學號：B04902099 系級： 資工三 姓名：黃嵩仁

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: 資工三 林政豪、資工三 林子雋、資工三 戴培倫)
答：
這次作業我所實作的 RNN model， 處理方式為：
(a) 首先，處理所讀入的 training data (labeled, non-labeled)，將每一筆 data 依空白字元切割(成為 2 維的 array)。
(b) 將處理好的資料作為 gensim 套件中 Word2Vec model 的 input，只考慮在 training data (labeled, non-labeled) 中出現次數大於 20 次的字詞，設定所轉換的 vector 維度 (72)，batch_size 設定為 10000，訓練 10 個 epoch 後將 model 儲存。
(c) 以上一個步驟中的 Word2Vec model 作為字典，將 labeled training data 依照字典，把每一個單字都轉換為一個 72 維的 vector (若字典中找不到對應的 vector，則輸出 72 為的 0 向量)，使整個 unlabeled training data 成為一個三維陣列 (data_size, 40, 72)，其中的 40 是假設有效句子不超過 40 個字。
(d) Model 訓練前，先將 labeled training data (input data) shuffle 後，80% 作為 training data，20% 作為 validation data。
(e) RNN model 的架構為：model 會先將 input_data 送進一層 GRU 後，再通過兩層 Dense，Dropout 皆設定為 0.2，並以 sigmoid function 作為 Activation function (compile 的 optimizer 使用 adam 演算法)。
訓練過程與準確率方面：
訓練時發現此次作業的 model 很容易 overfitting training data，通常訓練時，前 3~4 個 epoch 的 training accuracy 跟 validation accuracy 都會一起上升 (大約從 79%上升至 82%)，同時 loss 都會一起下降 (大約從 47%下降至 34%)，然而，再之後的 epoch，training accuracy 會不斷上升(至 95%)，但 validation accuracy 則是緩慢下降至 80%左右。

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: 資工三 林子雋)
答：
這次作業我所實作的 BOW model， 處理方式為：
(a) 同 1.(a)
(b) 計算 training data (labeled, unlabeled) 中每個字的出現次數，並將出現次數大於 20 次的字詞列入字典 (為求 RNN、BOW 兩 model 的公平性)
(c) 利用(b)所建立的字典，將 unlabeled training data 中每個字詞依照字典轉成數字 (若字典中找不到對應的值，則忽略)，進一步轉換成 one-hot，最後成為 bag-of-word，得到 (data_size, dict_size) 的二維陣列
(d) 將(c)的到的 data shuffle 後，80% 作為 training data，20% 作為 validation data。

(e) 將 input_data 丟入 3 層的 Dense 中 (unit 分別為 512, 256, 32)，Dropout 設定為 0.2，再傳遞至 output_layer，並以 sigmoid function 作為 Activation function (compile 的 optimizer 使用 adam 演算法)。

準確率方面：

與 RNN model 不同，BOW model 的 training accuracy 的上升幅度較小(大約從 76%~80%左右)，training loss 則緩慢下降(50%~45%)，而 validation accuracy 與 validation loss 則不像 RNN model，有明確上升、下降的分水嶺，是在某一個數值附近來回跳動 (accuracy 大約 78%，loss 大約 48%)。

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。

(Collaborators:)

答：

令 Sentience 1 = "today is a good day, but it is hot"

Sentience 2 = "today is hot, but it is a good day"

	Sentience 1	Sentience 2
RNN model	0.64503729	0.97500479
BOW model	0.63043880	0.65230935

由上面數據可以看出

RNN model 對於兩種句型，雖然都給出了 positive 的結果，但是兩個的信心程度差距極大，可以看出不同的句型組成會造成不同的結果，我認為 model 在看到句子中的 "but" 後，會影響 but 前句子含意的判斷，因此 Sentience 1 中的 today is a good day 正向的幫助下降了。而 Sentience 1 中 it is a good day 的正向的幫助則不受影響。

BOW model 對於兩句型的情緒分數則沒有差異，因為 BOW model 只會考慮每個字詞出現的次數，不考慮字詞在句子中的順序、位置。(表格中的差異是因為我使用空格來 split data，因此 Sentience 1 的 day 與, 會判定為一個字詞，Sentience 2 的 hot 與 ,會被判定為一個字詞，因此造成些微的差異)

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。

(Collaborators:)

答：

將 training data 去除標點符號後，並做與第一題相同的資料處理(a)~(d)後，送進與

第一題一樣的 RNN model (e)中，重複訓練幾次後發現，不包含標點符號的 data，其 accuracy (training、validation)與包含標點符號的 data 相比，都極為相近或稍微略低一點，而上傳至 kaggle 的 public score 也相差不遠(有：0.82444，無 0.82065)。推測標點符號對於這次 task 的判斷結果影響不大。

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。
(Collaborators: 資工三 林政豪、資工三 林子雋)
答：
我的 semi-supervised 方法是先利用 unlabeled training data 來產生最原始的 model (iteration = 5, 架構與第一題的 RNN model 一樣)，在產生的 5 個 model 中，挑選 validation accuracy (大約 82%)最高的 model (以下稱 origin model)來當作 predict 的基準，將 unlabeled training data 輸入，得到情緒分數，若情緒分數大於 95%(小於 5%)，則將該資料標記為 1(0)，並加進新 model 的 training data 中。接著，便以新的 training data 訓練出新的 model。
在重複上述的方法 3 次後(不包含最一開始的 model)，可觀察到 training 、validation、Kaggle 的正確率如下表。***(註：我每次所使用的 validation data 都是用同一組，也就是 unlabeled training data 的前四萬筆，以達到公平性)。***
可以看出第一次的 training accuracy 大幅提升，而 validation accuracy 則些微的下降，推測原因為新加入 training data 資料量極大，且與 model 的"方向一致"，因此 training accuracy 才會如此大幅的提升，然而 validation data 的 answer 不受 model 本身所影響，因此正確率不升反而有微幅的下降。而第二、三次的 training accuracy、validation accuracy 都只有些微的提升，推測原因為新加入的 data 相較於第一次所佔比例下降。
從 kaggle score 可以看出，第一次 semi-supervised 後，準確率有相當幅度的下降(相較於沒有 semi-supervised)，但是在第二次與第三次的 training 後，準確率逐步上升，最後甚至比沒有 semi-supervised 的狀況下高了 0.2%左右。推得，若使用 semi-supervised 的話，accuracy 是有可能上升的。

	Training accuracy	Validation accuracy	Kaggle score
沒有 semi-supervised	77.8%~87.0%	80.7%~82.0%	0.82033
第一次	92.4%~95.3%	80.9%~81.6%	0.81594
第二次	94.4%~96.6%	81.8%~82.2%	0.81997
第三次	95.3%~97.2%	81.8%~82.4%	0.82245