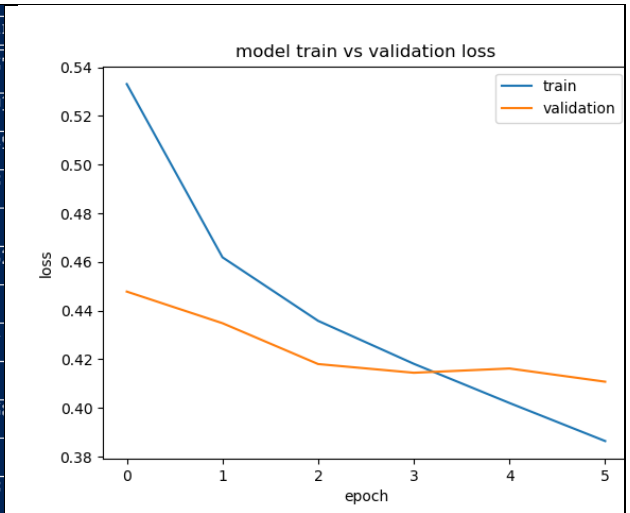
學號：R06921058 系級： 電機碩一 姓名：方浩宇

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？
答：

Layer (type)	Output Shape	Para
embedding_2 (Embedding)	(None, None, 200)	2805
bidirectional_1 (Bidirection	(None, None, 512)	9359
bidirectional_2 (Bidirection	(None, 512)	1574
batch_normalization_1 (Batch	(None, 512)	2048
dropout_1 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	1313
leaky_re_lu_1 (LeakyReLU)	(None, 256)	0
batch_normalization_2 (Batch	(None, 256)	1024
dropout_2 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 512)	1315
leaky_re_lu_2 (LeakyReLU)	(None, 512)	0
batch_normalization_3 (Batch	(None, 512)	2048
dropout_3 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 1024)	5253
leaky_re_lu_3 (LeakyReLU)	(None, 1024)	0
batch_normalization_4 (Batch	(None, 1024)	4096
dropout_4 (Dropout)	(None, 1024)	0
dense_4 (Dense)	(None, 2)	2050

Total params: 31,367,338
Trainable params: 3,305,730
Non-trainable params: 28,061,608



預處理：

(1) 把簡寫取代：例如 can’t 取代為 can not

(2) 連續出現的字母和符號都改為單一個：例如 hhhheeeeyyyyyy!!!!!改為 hey!

(3) 把一些縮寫改為原本的字：例如 2b 改為 to be

(4) 使用 gensim 的 stemmer 作 stemming，減少 token 數量

模型架構：

圖左為這次的 RNN 模型架構，一開始先用兩層 Bidirectional LSTM，然後 256、512、1024，三層 Dense，最後一層 Dense(2)使用 sigmoid 作輸出，其中每

一層 Dense 之間都有 BatchNormalization 和 Dropout。Activate Function 是使用 alpha 為 0.04 的 LeakyRelu

訓練過程：右圖為訓練時的 Loss 變化，不過這個 Loss 變化非最佳 Model 的，也沒有使用 Semi-supervised，這次的 Model 大部分都在第三個 Epoch 左右時達到最佳，超過之後幾乎都是 Overfitting。

output_10_Anotherreplace_semi_log9961_testReproduce.csv a day ago by r06921058_>.O add submission details	0.82395	0.82597	☐
output_004_Anotherreplace_ensemble_6Model.csv 19 days ago by r06921058_>.O add submission details	0.82948	0.83162	☐

準確率：單一 Model 的 public 準確率(有使用 Semi-supervised)可以到達 0.82597，而使用 10 個 Model 作 Ensemble 的話可以到達 0.83162 的準確率，另外，因為是 Semi-supervise，所以在訓練時 validation 的準確率幾乎都是 0.99 以上。

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？
答：

Layer (type)	Output Shape	Param
dense_1 (Dense)	(None, 512)	10245
activation_1 (Activation)	(None, 512)	0
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 2)	1026
activation_2 (Activation)	(None, 2)	0

Total params: 1,025,538
Trainable params: 1,025,538
Non-trainable params: 0

預處理：基本上同 RNN，不過是改成使用 keras 的 tokenizer.text_to_matrix，而我設定的維度是 2000 維，因此是使用出現頻率前 2000 高的字，設定 2000 是因為太大的話會 memory error。

模型架構：這次是簡單的一層 Dense 然後接 Relu 後 Dropout0.5，就直接接輸出層(dense(2),sigmoid)了。

準確率：

這個準確率是有使用 Semi-supervised 的，Replace 的方法都和 RNN 的一樣，準確率比起上面的 RNN 低非常多，不過可能是因為我模型作非常淺，但是因為 BOW 比起上面的 RNN 會流失一些可以判斷的資訊，因此即使把模型加深也會比 RNN 低很多。

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。
   答：

BOW	RNN
[0.65105045 0.34887126] [0.5133269 0.48659867]] Answer	[[0.90214014 0.09811708] [0.1989746 0.8012115]] Answer

兩張圖分別為 BOW 與 RNN 對這兩句的分數，上方為 today is a good day, but it is hot，下方為 today is hot, but it is a good day，左方為負面的情緒分數，右方為正面的情緒分數。

可以發現 BOW 的預測兩具都是偏負面，然而差別並不大，有可能是因為我 BOW 只有 2000 維，偏負面可能是因為 but 這個字出現時大部分為負面。而 RNN 預測第一個為負面，第二個為正面，可能是由於 Good 為一個正面的詞，而 But 為轉折，因此第一個先 Good 再 but 所以會預測為負面，第二個先 but 再 good 因此預測為正面。

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：無標點符號的方法是在 Replace 中去掉標點符號(例如把” , ”取代為””)，而去掉標點符號之後(其餘皆跟沒去掉的一樣)，準確率為 0.82071，而沒有去掉標點符號的有 0.82597。可以發現標點符號的確能夠幫助預測的結果，我猜測是類似有使用驚嘆號之類的句子通常比較容易是正面，而逗號也有助於分隔前後文，因此使用標點符號會讓結果比較好。

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。
   答：我使用的方法是先建立一個 RNN Model(準確率為 0.821)，然後用這個

Model 去 Predict 那些未標記的資料，然後經過我設定的門檻(正面和負面情緒差別要大於 0.4，而任一種的預測值要大於 0.8)保留比較有信心的資料，在把這些資料做標記之後儲存成跟 TrainData 一樣的格式，然後在訓練時讀取兩個 TrainData 和 UnlabeledData 的檔案後合併直接 Train。使用 Semi-supervised 之後可以讓模型的準確率從 0.821 上升到 0.825，Ensemble 後甚至可以到 0.831，有非常顯著的影響。

6. Reference:
   https://github.com/thtang/ML2017FALL/tree/master/hw4
   https://github.com/idea7766/ML2017FALL/tree/master/hw4
   https://github.com/keras-team/keras/blob/master/examples/reuters_mlp.py