Homework4 Report

系級:資工碩二 學號:R06922134 姓名:葉沛陽

Problem 1. (0.5%) 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 *。(0.5%) 請實作 BOW+DNN 模型,敘述你的模型架構,回報正確率並繪出訓練曲線。

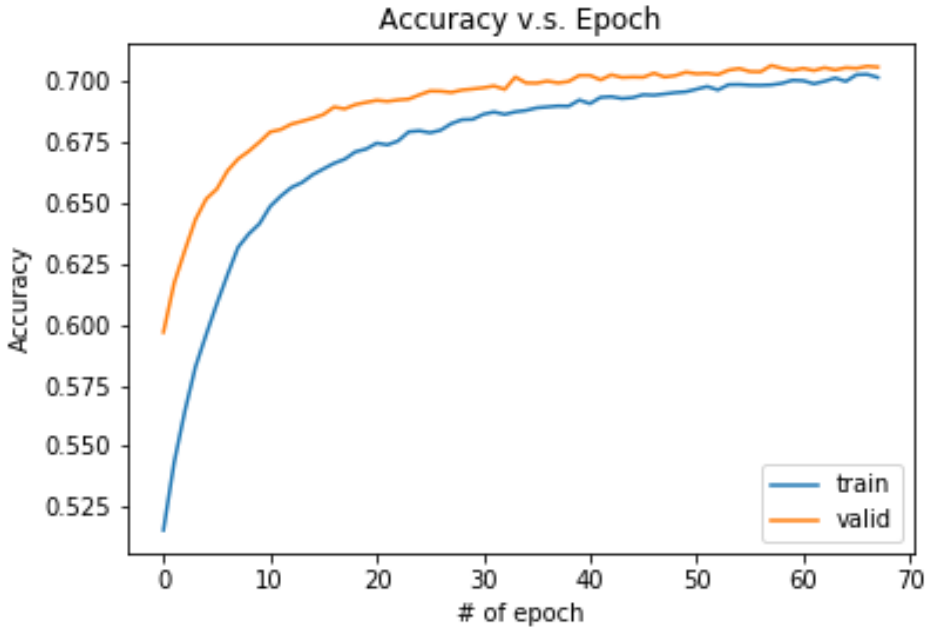
* 訓練曲線 (Training curve):顯示訓練過程的 loss 或 accuracy 變化。橫軸為 step 或 epoch,縱軸為 loss 或 accuracy。

RNN+ word embedding

word embedding 採用 genism 的 word2vector，參數為 size=256 維，min_count=5，iteration=5。然後將 train 的句子中的斷詞轉成 index 然後進去 embedding layer 再將它們依照每個 index 個別轉成對應的 256 向量，最後再丟進去 RNN 裡面。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 48)          5536176
_________________________________________________________________
gru_3 (GRU)                  (None, None, 128)         67968
_________________________________________________________________
gru_4 (GRU)                  (None, 256)               295680
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 514
=================================================================
Total params: 6,032,946
Trainable params: 496,258
Non-trainable params: 5,536,688
_________________________________________________________________
```

	Training set		Testing set(Kaggle)	
	Train	Validation	Public	Private
Accuracy	0.7029	0.70658	0.71195	0.70567
Epoch	Best at 67	Best at 58	at 58	at 58



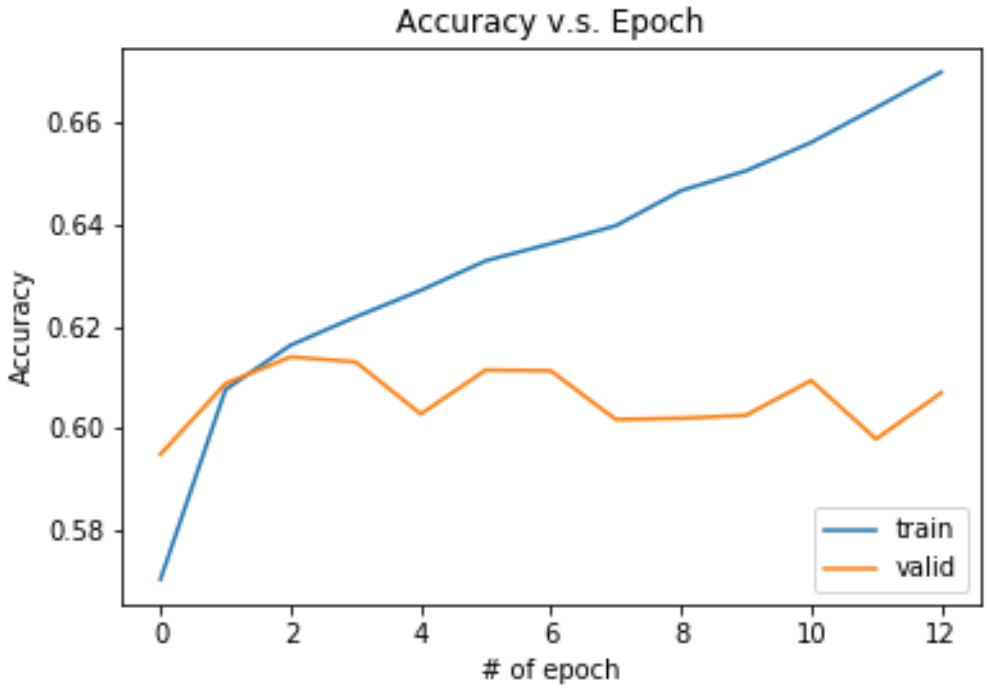
BOW+DNN

BOW 是先將所有出現的詞找出來，然後建立一個向量，每一維度就是代表那個詞出現在這個句子的次數。

DNN，最後再丟進 DNN 裡面。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               2936832
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 258
=================================================================
Total params: 2,953,602
Trainable params: 2,953,602
Non-trainable params: 0
_________________________________________________________________
```

	Training set		Testing set(Kaggle)	
	Train	Validation	Public	Private
Accuracy	0.6700	0.61400	0.60187	0.59650
Epoch	Best at 12	Best at 2	at 2	at 2



Problem 2. (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:

原本是將所有不是中文字的都去除，然後再分詞，不過這樣只能達到 0.72。後來又試著將 stopwords 去除，結果也是 0.72。反而學助教只去除數字跟逗號，就輕易達到 0.74 了。也許是因為中文甚至 dcard 上的人寫的文字，包含很多非中文字的符號，而他們都有其意思，所以如果隨便將他們去除的話，反而可能會降低準確度。

Embedding:

因為語言中的每個詞其實他們的關係不是獨立的，可以看到如果不加入 embedding，單純使用 BOW，不去考慮詞與詞之前的關係，準確度只能達到 0.6，而加入 embedding 後就輕易達到了 0.74。

不過這邊我有嘗試去 tune word2vector 的參數例如將 iteration 設成 3、5、10、15 甚至是 300，min_count=1、3、5，window=5 或 24，不過這些嘗試的結果，其準確度

都還是維持在 0.76，並沒有什麼用，只有如果將 size 設成 50，準確度只有 0.72，可能是因為 50 維相比 256 維並不能夠那麼完整的表達每個詞之間的關係，所以造成準確度下降。

架構:

原本只使用一層 LSTM 準確度只有 0.74。

後來在 LSTM 之後再加一層 GRU，準確度就來到的 0.75，因為通常來講一層 LSTM 可能不夠，再加一層 GRU，更多的參數能夠讓 model 更能去 fit 資料的分布，不過過多的參數也可能導致 overfitting，不過只有一層 LSTM 一層 GRU 應該是還不會到 overfitting 的地步。

之後嘗試先別更新 embedding layer 的 weights 但更新其他 layers 的 weights 兩個 epoch 後，再全部 layers 都能更新 weights，這樣讓準確度達到了 0.76，因為 embedding layer 的 weights 已經有經過 word2vector 訓練過了，因此想說先讓其他 layers 的 weights 也訓練的 2 個 epochs，有點讓他們站在同一起跑點的意思，然後在一起更新 embedding layer 的 weights，因為也許 word2vector 的訓練還不夠好，所以最後準確率能夠再提高。

Problem 3. (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。

	Training set		Testing set(Kaggle)	
	Train	Validation	Public	Private
不做斷詞	0.80070	0.75600	0.75107	0.74797
做斷詞	0.83320	0.76092	0.76142	0.75915

因為對於中文來說，詞比字更能代表意思，例如如果只看"白"、"痴"不會那麼直接有負面的意思，但如果是"白痴"就很確定是負面的詞語，所以有斷詞會比沒做斷詞來的準確，不過因為有時候 jieba 的斷詞並沒有分得那麼好，所以其實可以看到做斷詞跟不做斷詞的準確率只有差一點點。

Problem 4. (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	0.34403591	0.41651422
BOW	0.45266557	0.45266557

表格中的數字為 model 最後一層 sigmoid 後的分數。

可以看到 RNN 因為有考慮句子詞語出現的順序，所以造成兩個句子的分數不一樣。

而 BOW 因為是將句子中出現的詞語不考慮順序，只將句子每個詞語出現次數加進 vector，因為這兩句的每個詞都一樣，次數也一樣，所以他們的分數當然會一樣。

然後 RNN 雖然將兩個句子的 output 都小於 0.5，但也可以發現第二句比較接近 0.5，而依照人類判斷也是覺得第二句是比較有惡意傾向的句子。

Problem 5.

$$u_{t+1}^n = u_t^n * exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$$\alpha_t = ln(\sqrt{\frac{1-\epsilon_t}{\epsilon_t}})$$

$$\alpha_t = ln(\sqrt{\frac{1-\epsilon_t}{\epsilon_t}})$$

t=1:

$$u_1 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$$

$$\epsilon_1 = 0.2$$

$$\alpha_1 = ln2 = 0.6931$$

$$f_1(x) = \begin{cases} positive & x \leq 4 \\ negative & x > 4 \end{cases}$$

t=2:

$$u_2 = [0.5 \quad 2 \quad 0.5 \quad 0.5 \quad 0.5 \quad 0.5 \quad 0.5 \quad 2 \quad 0.5 \quad 0.5]$$

$$\epsilon_2 = 0.3125$$

$$\alpha_2 = ln\sqrt{\frac{11}{5}} = 0.3942$$

$$f_2(x) = \begin{cases} positive & x > 1 \\ negative & x \leq 1 \end{cases}$$

t=3:

$$u_3 = [0.337 \quad 2.966 \quad 0.742 \quad 0.742 \quad 0.742 \quad 0.337 \quad 0.337 \quad 2.966 \quad 0.337 \quad 0.337]$$

$$\epsilon_3 = 0.17$$

$$\alpha_3 = ln\sqrt{\frac{0.83}{0.17}} = 0.7928$$

$$f_3(x) = \begin{cases} positive & x \leq 1 \\ negative & x > 1 \end{cases}$$

$$u_4 = [0.153 \quad 6.526 \quad 1.632 \quad 1.632 \quad 1.632 \quad 0.153 \quad 0.153 \quad 6.526 \quad 0.153 \quad 0.153]$$

$$H(x) = sign(\sum_{t=1}^{3} \alpha_t f_t(x)) = [+ \ + \ + \ + \ + \ - \ - \ - \ - \ -]$$

Problem 6.

$$c_1 = 0$$

$$c_t' = g(wx^t + b) * f(w_i x^t + b_i) + c_t * f(w_f x^t + b_f)$$

$$y_t = h(c_t') * f(w_o x^t + b_o)$$

$$c_{t+1} = c_t'$$

依照上面式子可以求出

$$C = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 0.99986 \\ 4 \\ 3.99982 \\ 5.99964 \\ 5.99946 \\ 1 \end{bmatrix}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 0.000136 \\ 0.99986 \\ 4 \\ 3.99982 \\ 0.00027 \\ 5.99946 \\ 1 \\ 2.99995 \end{bmatrix}$$