學號：B03901076　系級：電機三　姓名：林尚謙

1. (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。

答：

模型架構：

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 306, 100)	5186700
gru_1 (GRU)	(None, 256)	274176
dense_1 (Dense)	(None, 256)	65792
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 192)	49344
dropout_2 (Dropout)	(None, 192)	0
dense_3 (Dense)	(None, 128)	24704
dropout_3 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 38)	4902

Total params: 5,605,618
Trainable params: 418,918
Non-trainable params: 5,186,700

我的模型架構是輸入 train_sequences 到一層 embedding layer，通過一層 GRU(RNN layer)以及三層 Hidden layer 後，Output layer 輸出 38 維的輸出，對應到 38 個 tags。

這次作業不適合使用 softmax 作為 output layer 的 activation，因為 softmax 會把所有輸出值做壓縮，使每個值的範圍在(0,1)之間，並讓全部的和為 1，而且 softmax 函數會壓低除了最大值以外的其他分量，使最大值特別凸顯出來，然而這次作業的一筆資料有可能同時屬於很多個種類，softmax 會讓最大值之外的其他種類很難分出來，所以不適合使用 softmax 作為 output layer 的 activation。

基於上述考量以及一些嘗試後，我最後是使用 sigmoid 作為我 output

layer 的 activation。因為 sigmoid 同樣會使輸出介於代表機率的(0,1)之間，但彼此之間不會互相影響，使一筆資料可以同時對應到多個不同的種類。

2. (1%)請設計實驗驗證上述推論。

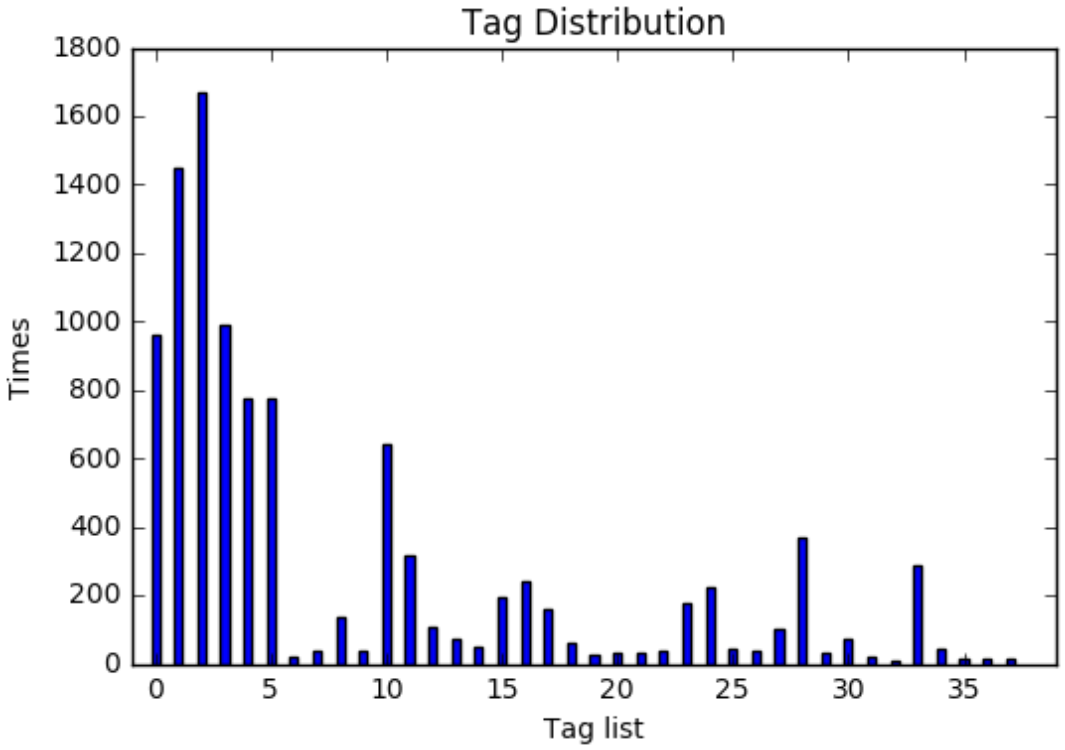
答：

Activation	Softmax(0.4)	Softmax(0.2)	Sigmoid
Kaggle public	0.15119	0.42251	0.50318

對於同樣的一個 model 架構，分別使用兩種 output layer 的 activation：softmax 與 sigmoid 來進行實驗，如果使用相同的 thresh(0.4 左右)，發現 softmax 幾乎沒辦法訓練出好的模型結果(val_f1_score 大約在 0.1~0.2 之間)，而且最後 test data 預測出來的結果，幾乎都是空白或只有唯一的一個 tag，跟上述推論 softmax 會無法分類 Multi-label 相符。而如果降低 softmax 的 thresh 到 0.2 左右，效果會比較提升，但還是以 sigmoid 的表現較好。

3. (1%) 請試著分析 tag 的分佈情況(數量)。

答：



這裡統計在 training data 中，所有(總共 38 種)Tags 出現的次數，發現出現次數最多的前三名分別是 FICTION(1672 次)、SPECULATIVE-FICTION(1448 次)和 NOVEL(992 次)，而最少的則是 UTOPIAN-AND-DYSTOPIAN-FICTION(11 次)。顯示在這些 Tags 中其實有涵蓋廣度的區分，例如有的 data 被分在 HISTORICAL-FICTION 的同時也有可能被分在 FICTION 下，但 HISTORICAL-FICTION 和 DETECTIVE-FICTION 就比較難同時出現在同一筆 data 的 tag 中，所以像 FICTION、NOVEL 這種涵蓋較廣的 tag 就會出現很多次，而其他涵蓋範圍較小(較有特殊性)的 tag 就出現較少次。

4. (1%)本次作業中使用何種方式得到 word embedding?請簡單描述做法。

答：

在這次作業中我是使用 GloVe 來得到 pre-trained 的 word vector，而 GloVe(Global Vectors for Word Representation)得到 word embedding 的方式是 Count based，計算在一個 corpus 之中每個 word 之間同時出現(co-occurrence)的次數，兩個 word 越常一起出現，就讓它們的 word vector 越接近(inner product 越大)，使每個 word 都有對應的 word vector，完成 word embedding。

5. (1%)請試著比較 bag of words 和 RNN 何者在本次作業中效果較好。

答：

bag of words 1 模型架構：

Layer (type)	Output Shape	Param #
dense_9 (Dense)	(None, 256)	13278208
dropout_7 (Dropout)	(None, 256)	0
dense_10 (Dense)	(None, 128)	32896
dropout_8 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 64)	8256
dropout_9 (Dropout)	(None, 64)	0
dense_12 (Dense)	(None, 32)	2080
dropout_10 (Dropout)	(None, 32)	0

```
dense_13 (Dense)             (None, 38)                1254
=================================================================
Total params: 13,322,694
Trainable params: 13,322,694
Non-trainable params: 0
_________________________________________________________________
```

bag of words 2 模型架構：

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 512)               26556416
_________________________________________________________________
dropout_11 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_15 (Dense)             (None, 256)               131328
_________________________________________________________________
dropout_12 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_16 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_13 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_17 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_14 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_18 (Dense)             (None, 38)                2470
=================================================================
Total params: 26,731,366
Trainable params: 26,731,366
Non-trainable params: 0
_________________________________________________________________
```

Model	Bag of words 1	Bag of words 2	RNN(best)
Kaggle public	0.46449	0.47119	0.50318

Bag of words 的方式透過將 data 的 sequences 轉成統計每個 word 出現次數的 matrix，之後透過 Neural Network 來 train，訓練過程中 val_f1_score 會穩定上升到 0.45 左右，不過之後就不太容易在繼續上升，很容易發生 overfitting(train 的 f1_score 還繼續上升)，拉高 dropout rate(約 0.5~0.6)後可以提升效果，但目前最好的分數仍較 RNN 來的差。

Bag of words 讓每一個 word(5 萬多個)在 matrix 中都有一個位置，然後用 NN 的方式去 train 它，所以在這次的 Multi-label 分類上，可能會有一些關鍵字，能夠讓模型去找到對應的 tag，會讓 train 的分數容易上升，但相較之下也容易發生 overfitting，所以 test 的分數到一定程度就難再繼續增加了。

除此之外，因為這次的 data sequences 是一段文字或句子，bag of words 的方式會忽略 word 出現在句子中的前後關係，相較之下 RNN 有考慮 word 的前後關係，所以這有可能是這次作業裡 RNN 效果較 bag of words 好的原因之一。