Machine Learning HW6 Report

學號：B05901005　系級：電機三　　姓名：賴沂謙

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*

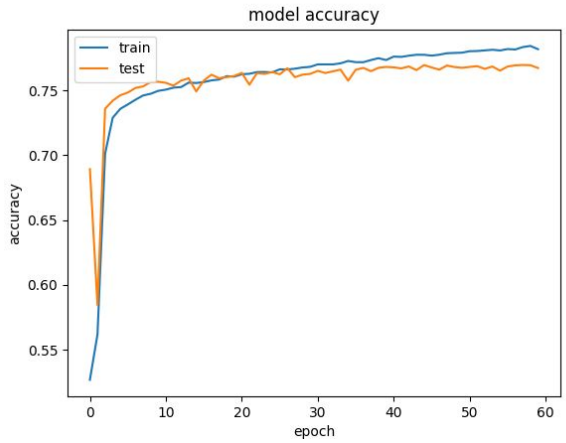
架構：

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, None, 128)	5837696
lstm_1 (LSTM)	(None, None, 128)	131584
lstm_2 (LSTM)	(None, 128)	131584
dense_1 (Dense)	(None, 128)	16512
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 2)	258

Word embedding的方法：使用gensim的word2vec，參數如下：

```
w2v_model = word2vec.Word2Vec(sentences, iter=16, size=128, min_count=3, workers=4, sg=1)
```

結果：



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*。

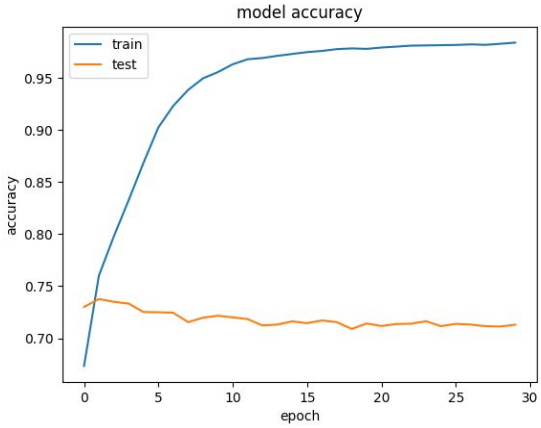
BOW採計出現過40次以上的詞

dimension=6933 (因為BOW實在太吃記憶體了，所以無法全部的詞彙都採計)

架構：

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 256)	1775104
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 2)	258

結果：



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。
   (1) 第一個改善performance的方法是增加word2vec模型的訓練次數，從原本iter=10增加為iter=16，結果就改善許多，後來嘗試使用iter=32。
   (2) 使用雙向的RNN，並且讓return_sequences=True
   (3) 模型的重點應該是在RNN，所以Dense不用加太多層，一層就可以了。

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。

	Public score
以詞為單位	0.75900
以字為單位	0.62849

   沒有斷詞的效果明顯比有斷詞的差很多，應該是因為以字為單位的話，會忽略掉很多詞義上的資訊，進而影響訓練結果。

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。

   非惡意的機率　惡意的機率

   RNN

   “在說別人白痴之前，先想想自己” [0.583348 0.416652]

   “在說別人之前先想想自己，白痴”[0.4437423 0.5562577]

   BOW:

   “在說別人白痴之前，先想想自己”[0.06069471 0.9393053]

"在說別人之前先想想自己，白痴" [0.06069471 0.9393053]
因為RNN會考慮到詞的順序，所以第二句會斷為惡意的留言，第一句不會，但BOW的話，這兩句的vector應該是長一樣的，因為句子的組成一樣，所以都會判斷為惡意留言。