學號：R05943139 系級：電子碩一 姓名：張育瑄

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。

我的 model 架構如下：

```
21
def build_model(input_dim, output_dim, embedding_layer):
    sequence_input = Input(shape=(input_dim, ))
    embedded_sequence = embedding_layer(sequence_input)
    x = GRU(512, dropout=0.5, return_sequences=False)(embedded_sequence)
    x = Dense(256, activation="relu")(x)
    x = Dropout(0.5)(x)
    x = Dense(128, activation="relu")(x)
    x = Dropout(0.5)(x)
    prediction = Dense(output_dim, activation="sigmoid")(x)

    model = Model(sequence_input, prediction)

    model.compile(
        optimizer=Adam(), loss="categorical_crossentropy", metrics=[f1_score])

    return model
```

我最後一層是 output 38 維的 NN，每一個 output 都是 binary，代表“有”或是“沒有”這個 tag，因此在我的架構下，使用 softmax 是不合理的。

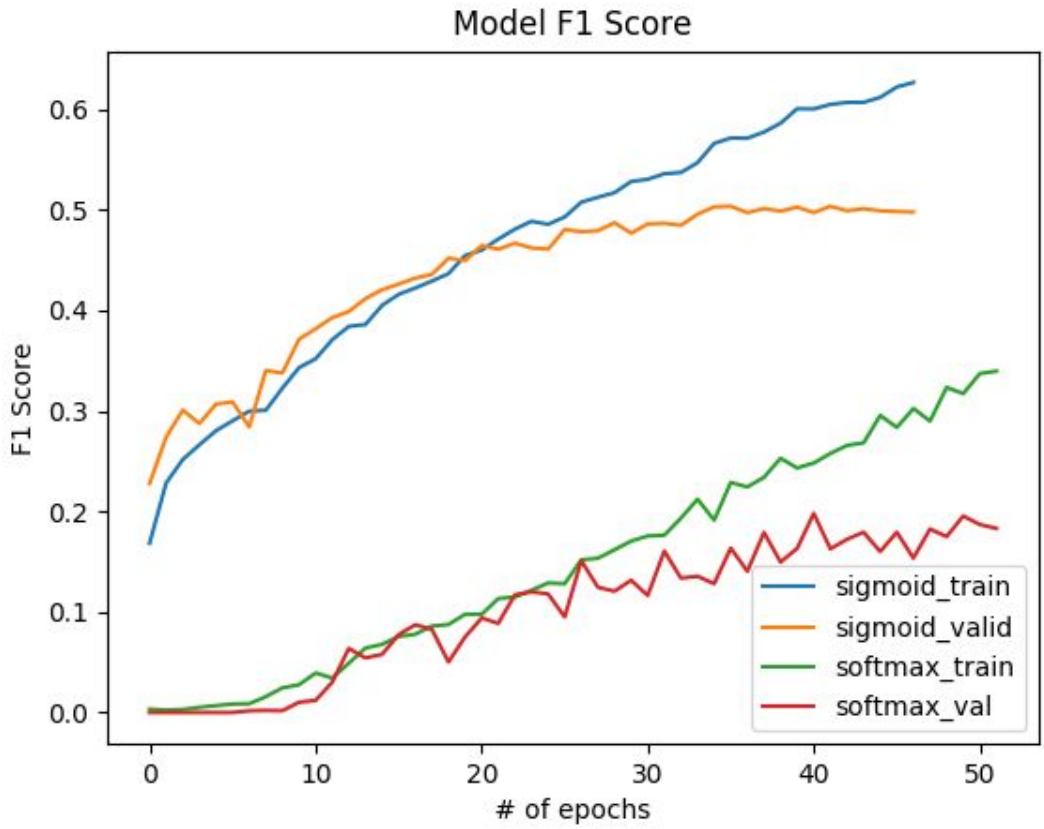
所以最後我選擇 sigmoid 作為 output 的 activation function。

最後 Kaggle 上成績最好的是 train 五個 model 然後做 bagging 得出。

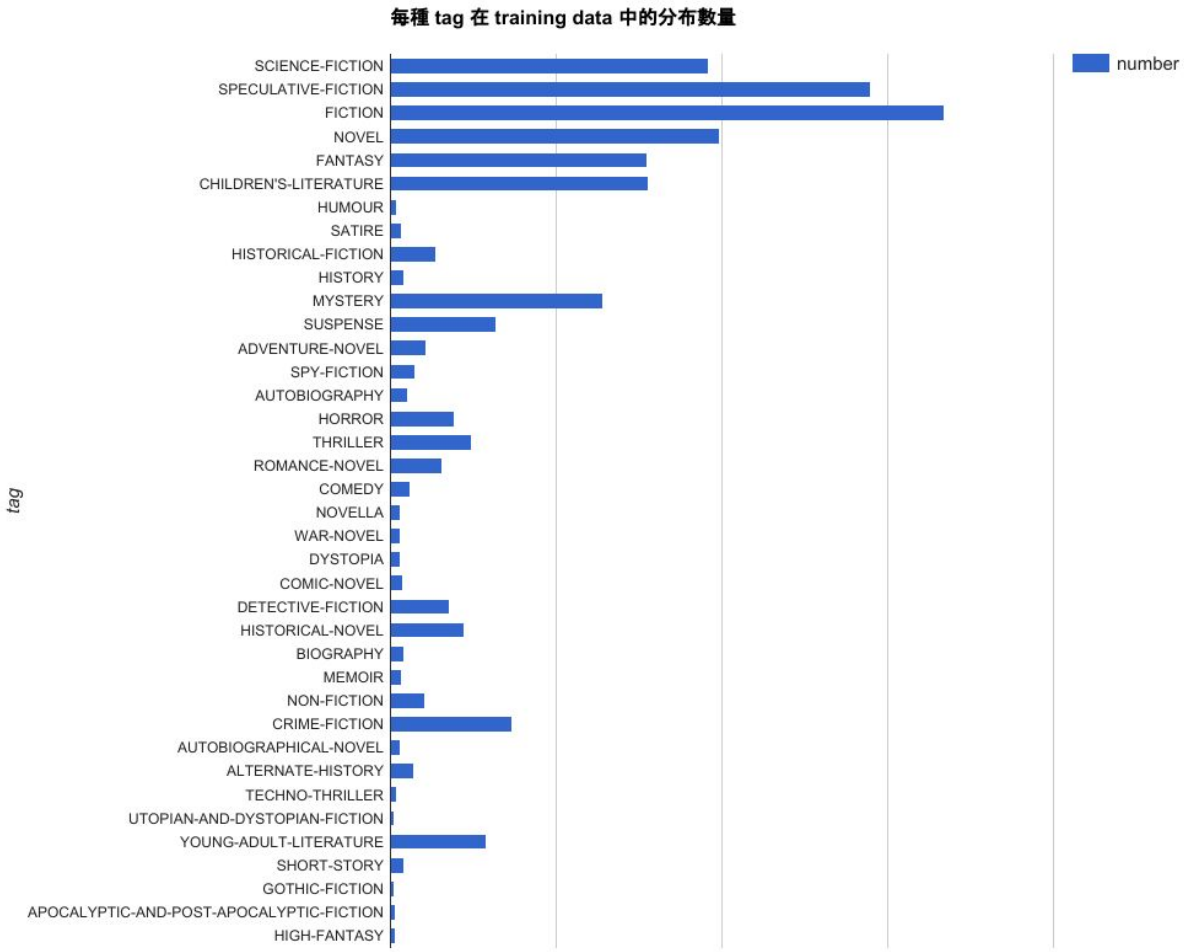
2. (1%)請設計實驗驗證上述推論。

若把 sigmoid 改為 softmax，則 performance 會差很多。

下圖為使用 sigmoid 及 softmax 的 training history 比較：



3. (1%)請試著分析tags的分布情況(數量)。



由圖表可以看出 tag 在 training data 中的分布極不平均，前六大的 tag 便佔了 65%，因此數量稀少的 tag 幾乎不會在 testing data 上被 predict 出來。

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

我使用 GloVe 作為 word embedding 的方法，並從其官網下載已經 gen 好的 word vectors。

跟之前作業使用的 word2vec 套件中的 CBOW 或是 Skip-gram 方法不同，GloVe 是 count-based 的 model。

它的步驟主要是先建出 input corpus 中的 words 的 co-occurence counts matrix，然後做 normalize 跟 log-smoothing，最後再對其做 dimension reduction 產生出 word vector。

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

```
19
1  def build_model(input_dim, output_dim):
2      bow_input = Input(shape=(input_dim, ))
3      x = Dense(1024, activation="relu")(bow_input)
4      x = Dropout(0.5)(x)
5      x = Dense(512, activation="relu")(x)
6      x = Dropout(0.5)(x)
7      x = Dense(256, activation="relu")(x)
8      x = Dropout(0.5)(x)
9      prediction = Dense(output_dim, activation="sigmoid")(x)
10
11     model = Model(bow_input, prediction)
12
13     model.compile(
14         optimizer=Adam(), loss="categorical_crossentropy", metrics=[f1_score])
15
16     return model
```

Bag of words 相比起來 train 的速度較快，但 performance 差一點。

直覺上來看，bag of words 不像 RNN 會利用到詞彙順序的資訊，當然 performance 就會差一點，但反過來說，其實文章的類型大部分只需要幾個 keywords 便可大概猜出，因此 bag of words 的方法其實也不會太差。

下圖為使用 RNN 和 bag of words 的 traing history 比較：