ML hw4 report

學號：b07901112　　系級：電機二　　姓名：劉聿珉

1. (1%) 請說明你實作的 RNN 的模型架構、word embedding 方法、訓練過程(learning curve)和準確率為何？ (盡量是過 public strong baseline 的 model)

   我的 RNN 架構為：

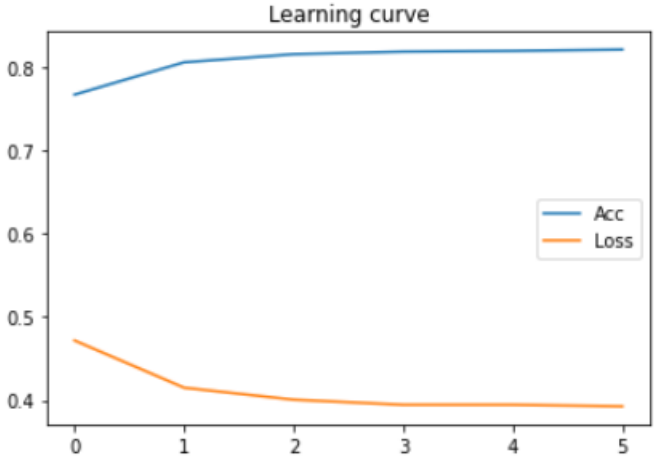
   sen_len = 35
   fix_embedding = True
   batch_size = 128
   epoch = 10
   lr = 0.0005

   我的 LSTM 模型參數為：

   embedding_dim=250,
   hidden_dim=250,
   num_layers=5,
   dropout=0.7，
   RNN 有使用 biderectional。

   word embedding 的方法則是使用 word2vector 函式，Word2Vec 的參數如下：size = 250, window = 5, min_count = 5, workers = 12, iter = 10, sg = 1。

   Learning curve 如圖：(此圖為 val 的 Acc 和 Loss)



   Kaggle 上最好的 val acc 為 82.538。

   經由多次實驗可以發現 Val Acc 每次大約在 3 次 epoch 之後就會飽和或是開始下降。但是 Training Acc 則會繼續上升，但是之後的 train 就對正確率沒有太大的幫助。

2. (2%) 請比較 BOW+DNN 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"這兩句的分數(過 softmax 後的數值)，並討論造成差異的原因。
   我 BOW+DNN 跑出來的分數兩句都是 0.61152，而 RNN 跑出來的分數第一句是 0.645332 第二句則是 0.97456。
   BOW 是 Bag of word 的簡寫，在此模型下會將一段文本（比如一個句子或是一個文檔）用一個裝著這些詞的袋子來表示，所以他完全不會考慮到任何的文法或是語句順句的問題，所以用 BOW+DNN 跑這兩句話的分數當然獲一樣(因為他們的字詞都一樣只是排序不同而已)。
   但是 RNN 的情形不一樣，之前的輸入會影響到接下來的輸出，他有類似人腦一樣的記憶功能，所以前後文會影響到最後的分數，而就這兩句來看，他們的雖然字一樣但是排序不一樣，所以丟到 RNN 裡面之後，分數自然就會不一樣。
3. (1%) 請敘述你如何 improve performance（preprocess、embedding、架構等等），並解釋為何這些做法可以使模型進步，並列出準確率與 improve 前的差異。（semi supervised 的部分請在下題回答）
   Biderectional：BRNN 的基本想法是每一個訓練列向前和向後分別是兩個 RNN，而且這兩個都連線著一個輸出。這樣的結構可以提供輸出層的輸入列完整的過去和未來的上下文資訊，使輸出的點不會只看見向前的文字。
   Word embedding：加入 nolabel data，可以讓 embedding 時的 data 變多，參考的 data 也變多，以得到比較好的結果。
   Trainning 時做 clipping：加 clipping 可以讓 model 在 train 時如果踩到懸崖可以不至於爆掉，但是但部分的時間都沒有發揮作用，所以我最後式沒有加。
   參數們：
   sen_len：一次考慮更長的句子可以得到更多的資訊，得到更完整的句子，但是如果調太長可能會稀釋掉每個字詞的重要性這樣反而會使正確率降低。
   hidden_dim, num_layers：我將這兩個參數調大，試圖 train 更多參數，以達到更好的結果，但在這同時，我也將 dropout 調大，這樣可以避免 overfitting。
4. (2%) 請描述你的 semi-supervised 方法是如何標記 label，並比較有無 semi-supervised training 對準確率的影響並試著探討原因（因為 semi-supervise learning 在 labeled training data 數量較少時，比較能夠發揮作用，所以在實作本題時，建議把有 label 的 training data 從 20 萬筆減少到 2 萬筆以下，在這樣的實驗設定下，比較容易觀察到 semi-supervise learning 所帶來的幫助）。
   這題的架構我是用第一題所用到的 model 下去改的，幾乎所有的參數都維持不變。在第四題的 model 中我設的 high_threshold 為 0.8，low_threshold 為

0.2，在第一次用有 label 的 data train 完一次之後，便拿他去測試那些沒有 label 的 data，並將分數超過 0.8 的 data 設成 1 低於 0.2 就設成 0，然後把界於 0.8 到 0.2 的 data 砍掉，經過這樣的步驟之後我們有 label 的 data 就變多了，於是將這及果得到的 data 再拿去 train 一次。我發現在沒有使用 unlabel 的 data 的狀況下，val acc 大約在 76.3422，train 第二次之後就變成了 77.4086，正確率有提高一些，但是沒有我想像中的顯著，我想如果 label 以及 trainning 的次數可以再多幾次或許可以拿到更好的結果(也有可能是我 threshold 參數調得不好)。

semi-supervised 的想法就像是用已經 train 好的 model 再對一群沒有 label 的 data 重新 label 一次，再把那些 data 拿來用，這樣我們就得到更多 data，更有機會得到正確率更高的結果。這種方法通常在有 label 的 training data 相對少的時候會有比較顯著的效果。