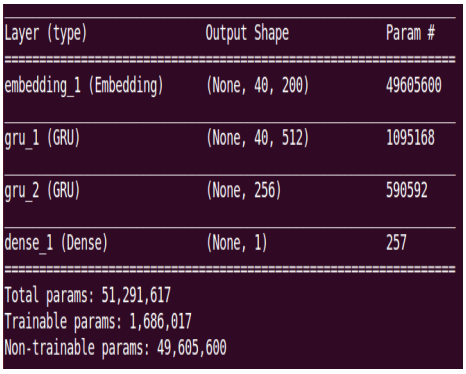
作業五 Text Sentiment Classification

學號：R06921002 系級： 電機所碩一 姓名：張哲誠

1. (1%) 請說明你實作的 RNN model，其模型架構、訓練過程和準確率為何？ (Collaborators: **吳睿哲**，幫忙檢查 parsing 的錯誤與提供 Word2Vec 想法) 答：

模型架構一：



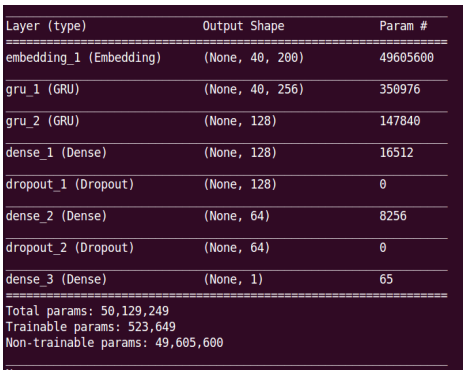
Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 40, 200)	49605600
gru_1 (GRU)	(None, 40, 512)	1095168
gru_2 (GRU)	(None, 256)	590592
dense_1 (Dense)	(None, 1)	257

Total params: 51,291,617
Trainable params: 1,686,017
Non-trainable params: 49,605,600

model1.hdf5

參數一欄表	
Dropout rate	0.4
Batch_size	512
epoch	50
Training_val_accuracy	0.82973
Kaggle_accuracy	0.82854

模型架構二：



Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 40, 200)	49605600
gru_1 (GRU)	(None, 40, 256)	350976
gru_2 (GRU)	(None, 128)	147840
dense_1 (Dense)	(None, 128)	16512
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
dropout_2 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 1)	65

Total params: 50,129,249
Trainable params: 523,649
Non-trainable params: 49,605,600

model2.hdf5

參數一欄表	
Dropout rate	0.4
Batch_size	512
epoch	80
Training_val_accuracy	0.83098
Kaggle_accuracy	0.82837

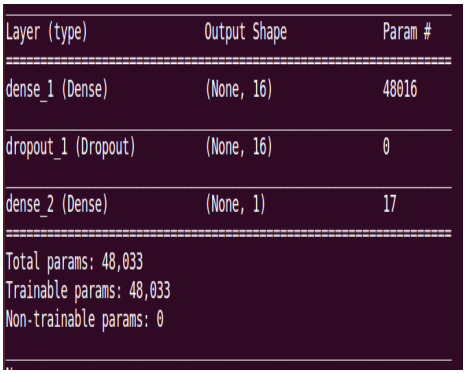
說明：

本次作業程式部分有參考助教的寫法(Sample Code)。除此之外，training 時我將 label_data 與 nolabel_data 丟進 tokenizer 裡面當作我的字典，在 padding 時把每句話的長度都補成向量長度 40。我還利用了 gensim 提供的 Word2Vec 做出一個 pretrained 好的 embedding_matrix，並把這個 matrix 丟進我模型的 embedding 層，並設定 trainable = false。標點符號部分，model1.hdf5 是沒有濾除、然而 model2.hdf5 有濾除(參考本次作業程式檔：hw5.py, util.py)

2. (1%) 請說明你實作的 BOW model，其模型架構、訓練過程和準確率為何？

答：

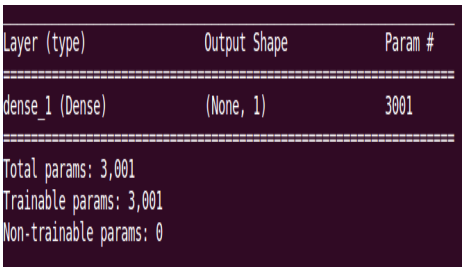
模型架構一：



model_bow_1.hdf5

參數一欄表	
Dropout rate	0.5
Batch_size	512
epoch	30
Training_val_accuracy	0.78685

模型架構二：



model_bow_2.hdf5

參數一欄表	
Dropout rate	0.5
Batch_size	512
epoch	30
Training_val_accuracy	0.78400

說明：

字典的作法與上題相同，也就是將 label_data 與 nolabel_data 丟進 tokenizer 裡面當作我的字典。但與上題不同的是，由於如果不限制 token 的大小，我的電腦會有記憶體不足(Memory Error)的問題，因此我有額外設定字典的最大字數為 3000，也有設定 filters 把標點符號濾除。將 tokenizer 建置好後，利用 test_to_matrix 的方式將每一句話都變成一個長度為 3000 (max_word)的向量，mode 為 tfidf。最後輸入到我的 model 做 training。(參考本次作業程式檔：hw5_bow.py, util_bow.py)

3. (1%) 請比較 bag of word 與 RNN 兩種不同 model 對於"today is a good day, but it is hot"與"today is hot, but it is a good day"這兩句的**情緒分數**，並討論造成差異的原因。

答：

本題我分別使用了 RNN 與 BOW 的第一個 model 來做測試，測試程式為 3.py (model1.hdf5 和 model_bow_1.hdf5)。

	RNN	BOW
today is a good day, but it is hot	0.30716628	0.5910252
today is hot, but it is a good day	0.979761	0.5910252

從 RNN 的分數來看，我認為是因為模型會考慮到”一句話順序”的關係。通常人類講出這句話時，我們習慣性會比較強調 but 後面的那句話，而相較於”today is a good day”，”it is hot”是較屬於負面的詞，因此若是”it is hot”在 but 的後面，RNN 會把他判斷成負面，反之亦然。然而 BOW，因為不會考慮到文字的順序問題，因此分數都是一樣的。

4. (1%) 請比較"有無"包含標點符號兩種不同 tokenize 的方式，並討論兩者對準確率的影響。

答：

此題的做法是在 tokenizer 的 initialize 中加上 filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~'如此濾掉標點符號，這樣在把每句話變成一個 vector 的時候就會因為找不到標點符號而用 0 去做 padding。其餘做法都跟第一題相同。

本實驗挑選與第一題的 model1.hdf5 同樣的模型架構，並調整 tokenizer 的標點符號濾波器後作訓練，結果如下表：

	model1.hdf5	model1(無標點符號)
Training validation accuracy	0.82973	0.83127
Kaggle accuracy	0.82854	0.82777

雖然在 kaggle 上的表現不如預期，但在 training 上卻有了進展。這個結果出乎我的意料之外，由於這個 dataset 來自 twitter，而我認為標點符號在打字中多少會透露出人類的情緒，而從情緒也會影響到這句話是正面還是反面。因此理論上，我認為有加標點符號是會影響到判斷結果而且是較精確的。但從 training 的準確度看起來，卻是沒有標點符號的準確率較高，看來標點符號也是有可能會造成判斷上的雜訊。

5. (1%) 請描述在你的 semi-supervised 方法是如何標記 label，並比較有無 semi-surpervised training 對準確率的影響。

Ans：

首先我先設定十個迴圈，迴圈的一開始都會先用 train 好的 model 對 semi_data 做標記的動作(predict)，接下來設定 2 個 epoch 做 training，每做完一次 training 就在 load 一次最新的 model，並且結束迴圈。如此一來，每結束一次迴圈，系統就會拿到一個剛更新過後的 model，並且重新 predict semi_data，並在做一次 training。

本次我同樣使用 model1.hdf5 當作 model 先匯入到程式內部，並且用此 model 先對 semi_data 做預測，接著重新 training 並記錄我的 model。以下為我 semi-supervised 在 training 與 kaggle 上的結果

	model1.hdf5	Model_semi.hdf5
Training validation accuracy	0.82973	0.82935
Kaggle accuracy	0.82854	0.82708

從結果上來看，不論是在 validation 的 accuracy 或者是在 kaggle 上的分數，都是經過 semi-supervised 的結果比較差，這個結果也出乎意料。我原先認為，dataset 變多了有可能可以讓整體的準確度上升；但從結果看起來，我想如果是本來 model 的強健性不夠，那麼在 label semi-data 的時候就會有一些問題，如此一來錯的 ground truth 拿來做 semi-supervised 的結果會變差、或者變好的程度有限，也是合理的。

本次作業參考網站：

(1) RNN model (助教提供)：https://ntumlta.github.io/2017fall-ml-hw4/RNN_model.html

(2) Using word2vec: http://www.orbifold.net/default/2017/01/10/embedding-and-tokenizer-in-keras/