學號：R05922085 系級： 資工碩一 姓名：周宗鴻

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：

我認為 softmax 不太適合用在本次作業的 output layer，因為 softmax 會讓最後的 ouput 加起來等於 1，但是這樣會導致在 training 的過程中提升某個類別的機率就勢必得讓其他類別機率下降，例如：一段話屬於 A 類別機率提升不代表一定會影響其他類別的機率。故我最終選擇的 output layer 是 sigmoid，理由是能夠個別看待每一類別的機率，比較合理。

此外我設計的模型架構與助教 sample code 類似，移除 hidden layer 的最後一層且再每層加上 BatchNormalization，下面為我使用 summary 印出我模型架構的結果：

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 306, 300)          15560400
_________________________________________________________________
gru_1 (GRU)                  (None, 128)               164736
_________________________________________________________________
dense_1 (Dense)              (None, 256)               33024
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
activation_1 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512
_________________________________________________________________
activation_2 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 38)                4902
```

2. (1%)請設計實驗驗證上述推論。

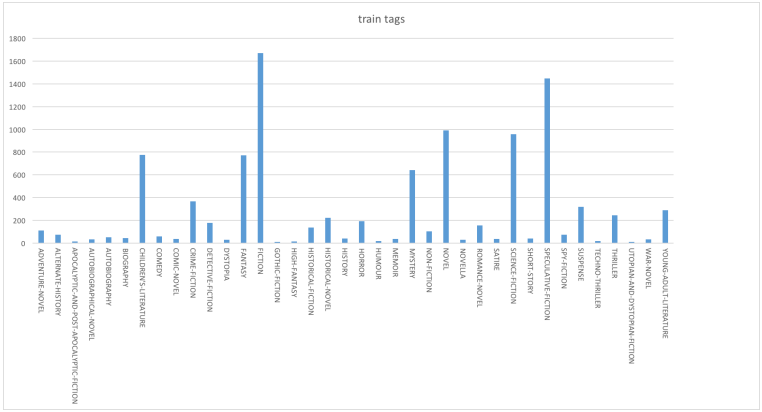
答：

首先我嘗試直接將最後一層設定為 softmax 與 sigmoid，softmax 丟到 kaggle 上的辨識率為: 0.23418、sigmoid 丟到 kaggle 上的辨識率為: 0.50343，我認為在不改變模型情況下直接抽換最後一層 output layer 不大有意義，因為最後一層 output 加總等於 1 然而 threshold 為 0.4 但若屬於 A、 B 、C Label 則有可能都是 0.33，最後 output 會為空。所以後來我是透過將每一個類別除以類別中最大的機率 (讓機率最大的變成 1) ，透過這種方式丟到 kaggle 上其辨識率為 0.41241，有提升但仍然比 sigmoid 差。

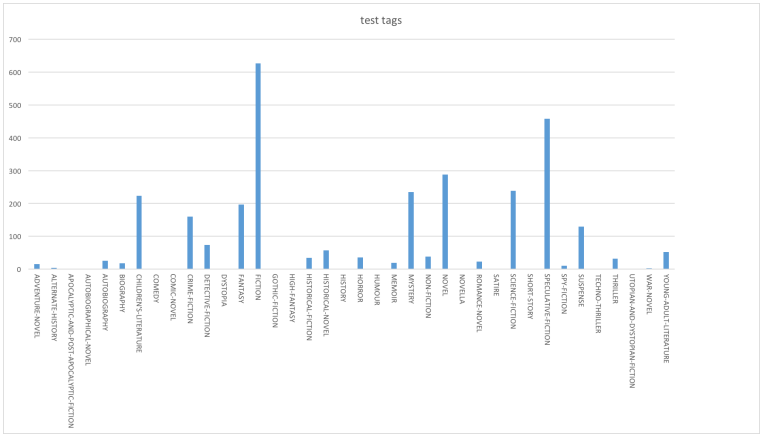
3. (1%)請試著分析 tags 的分布情況(數量)。

答：

train 資料內 tags 的分佈情況



預測 test 出來的 tags 分佈情況



觀察上述兩張圖片可以發現其分佈非常像（tag 在 train 中出現較多次則在 test 中也會出現較多次）。

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：

透過至 Stanford 大學網頁下載 Stanford 大學 NLP 組提供的 GloVe (Global Vectors for Word Representation)

GloVe 是透過整合基於統計 word vector 模型的方法以及基於預測 word vector 模型的非監督式學習方法，其 GloVe 的損失函數為：

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

其中 f 為函數用來降低高頻詞對模型的干擾。

上述方法為 Stanford 如何訓練出 glove.6B.XX.txt (XX 代表維度, ex: 50, 100, 200, 300) 的簡易介紹，要使用此 txt 很簡單。因為此 txt 使用空白分隔，故透過 python split 即可將一行切成 array, array[0] 代表 word, array[1 - XX] 則代表 word vector，即可使用。

5. (1%)試比較 bag of word 和 RNN 何者在本次作業中效果較好。

答：

在實作 bag of word 是紀錄 input 的每個 word 出現次數，並轉成一個維度很大的 vector，作為 input vector，再使用 fully connected neural network 的 model 下去訓練，最終透過此方法訓練出來的 model 丟到 kaggle 結果為 0.46121。而透過 RNN（第 1 題所使用的架構方法）訓練出來的 model 丟到 kaggle 結果為 0.50343。故我在本次作業中使用 RNN 得到的效果會比較。