Homework 4 Report - Malicious Comments Identification

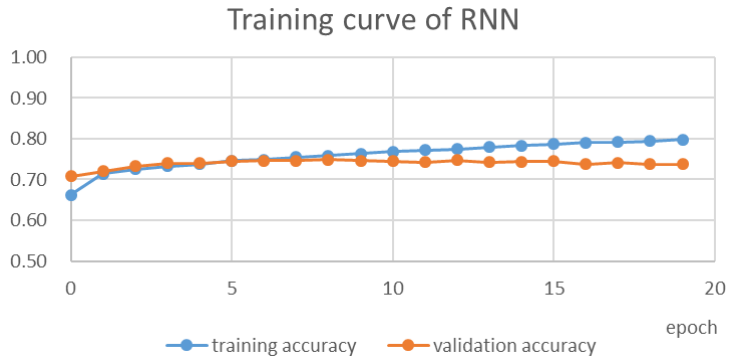
電機四 B04505025 陳在賢

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。

我所實作的 RNN model 共有 3 層 hidden layer，包含兩層 256 個節點的 LSTM layer 及一層 512 個節點的 Dense layer，三層的 dropout 設定分別 0.3、0.4、0.5（逐層增加的理由同作業三所述）。而 input 為 48 個 timesteps 的 256 維 vector，output 為整數（以 0.5 為 threshold 辨別輸出為 0 或 1）。

具體資料流為：第一層 LSTM 會從 input layer 序列讀進 256 維資料，並同樣序列輸出 256 維資料至下一層；而第二層 LSTM 也為序列讀進，但會在讀完 48 個 timesteps 後，才統一將 256 維結果輸出至下一層；最終的 Dense，會將 256 維資料展開至 512 維後傳至 output layer。

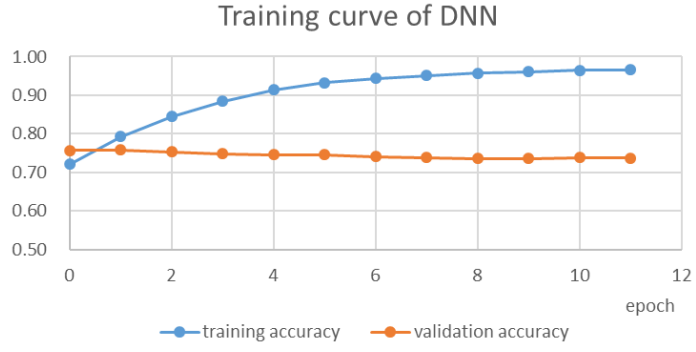
此 model 共計約 118 萬個參數，training curve 如下所示：



可看出 Training accuracy 為嚴格遞增。而 validation accuracy 在前 5 個 epoch 較 training 高（此為 dropout 的效果），但在第 8 個 epoch 到達極大值 0.7482 後就開始下降，此即發生了 overfitting（本次作業的 data 相當容易發生 overfitting，具體討論可見第 2 題）。

再來，關於 DNN 的實作共有 3 層 hidden layer，各為 256、128、64 個節點的 Dense layer，由於此 model 更容易 overfitting 故統一將三層的 dropout 設為 0.5。而 input 為 38790 維 vector（本次 data 共含此數量個詞語），output 同樣為整數。由於 DNN 的資料流相對單純，就不再贅述。

此 model 共計約 997 萬個參數，training curve 如下所示：



可看出 Training accuracy 仍為嚴格遞增，但最高值幾乎逼近 100%，我認為是因為此 model 的參數量遠多於 RNN，因此更有能力去擬合 training data。而 validation accuracy 在第 2 個 epoch 就達極大值 0.7586，之後就發生嚴重的 overfitting，而即便加了很重的 dropout 且試圖減少後幾層 layer 的節點數，仍無法解決此現象。

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。

在資料前處理部分，我統計了些負面關鍵字，此些字的特徵為：出現頻率高且包含此字的句子有八成以上為惡意評論（可以想像多為髒話或貶低他人之詞，如學店、甲甲、死一死、＊＊娘）。而由於此些字多為新興網路流用語，因此先將其加入辭典中，在 jeiba 切字時才不易被拆散。同時可用負面關鍵字做 pretraining，當 data 出現此些字時，先提升初分數，使其有較高機率被辨別為惡意留言。上述步驟約能提升 0.5%準確率。

至於 word embeding，考慮其為 unsupervised learning，因此我同時使用 training 及 testing data 進行 word2vec model 的訓練，此步有兩個優點：一來訓練過程讀到更多語句，因此更能精準做 embeding；二來在做 testing data 的 predict 時，也不容易遇到 OOV（罕見詞）。此步驟可提升約 1%準確率。

有關架構部分，由於本次使用的 data 為眾多網友的留言，用詞及語法變化十分多樣，因此在 training 過程會發現非常容易 overfitting（只在讀過的 training data 表現較好，但 validation data 仍讀不懂）。因此這此只用三層 layer 以減少參數量，同時使用了搭配 regularizer 及 dropout，以減少 overfitting 的可能性。

最終在預測結果部分，由於 RNN 及 DNN 等兩 models 是根據不同線索做判斷，前者看中語法及詞的順序性、後者看中句子使用到的詞，也使得兩者預測的結果相當不同。所以最後沿用了 Adaboost 的概念，將兩 model 的輸出分數根據其準確率作加權總合作為最終結果，此步提升 0.8%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。

在此以 RNN 為例，若以字為單位，訓練速度會明顯變慢（每個 epoch 的訓練時間變長，且 train 完每個 epoch 的 training accuracy 進步幅度相當小），同時準確率也會降低（validation accuracy 極大值約為 0.67）。

我認為原因有二：首先在語言中本來就是以「詞」為具意義的最小單位，若硬要斷為「字」，會強迫 RNN 得自行學會字跟字合併為詞所代表的意義，使得 training 速度緩慢。再者，以字斷句會使句子被拆為更多向量表示，若設定句子的 timestep 極大值為 48，會讓句子更容易因過長而被強迫斷句，在較少的資訊量做 training，也當然會使準確率降低。

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。

	前句（實為 0，非惡意）	後句（實為 1，惡意）
RNN	0.3340	0.5951
DNN	0.6327	0.6327

可看出 RNN 在兩句皆會做出正確的辨別，而 DNN 在兩句則皆辨別為惡意且分數相同。

深入探討，兩句經過 jieba 切詞後會有完全相同的結果，因此純粹以句中用詞來辨別而不考慮詞的順序性的 BOW+DNN，就會算出相同的分數，且因句中出現「白癡」等負面關鍵字，故會辨別為惡意。相反得，RNN 因為綜合考慮了詞的順序性，因此在此兩句會算出不同分數，且皆有正確的辨別！

然而這並不代表 BOW+DNN 一定表現得較差，如第 1 題所述兩者各有其優缺點，每句話要辨別對的關鍵點也不同，有些看中用詞、有些看中詞的順序性，因此綜合兩者結果才能得到最佳效果。

5. (1%)

t	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
x0	0	1	1	0	0	0	1	1
x1	1	0	1	1	1	0	1	0
x2	0	1	1	1	0	1	1	1
x3	3	-2	4	0	2	-4	1	2
g(z)	3	-2	4	0	2	-4	1	2
f(zi)	1	1	1	1	1	0	1	1
f(zf)	1	1	0	1	1	1	0	1
f(zo)	0	1	1	1	0	1	1	1
c	0	3	1	4	4	6	6	1
c'	3	1	4	4	6	6	1	3
y	**0**	**1**	**4**	**4**	**0**	**6**	**1**	**3**

6. (1%)

n	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
y	+	-	+	+	+	-	-	+	-	-
u1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
classifier 1	threshold =	4.5	(below: + / above: -)							
ŷ1	+	+	+	+	+	-	-	-	-	-
ϵ1 / α1	0.20	0.69								
u2	0.50	2.00	0.50	0.50	0.50	0.50	0.50	2.00	0.50	0.50
classifier 2	threshold =	1.5	(below: - / above: +)							
ŷ2	-	-	+	+	+	+	+	+	+	+
ϵ2 / α2	0.31	0.39								
u3	0.74	1.35	0.34	0.34	0.34	0.74	0.74	1.35	0.74	0.74
classifier 3	threshold =	0.5	(below: + / above: -)							
ŷ3	+	-	-	-	-	-	-	-	-	-
ϵ3 / α3	0.32	0.38								
final classifier	**x < 0.5: + / 0.5 < x < 1.5: - / 1.5<x<4.5: + / 4.5<x: -**									
	+	-	+	+	+	-	-	-	-	-