機器學習作業五 報告

學號	系級	姓名
B03902015	資工三	簡瑋德

1. 請問「softmax」適不適合作為本次作業的「output layer」? 寫出你最後選擇的「output layer」並說明理由。

- 模型的結構

Layer	Parameters
Embedding	`input_dim = 3000 , output_dim = 50 , input_length = 50`
LSTM	`units = 128 , dropout = 0.2 , recurrent_dropout = 0.2`
Dropout	`rate = 0.2`
Dense	`units = 64`
Dropout	`rate = 0.2`
Activation	`activtion = "relu"`
Dense	`units = 38 , activation = "sigmoid"`

- 「softmax」不適合作為本次作業的「output layer」
- 我選擇「sigmoid」作為最後一層的「activation function」，原因如下
  - 我希望「output layer」的38個「unit」，分別代表該文章與各個「tag」的相關程度
  - 使用「sigmoid」的話，相當於用一個「$0-1$」之間的數值來描述相關的程度，「1」表示非常相關，反之「0」表示無關。一篇文章可能和多個「tag」相關，即「multi-label」，因此「output layer」中可能有多個大於「0.5」的數值
  - 如果使用「softmax」，38個數值的總和會等於一，且每一個數值都不為負，分別代表該文章對應各個「tag」的機率，其中一個「tag」機率高，就表示其他「tag」機率必須較低。這種激活函數，比較適合「one-label」的資料，即各個「tag」間彼此互斥，每筆資料只會和其中一個「tag」相關

2. 請設計實驗驗證上述推論。

- 參數

epochs	batch_size	loss	optimizer	threshold
25	32	“binary_crossentropy”	“adam”	0.5

- 實驗結果 (Validation on 20% of Training Data)

Activation	Validation Loss	Validation F1 Score
Sigmoid	0.1242	0.2906
Softmax	0.1145	0.0662

- 觀察和解釋
  - 不論使用「sigmoid」或是「softmax」，「loss」都有降下來
  - 但是，「softmax」的「f1score」卻很慘，因為當答案有多個「tag」的時候，模型最多也只能猜出其中一個，甚至可能彼此競爭，最後四不像

3. 請試著分析「tags」的分布情況(數量)。

- 表格

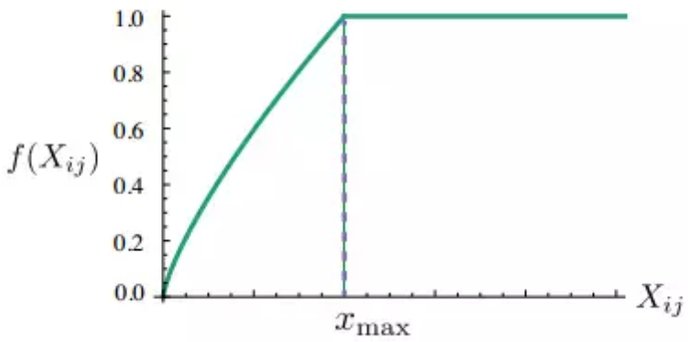
Tag	數量	Tag	數量	Tag	數量
FICTION	1672	SPECULATIVE-FICTION	1448	NOVEL	992
SCIENCE-FICTION	959	CHILDREN'S-LITERATURE	777	FANTASY	773
MYSTERY	642	CRIME-FICTION	368	SUSPENSE	318
YOUNG-ADULT-LITERATURE	288	THRILLER	243	HISTORICAL-NOVEL	222
HORROR	192	DETECTIVE-FICTION	178	ROMANCE-NOVEL	157
ROMANCE-NOVEL	137	ADVENTURE-NOVEL	109	NON-FICTION	102
SPY-FICTION	75	ALTERNATE-HISTORY	72	COMEDY	59
AUTOBIOGRAPHY	51	BIOGRAPHY	42	SHORT-STORY	41
HISTORY	40	COMIC-NOVEL	37	SATIRE	35
MEMOIR	35	AUTOBIOGRAPHICAL NOVEL	31	WAR-NOVEL	31
DYSTOPIA	30	NOVELLA	29	HUMOUR	18
TECHNO-THRILLER	18	HIGH-FANTASY	15	APOCALYPTIC-AND-POST-APOCALYPTIC-FICTION	24
GOTHIC-FICTION	12	UTOPIAN-AND-DYSTOPIAN-FICTION	11		

- 觀察與發現
  - 各類「tag」的數量非常不平均，從十幾個到一千六百個都有
  - 另外，在「test」的結果中，數量少的「tag」幾乎都沒有被抓出來
  - 有嘗試對數量少的「tag」做「up-sampling」，但「kaggle public score」反而下降

4. 本次作業中使用何種方式得到「word embedding」？請簡單描述做法。

- GloVe Embedding中維度等於50的版本，包含了40萬個詞彙
- 詞向量的訓練文庫包含「維基百科」和「Linguistic Data Consortium」提供的「Gigaword」資料庫
- 簡單來說，「GloVe」結合了「共現矩陣(如LSA)」和「局部語境窗口(如word2vec)」的優點，綜合運用「全局」和「局部」的統計訊息來生成詞向量。前者可以看作「count-base」的方法，後者則是「prediction-base」的版本，GloVe則是各取所長，嘗試建構更強大的詞向量。
- 至於做法，可以試著從「GloVe」的損失函數來理解
  - $J = \sum_{i,j=1}^{\left|V\right|} f(\mathrm{c}_{ij})(\mathrm{w}_i \cdot \tilde{\mathrm{w}}_j + b_i + \tilde{b_j} - \log \mathrm{c}_{ij})^2$

- w_i和b_i是當前詞t_i的詞向量和偏置，$\tilde{\mathrm{w}}_j$和$\tilde{b_j}$則是情境詞t_j的輔助向量和輔助偏置，而c_i則是從共現矩陣中得到的數值，代表當前詞t_i和情境詞t_j的相關程度，不會隨著訓練更新
- $f()$是加權函數，大於某個閥值就是1，如下圖所示



- 從損失函數中，可以發現它確實用到了全局資訊，即共現矩陣的c_{ij}，同時，又和「word2vec」同樣使用詞向量與情境輔助向量的內積作為指標

- 向量、偏置的更新方法就是普通的「gradient descent」，並採用「adgrad」來解決「不同的參數，更新、收斂的速度不一定一樣」的問題

5. 試比較「bag of word」和「RNN」何者在本次作業中效果較好

- 「Term Fequency Matrix」進「DNN」的模型

Layer	Parameters
Input	`shape = (3000,)`
Dense	`units = 1024`
Dropout	`rate = 0.2`
Activation	`activtion = "relu"`
Dense	`units = 512`
Dropout	`rate = 0.2`
Activation	`activtion = "relu"`
Dense	`units = 256`
Dropout	`rate = 0.2`
Activation	`activtion = "relu"`
Dense	`units = 38 , activation = "sigmoid"`

- 「Embedding」搭配「CNN」的模型 - 使用「Functional API」，考慮「Bigram」、「Trigram」和「4-gram」

Layer	Parameters	Name	Input
Input	`shape = (3000,)`	main_input	-
Embedding	`input_dim = 3000`, `output_dim = 50`, `input_length = 50`	emb	main_input
Dropout	`rate = 0.2`	emb_dropout	emb
Conv1D	`units = 128`, `filters = 2`, `activation = "relu"`, `strides = 1`	conv1	emb_drop
GlobalMaxPooling1D()	-	maxp1	conv1
Conv1D	`units = 128`, `filters = 3`, `activation = "relu"`, `strides = 1`	conv2	emb_drop
GlobalMaxPooling1D()	-	maxp2	conv2
Conv1D	`units = 128`, `filters = 4`, `activation = "relu"`, `strides = 1`	conv3	emb_drop
GlobalMaxPooling1D()	-	maxp3	conv3
concatenate	[maxp1, maxp2, maxp3]	merged	-
Dense	`units = 256`	dense	merged
Dropout	`rate = 0.4`	dropout	dense
Activation	`activtion = "relu"`	activation	dropout
Dense	`units = 38`, `activation = "sigmoid"`	main_output	activation

- 參數

epochs	batch_size	loss	optimizer	threshold
25	32	“binary_crossentropy”	“adam”	0.5

- 實驗結果 (Validation on 20% of Training Data)

Model	Validation Loss	Validation F1 Score
RNN-Problem1	0.1242	0.2906
DNN	0.1641	0.2588
CNN	0.1209	0.3079

- 觀察與發現
  - 「Term Fequency Matrix」接「DNN」的模型很直關，但參數數量很多，相當吃記憶體，「Validation」的表現反而不如原本的「RNN」，也許是因為參數需要再作調整
  - 「CNN」的版本表現最好，共用一個「embedding」矩陣，並交給三種「n-gram」的「CNN」，最後再將結果「concatenate」起來，進分類器判斷。