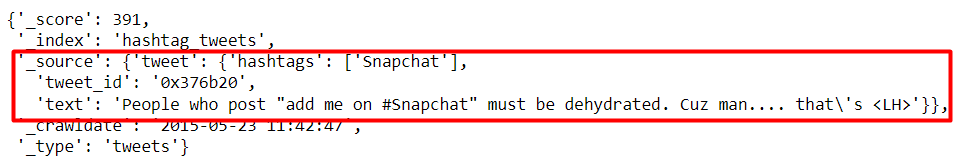
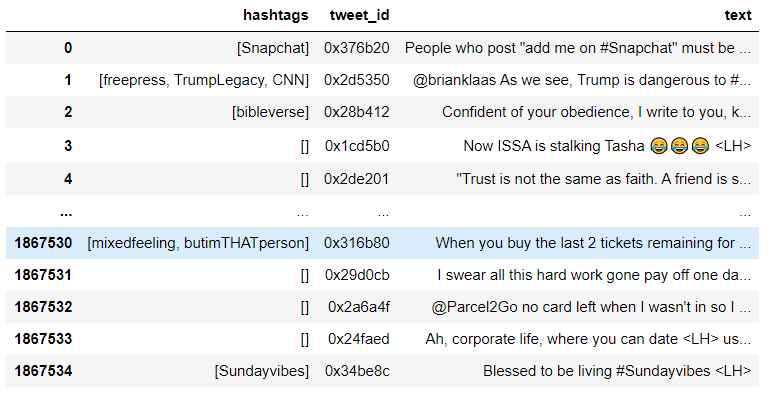
Data Mining Lab2

Kaggle competition report

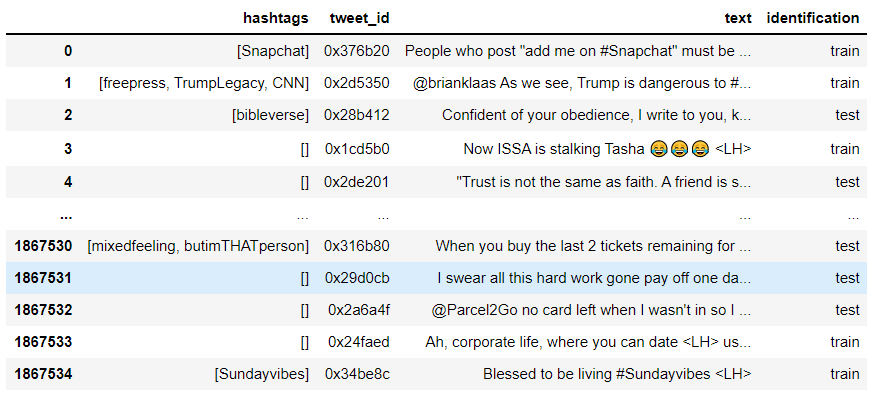
* **Data preparation (code in “Second\_part\_data\_preparation.ipynb”)**

首先在”tweets\_DM.json”裡，我只保留”source”的部分，圖為其中一筆。

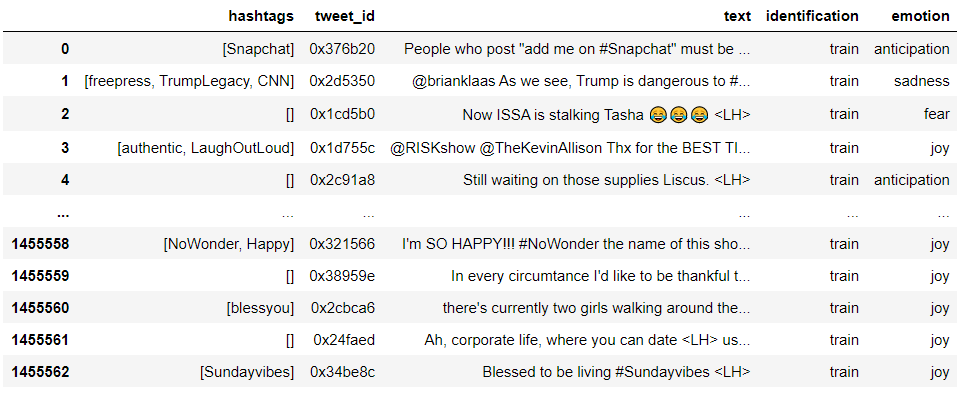
另外在”source”當中，也只保留”tweet\_id”, “hashtags”, “text”的部分。



透過”tweet\_id”，利用data\_identification.csv把資料分成training set和test set。



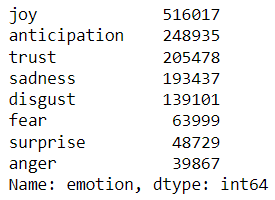
也透過”tweet\_id”，將emotion.csv併入training set裡標上label。



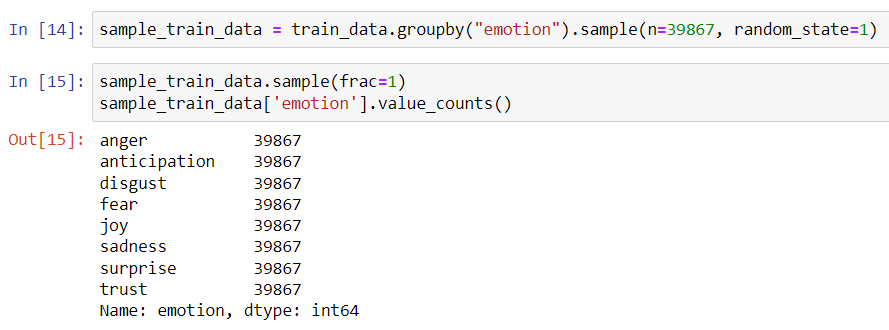
最後把training set和test set分別存成pkl檔，方便之後讀取。

* **Data preprocess**

在training set中，我發現資料有不平均的現象。



**我一開始有讓每個category用相同數量去訓練，但表現都變差，因此捨去此種方法。**

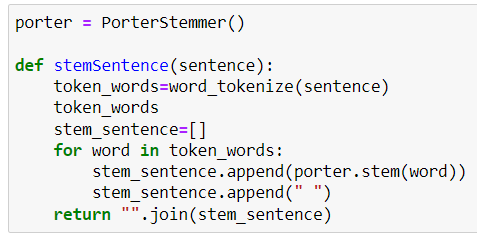


接下來決定model的input樣式，我有嘗試三種:

1. TFIDF with stemmer processing
2. Self-training W2V
3. Pretrained W2V model (glove-twitter 27B 100d)
   * TFIDF with stemmer processing:

Stemmer可以讓word還原成最基本的樣子，例如hid -> hide, children->child。

先將sentence tokenize，再把每個word丟進PorterStemmer進行stem。

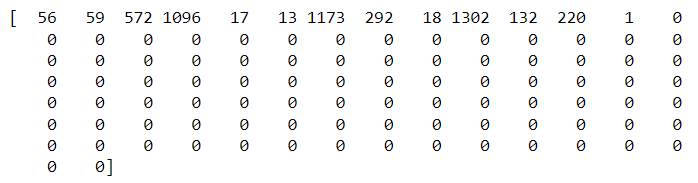


Stem完後，再利用TFIDF tokenizer計算TFIDF。

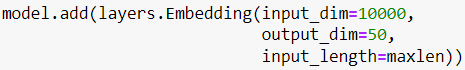
* + Self-trainging W2V:

我是利用keras的Tokenizer進行，Tokenizer會把所有sentence看過一遍，整理出由每個unique word組合而成的list，再將sentence裡出現過的word轉換成對應的index。



但每個sentence各有不同長度，model需要統一長度的input，這邊利用keras的pad\_sequences做padding。我設定長度為100，並在長度不滿100的句子後面補上0。

最後在model的第一層利用embedding layer進行w2v的訓練。因為我設Tokenizer的num\_word=10000，所以input維度為10000，input\_len為100，而output維度我則是設50。

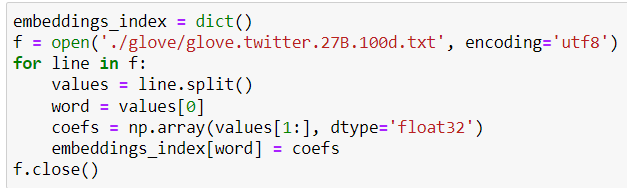


* + Pretrained W2V model (glove-twitter 27B 100d):

我利用glove的glove-twitter 27B 100d來做W2V，裡面有高達27 billion的word，而每個word會轉換成100 維度的vector。

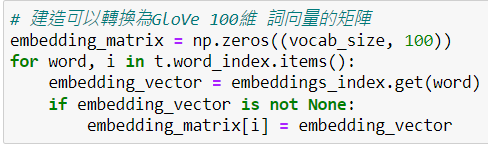
首先讀取glove準備好的txt並處理成可應用的形式。

embeddings\_index是一個dict，key為word，value為其轉換後的100d vectors。

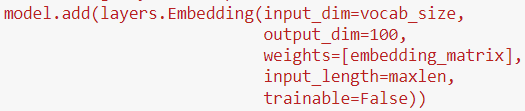


接下來先將每個sentence做tokenized，並做padding，如同之前所述。

最後我們要創造一個可以進行W2V的矩陣，row為tokenizer算出來的word size，column為100。依照每個word，將他對應到的vector整理成矩陣。那如果真的沒有對應到27 billion裡的任何一個word，就是0。



在model的embedding layer中，因為我們已經使用pretrained好的W2V model，所以trainable=False，而這邊多一個weights參數，就是剛剛整理出來的W2V矩陣。



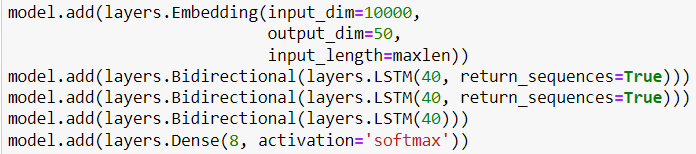
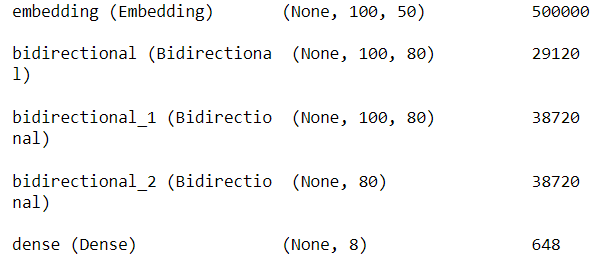
**這三種input轉變模式，最終是Self-trainging W2V通常都會有比較好的結果，因此捨棄另外兩個。**

轉變完後，進行train test split方便檢視模型表現，test size為0.03。

另外也需要對label進行one hot encoding，把每個label變成 1\*8 的矩陣。

* **Model training**

**模型訓練的部分，由於資料集過大，傳統的機器學習方法都沒獲得較好的結果，這邊直接跳至deep learning進行。**

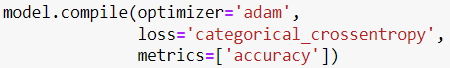
 

我是用**雙向的LSTM**進行訓練。LSTM是一種RNN的變化，RNN的特點是會考慮到循序序列的特徵，他會將hidden layer的output儲存在memory，當下一筆進來的時候，可以將這個output再拿來一起計算，達到考慮循序關聯的特點。

句子是由多個word組合而成，每個word都是互相關聯的，因此非常適合RNN這種模型進行訓練。而雙向LSTM是讓模型除了往前掃過句子，也可以將句子往反方向推論，這樣讓模型訓練到每個word的前後關係。

當LSTM訓練完成，最後套一層dense network來分類出8個emotion。

下面為hyper-parameter的參數設定



Epoch為20，batch\_size設10。

* **Conclusion**

在這次的訓練中，有許多較為反常的前處理讓我非常訝異，像是讓模型input的category用相同數量做計算會有比較差的結果，或是利用pretrained好的W2V model進行也是，在模型的部分，我也有嘗試加dropout層讓model不要過度overfit，表現也是降低，總總跡象讓我覺得其實稍微overfit是可以在競賽中取得較好的成績。

另外能夠大幅進步的方法也可以嘗試BERT進行訓練，並對BERT進行fine-tune。