Machine Learning HW6 Report

學號：R07942115　系級：電信碩一　　姓名：謝硯澤

1. (1%) 請說明你實作之 RNN 模型架構及使用的 word embedding 方法，回報模型的正確率並繪出訓練曲線*

RNN Model Architecture:

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, None, 250)	7371000
bidirectional_3 (Bidirection	(None, None, 512)	778752
bidirectional_4 (Bidirection	(None, None, 512)	1181184
time_distributed_2 (TimeDist	(None, None, 256)	131328
batch_normalization_3 (Batch	(None, None, 256)	1024
dropout_3 (Dropout)	(None, None, 256)	0
dense_5 (Dense)	(None, None, 16)	4112
batch_normalization_4 (Batch	(None, None, 16)	64
dropout_4 (Dropout)	(None, None, 16)	0
dense_6 (Dense)	(None, None, 2)	34

Total params: 9,467,498
Trainable params: 2,095,954
Non-trainable params: 7,371,544

<< loss = 'binary_crossentropy', optimizer = 'adam' >>

Word Embedding 實作方法:

（1）首先，使用套件 jieba 分割句子。

Ex: “人生短短幾個秋” => “人生”, “短短”, “幾個”, “秋”

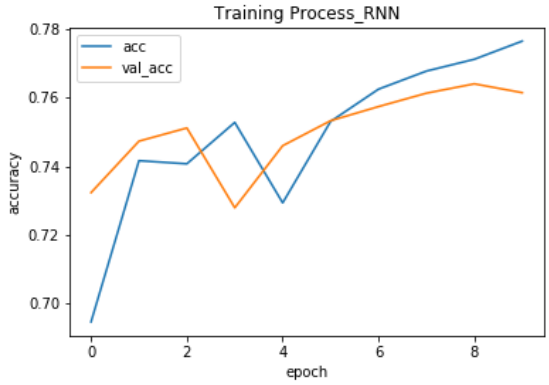
（2）再來，使用 gensim 套件中的 word2vec，訓練一個 word-to-vector model

word2vec.Word2Vec(seg_train_x, size=250, window=5, min_count=5, workers=4, iter=10, sg=1)

RNN 模型正確率：

Public score	0.75800
Private score	0.74950

使用 RNN 架構訓練 10 個 epoch 的訓練曲線:



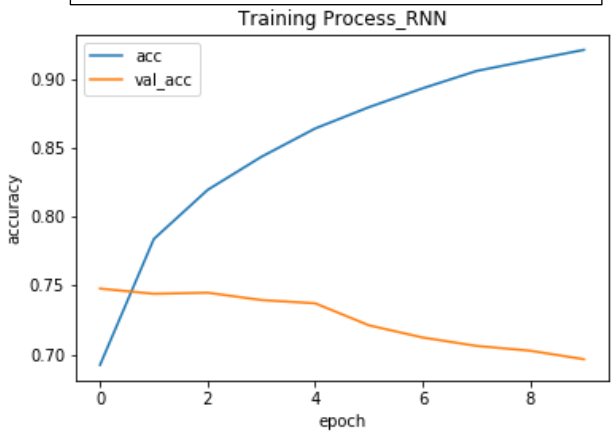
2. (1%) 請實作 BOW+DNN 模型，敘述你的模型架構，回報模型的正確率並繪出訓練曲線*。

BOW+DNN Model Architecture:

Layer (type)	Output Shape	Param #
dense_11 (Dense)	(None, 1, 512)	15095808
leaky_re_lu_9 (LeakyReLU)	(None, 1, 512)	0
batch_normalization_9 (Batch	(None, 1, 512)	2048
dropout_9 (Dropout)	(None, 1, 512)	0
dense_12 (Dense)	(None, 1, 128)	65664
leaky_re_lu_10 (LeakyReLU)	(None, 1, 128)	0
batch_normalization_10 (Batc	(None, 1, 128)	512
dropout_10 (Dropout)	(None, 1, 128)	0
dense_13 (Dense)	(None, 1, 64)	8256
leaky_re_lu_11 (LeakyReLU)	(None, 1, 64)	0
batch_normalization_11 (Batc	(None, 1, 64)	256
dropout_11 (Dropout)	(None, 1, 64)	0
dense_14 (Dense)	(None, 1, 32)	2080
leaky_re_lu_12 (LeakyReLU)	(None, 1, 32)	0
batch_normalization_12 (Batc	(None, 1, 32)	128
dropout_12 (Dropout)	(None, 1, 32)	0
dense_15 (Dense)	(None, 1, 2)	66

Total params: 15,174,818
Trainable params: 15,173,346
Non-trainable params: 1,472

使用 BOW+DNN 訓練 10 個 epoch 的 training accuracy& validation accuracy:



<< loss = 'binary_crossentropy', optimizer = 'adam' >>

BOW+DNN 模型正確率：

Public score	0.52590
Private score	0.52540

備註：BOW+DNN 的 performance 沒有很好，幾乎是用猜的，但 training acc.又有到 0.95，可能是 over-fitting 了，因為 valid. Acc.隨著 training acc.升高而下降。

3. (1%) 請敘述你如何 improve performance（preprocess, embedding, 架構等），並解釋為何這些做法可以使模型進步。
   Word embedding model 的訓練並沒有特別改什麼，主要更動的地方是 model 的架構。首先，我有把 word embedding 放在第一層一起進行訓練，再來我加了兩層 GRU(Gated Recurrent Unit)，我也有試過 LSTM 但發現效果差一點且訓練時間好像比較久一點？！而兩層 GRU 都是使用 Bidirectional 的 wrapper（考量到句子前後也具有高度的相關性），助教上課時也有講過疊兩層其實表現就已經很不錯了~我自己也有試過多疊幾層發現 performance 並沒有提升，所以只疊了兩層 GRU。最後則是三層的全連結層，第一層全連階層則使用 TimeDistributed() wrapper。
   << TimeDistributedDense applies a same Dense (fully-connected) operation to every timestep of a 3D tensor. >> 上網查使用 TimeDistributed 可以解決序列預測問題的多對多網路問題~

4. (1%) 請比較不做斷詞 (e.g., 以字為單位) 與有做斷詞，兩種方法實作出來的效果差異，並解釋為何有此差別。
   << 不斷詞做法：以字為單位，訓練一個 word embedding >>

	有斷詞	無斷詞
Public score	0.75800	0.75000
Private score	0.74950	0.74270

不做斷詞的表現差了一點，因為少了詞的結構後，有些語句的意思可能會完全不同了，例如：”啊不就好棒棒”拆成每個單詞後會變成[”啊”, “不”, “就”, “好”, “棒” , “棒”]跟理想情況[”啊”, “不”, “就”,“好棒棒”]的，解讀後的意思可能就完全不一樣了！但其實我用到的斷詞套件，其實也沒有每個斷詞都斷很精準，以至於有無斷詞的分數差異不是特別明顯。

5. (1%) 請比較 RNN 與 BOW 兩種不同 model 對於 "在說別人白痴之前，先想想自己"與"在說別人之前先想想自己，白痴" 這兩句話的分數（model output），並討論造成差異的原因。

	RNN	BOW+DNN
'在說別人白痴之前，先想想自己'	善意：58.73% 惡意：41.27%	善意：48.03% 惡意：51.97%
'在說別人之 前先想想自己，白痴'	善意：47.02% 惡意：52.98%	善意：48.05% 惡意：51.95%

RNN 的 model 兩個句子都判斷正確，而 BOW+DNN 則有一個判斷錯誤。我覺得 BOW+DNN 會判斷錯誤的原因是沒有加入 Bidirectional 的機制進去，學習表現上會比較差，因為前後句子其實是有很大的相關性在裡面，但 model 並沒有學習到。只會從前面的句子來做判斷。