學號：R04945008 系級： 生醫電資碩二 姓名：黃思翰

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。

本次作業探討的為一 Multti label & Multi class 的問題，合計共有 38 個 class，因此在設計 RNN 的 output layer 時，我使用了 38 個 neuron 來進行最終的分類。而 activation function 使用 softmax 時因會使所有輸出的機率相加為一，找出機率最高的來進行分類，當 data 僅屬於一 class 時使用 softmax 並不會有問題，但是當 data 屬於多重 class 時，就可能會出現明明屬於兩個不同的 class 但機率因為總和要為一的關係而使兩個 class 機率都降低的狀況。因此我 activation function 改用了 sigmoid 來代替，假設每個 class 都有相同的重要性，則經過 NN 運算後機率高的就判斷為此 class。

2. (1%)請設計實驗驗證上述推論。

為了要驗證上述假設，我比較於 output layer 使用不同的 activation function 時所得到的 validation data 的 F1 score 及 loss。而我的 RNN 架構如下，input layer 為一 pre-trained 的 glove word embedding，使用 wikipedia 100 維的 model。接著使用 hidden size 180 的 GRU，起始的 weight 使用了 Glorot uniform initialization，recurrent 的參數則是使用常態分佈 N(0, 0.01)，activation function = tanh，L2 norm(λ =0.001)，dropout = 0.35。再接上 3 層的 full-connected NN，activation function = relu，dropout = 0.35。最後使用 38 個 neuron 的 output layer，而所進行的實驗則是使用相同的 model 僅改變 activation function 來觀察訓練的結果。Optimizer 部分皆使用 adam (lr=0.00125,decay=1e-6,clipvalue=0.2)

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 306, 100)	5186700
gru_1 (GRU)	(None, 180)	151740
dense_1 (Dense)	(None, 256)	46336
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8256
dropout_3 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 38)	2470

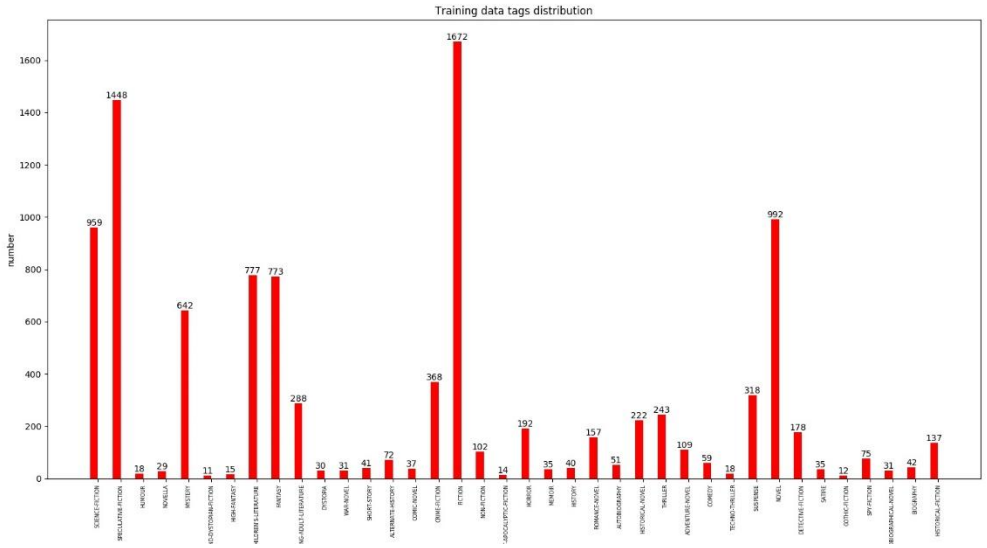
Total params: 5,428,398.0
Trainable params: 241,698.0
Non-trainable params: 5,186,700.0

實驗結果如下，可以觀察到 softmax 與 sigmoid 間 validation data 的 F1 score 有巨量的差異，而 sigmoid 很明顯的不論是 loss 或是 F1 score 皆比其餘的 activation function 好，顯見在 multi class 的問題中應使用 sigmoid。

Activation function	softmax	sigmoid	elu	relu	tanh
Val_F1_Score	0.0557	0.4826	0.2615	0.2513	0.1248
Val_loss	5.7457	4.9593	24.79	9.3699	15.635

3. (1%)請試著分析 tags 的分布情況(數量)。

將 training data 的所有 tag 切出來加總分析後如下圖所示，共有 38 個類別，其中數量佔最多的前三名分別為：Action(1672 個)、Speculative-Fiction(1448 個)、Novel(992 個)，而第四名的 Science-Fiction 也有 959 個。而數量最少的則是 Utopian-And-Dystopian-Fiction 僅有 11 個。



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

本次作業使用了 GloVe(Global Vectors for Word Representation)預先訓練好的 100 維 model 來進行 word to vec 的轉換。其原理如下，觀察每段 window 內取出的各個字在全部的文章內出現的頻率，利用上下文的內容來表現各個單詞，建出一 cooccurrence 矩陣但因為此矩陣為一超高維度的矩陣，利用 SVD base 的 latent semantic analysis(LSA)方法來進行向量分解並定義出個別的詞向量。

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

除了第一題&第二題的 RNN 模型外，此次作業我也試了 bag of word 的模型，與 RNN 架構最大的差別在於 training data 處理方式(word embedding 的方式)，bag of word 我使用了 tokenizer 的 tfidf 模式，會依據文章中字出現的頻率來評估字的重要性來建出 word to vec，轉成詞向量後，通過 3 層的 NN，neuron 數量從 256 遞減至 64，activation function 皆為 relu，dropout = 0.35，output layer 內含 38 個 neuron，activation function 為 sigmoid，optimizer 為 Adam(lr=0.001,decay=1e-6,clipvalue=0.5)。

而使用 RNN 和 Bag of word 於 validation data 得到的 F1 score 和 loss 如下，以 F1 score 來看的話，RNN 勝出，然而實際 train 時可以很明顯觀察到 bag of word 訓練起來快上許多也比較快收斂，RNN 則是滿慢的，犧牲速度換來準確性吧。

	RNN	Bag of word
Val_F1_Score	0.4995	0.4991
Val_loss	4.7626	5.3679
Kaggle F1 Score	0.5105	0.4932