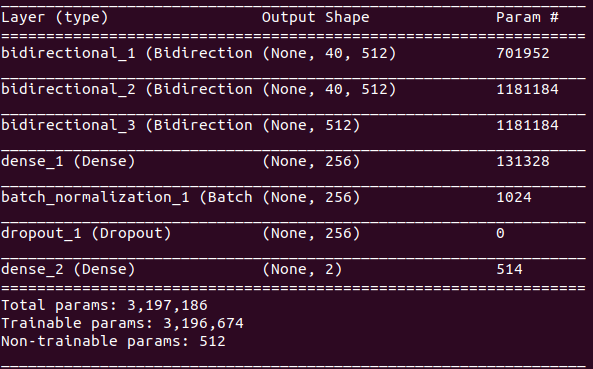
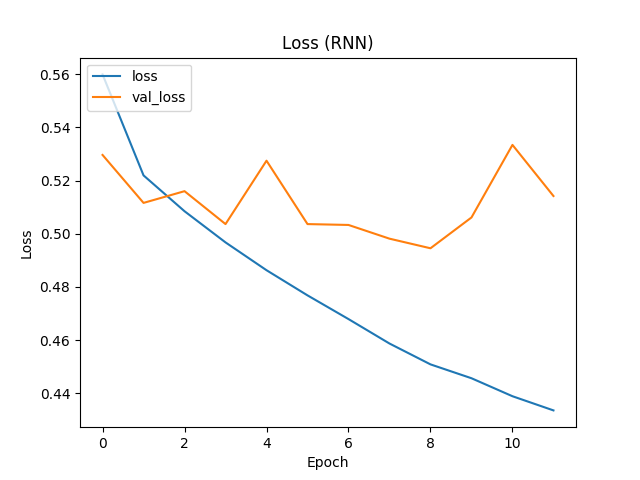
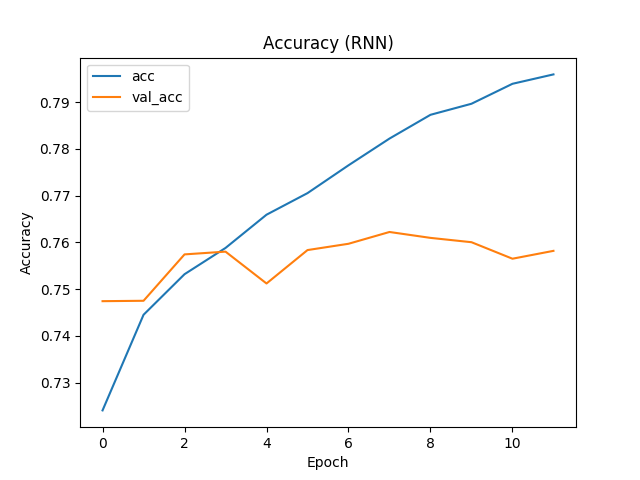
Machine Learning HW6 Report

學號：R07943107 系級：電子所碩一 姓名：徐晨皓

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\*。**
2. RNN模型架構及word embedding方法

Word embedding使用gensim的Word2Vec，細節在問題3解釋。RNN模型使用三層Bidirectional GRU layers再接著兩層dense layers。激活函數方面，GRU 使用tanh，dense使用relu，最後的output layer使用sigmoid。Dropout為0.3；epochs為12。Optimizer使用adam。

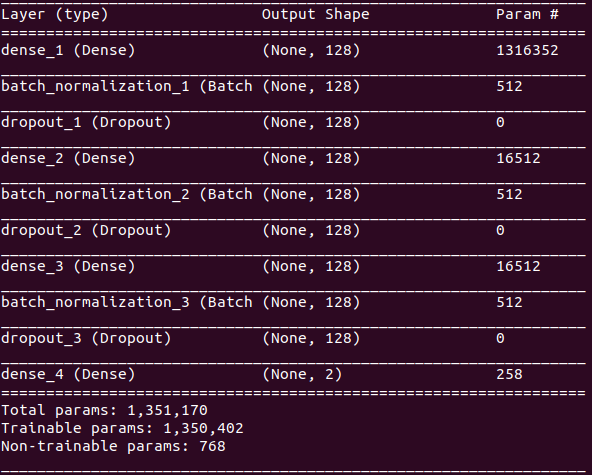
1. 模型正確率
2. Private score: 0.75980
3. Public score: 0.76450
4. 訓練曲線



(取10%的training data當作validation set)

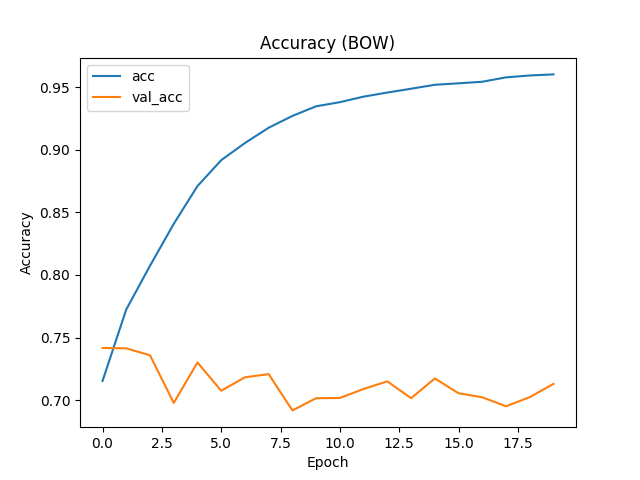
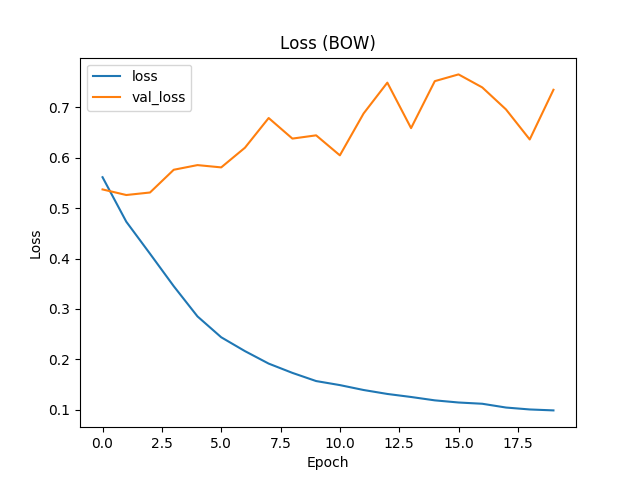
由上二圖可以觀察到在訓練過程中：

1. RNN model的training accuracy可以達到約0.81。但validation accuracy只落在0.75和0.76之間。
2. RNN model的training loss下降至約0.43。validation loss只落在0.50和0.54之間。
3. **(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\*。**
4. BOW+DNN 模型架構

BOW的字典是使用word2vec蒐集而成。為避免記憶體不足的問題，將min\_count設為20，最後word dictionary共蒐集10283個字詞。

DNN結構為三層dense layers，每層的cell數量為128。Iteration數為20；dropout為0.2。Epochs設為20。

1. 模型正確率
2. Private score: 0.71030
3. Public score: 0.71270
4. 訓練曲線



(取10%的training data當作validation set)

由上二圖可以觀察到在訓練過程中：

1. DNN model的training accuracy可以達到約0.96。但validation accuracy卻在0.68和0.74之間震盪。
2. DNN model的training loss下降至約0.10。但validation loss卻逐漸上升並劇烈震盪，可能是為了讓model更fit training data造成validation loss增加。
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。**
4. Preprocess

在preprocess中，基本想法是將留言裡不影響判斷惡意性的字詞去除。由於留言裡有Bxx與bxx對於各樓的回覆標記，這些標記並不影響留言的惡意性，故將這些字詞移除。原以為表情符號可以被移除，但經過實驗發現表情符號可以幫助判斷留言的惡意性，例如有一則留言只有一個比中指的符號，若將表情符號移除就無法判斷此留言的惡意。

1. Embedding

使用gensim的Word2Vec實作word embedding。參數為size=200, window=5, min\_count=3, iter=30。每則留言最大的word vectors數量為40。每個vector都有做normalization ()，從實驗發現若不做normalization，training accuracy只能落在0.7300~0.7400，但做完normalization後可以進步到0.7500以上。原因可能為未經normalized的word vector的數值差異可能很大，造成model不容易fit training data。

1. Ensemble

此方法是幫助performance進步最大的方法。共使用11個models，每個model皆隨機選取90%的data為training set，剩下的10%為validation set。這些model個別的public score基本上落在0.7520~0.7580之間。Ensemble方式為全部model的weight都相同來進行投票，以獲得最多票的label作為output label。以這樣多個model ensemble的方法可以避免單一模型的overfitting，進而改善performance，達到0.76450的public score。

1. **(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。**
2. 有做斷詞
3. Private score: 0.75300
4. Public score: 0.75520

(只使用單一RNN model)

1. 不做斷詞
2. Private score: 0.74080
3. Public score: 0.74830
4. 差異原因

由上面結果，可以發現不做斷詞的準確率較有做斷詞的低。可能是因為如果不做斷詞，雖然字的順序性還存在，但是一些詞的意義可能會變得模糊，使得RNN會判斷錯誤。

1. **(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。**
2. RNN分數
3. "在說別人白痴之前，先想想自己"：
4. "在說別人之前先想想自己，白痴"：

(只使用單一RNN model)

1. BOW分數
2. "在說別人白痴之前，先想想自己"：
3. "在說別人之前先想想自己，白痴"：
4. 差異原因

BOW蒐集到的這兩句話的字詞為

。

由於BOW並不會考慮字詞的先後順序，因此這兩則留言對於BOW來說，是完全一模一樣的input，導致BOW+DNN對於這兩句話的output是一模一樣的。另外，可能是因為有‘白癡’這個字詞，因此BOW+DNN model判定此二則留言為惡意留言。

由於RNN會考慮字詞的先後順序，因此儘管這兩則留言所包含的字詞是一樣的，但因順序不同，RNN會將這兩句話視為不同的input。從RNN output結果，"在說別人白痴之前，先想想自己"應為非惡意留言，而"在說別人之前先想想自己，白痴"為惡意留言，應為正確的。