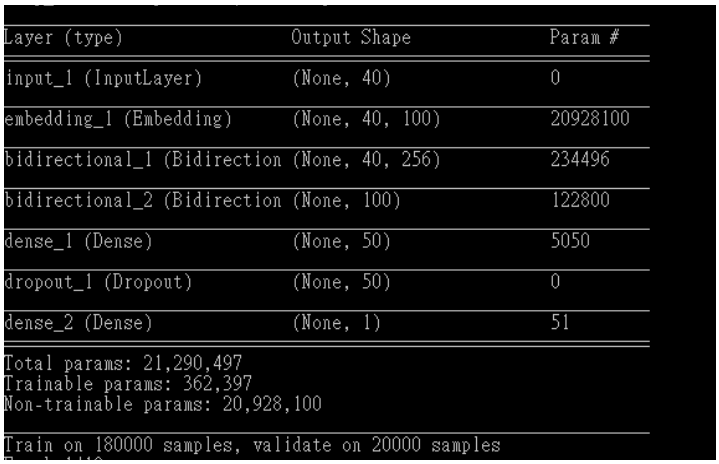
學號：B04902063 系級： 資工三 姓名：陳昱儒

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: 模型架構部分有參考去年修課的 湯忠憲 同學)

在進入訓練前有先使用gensim提供的PorterStemmer()提取詞幹，將一些英文單字詞性變化的差異去除。模型架構部分，首先使用了gensim提供的Word2Vec訓練好的100維詞向量丟進embedding layer（預設句子最長的長度為40個單字），再來分別使用了兩層的 Bidirectional LSTM，輸出維度分別為128和50，並分別使用了0.3以及0.2的dropout rate，activation 選用tanh。之後再接了兩層的Dense，輸出維度分別為50以及1，兩層Dense之間設了一個dropout，比率為0.3，而兩個Dense使用的activation分別為relu以及sigmoid，而model 訓練的 loss function 選用binary_crossentropy，optimizer選用adam，使用了10%的training data 做為validation data，model的summary如下:

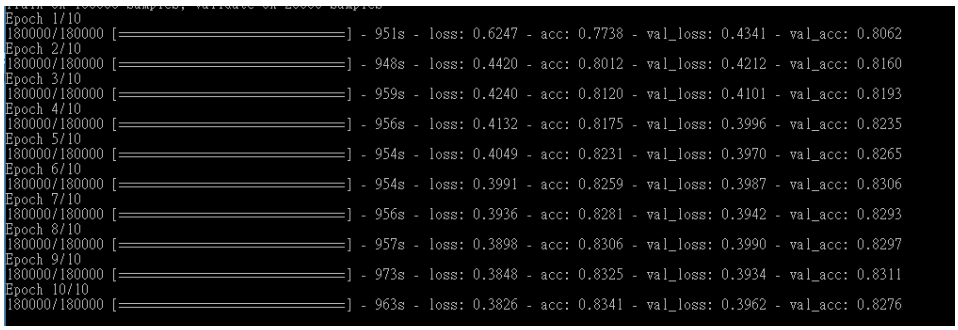


Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 40)	0
embedding_1 (Embedding)	(None, 40, 100)	20928100
bidirectional_1 (Bidirection	(None, 40, 256)	234496
bidirectional_2 (Bidirection	(None, 100)	122800
dense_1 (Dense)	(None, 50)	5050
dropout_1 (Dropout)	(None, 50)	0
dense_2 (Dense)	(None, 1)	51

Total params: 21,290,497
Trainable params: 362,397
Non-trainable params: 20,928,100

Train on 180000 samples, validate on 20000 samples

以下為 model 訓練過程，可以從中發現 model 其實收斂的滿快的，大概在第五六個 epoch validation data set 的準確度就開始在震盪了。



```
Epoch 1/10
180000/180000 [==============================] - 951s - loss: 0.6247 - acc: 0.7738 - val_loss: 0.4341 - val_acc: 0.8062
Epoch 2/10
180000/180000 [==============================] - 948s - loss: 0.4420 - acc: 0.8012 - val_loss: 0.4212 - val_acc: 0.8160
Epoch 3/10
180000/180000 [==============================] - 959s - loss: 0.4240 - acc: 0.8120 - val_loss: 0.4101 - val_acc: 0.8193
Epoch 4/10
180000/180000 [==============================] - 956s - loss: 0.4132 - acc: 0.8175 - val_loss: 0.3996 - val_acc: 0.8235
Epoch 5/10
180000/180000 [==============================] - 954s - loss: 0.4049 - acc: 0.8231 - val_loss: 0.3970 - val_acc: 0.8265
Epoch 6/10
180000/180000 [==============================] - 954s - loss: 0.3991 - acc: 0.8259 - val_loss: 0.3987 - val_acc: 0.8306
Epoch 7/10
180000/180000 [==============================] - 956s - loss: 0.3936 - acc: 0.8281 - val_loss: 0.3942 - val_acc: 0.8293
Epoch 8/10
180000/180000 [==============================] - 957s - loss: 0.3898 - acc: 0.8306 - val_loss: 0.3990 - val_acc: 0.8297
Epoch 9/10
180000/180000 [==============================] - 973s - loss: 0.3848 - acc: 0.8325 - val_loss: 0.3934 - val_acc: 0.8311
Epoch 10/10
180000/180000 [==============================] - 963s - loss: 0.3826 - acc: 0.8341 - val_loss: 0.3962 - val_acc: 0.8276
```

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？

首先找出labeled data中出現次數大於20次的字詞將它們作為辭典，並利用此辭典將labeled data中的每一個詞按照詞典用one-hot轉換，當作一個bag of word，若是沒有見過的字就忽略掉，最後會製作成一個data size * dictionary size 的二維陣列，在使用四層Dense layer(512,256,32,1)訓練，四層layer間各設置dropout，比率均為0.2，最後的output在使用sigmoid function作為activation function，model 訓練的部分則和RNN一樣 loss function 選用binary_crossentropy，optimizer選用adam，以下為model的summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               3339776
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 84)                43092
_________________________________________________________________
dropout_2 (Dropout)          (None, 84)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 85
_________________________________________________________________
activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 3,382,953
Trainable params: 3,382,953
Non-trainable params: 0
_________________________________________________________________
Train on 180000 samples, validate on 20000 samples
```

以下為 model 訓練過程，由圖中可以發現，model 從一開始 validation data 得準確率就一直在震盪，即使訓練到第十個 epoch 準確率都落在 0.78 附近，比起 RNN model 沒有什麼顯著的上升趨勢，而 loss 則是緩慢的下降。在 kaggle 得到的分數為 0.78728，也較 RNN 差些，但還是有一定的水準。

```
Epoch 1/10
2018-05-23 12:45:02.841812: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this
o use: SSE4.1 SSE4.2 AVX AVX2 FMA
180000/180000 [==============================] - 161s - loss: 0.4984 - acc: 0.7706 - val_loss: 0.4824 - val_acc: 0.7803
Epoch 2/10
180000/180000 [==============================] - 148s - loss: 0.4695 - acc: 0.7902 - val_loss: 0.4829 - val_acc: 0.7864
Epoch 3/10
180000/180000 [==============================] - 149s - loss: 0.4639 - acc: 0.7934 - val_loss: 0.4808 - val_acc: 0.7849
Epoch 4/10
180000/180000 [==============================] - 146s - loss: 0.4613 - acc: 0.7940 - val_loss: 0.4790 - val_acc: 0.7847
Epoch 5/10
180000/180000 [==============================] - 143s - loss: 0.4597 - acc: 0.7952 - val_loss: 0.4805 - val_acc: 0.7826
Epoch 6/10
180000/180000 [==============================] - 148s - loss: 0.4581 - acc: 0.7960 - val_loss: 0.4814 - val_acc: 0.7827
Epoch 7/10
180000/180000 [==============================] - 144s - loss: 0.4566 - acc: 0.7965 - val_loss: 0.4816 - val_acc: 0.7835
Epoch 8/10
180000/180000 [==============================] - 141s - loss: 0.4556 - acc: 0.7980 - val_loss: 0.4853 - val_acc: 0.7805
Epoch 9/10
180000/180000 [==============================] - 206s - loss: 0.4550 - acc: 0.7984 - val_loss: 0.4857 - val_acc: 0.7855
Epoch 10/10
180000/180000 [==============================] - 176s - loss: 0.4539 - acc: 0.7992 - val_loss: 0.4855 - val_acc: 0.7833
```

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。

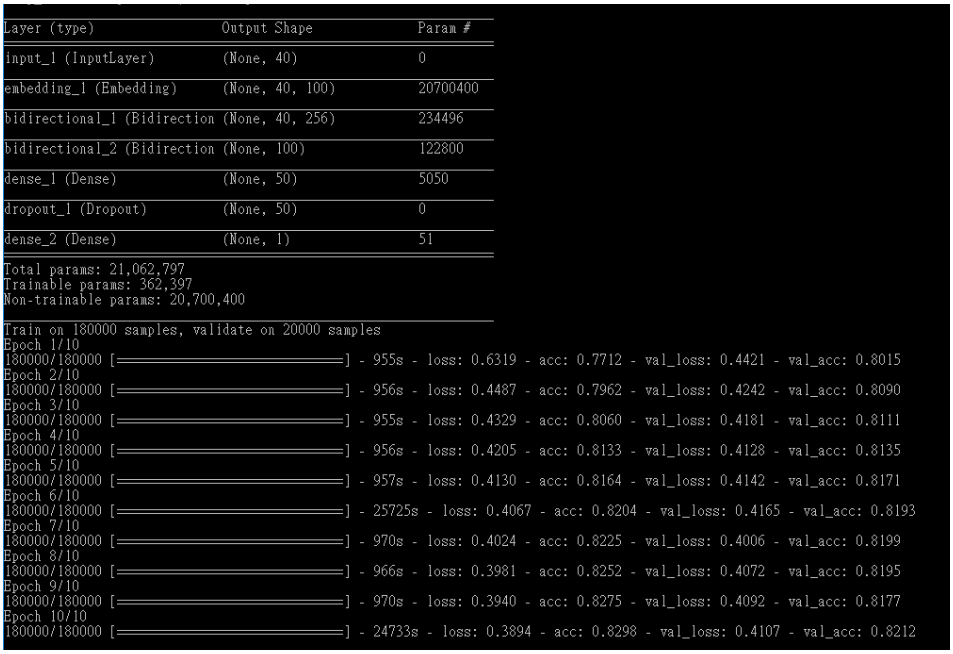
	RNN model	bag of word model
today is a good day, but it is hot	0.160979	0.542912
today is hot, but it is a good day	0.915438	0.594889

上圖為兩句話分別經過 RNN model 以及 BOW model 預測出的結果，可以看出在 BOW model 中兩個句子預測出的結果幾乎沒有差異，都給出 positive 但信心水準極低的答案，我想是因為 BOW model 不會考慮單詞在句中的位置，僅會考慮這個句子由什麼樣的單字組成，所以會造成這兩個句子被預測出來的結果幾乎無差異（我想數值會不一樣是因我是使用空格斷詞，所以 day,和 hot,會被當作一個單字，而造成兩句結果的些微差異）。而 RNN model 就對這兩句給出一反一正且信心水準滿高的答案，因 RNN model 會判斷單詞出現的先後順序，因此位置的調換對 RNN model 來說就是不一樣的句子了！比較不會有預測出來相似的狀況。

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。

將 training data 中所有的標點符號去掉後，使用和第一題一模一樣的 model 做訓練，發現在 kaggle 上的分數偏低了一點點，可以判斷標點符號對於句子正負面的判斷還是有一點點影響力的！

	有包含標點符號	去除標點符號
kaggle score	0.82751	0.82033



```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 40)                0
_________________________________________________________________
embedding_1 (Embedding)      (None, 40, 100)           20700400
_________________________________________________________________
bidirectional_1 (Bidirection (None, 40, 256)           234496
_________________________________________________________________
bidirectional_2 (Bidirection (None, 100)               122800
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 51
=================================================================
Total params: 21,062,797
Trainable params: 362,397
Non-trainable params: 20,700,400
_________________________________________________________________
Train on 180000 samples, validate on 20000 samples
Epoch 1/10
180000/180000 [==============================] - 955s - loss: 0.6319 - acc: 0.7712 - val_loss: 0.4421 - val_acc: 0.8015
Epoch 2/10
180000/180000 [==============================] - 956s - loss: 0.4487 - acc: 0.7962 - val_loss: 0.4242 - val_acc: 0.8090
Epoch 3/10
180000/180000 [==============================] - 955s - loss: 0.4329 - acc: 0.8060 - val_loss: 0.4181 - val_acc: 0.8111
Epoch 4/10
180000/180000 [==============================] - 956s - loss: 0.4205 - acc: 0.8133 - val_loss: 0.4128 - val_acc: 0.8135
Epoch 5/10
180000/180000 [==============================] - 957s - loss: 0.4130 - acc: 0.8164 - val_loss: 0.4142 - val_acc: 0.8171
Epoch 6/10
180000/180000 [==============================] - 25725s - loss: 0.4067 - acc: 0.8204 - val_loss: 0.4165 - val_acc: 0.8193
Epoch 7/10
180000/180000 [==============================] - 970s - loss: 0.4024 - acc: 0.8225 - val_loss: 0.4006 - val_acc: 0.8199
Epoch 8/10
180000/180000 [==============================] - 966s - loss: 0.3981 - acc: 0.8252 - val_loss: 0.4072 - val_acc: 0.8195
Epoch 9/10
180000/180000 [==============================] - 970s - loss: 0.3940 - acc: 0.8275 - val_loss: 0.4092 - val_acc: 0.8177
Epoch 10/10
180000/180000 [==============================] - 24733s - loss: 0.3894 - acc: 0.8298 - val_loss: 0.4107 - val_acc: 0.8212
```

以上為去除標點符號後的訓練過程以及 model summary。

5. (1%) 請描述在你的 semi-supervised 方法是如何標記 label，並比較有無 semi-supervised training 對準確率的影響。
(Collaborators: 參考助教投影片)

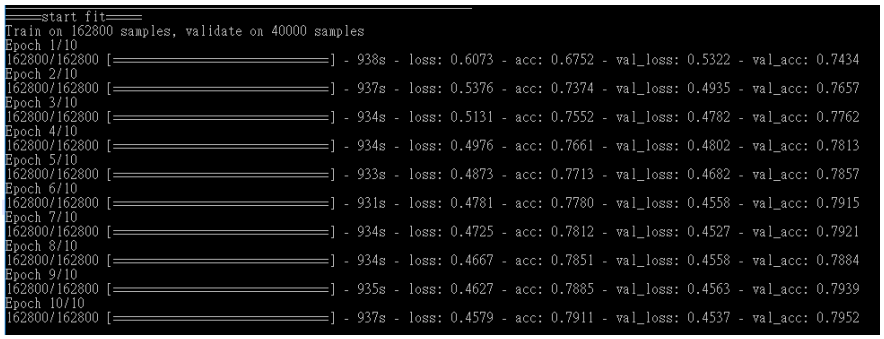
我選擇的方法是實作 self-training。先利用先前使用 labeled data 訓練出的 model 去預測 unlabeled training data，將 predict 出來結果大於 0.9 或小於 0.1 的 data 挑出來(也就是對於結果是 1 或 0 信心水準較高的 data)與原本的 labeled training data 接起來當做 training data 訓練 model，並記錄下已使用過的 data，讓下次預測時避免使用到已經被加入訓練過的 data，每次新加入 data 後會訓練 10 個 epoch(和原本的 model 一樣)，而 validation data 全都使用和訓練原本 model 一樣的 data set（labeled data 前 40000 筆）以拿來比較，這樣訓練了三輪(訓練數量分別為 162800/236949/250959 筆)，結果如下:

	semi-supervised	no semi-supervised
kaggle score (public)	0.73852	0.82751

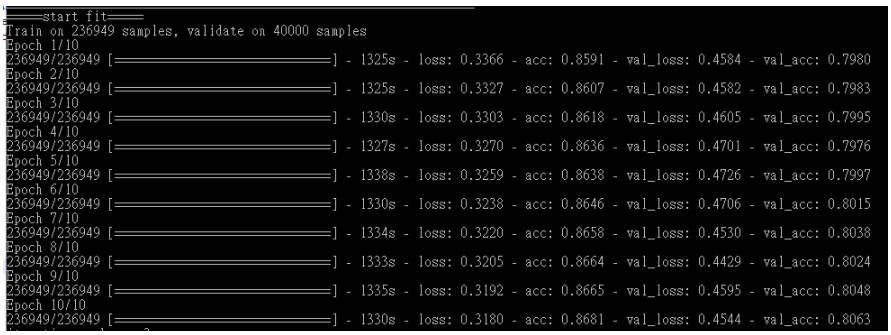
	First round	Second round	Third round
validation accuracy	0.74~0.79	0.798~0.806	0.802~0.806

由上表可以發現我的 model 再加入了 semi-supervised 後準確率反而變差了，我想可能是我所設的 threshold 較低篩進了比較多的劣質資料或是產生了 overfitting 的狀況(在第三輪的 training accuracy 都有在 87%以上)，所以造成訓練出來的結果反而比沒有實作 semi-supervised training 的 model 準確率還來得差。

第一輪訓練:



第二輪訓練:



第三輪訓練: