Deep Learning HW3

A061714 藍國瑞

A. RNN architecture:

1. The standard RNN:

i. Network architecture:

- Mini-batch size=32
- Sequence length=10
- Optimizer: Adam
- Hidden layers and hidden states:
  1. RNN with 75 hidden states
  2. Dropout=0.2
  3. RNN with 100 hidden states
  4. Dropout=0.2
  5. Dense with 67 neurons

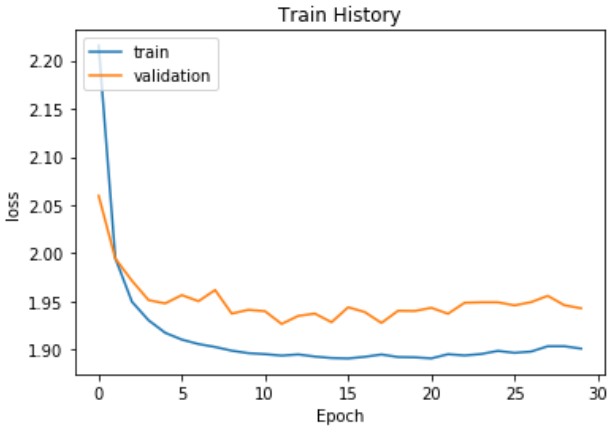
Layer (type)	Output Shape	Param #
simple_rnn_1 (SimpleRNN)	(None, 10, 75)	10725
dropout_1 (Dropout)	(None, 10, 75)	0
simple_rnn_2 (SimpleRNN)	(None, 100)	17600
dropout_2 (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 67)	6767

Total params: 35,092
Trainable params: 35,092
Non-trainable params: 0

`

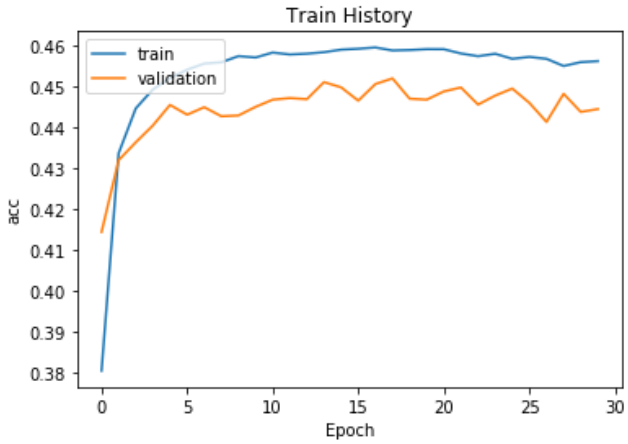
ii. Learning curve:

(loss V.S. epoch):



iii. Training error rate and valid error rate

(accuracy V.S. epoch):



2. Choose 5 breakpoints during training process. Feed some part of training text into RNN and show the text output:

Through the above loss_curve and acc_curve, we can observe when the epoch is 5 no matter the loss or the accuracy are gradually become saturated.

If I input the text " I "that was trained before, then the RNN model will return the text" **I for the way** ".It's likely the incomplete sentence.

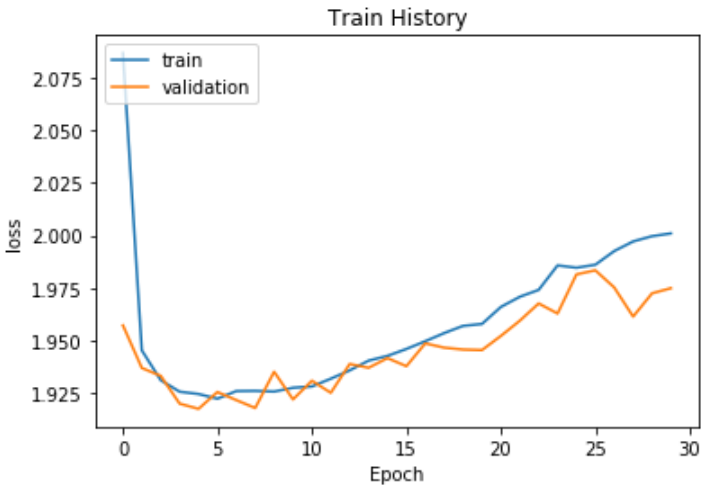
```
In [11]: print(generate_seq(model,mapping,10,'I',13))
I for the way
```

3. Compare the results of choosing different size of hidden states and sequence length by plotting the training curve V.S. different parameters:

- Network architecture:
  - Sequence length=5
  - Hidden layers and hidden states:
    1. RNN with 100 hidden states
    2. Dropout=0.2
    3. RNN with 150 hidden states
    4. Dropout=0.2
    5. Dense with 67 neurons

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_7 (SimpleRNN)     (None, 5, 100)            16800
_________________________________________________________________
dropout_7 (Dropout)          (None, 5, 100)            0
_________________________________________________________________
simple_rnn_8 (SimpleRNN)     (None, 150)               37650
_________________________________________________________________
dropout_8 (Dropout)          (None, 150)               0
_________________________________________________________________
dense_4 (Dense)              (None, 67)                10117
=================================================================
Total params: 64,567
Trainable params: 64,567
Non-trainable params: 0
```

  - Learning curve



B. LSTM architecture:

1. The standard LSTM:

ii. Network architecture:

- Mini-batch size=32
- Sequence length=10
- Optimizer: Adam
- Hidden layers and hidden states

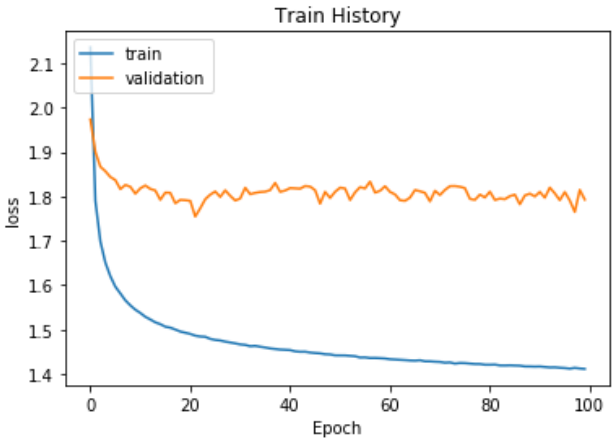
1. LSTM with 75 hidden states
2. Dropout=0.2
3. LSTM with 100 hidden states
4. Dropout=0.2
5. Dense with 67 neurons

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 10, 75)	42900
dropout_1 (Dropout)	(None, 10, 75)	0
lstm_2 (LSTM)	(None, 100)	70400
dropout_2 (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 67)	6767

Total params: 120,067
Trainable params: 120,067
Non-trainable params: 0

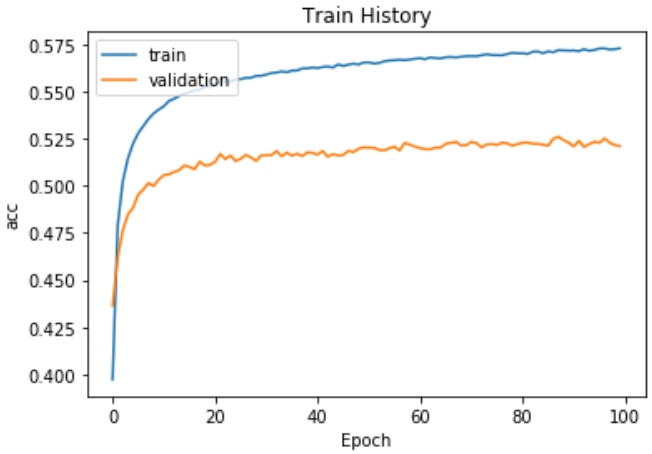
iii. Learning curve:

(loss V.S. epoch)



iv. Training error rate and valid error rate:

(accuracy V.S. epcoh)



2. Choose 5 breakpoints during training process. Feed some part of training text into LSTM and show the text output:

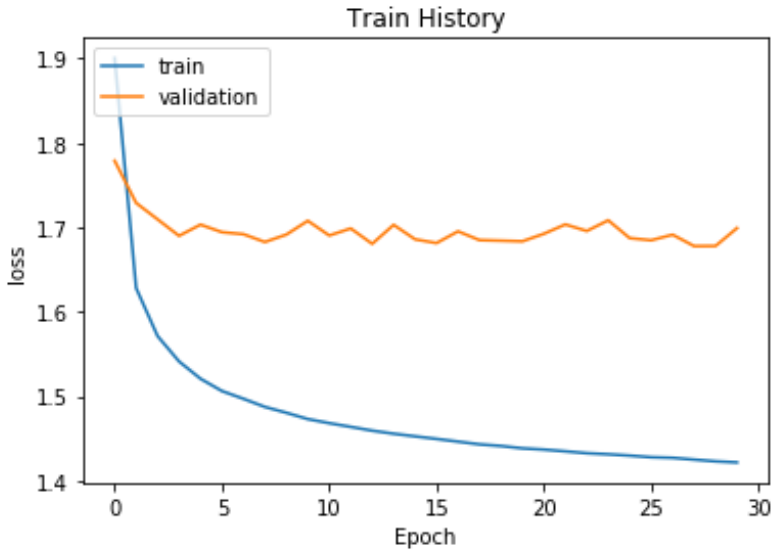
Through the above loss_curve and acc_curve, we can observe the validation data set becomes saturated when the epoch is 20. Hence in the next training process, I would like to set the epoch to be 20 instead of 100.

If I input the text " I "that was trained before, then the LSTM model will return the text" **I the common the senators of the courtesy**". Although the grammar is a little wrong, but we still can understand what it is talking about. I think the performance is better than the RNN model in PART A.

```
In [18]: print(generate_seq(model,mapping,10,'I',40))
I the common the senators of the courtesy
```

3. Compare the results of choosing different size of hidden states and sequence length by plotting the training curve V.S. different parameters:

- Network architecture
  - Sequence length=5
  - Hidden layers and hidden states
    1. LSTM with 100 hidden states
    2. Dropout=0.2
    3. LSTM with 150 hidden states
    4. Dropout=0.2
    5. Dense with 67 neurons
  - Learning curve:



C. Use LSTM to generate some words by priming the model with a word related to your dataset.

I use the LSTM with following architecture to generate the word:

- Network architecture
  - Sequence length=5
  - Hidden layers and hidden states
    1. LSTM with 100 hidden states
    2. Dropout=0.2
    3. LSTM with 150 hidden states
    4. Dropout=0.2
    5. Dense with 67 neurons

When I input the text ”Before” into the model, the model will output the text “ Before the state”.

```
In [33]: print(generate_seq(model,mapping,5,'Before',10))
Before the state
```