HW4 Report

學號：R06942074 系級：電信所碩一 姓名：李宇哲

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？

我實作的 RNN model 主要是先使用 gensim 的 library pretrain training 和 testing data，各有 20 萬筆資料，然後將近 20 萬個字，不過我只取其中的 20000 個字當作 feature，然後把用 pretrain 的 weights 塞進 RNN Model，再來就是用接上 Bidirectional 的 LSTM，總共接上兩層，然後 Dense 兩層出來，整體的架構如下圖。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 36, 128)           2560128
_________________________________________________________________
bidirectional_1 (Bidirection (None, 36, 512)           788480
_________________________________________________________________
bidirectional_2 (Bidirection (None, 256)               656384
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 4,046,209
Trainable params: 1,486,081
Non-trainable params: 2,560,128
```

訓練的過程使用使用 adam optimizer 用 learning rate = 0.01 去 minimize loss

最後的整確率如下：

Name	Submitted	Wait time	Execution time	Score
ans_pun1.csv	a few seconds ago	2 seconds	2 seconds	0.82571

Complete

2. 請說明你實作的 BOW model，其模型架構、訓練過程和準確率為何？
我的 BOW model 使用 tokenizer 的涵式去將每一筆資料轉成 20000 個 features（因為我只使用 20000 個字），然後丟到 4 層 DNN 去做 training，架構如下：

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               10240512
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 65
=================================================================
Total params: 10,413,057
Trainable params: 10,413,057
Non-trainable params: 0
_________________________________________________________________
```

發現 DNN 的參數明顯多很多，但是準確率卻不到 79%，如下圖

其中在訓練的時候一樣使用 0.01 lr 的 adam 去 minimize loss

```
190000/190000 [==============================] - 42s - loss: 0.0907 - acc: 0.9665 - val_loss: 0.8650
 - val_acc: 0.7892
Epoch 11/200
189952/190000 [============================>.] - ETA: 0s - loss: 0.0808 - acc: 0.9701Epoch 00010: va
l_acc did not improve
190000/190000 [==============================] - 42s - loss: 0.0808 - acc: 0.9701 - val_loss: 1.0274
 - val_acc: 0.7845
Epoch 12/200
189952/190000 [============================>.] - ETA: 0s - loss: 0.0756 - acc: 0.9722Epoch 00011: va
l_acc did not improve
190000/190000 [==============================] - 41s - loss: 0.0756 - acc: 0.9722 - val_loss: 0.9387
 - val_acc: 0.7839
Epoch 13/200
189824/190000 [============================>.] - ETA: 0s - loss: 0.0687 - acc: 0.9746Epoch 00012: va
l_acc did not improve
190000/190000 [==============================] - 41s - loss: 0.0687 - acc: 0.9746 - val_loss: 0.9470
 - val_acc: 0.7822
Epoch 14/200
189824/190000 [============================>.] - ETA: 0s - loss: 0.0645 - acc: 0.9762Epoch 00013: va
l_acc did not improve
190000/190000 [==============================] - 41s - loss: 0.0645 - acc: 0.9762 - val_loss: 0.9776
 - val_acc: 0.7837
Epoch 15/200
189824/190000 [============================>.] - ETA: 0s - loss: 0.0595 - acc: 0.9784Epoch 00014: va
l_acc did not improve
190000/190000 [==============================] - 41s - loss: 0.0596 - acc: 0.9784 - val_loss: 1.0122
 - val_acc: 0.7846
Epoch 16/200
 28672/190000 [===>..........................] - ETA: 34s - loss: 0.0497 - acc: 0.9814
[2] 0:yujheli@localhost:~/ML2017/HW4*                  "localhost.localdomain" 18:04 07-Dec-17
```

從上圖可以看出 training acc 已經到 0.984 時 validation 還在 0.789。

3. 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。

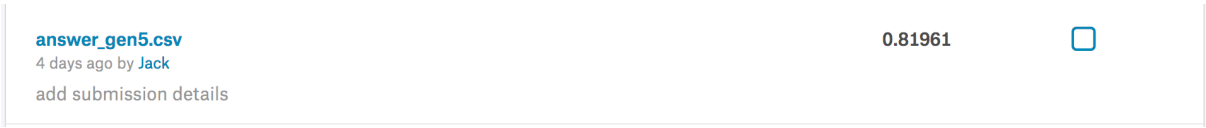
	"today is a good day, but it is hot"	"today is hot, but it is a good day"
BOW	0.66658622	0.66658622
RNN	0.16137353	0.97901684

可以看出 BOW 的方法無法看出時間上的關係，畢竟統計次數所 represent 的 feature 還是沒有 word embedding+RNN 來得有效。

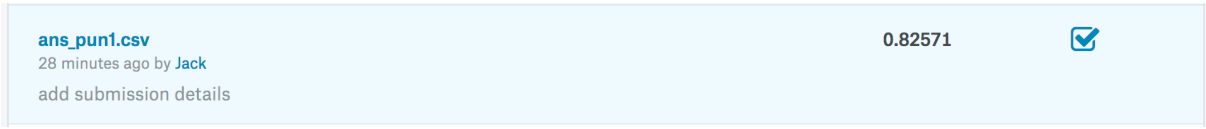
4. 請比較"有無"包含標點符號兩種不同 tokenize 的方式，並討論兩者對準確率的影響。

我使用與第一題一樣的架構去改變 tokenizer 的 filter，將所有的標點符號就去除，但是準確率如下：

沒有標點符號：



有標點符號：



可以發現保留標點符號的 model 真的有比較高的準確率

5. 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。

我的標記方法是對 predict 出來的 score 設定一個 threshold，大於 0.85 給 1，小於 0.15 給 0，但是發現做出來的效果沒有比較好，training 的截圖如下，最後總共有 602410 的 sample 進入第二階段的 training，但是 acc 只有 0.79……