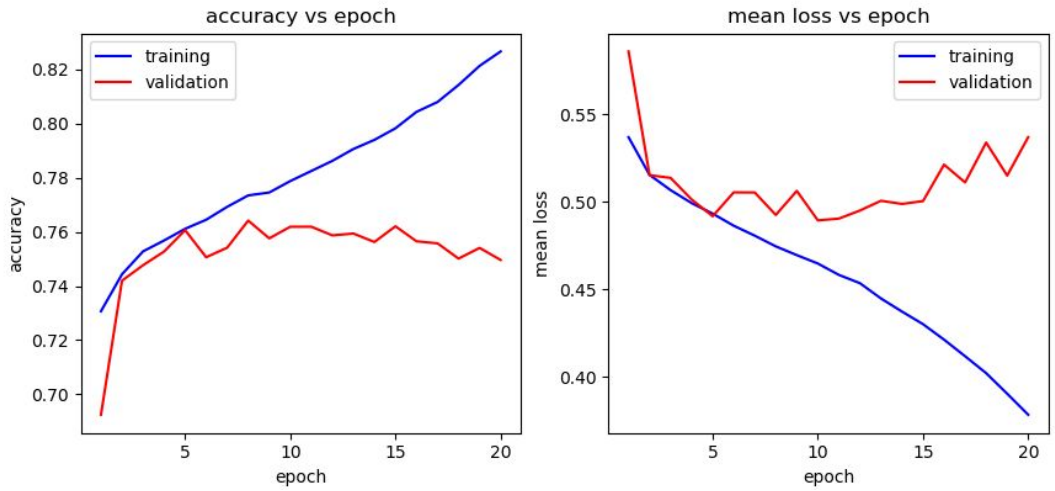
Machine Learning HW6 Report

學號：R07922004　系級：資工碩一　　姓名：吳星耀

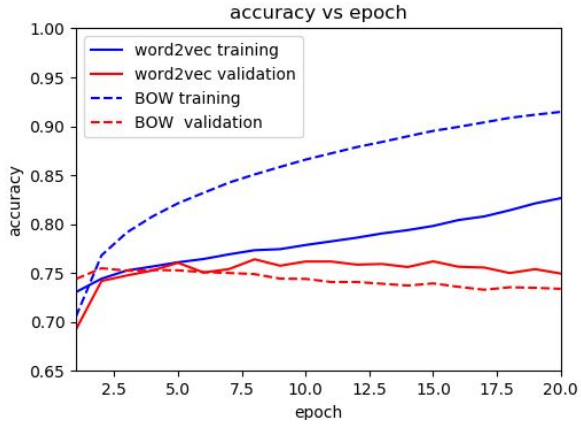
1. (1%) 請說明你實作之 RNN 模型架構及使用的 word embedding 方法，回報模型的正確率並繪出訓練曲線*

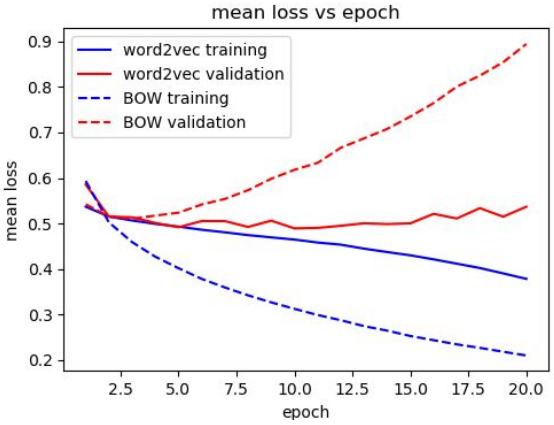
   我把所有句子先用jieba切出詞，並且每句話都只取前128個詞作為RNN的input。模型使用兩層bidirectional的LSTM，hidden layer大小為128，dropuout比例為0.3。LSTM後接兩層fully connected layer(weight數目分別為256*128、128*1)，中間以ReLu連接，並且LSTM與fully connected layer間跟兩個fully connected layer之間都有作batchnorm與dropout(0.3)，最後再接一個Sigmoid後輸出。我用gensim套件的word2vec實作word embedding，我將training與testing set裡面的所有句子都用jieba來切成詞，並且只取出現頻率超過五次的詞作為input，用skip-gram方法來算出每個詞對應的向量，向量維度取150。這個模型在public testing set達到0.76110，private達到0.76030的準確度。下圖為訓練曲線：



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*。

   我仿照我的LSTM模型來實作BOW+DNN模型，只取出現次數5次以上的詞，因此將接近十二萬的向量維度縮減為31429，每個向量維度的大小代表該詞在該句出現的次數。我的DNN架構為：fully connected layer (weight數目為31429 * 22)、ReLu、比例為0.3的Dropout、fully connected layer (weight數目為22 * 1)、Sigmoid。如此設計模型的原因是為了要讓參數數目與我的RNN模型接近。這個模型在public testing set達到0.75660，private testing set則達到0.75320的準確度，略低於RNN。下圖為訓練曲線：





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。

   我原先的架構為兩層hidden size為256的LSTM，我發現在這次作業裡hidden size大小太大validation反而準確度會下降，因此後來將hidden size減小為128，這應該是因為過大的function set導致training太容易overfitting的關係。

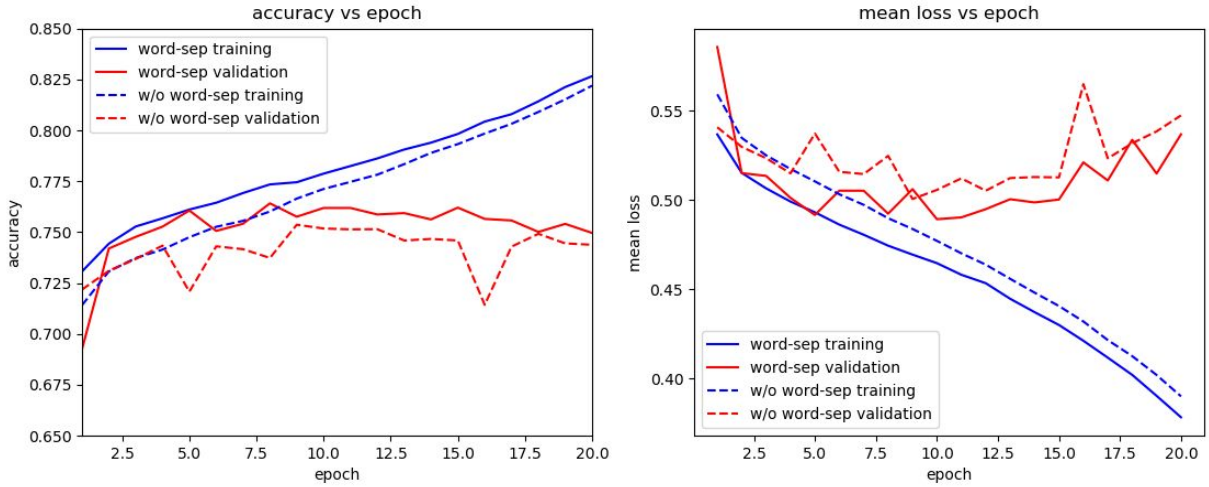
   我接著將embedding的方法從CBOW改成skip-gram，發現準確度大幅度提昇，這可能是因為skip-gram為由每個中心的詞去推測周圍詞，而CBOW是由周圍詞去推測中心詞，因此skip gram每個詞受周圍詞的影響較大，調整的次數也相對較多，因此對一些出現次數少的詞能夠更準確預測。

   我把原本只要出現過一次就當作詞改成如果出現五次才當作一個詞也會讓perfrormance變好，應該是因為能夠避免把一些jieba分錯的字一起放進去train。

   另外在讀字串時，我原本使用pandas的read_csv()來讀取，可是發現它會刪除掉每句話最後的'\n'，而改用其他方法讓它讀到'\n'後，也可以得到較好的performance。這可能是因為我每句話都擷取取固定長度的詞數，因此當機器讀到'\n'時，就帶有『機器不用去預測說後面還會有其他字』的訊息，可能因此更能準確判斷。

   input data從每句話擷取的詞數越多performance越好，這是因為每句話的訊息量變多。但是word embedding的word2vec向量維度取在一定範圍內效果較好，這應該是因為在維度過低時，每個詞就會位置都很接近，但在維度過高時，每個詞又會分得過散，詞之間的相關性又會下降，因此我取150，發現在這個時候的performance最好。

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。



由上圖可以發現不做斷詞的performance較低。原因應該是因為中文的句子組成是以詞為單位，每個『詞』才會有完整的意思，做word2vec的時候也比較能統計出明顯的規律。如果改成用『字』去分的話，word2vec在判斷一個字跟另一個字的關係的同時，就比較難找出每個字之間的對應關係，因為字在不同詞的意義通常會有差距。因此，每個句子的組成單位由詞變成字，造成RNN較難準確預測。

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	0.4015	0.4100
BOW + DNN	0.5713	0.5713

『在說別人之前先想想自己，白痴』和『在說別人白痴之前，先想想自己』兩句話相比，由於前者有直接的罵人，所以應該是較後者更為負面的句子。由RNN的結果可以發現RNN有把兩者之間的相對關係找出來，但是並沒有把『在說別人之前先想想自己，白痴』成功判斷為負面句子，我由training data裡面比對多個有『白痴』的資料發現許多都沒有被判斷為負面發言，這可能是讓RNN對這個詞比較不會有負面判斷的其中一個原因。BOW+DNN將兩個句子都判斷為負面句子，且分數一樣。這是因為Bag of Words只能算出每句的字數，而沒有考慮其連接的關係，因此兩個句子會是一樣的input，自然就是一樣的output。所以它無法成功辨別出『在說別人白痴之前，先想想自己』並不算一個負面句子。