HW4 - Malicious Comments Identification

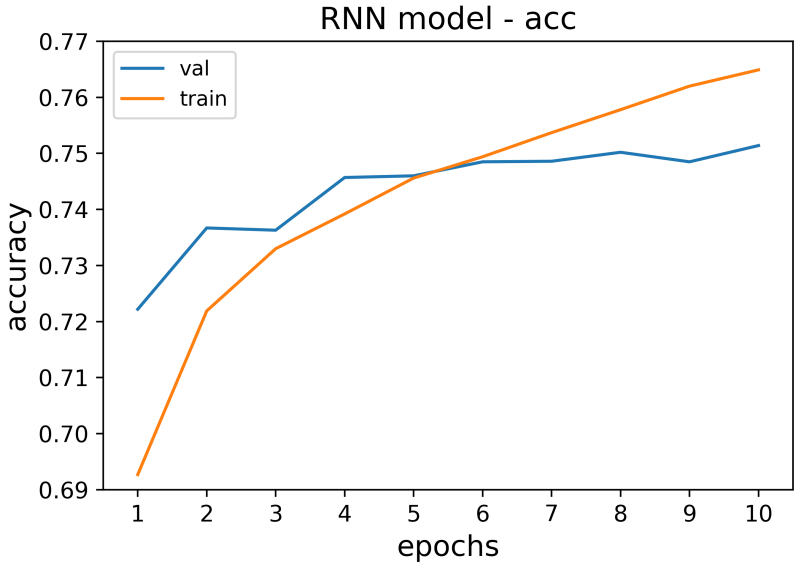
學號：B06209027 系級：大氣二 姓名:李冠勳

1-1 (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。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_19 (Embedding)     (None, 100, 100)          5645200
_________________________________________________________________
lstm_30 (LSTM)               (None, 100, 256)          365568
_________________________________________________________________
lstm_31 (LSTM)               (None, 128)               197120
_________________________________________________________________
batch_normalization_6 (Batch (None, 128)               512
_________________________________________________________________
dense_48 (Dense)             (None, 256)               33024
_________________________________________________________________
dropout_28 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_49 (Dense)             (None, 1)                 257
=================================================================
Total params: 6,241,681
Trainable params: 596,225
Non-trainable params: 5,645,456
```

- 最終的結果是用五種類似的model ensemble而成，僅僅是LSTM改用GRU或是運用Bidirectional 或是將 word2vec的model 加入Embedding中訓練而已，每個model也只有在RNN和DNN之間加入BatchNormalization。



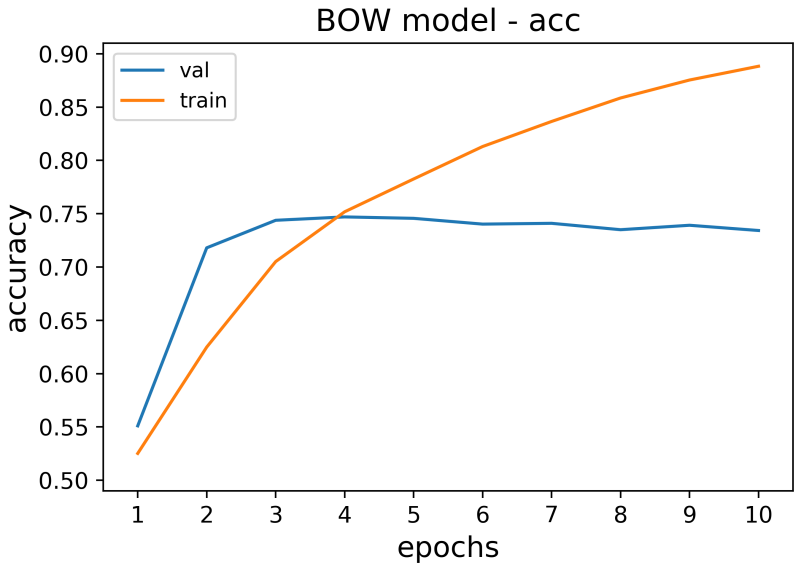
- 訓練一個epoch 就可以在val data取得不錯的結果，而一開始的train accuracy低於val accuracy，應該是有dropout的原因，但經過4個epoch之後val的準確度就不再上升因此，隨後train accuracy就超過val accuracy了(以上五個model都有相同的情形
都有相同的情形)

1-2(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。

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 1024)              40060928
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_8 (Dense)              (None, 256)               262400
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_9 (Dense)              (None, 64)                16448
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_10 (Dense)             (None, 16)                1040
_________________________________________________________________
dropout_7 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 17
=================================================================
Total params: 40,340,833
Trainable params: 40,340,833
Non-trainable params: 0
```

- 總共疊4層的DNN在BOW model,中間都有加入dropout



- 即使加入dropout，BOW的模型最後一樣可以在training 得到很高的acc，val一開始比train高但大概收斂在0.73 0.74左右
- 比較兩種模型
  - 共同點：val一開始都比train高但3-4個epochs就會收斂在0.74左右
  - 相異點：BOW的training acc一開始雖然比較低但上升速度比較快很快就大幅超越val acc。

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。

- pre-processing
  - 將兩個以上的特殊字元合併成一個(ex:!!! ⇒ !)
  - 將重複超過三個以上的文字砍掉(ex:阿阿阿 ⇒ 阿阿)
  - 將長度超過上限(我的設定為100個字詞)的data捨棄，因為無法判斷其惡意留言是出現在整個留言的哪一部份，故刪除
- model
  - embedding:有train的結果稍微好一點點，可能是word2vec的判斷有一點點進步的空間
  - 使用雙向的LSTM或是GRU結果沒有差，但最後把五個類似的model ensembel後結果好一點點(不如預期)。
  - RNN + DNN的model比RNN好一點點，可以對RNN的結果再進行調整所以結果比較好
- test data
  - 直接砍掉剩100個字詞進行判斷

- 把長度超過100的字詞分段進去判斷只要其中有一部份是惡意留言就斷定為惡意留言，結果沒有差，可能是直觀將某段是被判斷是惡意留言就判斷整段話是惡意留言的判斷方式太過於武斷，會放大model誤判的機率，因此，結果差不多。可能要加個分數的評比等等才能提高準確率

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。

model	val acc	public	private
use jieba	0.75270	0.75540	0.75137
without jieba	0.74008	0.74317	0.74010

- 使用jieba的結果只比沒用jieba的結果好一點點，結果令我有點訝異！可能的原因有以下兩點
  - 文字長度沒有很長因此沒有做斷詞一樣可以判斷出語意，進而判斷是否為惡意留言
  - 在社群網站中新潮的字詞很多，這些字詞無法準確的切割出來，照成誤判進而影響結果

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。

sentence1:"在說別人白痴之前,先想想自己"
sentence2:“在說別人之前先想想自己,白痴”

model/score	sentence1	sentence2
RNN	0.4247349	0.5119428
BOW	0.76166123	0.76166123

- 在RNN model中字詞的順序會影響判斷結果，因此，兩句得到不同的分數，也才勉強有把第二句判斷成惡意留言
- 在BOW model中只判斷字詞在一段話中出現的數量，沒有先後關係，因此，兩句字詞完全相同、順序不同的話，會得到一樣的分數，也因為如此，判斷到「白痴」這個字詞就把這句話當作惡意留言了吧！

5 (1%)

x	0	1	2	3	4	5	6	7	8	9
y	＋	－	＋	＋	＋	－	－	＋	－	－

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

- t=1:在4-5中間切下0-4為＋，5-9為－

x	0	1	2	3	4	5	6	7	8	9
y	＋	－	＋	＋	＋	－	－	＋	－	－
預測	＋	＋	＋	＋	＋	－	－	－	－	－
u_1^x	1	1	1	1	1	1	1	1	1	1

  $\epsilon_1 = 0.2 \quad d_1 = 2 \quad \alpha_1 \approx 0.69$

- t=2:在0-1中間切下0為＋,1-9為－

x	0	1	2	3	4	5	6	7	8	9
y	＋	－	＋	＋	＋	－	－	＋	－	－
預測	＋	－	－	－	－	－	－	－	－	－
u_2^x	0.5	2	0.5	0.5	0.5	0.5	0.5	2	0.5	0.5

$\epsilon_2 \approx 0.44 \quad d_2 \approx 1.13 \quad \alpha_2 \approx 0.12$

- t=3:在7-8中間切下0-7為＋,8-9為－

x	0	1	2	3	4	5	6	7	8	9
y	＋	－	＋	＋	＋	－	－	＋	－	－
預測	＋	＋	＋	＋	＋	＋	＋	＋	－	－
u_2^x	0.44	1.77	0.57	0.57	0.57	0.44	0.44	2.26	0.44	0.44

$\epsilon_2 \approx 0.33 \quad d_2 \approx 1.42 \quad \alpha_2 \approx 0.35$

- 總結
  $H(x) = sign(0.69f_1(x) + 0.12f_2(x) + 0.35f_3(x))$

x	0	1	2	3	4	5	6	7	8	9
y	＋	－	＋	＋	＋	－	－	＋	－	－
預測結果	＋	＋	＋	＋	＋	－	－	－	－	－

  - 預測結果與t=1的結果相同，其他的function結果太差因此不能使t=1的結果進步

6 (1%)

t	1	2	3	4	5	6	7	8
$g(z)$	3	-2	4	0	2	-4	1	2
$f(z_i)$	1	1	1	1	1	0	1	1
$f(z_f)$	1	1	0	1	1	1	0	1
c	1	4	2	4	4	6	6	1
c'	4	2	4	4	6	6	1	3
$h(c')$	4	2	4	4	6	6	1	3
$f(z_o)$	0	1	1	1	0	1	1	1
y	0	2	4	4	0	6	1	3

- $f(z) = \frac{1}{1+e^{-z}} \quad g(z) = z \quad h(z) = z$
- c為前一時間的c'
- $c' = g(z)f(z_i) + cf(z_f)$
- $y = h(c') + f(z_o)$
- 分別帶入公式即可得到上表$y^t = (0, 2, 4, 4, 0, 6, 1, 3) \qquad (t = 1 \sim 8)$