Machine Learning HW6 Report

學號：B05901040　系級：電機三　姓名：蔡松達

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。

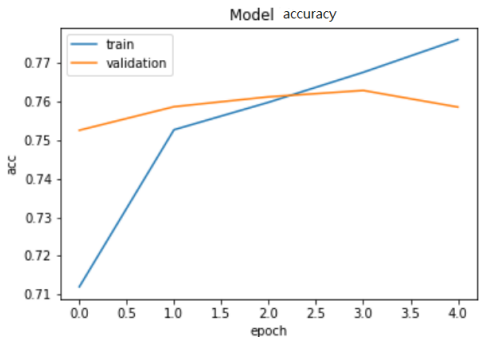
答：

Word embedding 方法：word2vec (size=250, window=5, min_count=5, iter=10, sg=1)，使用 Skip-Gram(sg=1)方式訓練，詞向量的維度為 250(size)，被納入的詞必須至少出現五次(min_count)，預測詞可以向左右看 5 個詞(window)，總共迭代 10 次(iter)，padding length: 200（每筆 data 只取前 200 個詞語）。

模型架構與 epoch vs accuracy：

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, None, 250)	7199750
bidirectional_1 (Bidirection	(None, None, 512)	778752
time_distributed_1 (TimeDist	(None, None, 256)	131328
dropout_1 (Dropout)	(None, None, 256)	0
dense_2 (Dense)	(None, None, 16)	4112
dropout_2 (Dropout)	(None, None, 16)	0
dense_3 (Dense)	(None, None, 2)	34

Total params: 8,113,976
Trainable params: 914,226
Non-trainable params: 7,199,750



第一層使用 embedding layer；第二層使用 Bidirectional 的 GRU(256)；第三層使用 Time_distributed dense layer(256, activation=relu)；第四層與第六層使用 dropout(0.1)；第五層使用 Dense(16, activation=relu)；最終 output 層為 Dense(2)，接上 activation 為 softmax 便能得到結果為 0 或 1 各自的機率，較大者便是預測的結果。(loss='categorical_crossentropy', optimizer='adam', epoch: 5, batch size:100)

正確率：

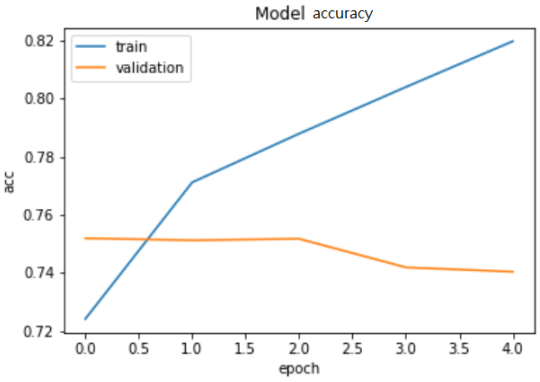
Public accuracy	Private accuracy	Testing average accuracy
0.75750	0.74890	0.75320

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。

答：模型架構與 epoch vs accuracy：

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 500)	5298000
batch_normalization_1 (Batch	(None, 500)	2000
leaky_re_lu_1 (LeakyReLU)	(None, 500)	0
dropout_3 (Dropout)	(None, 500)	0
dense_5 (Dense)	(None, 100)	50100
batch_normalization_2 (Batch	(None, 100)	400
leaky_re_lu_2 (LeakyReLU)	(None, 100)	0
dropout_4 (Dropout)	(None, 100)	0
dense_6 (Dense)	(None, 25)	2525
batch_normalization_3 (Batch	(None, 25)	100
leaky_re_lu_3 (LeakyReLU)	(None, 25)	0
dropout_5 (Dropout)	(None, 25)	0
dense_7 (Dense)	(None, 1)	26

Total params: 5,353,151
Trainable params: 5,351,901
Non-trainable params: 1,250



Word embedding 方法：word2vec (size=250, window=5, min_count=20, iter=10, sg=1)。模型中 Leaky Relu(alpha=0.3), Dropout rate 皆為 0.2，最後一層 output 為 1 接上 sigmoid 層，使用 loss='binary_crossentropy', optimizer='adam'，epoch: 5，batch size:100。

正確率：

Public accuracy	Private accuracy	Testing average accuracy
0.74100	0.73280	0.73690

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。

答：我試著將架構中的 GRU 由一層換成兩層，以及試著將 GRU 換成 LSTM 進行實作得到下列結果：

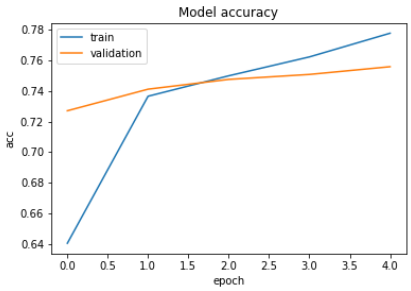
方式	Public score	Private score	Average score
第一題架構	0.75750	0.74890	0.75320
兩層 GRU	0.75430	0.75570	0.75500
GRU 換成 LSTM	0.75580	0.75820	0.75700
三層做 ensemble	0.75950	0.75610	0.75780

可以看出只使用一層的 LSTM 之結果能夠比起一層或兩層 GRU 來的好，而兩層的 GRU 也比起只有一層來的好，原因可能是因為 LSTM 使用的閘數比較多，GRU 只使用 update gate；兩層比一層好可能是架構上較能夠學得起來，但其實進步的幅度皆不算非常大；若是希望可以得到進一步的 improvement，使用 ensemble 也是不錯的方式，可以結合各模型的力量得到預測結果。

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。

下面兩者都是根據第一題的 RNN 架構訓練而成，基本上兩者之間的效果幾乎沒有太大差別，主要原因是因為每個字基本上都能表達出一定的意義，尤其是針對是不是罵人這件事情，只要是特定的字眼出現在特定的位置的時候就得以順利辨別，不過以詞彙來進行訓練仍舊可以得到稍微好一點點的結果。

不做斷詞的訓練過程如下：



方式	Public score	Private score	Average score
不做斷詞	0.75290	0.75100	0.75195
有做斷詞	0.75750	0.74890	0.75320

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。

	在說別人白痴之前，先想想自己	在說別人之前先想想自己，白痴
RNN	0.03（預測值為 0）	0.985（預測值為 1）
BOW	0.62（預測值為 1）	0.59（預測值為 1）

此處分別使用第一題以及第二題模型進行預測，RNN 之結果為前者預測值為 0（0.03）；而後者預測值為 1（0.985），可見 RNN 幾乎能夠精準預測出結果，主要原因就是因為此處使用到 Bidirectional 的 GRU，能夠觀察前後文，比較能得知語意；而 BOW 預測的狀況則不理想，由於無法判斷上下文的關聯性，所以兩句話預測值相當接近，即便兩句話的意義完全不同，卻因為兩句話所用的詞語完全相同（只是順序不同），預測值相當接近，且機率都落在 0.6 左右，也就是並不是很明確是 0 或是 1，足可見使用 BOW 對於文字調換造成語意不同會有相當大的問題存在。