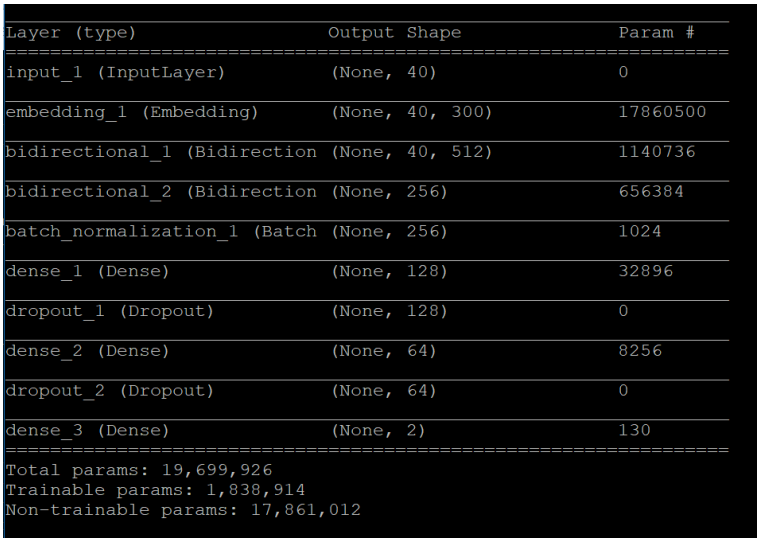
學號：B04505026 系級： 工海三 姓名：蔡仲閔

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:)
   答：



Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 40)	0
embedding_1 (Embedding)	(None, 40, 300)	17860500
bidirectional_1 (Bidirection	(None, 40, 512)	1140736
bidirectional_2 (Bidirection	(None, 256)	656384
batch_normalization_1 (Batch	(None, 256)	1024
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, 2)	130

Total params: 19,699,926
Trainable params: 1,838,914
Non-trainable params: 17,861,012

Param	*optimizer*	*epoch*	*loss function*
Value	Adam	20	categorical_crossentropy

RNN Model

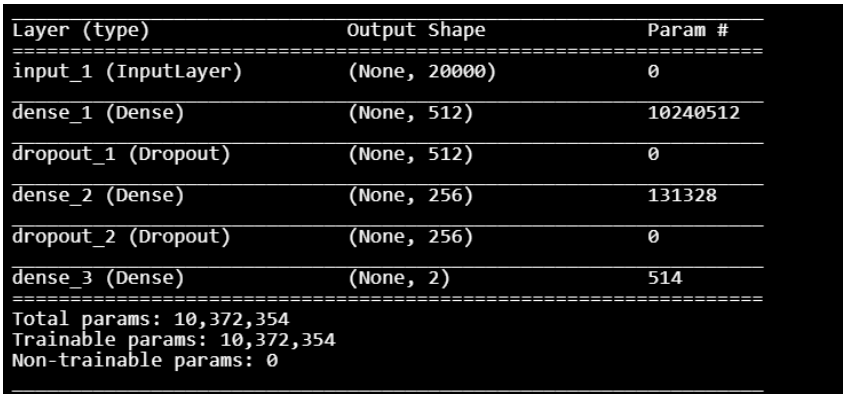
這次作業最後使用的是兩層架構的 Bidirectional LSTM model，total params 有 19M 個，其中有 17.8M 個是 pre-train 的 Embedding layer，並在後面接上三層的 DNN layer，在實驗過程中發現使用 Bidirectional layer 可以提升 performance 但相對會花比較多的時間，而後面的 DNN layer 對於最後的結過影響則較小。而實際上對於結果影響較大的則是在 Embedding layer，這次使用的是 Gensim 這個套件，並使用 Skip-gram 演算法，比起直接放到 model 中直接 train，使用 skip-gram 來 pre-train Embedding layer 可以上升約兩個百分點的準確率，而 window size 的增加也可以提升準確率，然而也會造成花費時間提高。最後在單一 model 的情況下可以在 Public Score 上達到 0.824 的準確率。

Param	*window*	*sg*	*size*
Value	10	Skip-gram	60000

Gensim Word2Vector Model

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:)

答：



Param	*optimizer*	*epoch*	*loss function*
Value	Adam	20	categorical_crossentropy

DNN Model

BOW model 在透過 BOW 取 feature 後使用三層 fully connected layer 進行訓練，這邊使用 Keras.Tokenizer 的 text_to_matrix 實作，並選用 count mode。而 DNN model 相對 RNN model 來說訓練時間相當短，一個 epoch 約只需要 1/10 的時間，但最後的準確度也不盡理想，在類似條件下只能得到約 0.7548 的準確率。

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:)

答：

	today is a good day, but it is hot	**today is hot, but it is a good day**
RNN	(0.6280, 0.3719)	(0.0145, 0.9854)
BOW	(0.4789, 0.5210)	(0.4541, 0.5458)

從這上表可以看出 BOW 有明顯的缺點，在這語序互換的情況下，在 BOW 的演算法統計起來結果是相似的，因此他無法判斷前後文造成的影響。而 RNN 則是其擁有記憶性的特色，讓在這個 task 中成功判斷出這兩句話的差異。

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。

   答：

Tokenize	Public Score
包含標點符號	0.82481
不含標點符號	0.81692

   Tokenize 的調整我們可以從 keras 的 Tokenizer 的 filter 更改，他預設為 `filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~\t\n'`，基本上已經濾掉所有標點符號，因此我們只要把 filter 值改掉後再做一次實驗即可。意外的加入標點符號後，accuracy 有明顯的上升，本來容易被認為是雜訊的標點符號竟然增進了最後的表現，實際原因不得而知，我們只能推斷也許標點符號帶來時序上的差異具有其效果，或是在這個 task 中也許驚嘆號之類的符號對於情緒判斷有相當大的影響力。

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:)

   答：在 semi-supervised 的部分我使用上面做出的 best model 為基礎進行 predict，並設定 threshold 為 0.1，將預測過後符合條件的加入 train set，並進行最多 10 次的疊代。在這個實驗中相當的耗時，一方面因為 train data 不斷的增加，一方面 train 的次數也相當多次，但令人遺憾的在結果上似乎並不如想像的有效，從 0.81692 進步到 0.81926，僅有微小的進步。