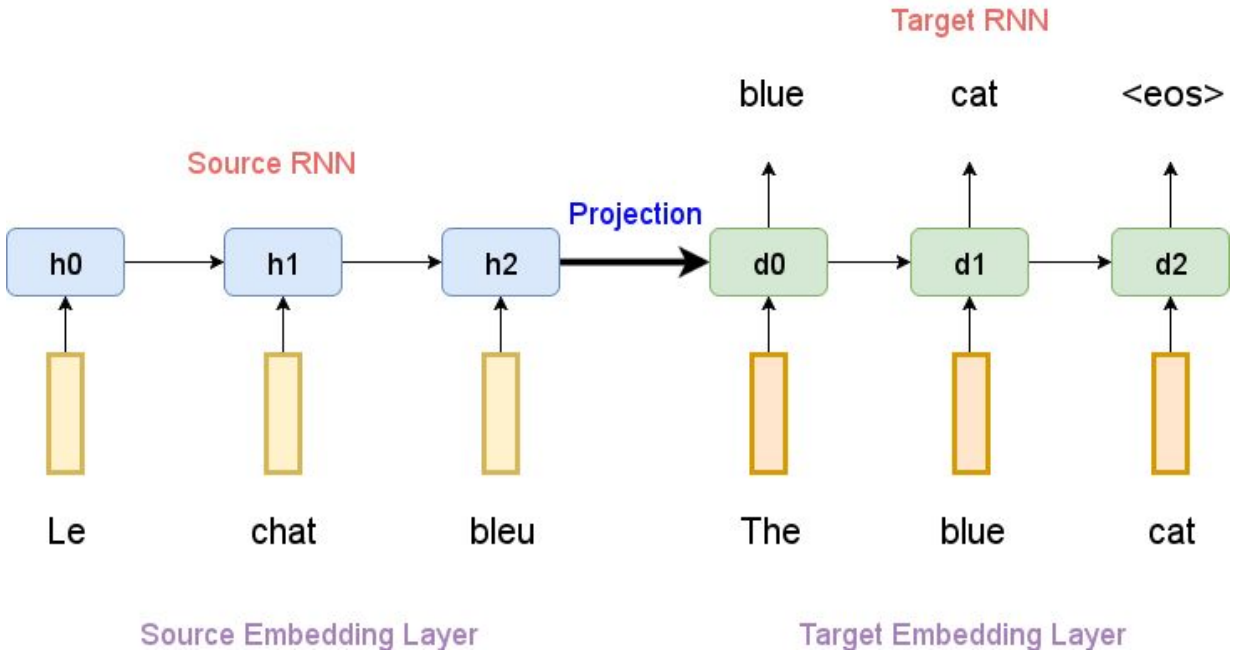
Lab4

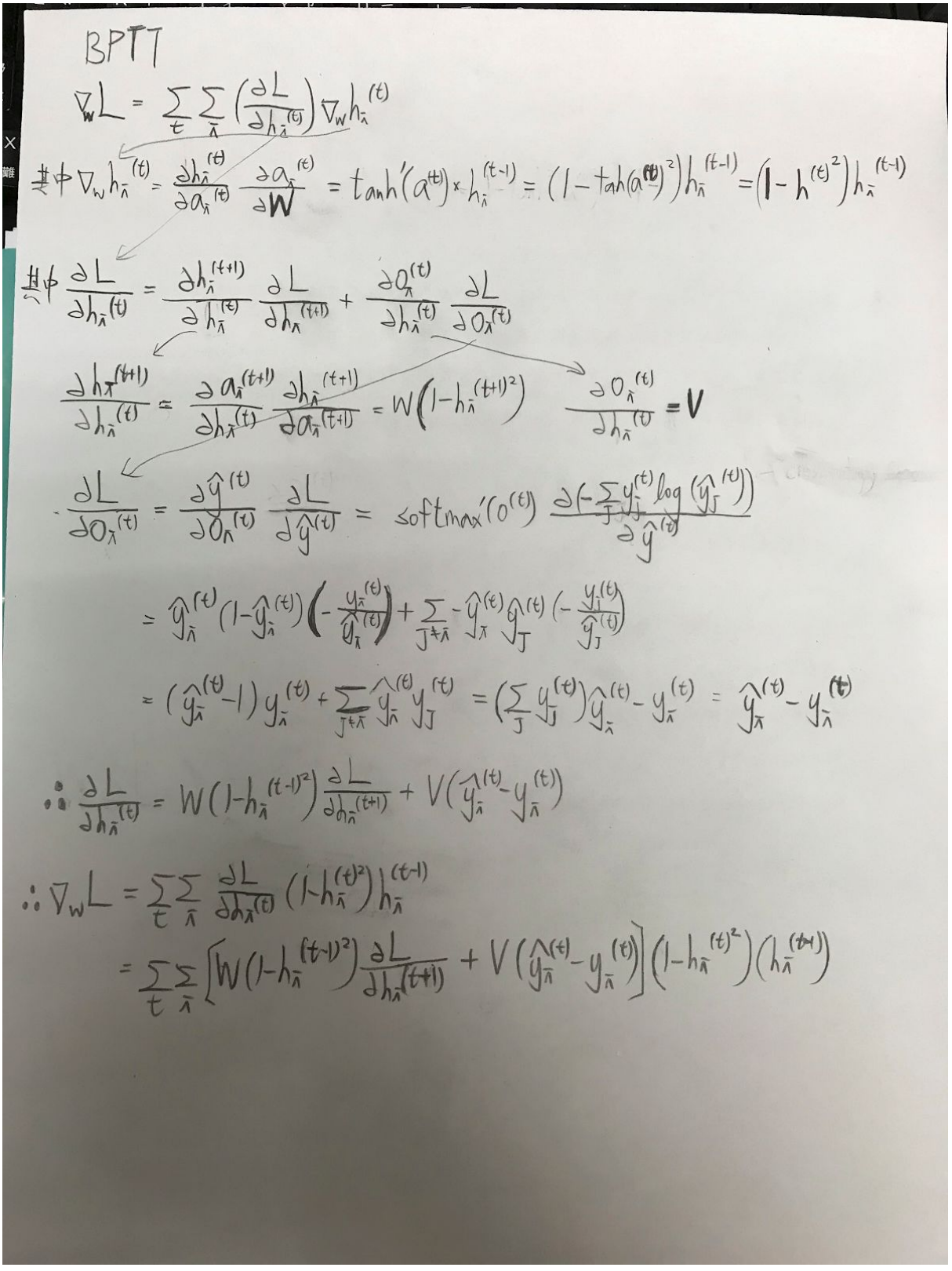
資工碩一 吳承翰 0856105

1.Introduction

用seq2seq Network實做英文(含錯誤字元)翻英文問題，完成spelling correction。
seq2seq有encoder與decoder，encoder負責壓縮資料成一個vector，decoder負責解析vector
。
圖例如下: 不過圖片中的word在這個task中皆為英文字母(a~z)



2.Derivation of BPTT



3.Implement details

(A)how I implement my model

要先建一個字母與整數的對應表：{'SOS':0,'EOS':1,'UNK':2,'a':3,'b':4 ... 'z':28}
當把每一個word丟進model時，要先把word變成一個整數序列，
像:'apple'+'EOS' -> [3 18 18 14 7 0]

```
def sequence2indices(self,sequence,add_eos=True):
    """
    :param sequence(string): a char sequence
    :param add_eox(boolean): whether add 'EOS' at the end of the sequence
    :return: int sequence
    """
    indices=[]
    for c in sequence:
        indices.append(self.char2idx[c])
    if add_eos:
        indices.append(self.char2idx['EOS'])
    self.MAX_LENGTH = max(self.MAX_LENGTH, len(indices))
    return indices
```

有了整數序列後再把每個整數值透過torch的embedding API轉成一個高為度的tensor，透過encoder中LSTM的多次time step forwarding，便可以得到context vector (hidden_state & cell_state)

```
#Encoder
class EncoderRNN(nn.Module):
    def __init__(self,input_size,hidden_size):
        """
        output of rnn is not used in simple decoder,but used in attention decoder
        :param input_size: 29 (containing:SOS,EOS,UNK,a-z)
        :param hidden_size: 256
        """
        super(EncoderRNN, self).__init__()
        self.hidden_size = hidden_size
        self.embedding = nn.Embedding(input_size, hidden_size)
        self.rnn = nn.LSTM(hidden_size, hidden_size)

    def forward(self, input, hidden_state, cell_state):
        """
        batch_size here is 1
        :param input: tensor
        :param hidden_state: (num_layers*num_directions=1,batch_size=1,vec_dim=256)
        :param cell_state: (num_layers*num_directions=1,batch_size=1,vec_dim=256)
        """
        embedded = self.embedding(input).view(1, 1, -1)  # view(1,1,-1) due to input of rnn must be (seq_len,batch,vec_dim)
        output,(hidden_state,cell_state) = self.rnn(embedded, (hidden_state,cell_state) )
        return output,hidden_state,cell_state
```

有了encoder輸出的hidden_state與cell_state，便可以把他們作decoder的輸入

```python
#Simple Decoder
class SimpleDecoderRNN(nn.Module):
    def __init__(self,input_size,hidden_size):
        super(SimpleDecoderRNN, self).__init__()
        self.hidden_size = hidden_size
        self.embedding = nn.Embedding(input_size, hidden_size)
        self.rnn = nn.LSTM(hidden_size, hidden_size)
        self.out = nn.Linear(hidden_size, input_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden_state, cell_state):
        output = self.embedding(input).view(1, 1, -1)
        output = F.relu(output)
        output, (hidden_state,cell_state) = self.rnn(output, (hidden_state,cell_state) )
        output = self.softmax(self.out(output[0]))
        return output,hidden_state,cell_state
```

forward()回傳的output就是predict的結果了。

(B)evalutation part do not use ground truth

在decoder中forwarding時decoder_output作為下一個time step的decoder_input，當decoder_output為EOS時就可以break結束這個word的回合了。

不像training時使用teacher_forcing，可以讓target作為decoder_input。

```python
def evaluate(decoder_type,input_tensor,encoder,decoder,max_length,device):
    """
    :param decoder_type: 'simple' or 'attention'
    :param input_tensor: (time,1) tensor for encoder
    :param max_length: word maximum length in training data
    :return: predicted indices list
    """
    predicted=[]
    input_length=input_tensor.size(0)
    encoder_outputs=torch.zeros(max_length,encoder.hidden_size,device=device)
    """
    encoder forwarding
    """
    encoder_hidden_state=encoder.init_h0()
    encoder_cell_state=encoder.init_c0()
    for ei in range(input_length):
        encoder_output,encoder_hidden_state,encoder_cell_state=encoder(input_tensor[ei],encoder_hidden_state,encoder_cell_state)
        encoder_outputs[ei]=encoder_output[0,0]
    """
    decoder forwarding
    """
    decoder_input=torch.tensor([[SOS_token]],device=device)
    decoder_hidden_state=encoder_hidden_state
    decoder_cell_state=encoder_cell_state
    for di in range(max_length):
        if decoder_type=='simple':
            decoder_output,decoder_hidden_state,decoder_cell_state=decoder(decoder_input,decoder_hidden_state,decoder_cell_state)
        else:  # decoder_type=='attention'
            decoder_output,decoder_hidden_state,decoder_cell_state,_=decoder(decoder_input,decoder_hidden_state,decoder_cell_state,encoder_outputs)
        topv,topi=decoder_output.data.topk(1)
        decoder_input=topi.squeeze().detach()
        if decoder_input.item()==EOS_token:
            break
        else:
            predicted.append(decoder_input.item())
    return predicted
```

4.Result and discussion

我實驗了0.3,0.5,0.7三種不同的teacher_forcing_ratio訓練50個epoch，最終的BLEU-4 score皆達到0.97或0.98，training過程中loss也很穩定的下降。

```
============================
input:  poartry
target: poetry
pred:   portty
============================
input:  leval
target: level
pred:   level
============================
input:  basicaly
target: basically
pred:   basically
============================
input:  triangulaur
target: triangular
pred:   triangular
============================
input:  unexpcted
target: unexpected
pred:   unexpected
============================
input:  stanerdizing
target: standardizing
pred:   standardizing
============================
input:  varable
target: variable
pred:   variable
```

```
============================
input:  mantain
target: maintain
pred:   maintain
============================
input:  miricle
target: miracle
pred:   miracle
============================
input:  oportunity
target: opportunity
pred:   opportunity
============================
input:  parenthasis
target: parenthesis
pred:   parenthesis
============================
input:  recetion
target: recession
pred:   recession
============================
input:  scadual
target: schedule
pred:   schedule
============================
BLEU-4: 0.98
```