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:

我最後使用 sigmoid 作為最後的 output layer，我認為在分類文章屬性時，每一種屬性應為獨立判斷，例如 Fiction 與 Thriller 並無對立關係，一篇文章屬於 Fiction 並不會使其屬於 Thriller 的可能性增加或減少，因此我認為應該用 sigmoid 較合適。

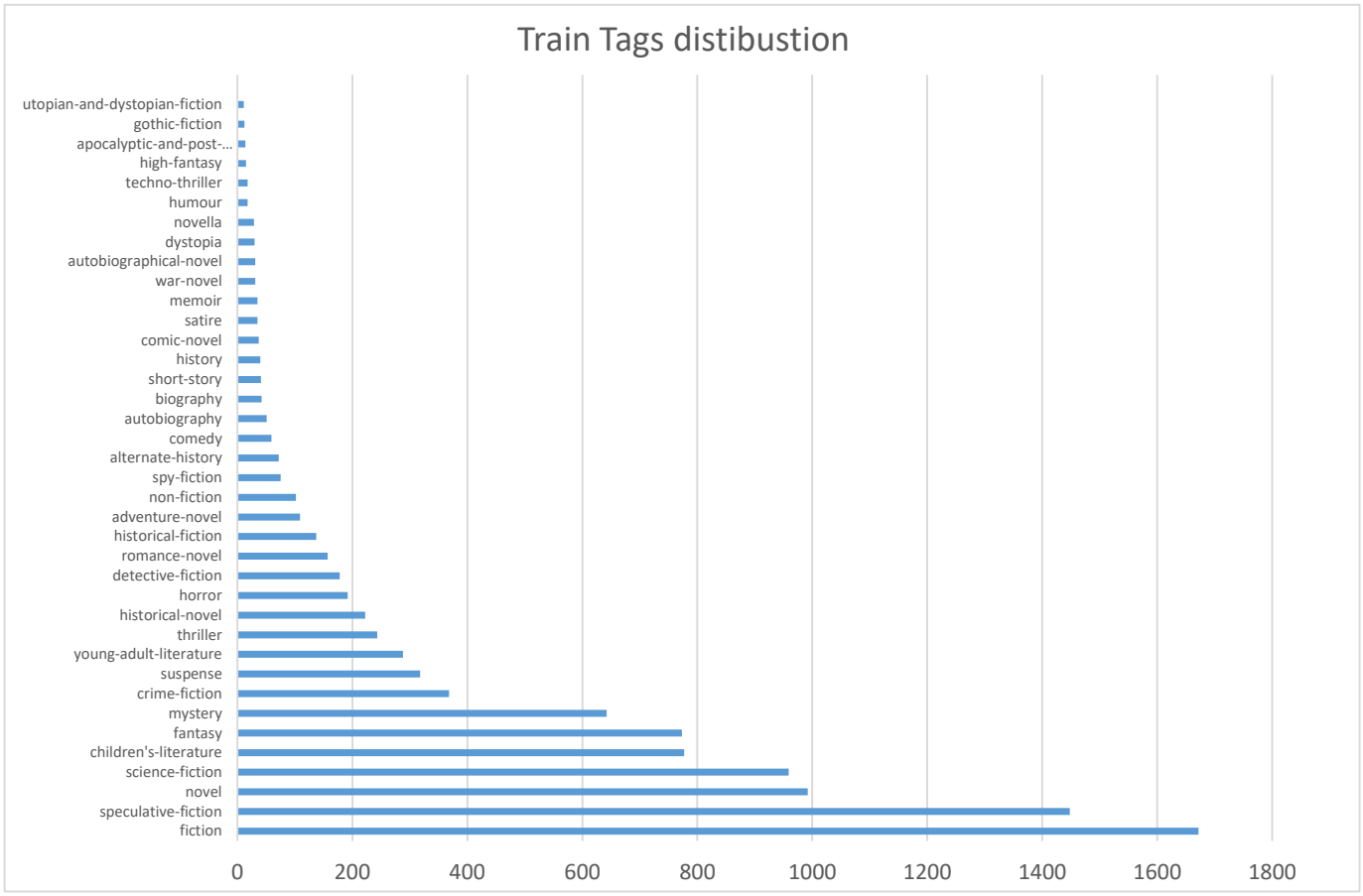
2. 請設計實驗驗證上述推論：

我維持 model 的層數與每一層的 cell 數量，僅將最後的 sigmoid 改成 softmax，本地 f1_score 從 0.4 降到 0.35，kaggle 則從 0.52 降到 0.46。由於 softmax 會機率分布總和為一，因此最後 predict threshold 也往下調低許多，儘管如此還是 sigmoid 做出來的結果較佳

```
model = Sequential()
model.add(embedding_layer)
model.add(GRU(512, dropout = 0.4, recurrent_dropout = 0.4, return_sequences=True, implementation=2))
model.add(GRU(256, dropout = 0.4, recurrent_dropout = 0.4, return_sequences=True, implementation=2))
model.add(GRU(128, dropout = 0.4, recurrent_dropout = 0.4, return_sequences=False, implementation=2))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(39, activation='softmax'))
model.load_weights("temp.hdf5")
opt = optimizers.rmsprop(lr = 0.001, clipvalue=0.5, decay=1e-6)
model.compile(loss='binary_crossentropy', optimizer=opt, metrics = [fscore])
```

3. 請試著分析 tags 的分布情況(數量):

統計訓練資料的 tags 分布可以發現出現次數最高的種類為 fiction，其次為 speculative-fiction，最少者為 gothic-fiction 與 utopian-and-dystopian-fiction。平均一篇文章會有兩個 tag 左右。



4. 本次作業中使用何種方式得到 word embedding?請簡單描述做法:

一開始我嘗試從 training data 和 test data 做 word2vec，然而做出來的結果並和 keras 內建的 embedding layer 效果相差無幾(甚至更差一點)。因此改用 Glove 的”glove.42B.300d” 向量集。Glove 本身已經是 vector 形式的 dictionary，每一個字詞都有對應的 vector，只要將我處理過的 input 字詞對應之 vector 放入 embedding_matrix，即可做為 model 中的 embedding layer 來訓練。

5. 試比較 Bag of Word 和 RNN 何者在本次作業中效果較好：

以下是我的 Bag of Word 模型，將訓練資料做 text to sequence 之後再做 sequence to matrix，以此為 input。採用四層 hidden layer，ADAM optimizer，本地端測試最高 f1_score 約 0.37，上傳 kaggle 後為 0.46，比 RNN 低上一些。我認為主要原因是 Bag of Word 無法考慮字跟字之間前後文之關係，只能憑詞頻作判斷，故無法達到更精準的預測。

```
model = Sequential()
model.add(Dense(4086, activation='relu',input_dim=3000))
model.add(Dropout(0.5))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(39, activation='sigmoid'))
model.load_weights("temp.hdf5")
opt = optimizers.adam(lr = 0.0003)#, clipvalue=0.5, decay=1e-6)
model.compile(loss='binary_crossentropy', optimizer=opt, metrics = [fscore])
```