Machine Learning HW6 Report

學號：B06901087 系級：電機二 姓名：翁瑋襄

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模型架構:

   一層embedding

   一層lstm (128 units)

   一層bidirectional lstm (128*2 units)

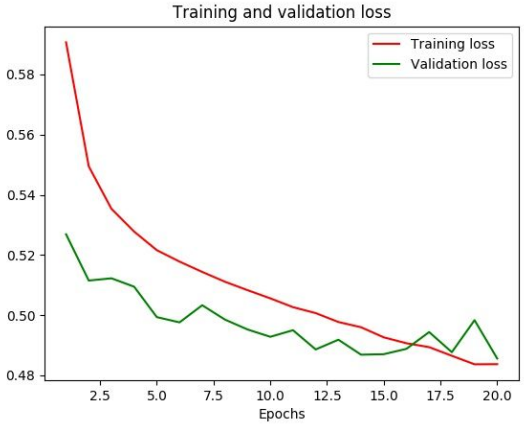
   兩層dense (64 units , 1 units)

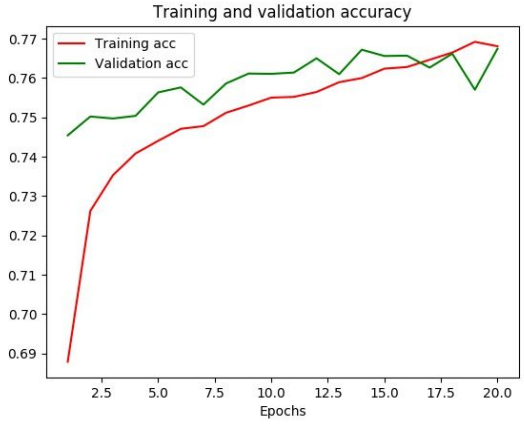
   word embedding的方法:

   先把jieba切好的詞彙train word2vec model， 得到word2index和index2vector的資訊，然後把句子裡前80個word轉成index後作為RNN的input，再讓RNN embedding層依據index2vector轉成vector下去train。

   kaggle上的public score: 0.7619, private score: 0.7620

   本機測試: acc = 0.7681 , val_acc = 0.7674

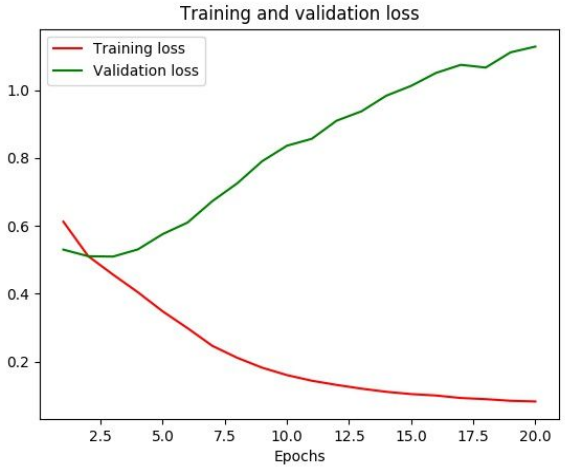


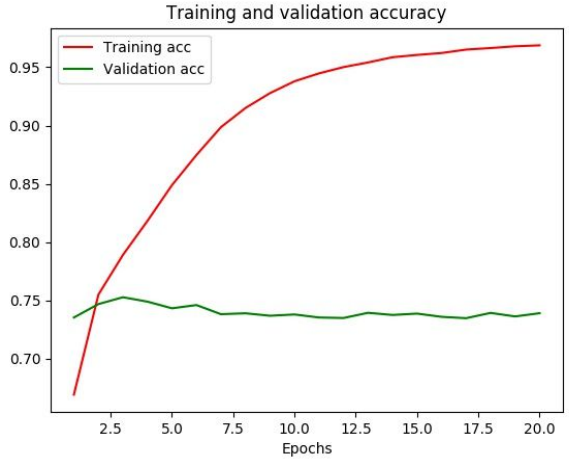


2. (1%) 請實作 BOW+DNN 模型，敘述你的模型架構，回報模型的正確率並繪出訓練曲線*。
   BOW實做方法:
   先把jieba切好的詞彙train word2vec model， 得到word2index，接著開一個跟index一樣長度的list，紀錄每一個index對應的詞出現的次數，接著把這個list作為DNN的input。

   DNN模型架構:
   三層dense (256 units, 64 units, 1 units)

   本機測試: acc = 0.9688 , val_acc = 0.7390



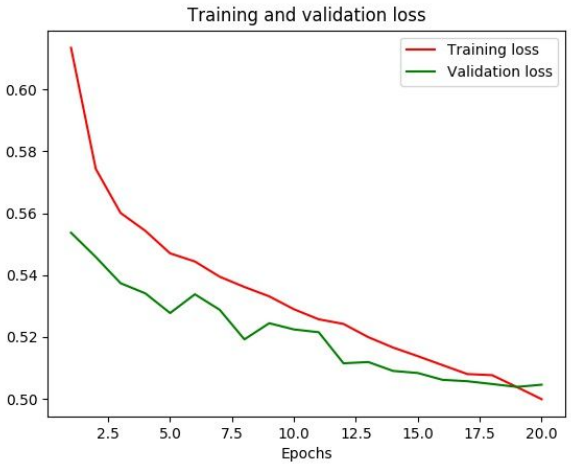


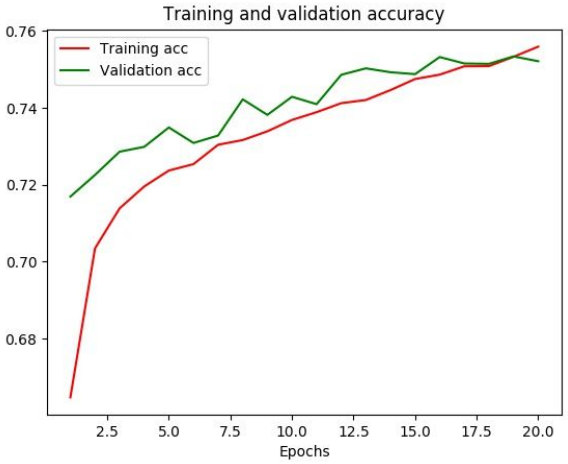
3. (1%) 請敘述你如何 improve performance（preprocess, embedding, 架構等），並解釋為何這些做法可以使模型進步。
   preprocess:
   我的對原本的字串處理包含: 把emoji轉成英文、jieba切完的句子的空格刪除、刪除相鄰重複的詞彙。

embedding的過程:
先把jieba切好的詞彙train word2vec model， 得到word2index和index2vector的資訊，然後把句子裡前80個word轉成index後作為RNN的input，再讓RNN embedding層依據index2vector轉成vector下去train。其中我調了word2vector model的一些參數， 如下: Word2Vec(corpus, size=128,sg=1, window = 5, min_count=3,compute_loss = True, iter = 27, batch_words = 64)
model 架構:
一層embedding
一層lstm (128 units)
一層bidirectional lstm (128*2 units)
兩層dense (64 units , 1 units)

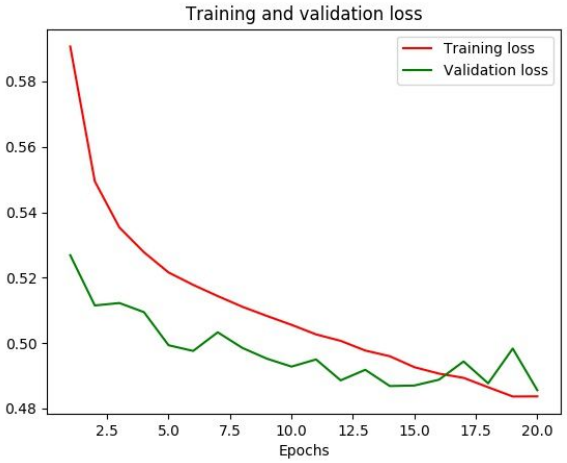
我把所有資料做完jieba的詞彙分割後， 發現長度>80的句子不多，而且通常很長的句子都是因為複製大量重複的資訊，因此我做了句子分割，還有把相鄰重複的詞刪除；我一開始只有用單向lstm train，效果好像不太顯著，於是後來決定使用雙向的試試，結果進步滿多的，我認為是因為分別從兩端判斷對結果預測更有幫助。

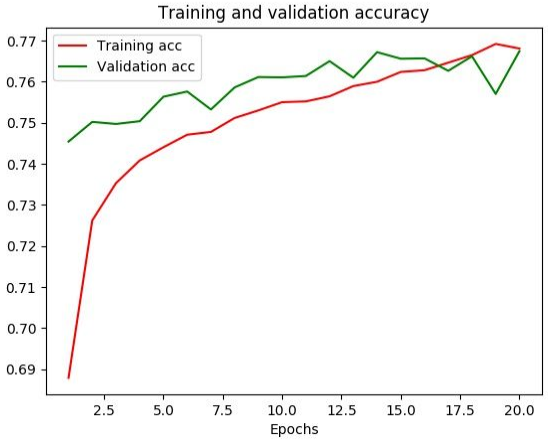
4. (1%) 請比較不做斷詞 (e.g., 以字為單位) 與有做斷詞， 兩種方法實作出來的效果差異，並解釋為何有此差別。
不做斷詞:





有做斷詞:





很明顯可以看到沒做斷詞的版本acc和val_acc難以超過0.76，但有做斷詞的版本其實acc和val_acc滿容易就超過了。我認為差異在於，中文有時候單一個字無法呈現出正面或負面的意思，但是連成詞語後就有直接的意義，因此做斷詞後，能讓model有更準確的判斷。

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。

BOW做出來兩句取sigmoid的output都是0.9617003(惡意)，而RNN的前一句output是0.34480262(非惡意)，後面那句是output是0.5571045(惡意)。

我認為造成這樣的結果是因為BOW只看每一個詞出現的次數，沒有考慮順序，所以做出來的兩句預測結果都是惡意攻擊；而RNN會看每一個詞之間連接的關係，才會造成第一句跟第二句預測出來的結果不同。