

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。

Softmax 的方程式裡頭會將所有的 label 依照權重分配比值，而這些筆直的加總為 1，代表著其他 label 的權重都會影響到關鍵 label 的比值。因此很難給定一段文字到底該有幾個 label，因為同樣是 fiction 或 novel 為 dominate label，有些文字獲得 0.4 的權重，有些只有 0.2，因此很難給定參數。我使用的是 sigmoid 當作 activation function，就沒有 softmax function 那樣每個 label 互相牽制的問題，在 kaggle 上的效果也較好。

2. (1%)請設計實驗驗證上述推論。

我的 GRU 的 Model 如下，當 activation 為 sigmoid 時，kaggle 分數可達 0.5132。

```
model.add(GRU(304,activation='tanh',recurrent_dropout=0.4,dropout=0.4))
model.add(Dense(304,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(152,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(76,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(38,activation='sigmoid'))
```

我另外一個 train 了一個 model 把 activation function 改為 softmax，threshold 一樣是 0.4，kaggle 上的成績只有 0.3135。從結果發現大部分的的文章都只有一個 label，至多兩個，有些甚至連 label 都沒有。從這裡就可以看出他把 label 的權值加總為 1 是個不大可靠的方法。

3. (1%)請試著分析 tags 的分布情況(數量)。

簡單寫了一個 word count 分析 testing data tags 的分布如下圖左:

而 training data tags 的分布如下圖右:





```
In [24]: word_count(s1)
Out[24]: {'ADVENTURE-NOVEL': 15,
'ALTERNATE-HISTORY': 10,
'AUTOBIOGRAPHY': 19,
'BIOGRAPHY': 6,
'CHILDREN'S-LITERATURE': 237,
'CRIME-FICTION': 161,
'DETECTIVE-FICTION': 60,
'FANTASY': 272,
'FICTION': 829,
'HISTORICAL-FICTION': 46,
'HISTORICAL-NOVEL': 92,
'HISTORY': 2,
'HORROR': 60,
'MEMOIR': 8,
'MYSTERY': 243,
'NON-FICTION': 37,
'NOVEL': 370,
'ROMANCE-NOVEL': 20,
'SCIENCE-FICTION': 259,
'SPECULATIVE-FICTION': 525,
'SPY-FICTION': 28,
'SUSPENSE': 152,
'THRILLER': 67,
'WAR-NOVEL': 4,
'YOUNG-ADULT-LITERATURE': 45}

Out[42]: {'ADVENTURE-NOVEL': 109,
'ALTERNATE-HISTORY': 72,
'APOCALYPTIC-AND-POST-APOCALYPTIC-FICTION': 14,
'AUTOBIOGRAPHICAL-NOVEL': 31,
'AUTOBIOGRAPHY': 51,
'BIOGRAPHY': 42,
'CHILDREN'S-LITERATURE': 777,
'COMEDY': 59,
'COMIC-NOVEL': 37,
'CRIME-FICTION': 368,
'DETECTIVE-FICTION': 178,
'DYSTOPIA': 30,
'FANTASY': 773,
'FICTION': 1671,
'GOTHIC-FICTION': 12,
'HIGH-FANTASY': 15,
'HISTORICAL-FICTION': 137,
'HISTORICAL-NOVEL': 222,
'HISTORY': 40,
'HORROR': 192,
'HUMOUR': 18,
'MEMOIR': 35,
'MYSTERY': 642,
'NON-FICTION': 102,
'NOVEL': 992,
'NOVELLA': 29,
'ROMANCE-NOVEL': 157,
'SATIRE': 35,
'SCIENCE-FICTION': 959,
'SHORT-STORY': 41,
'SPECULATIVE-FICTION': 1448,
'SPY-FICTION': 75,
'SUSPENSE': 317,
'TECHNO-THRILLER': 18,
'THRILLER': 242,
'UTOPIAN-AND-DYSTOPIAN-FICTION': 11,
'WAR-NOVEL': 31,
'YOUNG-ADULT-LITERATURE': 288}
```

可以輕易看出 Testing data 中電腦判讀出最大宗的 label 為 Fiction, Speculative-Fiction, Novel。在 Training data 中 Fiction, Speculative-Fiction, Novel 也是最多，因此在判讀上電腦有較多機會學習到 Fiction novel 的文章寫作樣本。在 Training 時有出現的 Label 在 Testing label 確沒出現的，例如 Comedy 還有 Dystopia 顯示了 Comedy 或者 Dystopia 的用字遣詞可能還沒有足夠的樣本數讓電腦能夠理解這類型的文章的雛型架構(其實 Dystopia 很多人也沒辦法分辨出來)。

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

本次作業我使用了助教提供的 glove 來當我的 word embedding，其原理有二。其中一個原理是 Matrix Factorization，就如同老師上課所講的涼日春宮公仔與阿宅的兩個屬性(如下圖)，只是在這之中 ABCDE 代表了關鍵文字，各動漫公仔代表了各種 label。

				
A	5	3	0	1
B	4	3	0	1
C	1	1	0	5
D	1	0	4	4
E	0	1	5	4

另一個方法是使用 Shallow Window-based，其觀念比較像之前作業四做的 word2vec 的練習，把有關聯的文字聚集在一起討論他們的屬性。他們還有用到

像 woman-man = queen-king 這樣的文字類比方式，整體來說這是一個十分好的架構，所以檔案才會那麼大。

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

用 RNN 的 GRU 來 train 我試了許久才達到 validation set 大於 0.5，但用 bag of words 來 train，隨便丟 2200 個字再用 5 層 Dense Layer 就可以讓 validation set 超過 0.5 的程度。且 Bag of word 訓練起來速度超快，每個 epoch 只要 3 秒，如果沒有助教提供的 RNN 訓練的 sample code，一般好上手的方式可能還是以 bag of words 較簡單。