(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。

不適合。我選擇的是 sigmoid 函數作為 output layer，因為，softmax 的結果是和為 1 的輸出函數，可以認為是一種概率分佈，很難找一個合適的 threshold 把 multi-label 的個數和類型確定下來。而 sigmoid 函數，是把每個 label 的 weight 映射到 0-1 之間，然後我們可以用一個合適大小的 threshold 進行分類，是否選取這個 label 作為最終的 label。

(1%)請設計實驗驗證上述推論。

使用相同的 layer，僅僅修改最後一層 Dense 的 activation，然後對比在 public 上的 F1 score 結果來判斷哪種方式是否合適。Eartly stop 均設置 patient = 15。

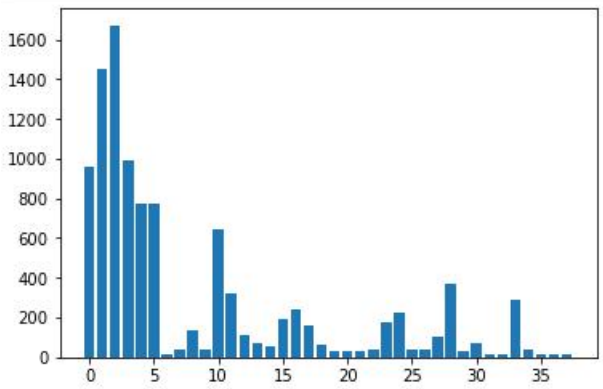
```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 306, 100)          5186900
_________________________________________________________________
gru_1 (GRU)                  (None, 306, 128)          87936
_________________________________________________________________
gru_2 (GRU)                  (None, 128)               98688
_________________________________________________________________
dense_1 (Dense)              (None, 512)               66048
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 38)                19494
=================================================================
Total params: 5,459,066
Trainable params: 272,166
Non-trainable params: 5,186,900
```

Last layer Activation	Loss(validation)	F1 score(validation)	F1 score(public)
Sigmoid	3.956	0.516	0.505
Softmax	4.835	0.202	0.180

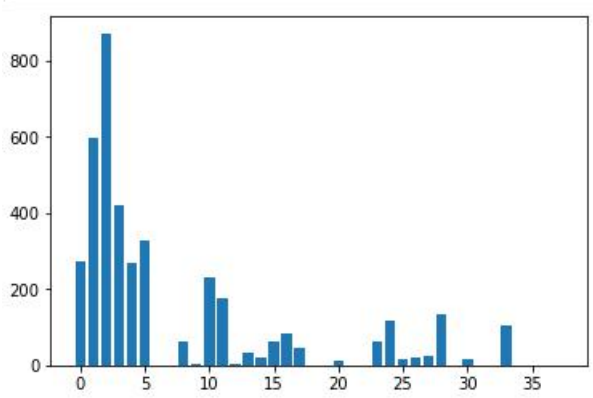
很明顯，sigmoid 作為 last layer 的情況下，得到了比 softmax 高太多的 F1 score。而且，Loss 相對比較小。所以上題的推論結果是正確的。

(1%)請試著分析 tags 的分佈情況(數量)。

Training data 的 Tags 分佈情況：



Prediction 的 Tags 分佈情況：



Tag_list=

['SCIENCE-FICTION', 'SPECULATIVE-FICTION', 'FICTION', 'NOVEL', 'FANTASY', "CHILDREN'S-LITERATURE", 'HUMOUR', 'SATIRE', 'HISTORICAL-FICTION', 'HISTORY', 'MYSTERY', 'SUSPENSE', 'ADVENTURE-NOVEL', 'SPY-FICTION', 'AUTOBIOGRAPHY', 'HORROR', 'THRILLER', 'ROMANCE-NOVEL', 'COMEDY', 'NOVELLA', 'WAR-NOVEL', 'DYSTOPIA', 'COMIC-NOVEL', 'DETECTIVE-FICTION', 'HISTORICAL-NOVEL', 'BIOGRAPHY', 'MEMOIR', 'NON-FICTION', 'CRIME-FICTION', 'AUTOBIOGRAPHICAL-NOVEL', 'ALTERNATE-HISTORY', 'TECHNO-THRILLER', 'UTOPIAN-AND-DYSTOPIAN-FICTION', 'YOUNG-ADULT-LITERATURE', 'SHORT-STORY', 'GOTHIC-FICTION', 'APOCALYPTIC-AND-POST-APOCALYPTIC-FICTION', 'HIGH-FANTASY']

可以看出，tag 分佈最密集的部分在前 5 個，['SCIENCE-FICTION', 'SPECULATIVE-FICTION', 'FICTION', 'NOVEL', 'FANTASY']，無論是之前 Tag 數量的分佈，還是 predict 的結果。而且 training data 和 prediction 的 tag 分佈非常相似。Tag_list 後面部分在 training data 中分佈數量很少的，在 predict 的 data 里就幾乎沒有出現。

可能因為 training 中，一定 tag 類別的數量太少，在 word embedding 中找不到很好的 feature 可以判斷是該 tag。以及，只要 training 時，某些 tag 有大量數據的，prediction 中都有比較好的結果，說明 training 中不同 tag 的數量對於結果也很重要。如果都有一定數量的 data 去訓練，得到的結果會更好。這些推論是基於 test data 的分佈和 training data 分佈接近的情況。

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

本次作業使用 GloVe 100d 的 model 作為 embedded layer 的 model，text 經過分類器分割成多個詞組，然後每個詞組通過 texts_to_sequences 根據出現頻率量化為數字 vector，然後用 pad_sequences 用 0 填充成相同長度的序列。經過在 model 中相同單詞的賦值，轉換成 GloVe 100d 的 word embedding 表示。GloVe 模型對原理是對全局字詞矩陣的非零項進行訓練，字詞矩陣列出了給定語料庫（Wikipedia 2014+Gigaword 5）中單詞在彼此共同出現的頻率。尋找出兩

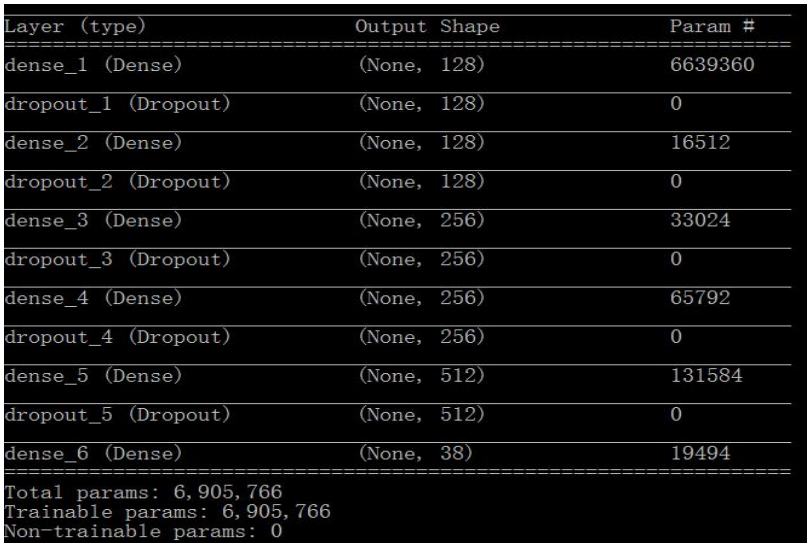
個 word vector，表示兩個詞組，使得 inner product 和他們共同出現的頻率 N 越接近越好。如果他們共同出現的頻率很大，那麼就說明，他們的 word vector 很相似。

GloVec 模型結合了兩種主要模型的優勢：Global matrix factorization 和 Local context window methods 方法。利用了詞全局統計資訊，模型通過訓練 word-word co-occurrence matrix 裏非零元素

(1%)試比較 bag of word 和 RNN 何者在本次作業中效果較好。

	Loss（train）	F1 score（train）	Loss（val）	F1 score（val）	F1 score（piblic）
bag of word	3.47	0.65	5.51	0.48	0.4954
RNN（GRU）	3.96	0.52	3.98	0.51	0.5050

Eartly stop 均設置 patient = 15。Bag of word 我使用的結構是：



Bag of word 的在本次實驗中表現的訓練速度十分的快，在 train 的時候，F1 score 上升的很快，training data 的 Loss 下降的很快。但是很快就出現了 overfitting 的情況，而且不管怎麼加 dense 層都提升不大。雖然，很容易就超過了 baseline，但是應該在 private 上會很差。可能因為在有相同 word 的情況下，有些句子表達的含義不同，導致誤判。

這裡 RNN 使用的是 GRU。訓練速度比較慢，但是在 validation 上表現比較穩定，不會很大的 overfitting。不過到了後面就提升非常緩慢了。在 validation 的 Loss 上比 Bag of word 好比較多。而且最後的得分也比較好一些。

依據 validation F1score 的好壞去判別,RNN 的效果在本次實驗中會比較好。