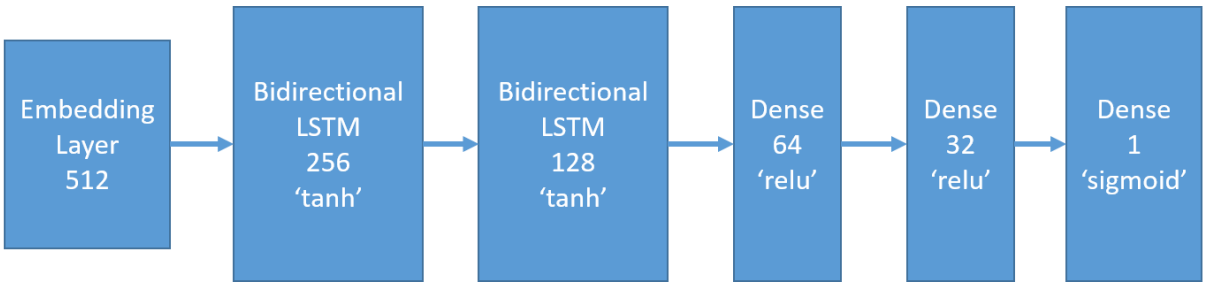
學號：R06942143 系級： 電信所碩一 姓名：籃聖皓

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？
   答：



模型架構

Embedding layer：

這是我 RNN 實做的模型，其中 embedding 所使用的 weight 是來自 gensim 對 train_nolabel.txt train 出來的，而 gensim 這邊我將出現次數少於 1 次的字都濾掉之後當成 unknown 再去 train Vector，這樣可以濾掉一些奇怪的符號跟鮮少出現的字眼，判斷是否是負面或是正面，鮮少字眼與奇怪符號造成的效果是沒有那麼顯著的。而因為我有刪減調這些字眼，於是我把這層的 trainable 設為 False，因為在 train 的時候，被濾掉的字還是會再出現，這樣前面濾掉這些字就沒有太大意義了，所以這層設為 False。

bidirectional layer：

層數的設計就是簡單的每層都除以二，而實作了 GRU 以及 LSTM，使用 LSTM 的結果會比較好大概 0.01~0.02 左右，因此選用了 LSTM，而使用 relu 準確度大概只會卡到 0.5~0.6 左右，因此也嘗試了 sigmoid，不過結果都沒有比 tanh 好。後來接兩層 dense 之後用 sigmoid 判斷是正面或是負面。

訓練過程

optimizer 使用 Adam、lr=0.001，大概都在第 20 個或是第 30 個 epoch 之後可以得到最高的 val_acc。

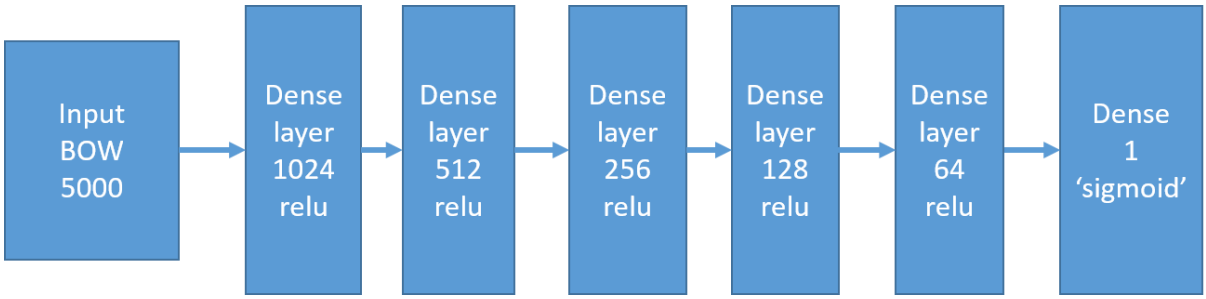
1. 因為看了 data 裡面有些人會打!!!!!!!!!!!!或是?????????????或是 sooooooooooooo 等等的字眼，不過這邊就把這些重複的地方都改掉了，變成!以及?以及 so，不然其實他要表達的東西是一樣的意思，不過卻會被判定成不一樣的 token 數量，雖然這樣好像少了一些語意(可能看到一堆!!!!!就可以知道是 super 驚訝等等)，不過實做下來的結果有提升 0.01~0.03 還不錯。
2. 有看到有些人 fuck 打成 fxck，因此把 fxck 都換成 fuck 了，不過準確度沒有大差別。
3. 把所有的複數都刪掉了，這邊用的是比較簡單的方法，如果第三個字之後有 s 的話，就把這個 s 刪掉，準確度也有提升，而為甚麼是第三個字之後因為怕把 is 變成 i，或是 his 變成 hi，所以從第三個之後開始。

準確度

0.81901

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？(Collaborators:參照助教投影片用 tokenizer.texts_to_matrix 實現)

答：



模型架構

BOW：

把這些詞透過 Bag of word 轉換成 5000 維的 vector，參考助教再手把手時間介紹的 tokenizer.texts_to_matrix 實作。

Dense：

這邊就簡單的每層都除以二最後接到 sigmoid 算機率是多少。

訓練過程

這邊的 train_nolabel.txt 也照一樣的處理方式，處理了上面三點

大概結果再 0.78~0.79 左右，可能因為沒有考慮順序關係，所以準確度不會像 RNN 一樣那麼高。

準確度

0.785

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：

用 RNN predict 出來的結果是：

"today is a good day, but it is hot"：0.3203351

"today is hot, but it is a good day"：0.9888482

因為 RNN 會考慮到順序的關係，所以有把上面那句的 but 語氣考慮進去，因此上面那句的意思被判為比較接近負面的。

用 Bag Of word 出來的結果是：

"today is a good day, but it is hot"：0.74828357

"today is hot, but it is a good day"：0.7629782

因為 bag of word 比較沒有考慮順序問題，因此看到有 good，可能就將這句的意思判斷成是正面的句子。

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：

這邊是把用有標點符號的 model 來作測試，因此有標點符號的樣本與用 re.sub 濾掉標點符號的樣本。
有包含標點符號的 training acc 是 0.85658
沒包含標點符號的 training acc 是 0.6566600000190734
因為”, . ? !”，其實也是包含很重要的意思，很多句子也會因為加上一個標點符號意思就轉變了，像是：
　你好厲害!　你好厲害?
這兩句的意思就會因為標點符號有了不一樣的變化，因此標點符號也是很重要的。

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。
答：
這邊先 train 出一個 RNN 的 model，用 RNN 的 model 去作 predict 之後，用 hard label，把這個 predict label，跟原本的 train_label.txt，用 np 的 concatenate 在一起之後，再放入 np.random.shuffle，當成下一次模型的 input，再丟到 model 裡面去 train 2 個 epoch 之後再 predict 一次，然後再把這個 predict 跟原本的 ground truth 疊加在一起之後，在丟到模型 train 2 個 epoch，如此循環 10 次。
還沒 semi-supervised 的準確度是：0.8191
加上 semi-supervised 的準確度是：0.8202
準確度有提升一些，不過好像不是那麼顯著，感覺是因為在 train_no_label.txt 裡面的這個檔案，有很多字是沒看過的，這樣在都被歸類到 unknown，其實應該是很多可以來分辨意思是甚麼的，這邊沒有把 train_no_label 與原本的 training data 丟到 tokenizer 以及 word2vec，只有給最後的 model 去 improvement 而已，應該可能是需要把這個也丟到 word2vec 來做 training 可能可以更提高準確度。