DEEP LEARNING

Homework 3

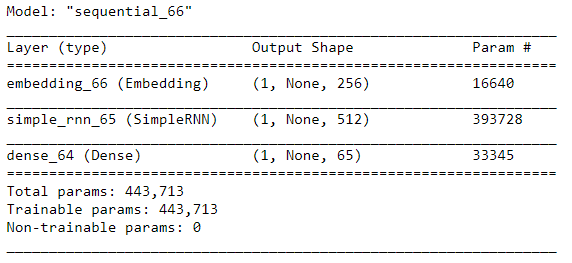
309511045 黃麟凱

Train Data：9938000 characters

Validation Data：110500 characters

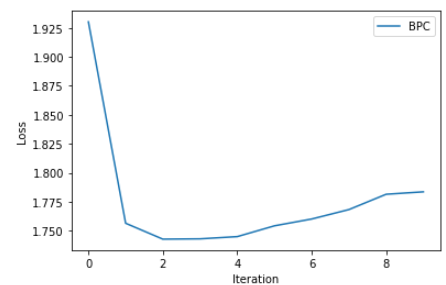
1. RNN：

Network architecture



這次使用的網路架構如上圖。除了使用RNN以外，還加上了Embedding layer 以及 Dense layer。Dense layer 的用意是在於將最後的輸出能壓縮到一個範圍，原因在於因為文本中所用到的字只有65個，因此需要將輸出大小壓至65。而Embedding的用意在於，將原本的字的array轉換到空間中，讓字跟字之間比較有相關性。

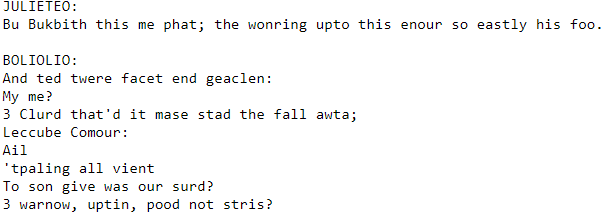
Learning curve



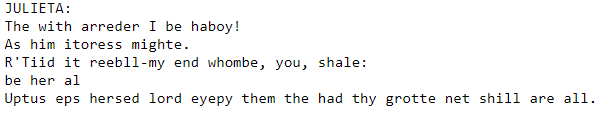
Training error rate and validation error rate

|  |  |
| --- | --- |
| Training | 1.789937 |
| Validation | 1.785276 |

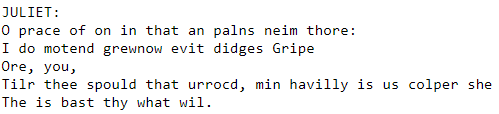
2/10 epochs:



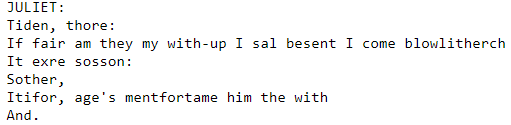
4/10 epochs:



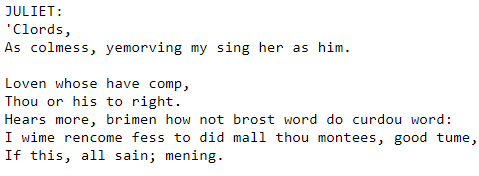
6/10 epochs:



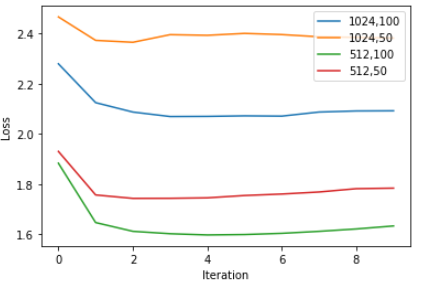
8/10 epochs:



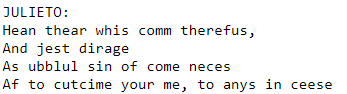
10/10 epochs:



我使用的輸入資料為”JULIET”，可以看出當訓練的時數越久，預測出的的結果越好。如在epochs 2 時，雖然可以看出部分字句了，但還是有很多字很奇怪。就已輸入的”JULIET”來看，照理來說下一個字應該是冒號，而成功預測出冒號的是從第6個epochs開始。且越後面的epochs，所產生的字也較符合文本。



我調整了size of hidden states 以及 sequence length的長度，分別使用了1024 或 512 個hidden states，以及100 或 50 的sequence length。結果如上圖所示，可以發現長度為100的sequence訓練的比長度為50的好。而hidden states的size，根據loss的結果，反而是512比1024好。我一開始以為1024的效果會比512好，但測試了好幾次產生出來的文章，可以發先1024的比較容易產生出新的字，如下圖：



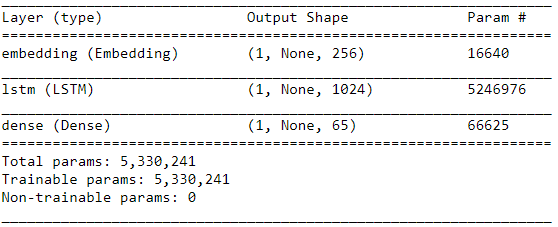
就會很容易在JULIET後面多一個 ’O’，反而512的就幾乎沒有這個狀況。如下圖：



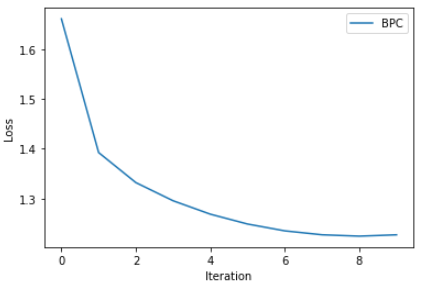
雖然大部分的字的結果都跟1024的差不多，但1024比較容易產生出文本沒有的字。

1. LSTM：

Network architecture



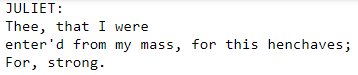
Learning curve



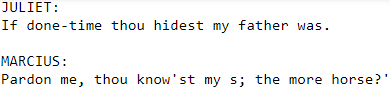
Training error rate and validation error rate

|  |  |
| --- | --- |
| Training | 1.1934 |
| Validation | 1.1972 |

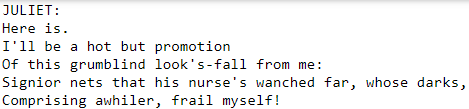
2/10 epochs:



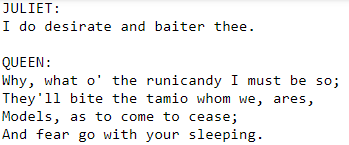
4/10 epochs:



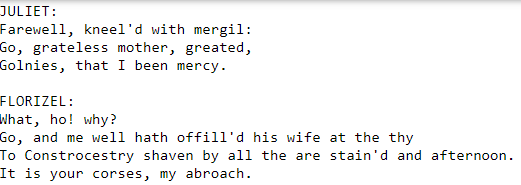
6/10 epochs:



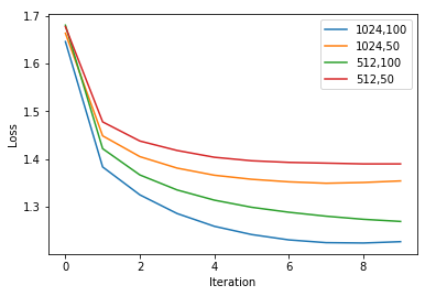
8/10 epochs:



10/10 epochs:

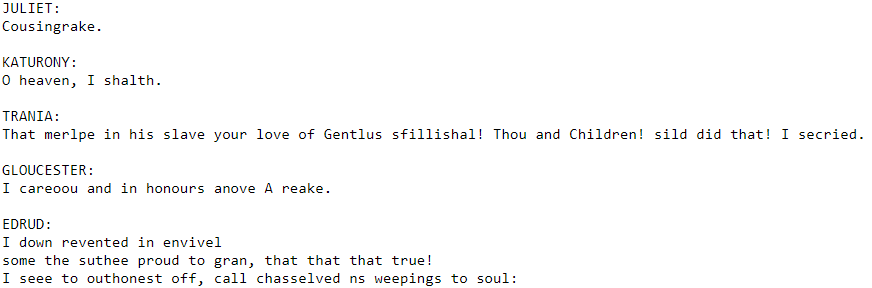


我使用的輸入資料一樣為”JULIET”，可以看出LSTM的結果明顯比RNN好很多，且大概從第6個epochs開始，就能做出比較好的句子了。

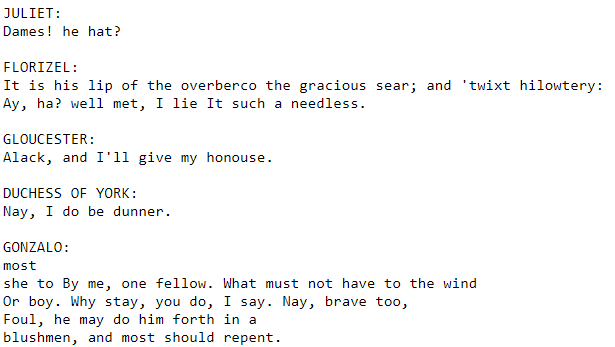


這次一樣調整hidden state的數量以及sequence length。一樣採用1024 or 512 兩種state 數量，以及100 or 50兩種length。跟RNN實驗的結果不太一樣，state的數量以及sequence length都扮演重要的角色。 同一種state，較長的length效果較好。反之亦然，同一種length，state越多效果越好。

1. Use RNN or LSTM to generate some words
   1. RNN



* 1. LSTM



RNN 以及 LSTM都是選擇訓練中最好的出來測試，可以看出LSTM的效果比RNN好，因為在單字的產生上，LSTM產生出來的字比較正常一點。除了這點以外，當我在用RNN產生字的時候，時常會有跑出奇怪的字( ex：JULIETTTEERT)，感覺是有點overfitting。相反的LSTM就不會有這樣的狀況。

Reference: <https://www.tensorflow.org/api_docs/python/tf/keras/layers/SimpleRNN>

<https://www.tensorflow.org/guide/keras/rnn>

<https://www.tensorflow.org/tutorials/text/text_generation>