學號：R06922048 系級： 資工所碩一 姓名：陳柏堯

1. (1%) 請說明你實作的 RNN model，其模型架構、訓練過程和準確率為何？(Collaborators: 無)

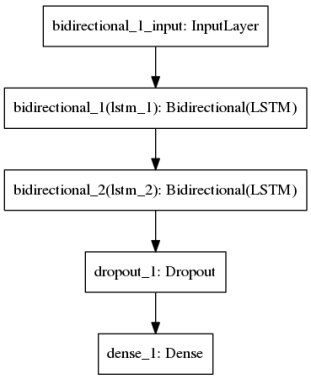
答：

模型架構：

首先讀入 data, 去掉 stop words('is', 'are', 'a', 'the', 'm', 'was', 'were), 再利用 gensim 的 word2vec 將每一個 word 轉成 80 維、windows size 5 的 vector，變成 vector sequence 之後再丟入 RNN Model。架構中有兩層 bidirection 的 RNN model，然後接 dropout 及輸出到 2 元分類。

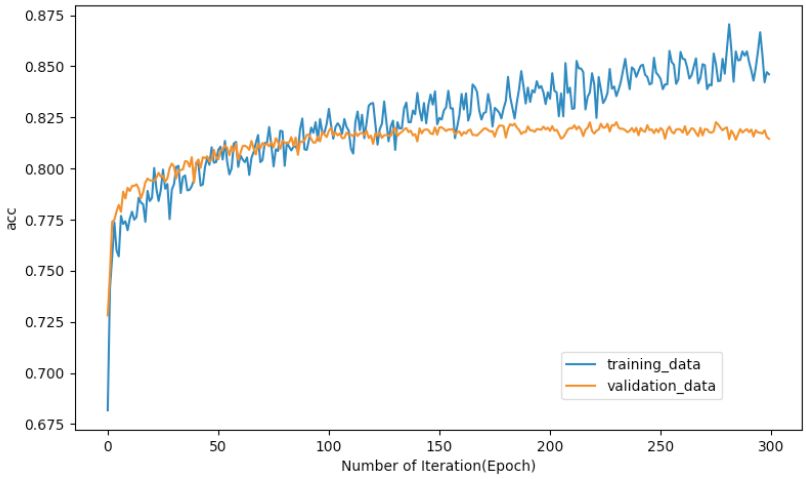
Layer (type)	Output Shape	Param #
bidirectional_1 (Bidirection	(None, 35, 256)	214016
bidirectional_2 (Bidirection	(None, 128)	164352
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 2)	258

Total params: 378,626
Trainable params: 378,626
Non-trainable params: 0



訓練過程：

因為往往 5~10 個 epoch 左右之後，RNN 的 training 就會 overfitting，我認為這個過程中可能會錯過 acc_val 的高點，所以我自己實作了 data_generator，他每次吐出 batch128 的 training data，以一次 30 個 steps 作為 epoch 來 train。這樣 model 真正看過所有 data 一次實際大概是 60 個 epoch 左右。 如下圖，大概再 100 個 epoch 左右，val acc 就會在 0.81~0.82 左右振動。



（validation data 是從 training data 中 10%隨機切出來的）

準確率：

在第 224 個 epoch 處，得到最好的 validation acc 0.824100, 上傳到 Kaggle 得到

public acc 0.82327

2. (1%) 請說明你實作的 BOW model，其模型架構、訓練過程和準確率為何？ (Collaborators: 無)

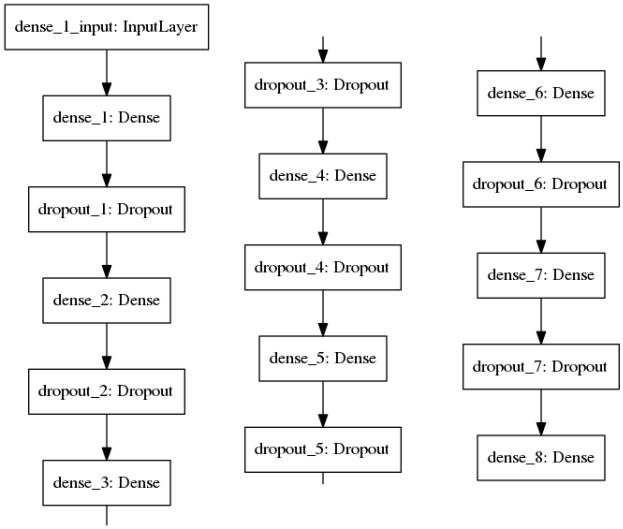
答：

模型架構：

前處理的做法和 RNN 第一題差不多，只是 tokenizer 是用助教範例的 keras.texts_to_matrix 來 count 詞語的出現次數作為 vector 來 embedding。最後再放入到 DNN 中來 training。

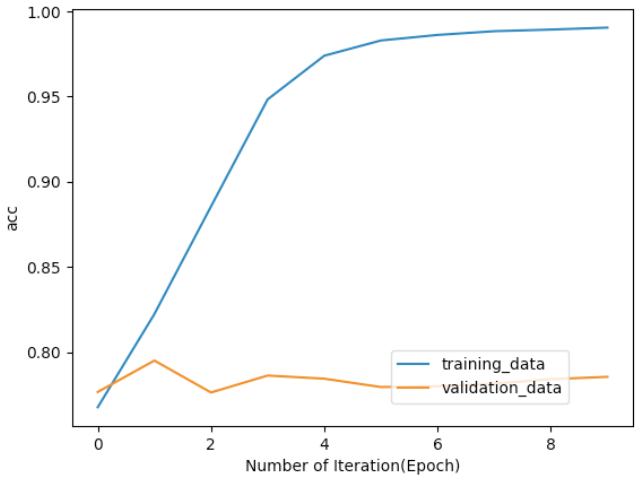
Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 4096)	40964096
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 2048)	8390656
dropout_2 (Dropout)	(None, 2048)	0
dense_3 (Dense)	(None, 1024)	2098176
dropout_3 (Dropout)	(None, 1024)	0
dense_4 (Dense)	(None, 521)	534025
dropout_4 (Dropout)	(None, 521)	0
dense_5 (Dense)	(None, 128)	66816
dropout_5 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 32)	4128
dropout_6 (Dropout)	(None, 32)	0
dense_7 (Dense)	(None, 8)	264
dropout_7 (Dropout)	(None, 8)	0
dense_8 (Dense)	(None, 2)	18

Total params: 52,058,179
Trainable params: 52,058,179
Non-trainable params: 0



訓練過程：

因為覺得 BOW 的 performance 不會太好，所以並沒有寫 data generator，直接 train 下去，training data 很快就上去到 0.9，而 val acc 大概再 0.79 左右振動。



準確率：

在第 1 個 epoch 處，得到最好的 validation acc 0.795100, 上傳到 Kaggle 得到 public acc 0.79747

3. (1%) 請比較 bag of word 與 RNN 兩種不同 model 對於"today is a good day, but it is

hot"與"today is hot, but it is a good day"這兩句的情緒分數，並討論造成差異的原因。

(Collaborators: 無)

答：

如下圖，分別將兩句話作前處理再分別丟入兩個 model，RNN 會考慮句子前後出現順序的影響，如果 but 後面是 hot 則預測負面情緒的分數較大，如果 but 後面是 good day，則預測正面情緒的分數較大。而 BOW 的預測結果，或許只考慮單詞出現與否，可能對於出現“good”這個詞，model 都給正面情緒較大的分數。因此，就 Sequence 的 word 前後順序，RNN 表現較好。

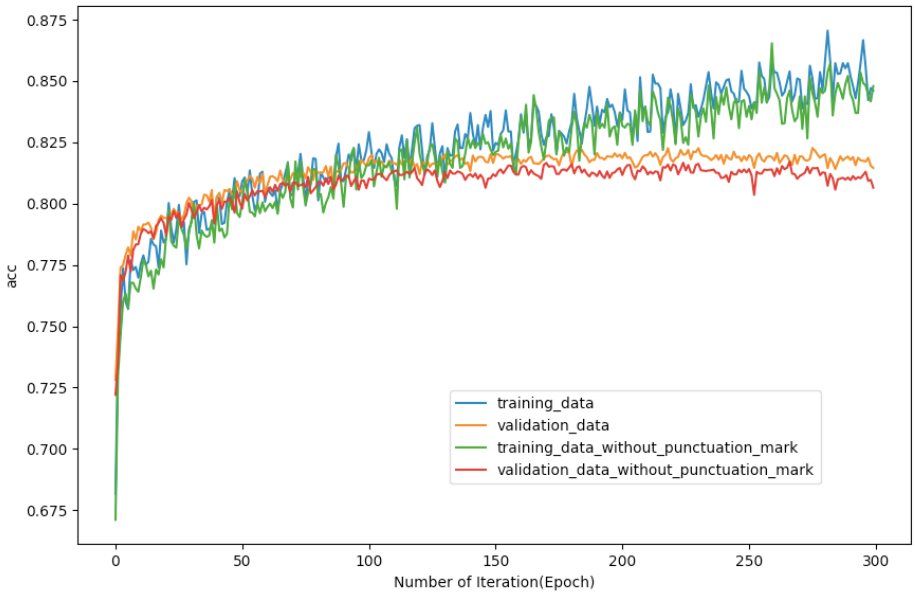
	RNN (Word2Vec)		BOW	
	0	1	0	1
today is a good day, but it is hot	0.80578971	0.19421022	0.30266365	0.69733632
today is hot, but it is a good day	0.08541219	0.91458774	0.18986592	0.81013411

4. (1%) 請比較"有無"包含標點符號兩種不同 tokenize 的方式，並討論兩者對準確率的影響。

(Collaborators: 無)

答：

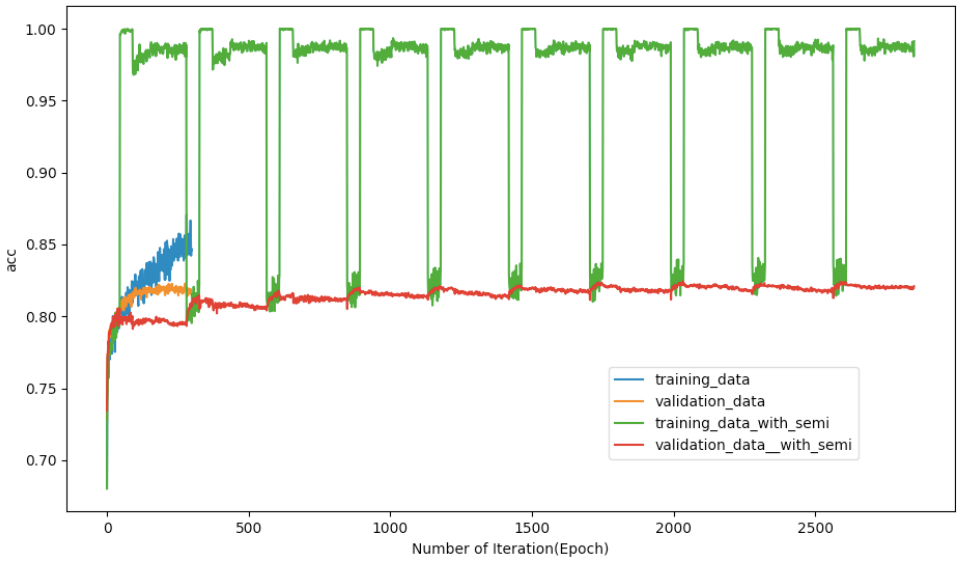
在第 1 題的 RNN 中，我本來就有去掉部分標點符號（#$%&()*+-/:;<=>@[\\]^_`{|}），但是保留一些符號（!.?,）,因為我覺得驚歎號之類的符號有表達一種情緒，而在本題中我在第一題的基礎上再去掉了（!.?,）這些符號，也就是說完全沒有符號了。架構和第一題一樣下去 train，得到 performance 如圖：



可以發現有無標點符號的兩個版本，訓練過程非常相似，但是保留驚歎號、問號的版本無論在 training data 還是 validation data 都有比較好的表象。去掉所有符號的 model 的 validation acc 在 100 個 epoch 之後大概在 0.80~0.81 左右振動，略低于 Strong baseline。因此，保留有一些符號的 tokenize 方式較好。

5. (1%) 請描述在你的 semi-supervised 方法是如何標記 label，並比較有無 semi-surpervised training 對準確率的影響。
   (Collaborators: 無)
   答：
   我使用的 Semi-supervised 方法是老師上課說的 self-training，首先依然是先切出 10%的 validation data, 架構和第一題的 RNN、word2vec 一樣，然後 training 的方法是：
   * train 一遍所有的 labeled data (因為是自己實作 data generator，所以 train 一遍是 60 個 epoch)
   * 分 4 次從 unlabeled data 抓出 sentence
   * 用目前的 model 去 predict 目前抓出來的 sentences
   * predict 好的結果再回去 train model，設定的 threshold 是 0.25

   Train 一遍 labeled data 和抓 4 次 unlabed 作為一個 round，總共 repeat 10 round。(分成 4 次抓出 unlabeled data 是為了使用較少的記憶體，如果全部 100 萬筆資料抓出來做 word2vector 會出現 memory error).



如上圖，使用 semi-supervised learning，我覺得 performance 和沒有使用是差不多的，但如果長遠下來多一些 epoch，semi-supervised 會較好一點。

沒有使用 self-training 的 model，training data 的 acc 很快就會爬上去，會容易 overfitting，而使用 self-training 的話，training-acc 會爬得比較慢.

這樣可以多跑幾個 epoch 讓 validation acc 再進一步往上升。我認為 semi-supervised learning 對於 model 的穩定度有一些幫助。

結果：
沒有使用 semi-supervied learning 的第一題中我最好的 validation acc 是 0.824100，這題中使用 semi-supervised learning 我最好的 validation acc 是 0. 824150，semi 的準確率略高一些，但其實差不多。