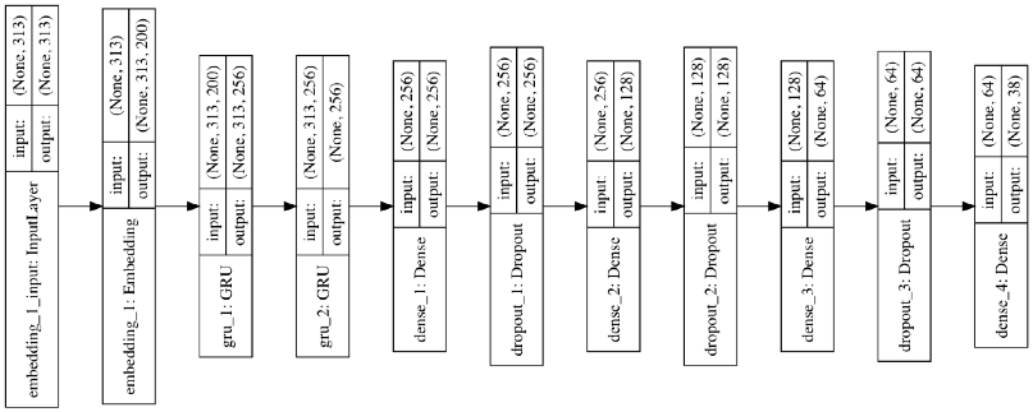
學號：B03901039　系級：電機三　姓名：童寬

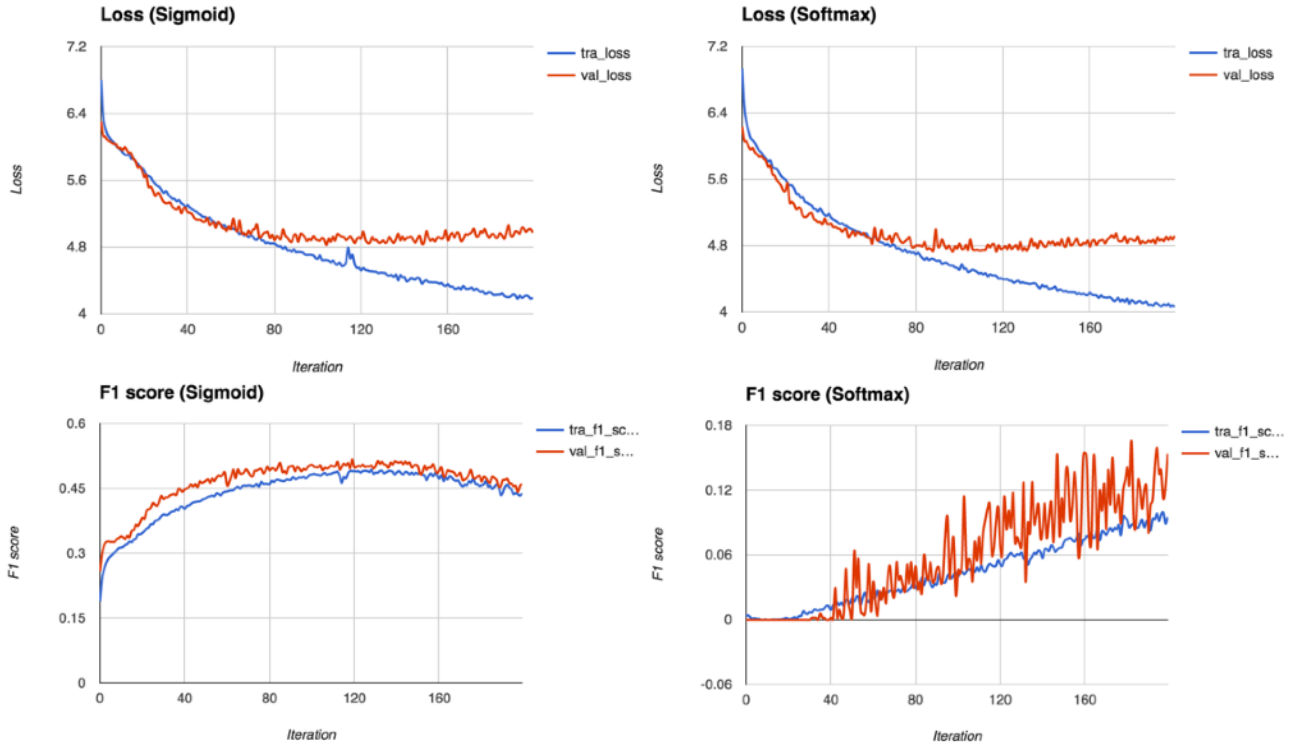
1. (1%) 請問 softmax 適不適合作為本次作業的 output layer？寫出你最後選擇的 output layer 並說明理由。

不適合。我最後選擇 sigmoid 作為 output layer 的 activation function。Softmax 會把最後 output 變成機率分佈，也就是總和為 1，但這並不適合這次的問題。這次的問題屬於 multi-class，同一筆資料可能同時屬於很多個種類，也就代表最後的 output 有可能會是同時有好幾個 1 的向量，softmax 將總和 normalize 到 1 的特性不適合用在這裡。Sigmoid 相對而言就很適合，因為它會盡可能把每個 output 都分到 0 跟 1 兩邊，而這正是這個問題需要的。

2. (1%) 請設計實驗驗證上述推論。

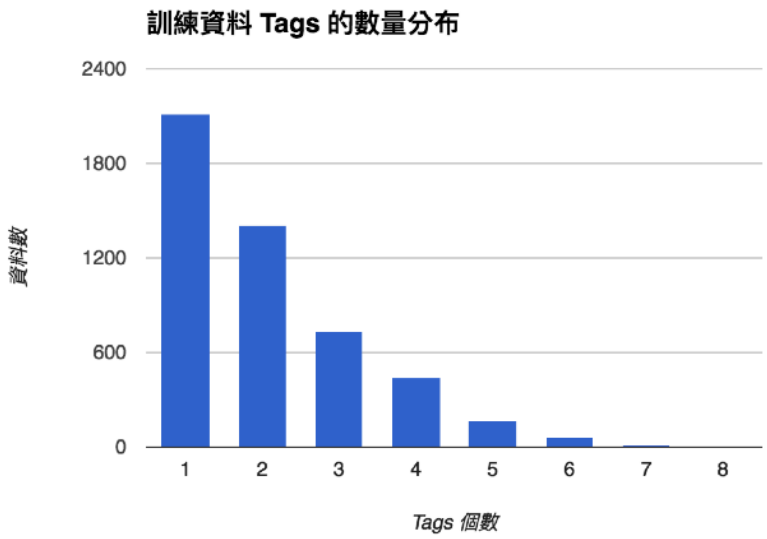


我以上圖的模型進行兩個實驗，兩者的差別只有在最後的 Dense layer 用不同的 activation function，分別為 sigmoid 跟 softmax，訓練過程如下(左圖為 sigmoid，右圖為 softmax)：



- 取 validation 最好的去做 testing 結果為：0.51188 (Sigmoid)，0.16016 (Softmax)
- 使用 Softmax 的模型，雖然 loss 跟使用 Sigmoid 的差不多，但 F1 score 卻差很多，因為在最後判斷屬不屬於該種類是以 0.5 作為 threshold，Softmax 的 output 基本上不可能出現好幾個 0.5。

3. (1%) 請試著分析 tags 的分布情況(數量)。



- 越少 Tags 的資料量越多
- 有兩筆資料有最多的 Tags 數 (8)

訓練資料 Tags 種類分布			
種類	個數	種類	個數
ADVENTURE-NOVEL	109	HORROR	192
ALTERNATE-HISTORY	72	HUMOUR	18
APOCALYPTIC-AND-POST-APOCALYPTIC-FICTION	**14**	MEMOIR	35
AUTOBIOGRAPHICAL-NOVEL	31	MYSTERY	642
AUTOBIOGRAPHY	51	NON-FICTION	102
BIOGRAPHY	42	NOVEL	992
CHILDREN'S-LITERATURE	777	NOVELLA	29
COMEDY	59	ROMANCE-NOVEL	157
COMIC-NOVEL	37	SATIRE	35
CRIME-FICTION	368	SCIENCE-FICTION	959
DETECTIVE-FICTION	178	SHORT-STORY	41
DYSTOPIA	30	SPECULATIVE-FICTION	**1448**
FANTASY	773	SPY-FICTION	75
FICTION	**1672**	SUSPENSE	318
GOTHIC-FICTION	12	TECHNO-THRILLER	18
HIGH-FANTASY	15	THRILLER	243
HISTORICAL-FICTION	137	UTOPIAN-AND-DYSTOPIAN-FICTION	**11**
HISTORICAL-NOVEL	222	WAR-NOVEL	31
HISTORY	40	YOUNG-ADULT-LITERATURE	288

- 總共 38 個種類
- FICTION 跟 SPECULATIVE-FICTION 是最常見的種類
- UTOPIAN-AND-DYSTOPIAN-FICTION 跟 APOCALYPTIC-AND-POST-APOCALYPTIC-FICTION 是最少見的種類

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

我是使用別人訓練好的 word embedding (Glove 的 200 維 Wikipedia 2014 + Gigaword 5)，它是以 global log-bilinear regression model 根據字跟字在 global 出現的次數去訓練的，它能兼顧 LSA 在統計全部 corpus 上所保存的訊息跟 local context window 在字跟字類比關係的優點 。取得方法如下：

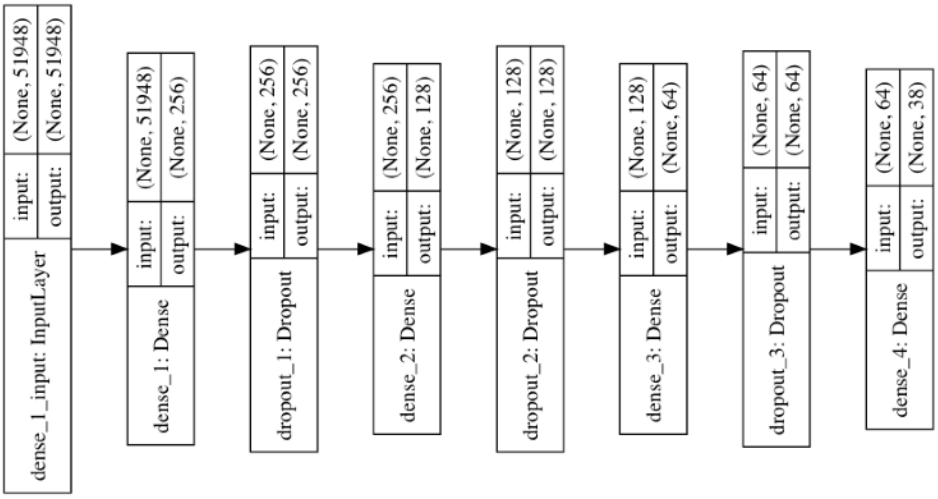
1. 下載 Glove 的 word embedding (glove.6B.zip)
2. 將 word embedding 讀進來 (glove.6B.200d.txt)
3. 宣告一個矩陣將訓練資料裡的字 map 到對應的 word vector
4. 用 Keras 的 Embedding layer 當作模型的輸入層，並將 weights 設為上一步宣告的矩陣
5. 將經過 Keras 的 tokenizer 處理過的訓練資料送進去，就能在這一層被轉成對應的 word vector

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

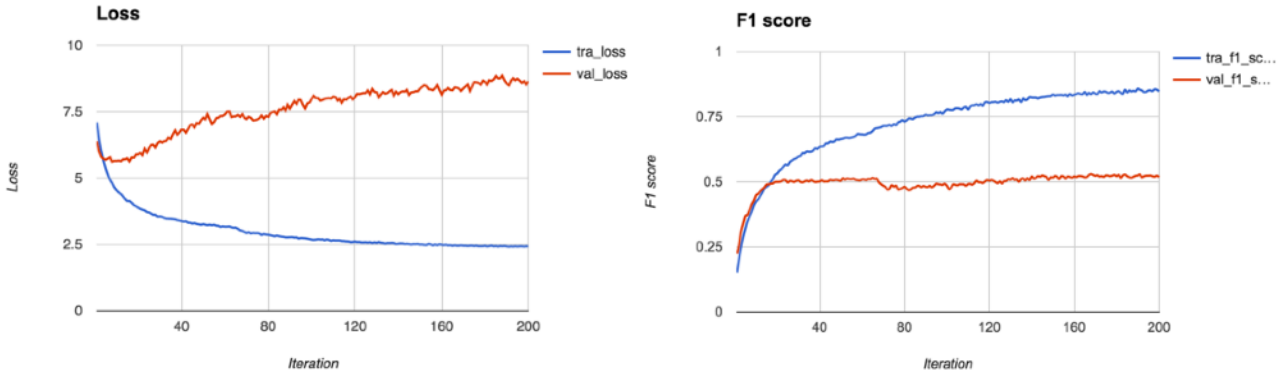
我分別以兩種方式進行訓練 (模型的參數量盡量接近，約 1200 萬個)，訓練方式跟結果如下：

Bag-of-words

用 Keras tokenizer 的 sequences_to_matrix (mode = tfidf) 將每一筆 summary 轉換成單一 matrix，每次都將整個 matrix (bag of words) 送進模型訓練。下圖是模型架構：



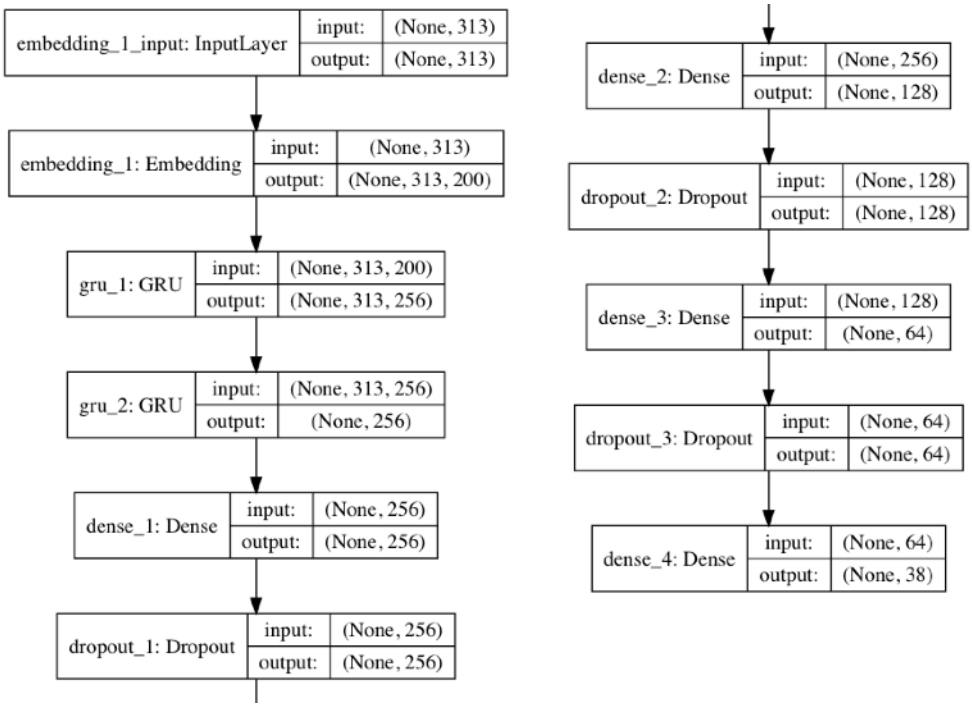
訓練過程：



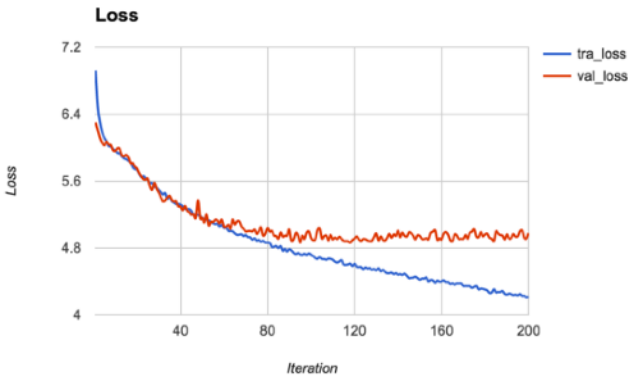
- 選 validation 最好的去做 Testing，F1 score (Kaggle 的結果) 為 0.48719
- Validation 的 loss 一直都下不來，但 f1 score 還不錯，在 0.5 左右振盪

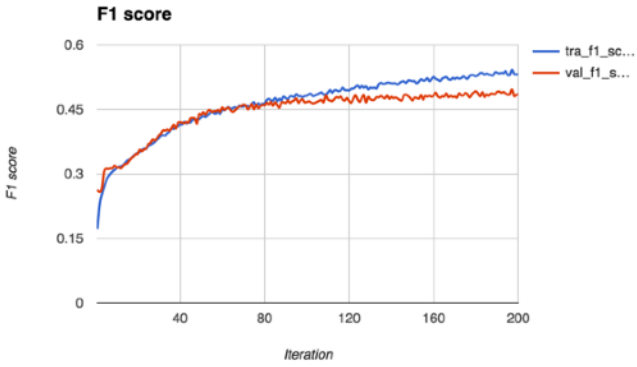
RNN

用 Keras 的 GRU (Gated Recurrent Unit) 當作 RNN 的 unit。透過 Embedding layer 將一個一個字的 word vector 送進模型裡。下圖是模型架構：



訓練過程：





- 選 validation 最好的去做 Testing，F1 score (Kaggle 的結果) 為 0.51196
- Validation loss 不像 bag-of-words 那樣一直下不來，直到 80 iteration 才開始停止下降

結果討論

- RNN 的效果會稍微好一點，但 bag-of-words 也不會太差
- 我覺得這樣的結果是合理的，因為這個分類問題，並不需要太多時間歷史的資訊，因此使用 RNN 沒辦法看到非常明顯的優勢