Machine Learning HW5

學號：R04922169 系級：資工所碩二 姓名：楊智偉

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：

我的 model 設計如下圖表示。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_10 (Embedding)     (None, 306, 100)          5186700
_________________________________________________________________
gru_10 (GRU)                 (None, 256)               274176
_________________________________________________________________
dense_41 (Dense)             (None, 512)               131584
_________________________________________________________________
dropout_31 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_42 (Dense)             (None, 256)               131328
_________________________________________________________________
dropout_32 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_43 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_33 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_44 (Dense)             (None, 38)                4902
=================================================================
Total params: 5,761,586
Trainable params: 574,886
Non-trainable params: 5,186,700
_________________________________________________________________
```

我認為 softmax 不適合作為本次作業的 output layer。softmax 的計算方式是將一個 K 維的任意實數向量 mapping 成另一個 K 維的實數向量，其中向量中的每個元素取值都介於 (0, 1) 之間，且總和等於 1。而這次的問題是 multi-label，如果使用 softmax 的話會將 value 壓縮到總和為 1，這樣就不利於 multi-label 的分類了。

我最後選擇 sigmoid 作為 model 的 output layer，因為 sigmoid 函數的所有 output 都會是介於 (0, 1) 之間，這樣就能得到每一維都是 (0, 1) 之間的 38 維 output，對於此題的 multi-label 會比較好處理。

2. (1%)請設計實驗驗證上述推論。

答：

	Validation F1 Score	Public F1 Score	Private F1 Score
Softmax 1	0.24220	0.25028	0.22373
Softmax 2	0.22672	0.25595	0.23501
Sigmoid 1	0.48160	0.48814	0.45486
Sigmoid 2	0.48503	0.50425	0.48209

我將上述的差別分做「softmax」,「sigmoid」兩組，每組做兩次實驗並紀錄分數。其實在使用 Softmax 來 training 的時候，都會發現 training 的 score 接近 0.48，但是 validation 的 score 則是都在 0.2~0.3 的地方徘徊。

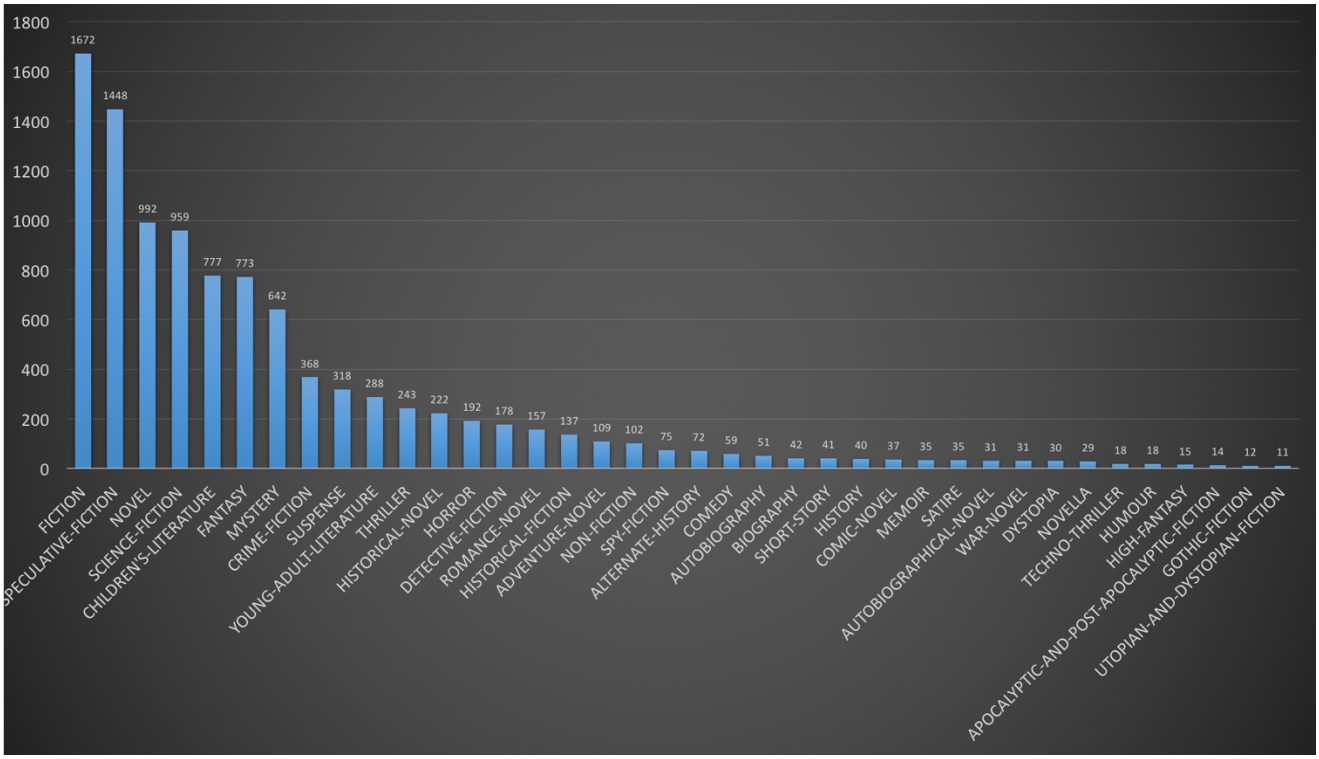
由以上的結果可以得到一個結論，我們在使用 sigmoid function 作為 output layer 會比使用 softmax 的效果還要好得多。

3. (1%)請試著分析 tags 的分布情況(數量)。

答：

我將所有 data 中的 tags 統計如下表，出現頻率比較高的是 Fiction 和 Speculative-Fiction，接著還有 Novel, Science-Fiction, Children's-Literature, Fantasy, Mystery，剩下的 tags 都是出現次數比較少的部分。

由此統計表可以發現，其實 data 中的大部分 tags 都是集中在某幾個大分類之中而已，所以我認為可以將 training, testing 的步驟中特別處理這些分類（例如使用不一樣的 weights），也許能夠有比較好的分類結果。



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：

這次的作業我使用 GloVe 作為 word embedding 的方法，它是屬於 Count based 的做法。GloVe 是使用一個 global log-bilinear 的 regression model，主要是由兩種 model 的優勢結合而成，分別是「global matrix factorization」以及「local context window methods」。它們的 model 是利用了 statistical

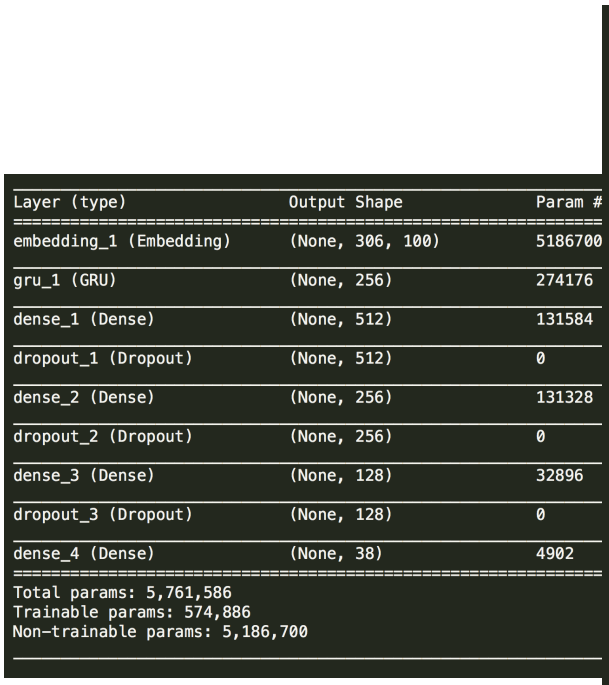
information 的資料，model 只使用 word-word co-occurrence matrix 中的 non-zero element 有效率的來 training。而作業中我用到的 GloVe 是 100 維度的 data 來建立 embedding dictionary, embedding matrix。

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

答：

在我的測試當中，我覺得 RNN 的效果都比 bag of word 來得好。bag of word 的方法很容易 over fitting，在 validation set 只有 0.3~0.4 準確率的時候 training set 能夠跑到 0.6~0.7 甚至還有看過 0.9 的情況。而且我的 bag of word 還沒有辦法輕鬆過 simple baseline，而 RNN 的方法只要稍微 tune 一下參數，讓程式多跑幾次就能拿到較高的分數。

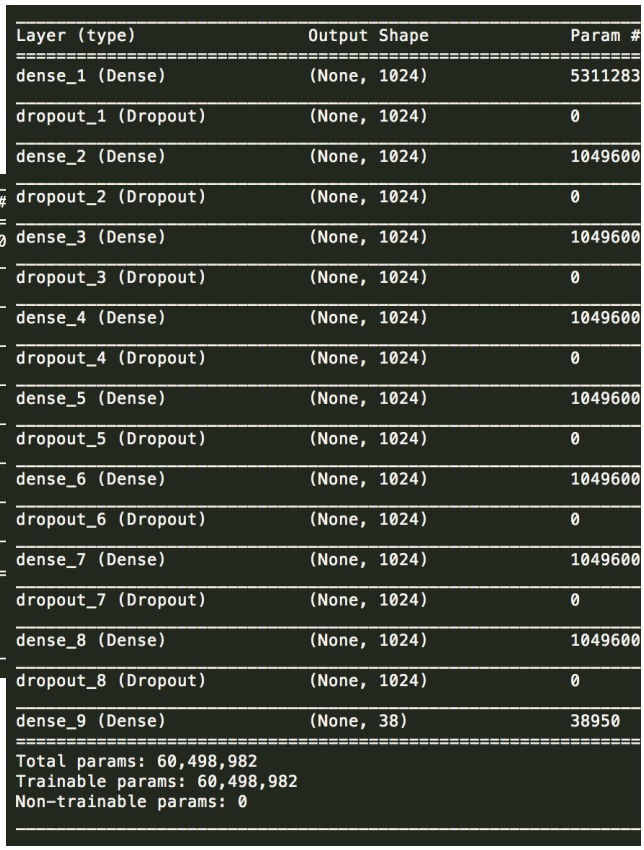
以下是我 RNN、bag of word 的架構圖。



Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 306, 100)	5186700
gru_1 (GRU)	(None, 256)	274176
dense_1 (Dense)	(None, 512)	131584
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131328
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
dropout_3 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 38)	4902

Total params: 5,761,586
Trainable params: 574,886
Non-trainable params: 5,186,700

RNN model



Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1024)	5311283
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1024)	1049600
dropout_2 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 1024)	1049600
dropout_3 (Dropout)	(None, 1024)	0
dense_4 (Dense)	(None, 1024)	1049600
dropout_4 (Dropout)	(None, 1024)	0
dense_5 (Dense)	(None, 1024)	1049600
dropout_5 (Dropout)	(None, 1024)	0
dense_6 (Dense)	(None, 1024)	1049600
dropout_6 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 1024)	1049600
dropout_7 (Dropout)	(None, 1024)	0
dense_8 (Dense)	(None, 1024)	1049600
dropout_8 (Dropout)	(None, 1024)	0
dense_9 (Dense)	(None, 38)	38950

Total params: 60,498,982
Trainable params: 60,498,982
Non-trainable params: 0

bag or word (bow)

接下來是實際執行的實驗記錄。我針對 RNN 以及 bag of word 兩種方式做實驗，兩組我都各用了 5 次來評估。其實使用 bag of word 的效果還蠻好的，而且他的實行速度飛快，都已經接近了 simple baseline，但是 bag of word 的方法忽略了 words 之間的前後關係，只剩下「有沒有這個 word」的資訊。至於使用 RNN 的方法，就不會忽略這些資訊，只需要多 tune 一下參數，多跑幾次的 training, testing 結果就能夠有好的分數通過 simple baseline 了！實驗的分數記錄如下表：

	Public F1 Score	Private F1 Score
RNN 1	0.47770	0.47429
RNN 2	0.49544	0.46433
RNN 3	0.48478	0.48352
RNN 4	0.48426	0.47297
RNN 5	0.50628	0.48615
bow 1	0.45601	0.44715
bow 2	0.46277	0.45400
bow 3	0.47009	0.46556
bow 4	0.47557	0.47485
bow 5	0.47126	0.45421