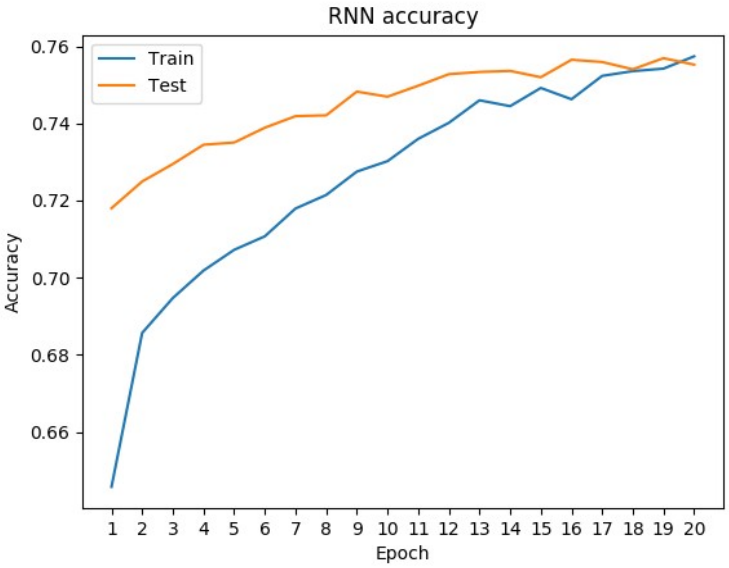
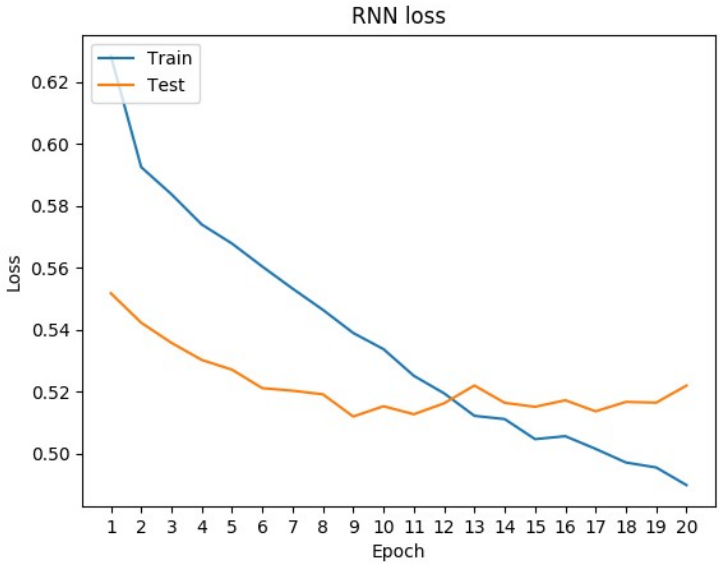
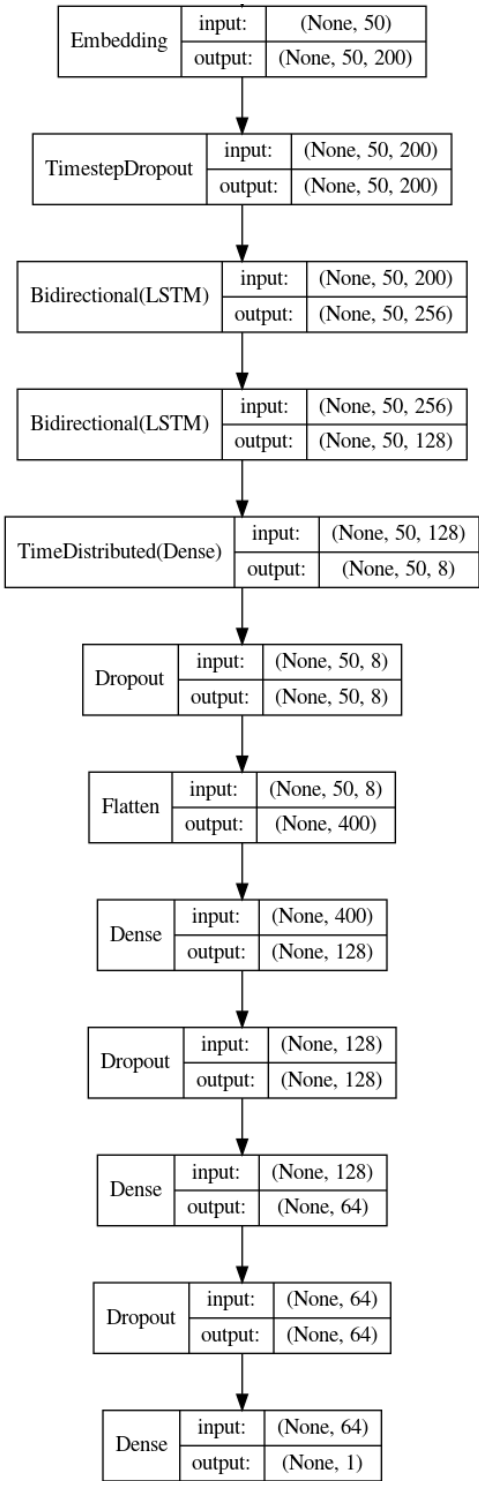
Machine Learning HW6 Report

學號：B07901069 系級：電機一 姓名：劉奇聖

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。







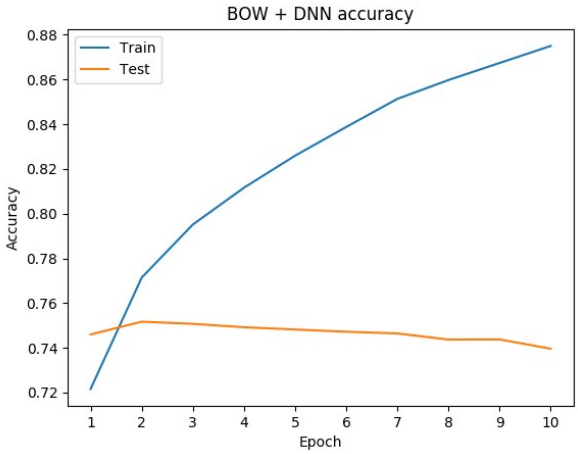
上頁左上圖爲 RNN 訓練時準確率的訓練曲線，左下圖爲 loss 的訓練曲線，右圖爲模型架構。我使用的 word embedding 爲 gensim 的 Word2Vec，embedding 的 size 設爲 200，window 設爲 5，min_count 設爲 6。模型裡第一層 Embedding Layer 的參數以上述 gensim 訓練出來的 embedding matrix 初始化但設爲 trainable 使它能和 RNN 一起被訓練。TimestepDropout 爲一個特殊的 Dropout Layer，用來 drop 掉 Embedding Layer 裡的特定 id 以防其 overfitting，參考自 https://github.com/keras-team/keras/issues/7290。之後是兩層的 Bidirectional LSTM，再加上一層 TimeDistributed Dense 以利用每一個 Time State，最後 Flatten 並通過數層的 Dense 和 Dropout Layer，除了最後一層的 Dense 使用 sigmoid 當作 activation function 之外，其他的 Dense 都使用 swish 作爲 activation function。

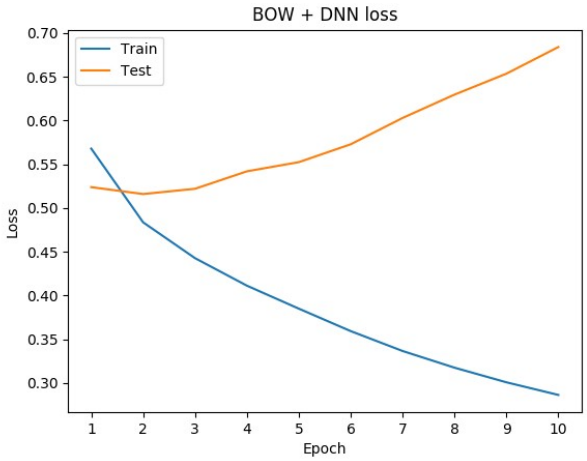
我最好的模型是用上述模型做 bagging，以 50 個 model ensemble 起來的結果，每個模型的訓練過程爲自所有資料(約十二萬筆)中重複選取三十萬筆資料當作 training data，剩餘資料(以機率計算約九千筆)作爲 validation data。訓練到 validation accuracy 達到最高點時即停止訓練。使用的 optimizer 爲 Adam。

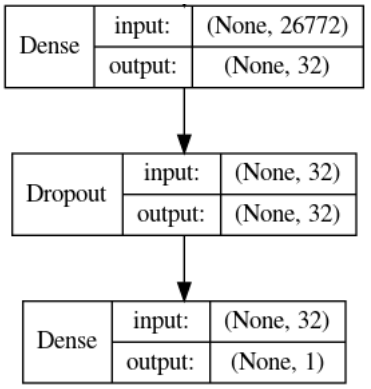
上面呈現的訓練曲線圖沒有使用 bagging，以便和下面問題裡問的模型做比較。以下是我的模型在 kaggle 上的正確率。

	public	private
best model(ensemble 50 models)	0.76030	0.75670
single model in the 50 models	0.75120	0.75450
single model without bagging(方便與下面問題比較)	0.75520	0.75460

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。







上頁的 3 個圖從左至右依序是 BOW+DNN 訓練時準確率的訓練曲線、loss 的訓練曲線及模型架構。因爲發現很容易 overfitting，我選用了一個較小的 model，但由圖可看出仍在第二個 epoch 後開始 overfitting。於是我將第二個 epoch 的參數抽出當作訓練結果，預測 test_x.csv 上傳 kaggle 得到了 public 0.75580，private 0.75090 的分數，和單個 RNN 模型的表現相近。input dimension 26772 這個數字是爲了和 RNN 比較所得的，在 RNN 的 word embedding 裡將 min_count 設爲 6 後 word 的數量總共爲 26771，因此在做 bag of word 的時候多使用一個維度代表 oov(out of vocabulary，即不在 word vector 裡的字)的數量。

3. (1%) 請敘述你如何 improve performance（preprocess, embedding, 架構等），並解釋爲何這些做法可以使模型進步。

我原本做了很多的 preprocess，目的是爲了減少 token 的數量，例如將所有符號移除，只留中文、英文和數字，將英文全部變成小寫，將 B07、B168 此類的詞全部移除，再把所有連續出現的字或詞變成一個，最後將所有空白去除。後來我減少 preprocess，只將英文變成小寫、 將 B07、B168 此類的詞全部移除及將所有空白去除，model 的表現反而變好，推測是因爲在這個 dataset 裡標點符號和表情符號也可傳達一定程度的情緒，故將它們全部移除反而會使整體表現下降。

我原本將 embedding 設爲 untrainable，後來將其改成 trainable 發現正確率提升，但在一個 epoch 後便會 overfitting，因此再加上一層 TimestepDropout，整體正確率因而提升，推測是因爲讓 embedding 和 RNN 一起訓練可以讓 embedding 學到更多字和字的關係，甚至讓惡意的詞和惡意的詞的向量更接近，因此表現提升。

在我原本的模型架構裡我沒有加 Bidirectional 和 TimeDistributed 這兩個 layer wrapper，加了之後發現表現變好，推測是 Bidirectional 讓模型可以考慮句子倒過來的情形、TimeDistributed 可以讓模型考慮各個 Time State 的輸出而非最後一個單一 state，有了更多面向的資訊使模型可以學得更好。

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

	public	private
不做斷詞	0.73850	0.73880
有做斷詞	0.75520	0.75460

上面的表格是不做斷詞和有做斷詞的 RNN 上傳 kaggle 後的結果，這兩個模型的訓練過程完全相同(20 個 epoch)，word embedding 設置的參數也完全相同 (min_count = 6)，但不做斷詞的模型在訓練時正確率上升緩慢，且最終的正確率也較有做斷詞的模型低，推測是不做斷詞直接以字的等級做訓練會少了很多語義，模型也因而學不到太多的資訊。

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。

	第一句話	第二句話
RNN	0.57105	0.67539
BOW	0.56502	0.56502

上面的表格是 RNN 和 BOW 對於第一句話和第二句話的輸出，數值代表爲惡意留言的機率。對於 BOW 模型而言，因爲組成這兩句話的詞完全相同，預測的機率也都相同；對於 RNN 而言，雖然兩句話爲惡意留言的機率仍都大於 0.5，第二句話爲惡意留言的機率卻較第一句話高，造成此種差異的原因是 RNN 會因爲詞的輸入順序不同而產生不同的結果，可看出 RNN 有學到某種程度的順序代表語義。