學號：B02404002 系級： 醫技四 姓名：葛竑志

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。

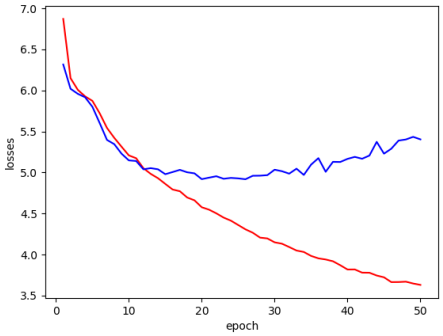
1. 建立的 model 架構：

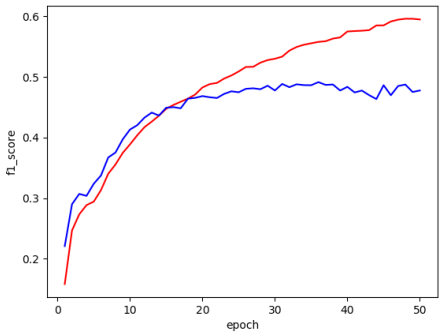
GRU: 128 (activation= ‘tanh’) -> Dropout:0.2 -> Dense:256 (activation= ‘relu’) -> Dropout:0.2 -> Dense:128 (activation= ‘relu’) -> Dropout:0.2 -> Dense:64 (activation= ‘relu’) -> Dropout:0.2 -> Dense:38 (activation= ‘sigmoid’)。

2. 由於題目做「可具有 multi-label」的 multi-class classification，相當於在此 38 個 label 中各自做 binary 的 classification，來判斷文章的大綱適合編類在哪種類型的小說。Sigmoid 適用在 binary classification 的情形，其假設是各個分類之間是獨立關係，這個假設同樣適合在這種我們對於 articale classification 不具有 domain knowledge 的情形，而 output layer 的每一個 element 都是對該指向特定 label 的分類之 classification，因此可加上 threshold 去篩出最佳解答。Softmax 雖然有適用於 multi-class classification 的情形，但僅限在選出最佳解的問題，意即 label 只能有一個，顯然無法在此題目中顧及具有 multi-label 的 data，再者，Softmax 有將整個 output layer 的總和限制在 1 的條件，且無對各個分類之間的獨立假設，因此我認為不適合用在此處。

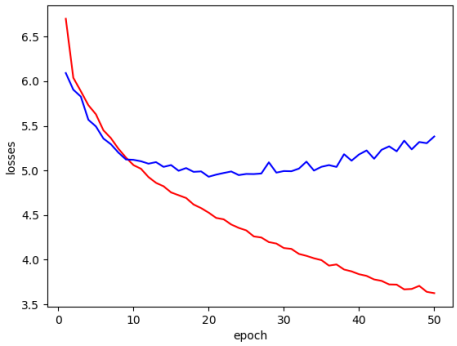
2. (1%)請設計實驗驗證上述推論。

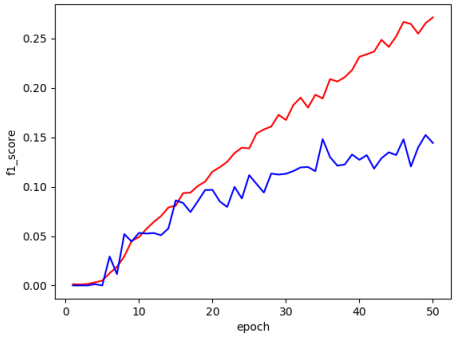
1.使用上題架構(with Sigmoid output layer)做出來的 categorical_loss 與 f1_score（紅線：training data，藍線：Validation data）





2. 使用同樣架構與相同參數(with Softmax output layer) 做出來的 categorical_loss 與 f1_score（紅線：training data，藍線：Validation data）





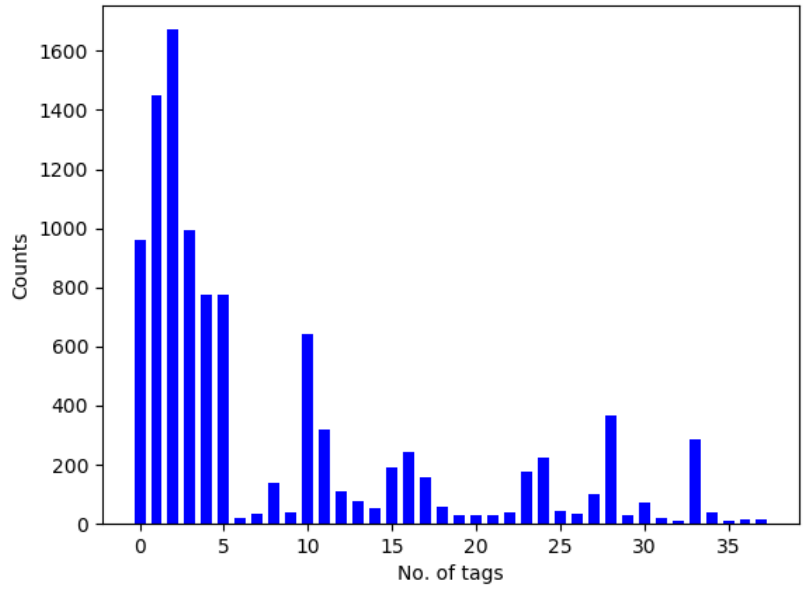
3. 觀察：

(1)兩者在 categorical_loss 上的差異不大，大約在 epoch= 30 時，到達 loss 為 5.0 時即會收斂。

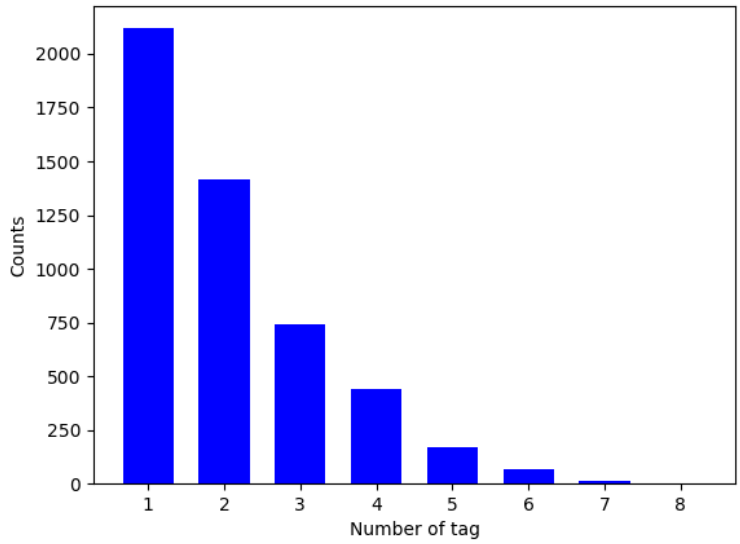
(2)在 f1_score 的表現上，Sigmoid 可以達到 0.49，而 Softmax 在 0.14 的位置劇烈震盪且 learning 的速率較慢，因此 Sigmoid 的表現較 Softmax 為佳。

(3)兩者在僅改變 output layer 的情況下，loss 皆能夠達到差不多的低點，代表如此改變對 model 本身不會造成顯著的影響而 train 好或 train 壞。然而僅在 loss 上出現明顯差異，可能代表是 ouput layer 的問題。推測是在最後判斷 tag 是否要列進去時，採用到的 threshold 定義（在此處是單一 element 超過 0.4 即列入）可能較適合使用在 Sigmoid 的情況中，而 Softmax 因為有 regulation 存在，導致應有複數個 tag 的 data 無法選擇出最適合的 tag 做 output。（觀察 Softmax 的 prediction 可以看到有約三分之二的 data 並沒有出現 tag，data not shown。）

3. (1%)請試著分析 tags 的分布情況(數量)。



統計 Train_data 的出現 tag 的次數，做統計如上圖。（x-axis：將 38 種 tag 做 0-based labeling，y-axis：出現次數）。出現次數最高的前五者依序為：FICTION、SPECULATIVE-FICTION、NOVEL、SCIENCE-FICTION、CHILDREN’S-LITERATURE。



統計同一筆 data 會出現多少個 tag，做統計如上圖。（x-axis：Tag 數量，y-axis：出現次數）。一筆資料最常出現只有一個 tag 的情形，而 data set 中一筆資料最多有 8 個 tag。

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

本次作業中使用的是 glove，屬於 count based method。

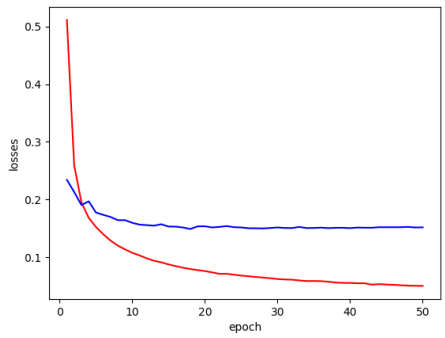
1. 建立 word-word co-occurrence counts matrix，一個 element 代表一個特定的 word 在有另一個特定的 word 的 window 中出現的次數，這個概念與 latent semantic analysis 是很類似的。

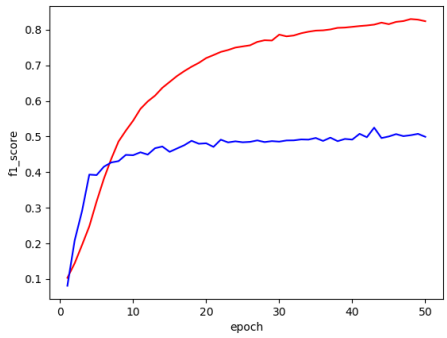
2. 在 co-occurrence probabilities 的基礎下，從相對機率(ratio)的概念去區分不相干與相對的 words。

3. 為了處理相對 rare 而 noisy 的 words，使用一個自定義的 weighted least squares regression model 加到 cost function 內，在使用 gradient descent 的方法去得到最適合的 vector。

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

1. Bag of word model 架構：
   (1)Preprocessing：將 training 與 testing data stack 成 all_corpus -> NLTK-Stop words (‘english’)去除掉功能性普遍、相對不具意涵的字 -> Keras Tokenizer 去 fit corpus 獲得 word index，產生 47905 個 vocabulary -> texts_to_sequences -> sequences to matrix (mode= ‘tfidf’) -> 切割成 training、testing、validation data set。
   (2)DNN model：
   Dense: 256 (activation= ‘relu’) -> Dropout:0.2 -> Dense:128 (activation= ‘relu’) -> Dropout:0.3 -> Dense:64 (activation= ‘relu’) -> Dropout:0.3 -> Dense:32 (activation= ‘relu’) -> Dropout:0.3 -> Dense:38 (activation= ‘sigmoid’)。
2. Optimizer 參數：
   Adam(lr=0.00125,decay=1e-4,clipvalue=0.2)
3. BOW 訓練過程：使用上題架構(with Sigmoid output layer)做出來的 binary_loss 與 f1_score（紅線：training data，藍線：Validation data）





   從 f1_score 來看，可以看到在 epoch= 50 時，Training set 可以達到 0.8，而 Validation set 也可以達到 0.51。與題目一的 loss 和 f1_score 的曲線比較，可以看到 BOW 比較不容易 overfit，達到收斂的速度也較快。此外，在調整 RNN model 的參數時，由於 loss plane 很陡峭，會發現 learning rate 必須要微幅調整，否則會造成 loss 一直震盪無法收斂，BOW model 則震盪幅度不大。
4. 結論：與題目一比較，認為 BOW model 比較容易 train 到較佳的分數。