

1. (1%) 請說明你實作之 RNN 模型架構及使用的 word embedding 方法，回報模型的正確率並繪出訓練曲線*

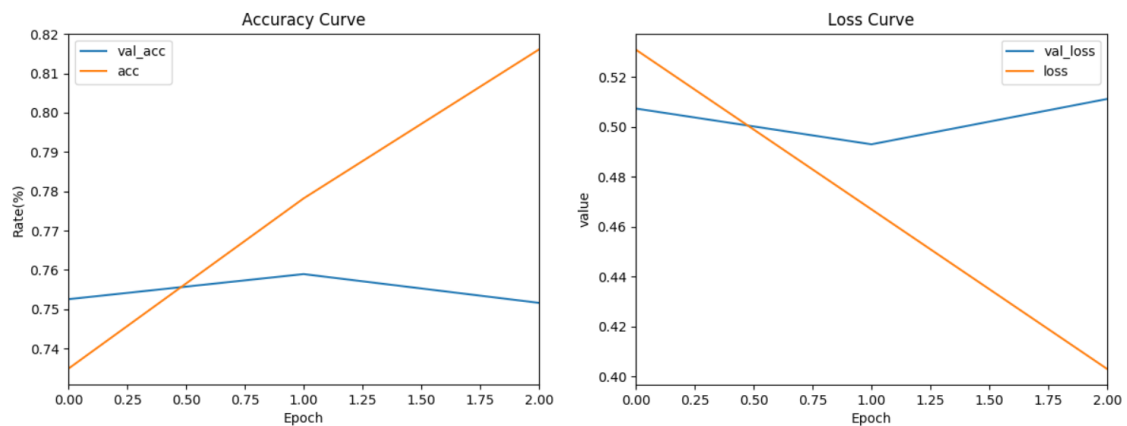
模型：

⇒ 資料經過預處理(jieba 與 word2vec)後就進下圖的模型中訓練，而模型架構主要有兩層雙向的 LSTM 再加上 dense 做訓練及預測。

⇒ Word embedding 的方法為用 word2vec 將句子中斷詞轉為 word vector

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, None, 250)	7889000
bidirectional_1 (Bidirection	(None, None, 512)	1038336
bidirectional_2 (Bidirection	(None, 256)	656384
dense_1 (Dense)	(None, 16)	4112
dense_2 (Dense)	(None, 2)	34
Total params: 9,587,866		
Trainable params: 9,587,866		
Non-trainable params: 0		

訓練過程：



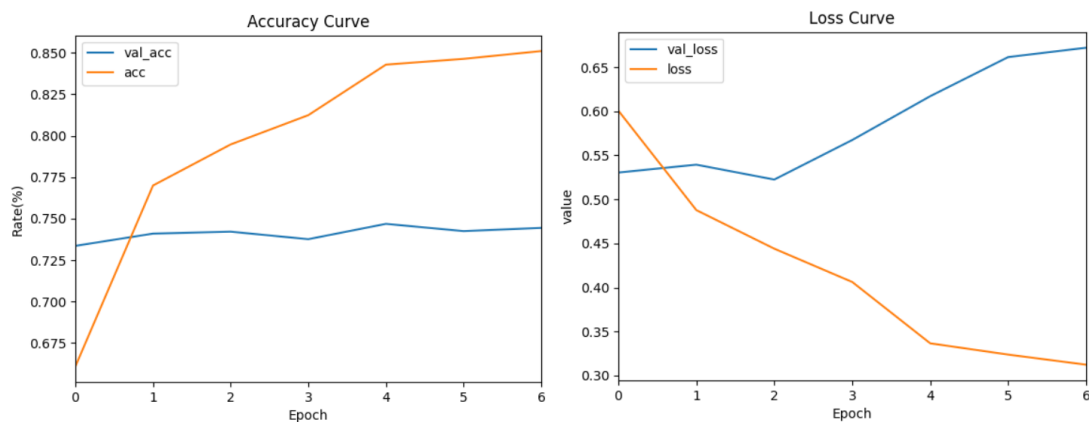
2. (1%) 請實作 BOW+DNN 模型，敘述你的模型架構，回報模型的正确率並繪出訓練曲線*。

模型:

⇒ 資料經過預處理(jieba 後轉為 bag of word 的型式)後就進下圖的模型中訓練，而模型架構主要有五層 Dense 做訓練及預測。

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1024)	32314368
dense_2 (Dense)	(None, 1024)	1049600
dense_3 (Dense)	(None, 512)	524800
dense_4 (Dense)	(None, 512)	262656
dense_5 (Dense)	(None, 2)	1026
Total params: 34,152,450		
Trainable params: 34,152,450		
Non-trainable params: 0		

訓練過程:



⇒ 最後結果較差，原因是不同語意會因缺乏順序而混淆。

3. (1%) 請敘述你如何 improve performance (preprocess, embedding, 架構等)，並解釋為何這些做法可以使模型進步。

⇒ 我這邊採用 ensemble 去作處理，因為語意的分析容易 overfitting，所以我們可以採用多個模型作預測並平均，如此一來便會有更加 global 的表現。

4. (1%) 請比較不做斷詞 (e.g., 以字為單位) 與有做斷詞，兩種方法實作出來的效果差異，並解釋為何有此差別。

⇒ 在不使用斷詞(不經 jieba 直接丟入 word2vec)的情況下，在 test set 的準確度約下降 1%，而這個原因我推測與詞意有關，在中文裡詞義會因組合而有所不同，所以好的斷詞確實可以將語意更好地表達出來。

5. (1%) 請比較 RNN 與 BOW 兩種不同 model 對於 "在說別人白痴之前，先想想自己" 與 "在說別人之前先想想自己，白痴" 這兩句話的分數（model output），並討論造成差異的原因。

⇒ 因為 RNN 會是有順序的，所以這兩句所對應的分數會不同，但是 BOW 不一樣，我只在乎這個斷詞是否存在，故這兩句對應到相同的分數。