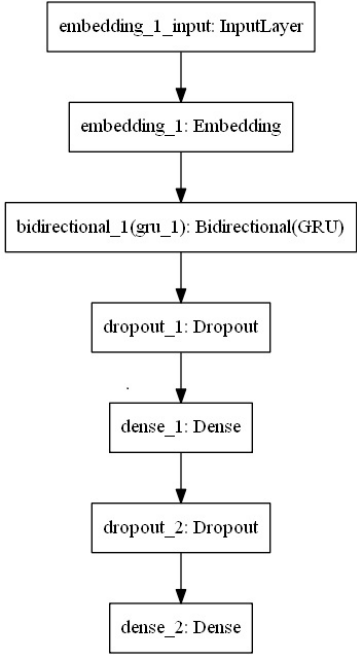
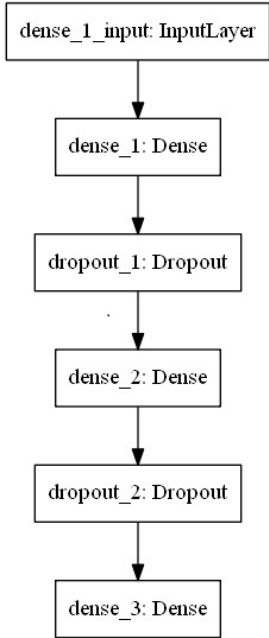
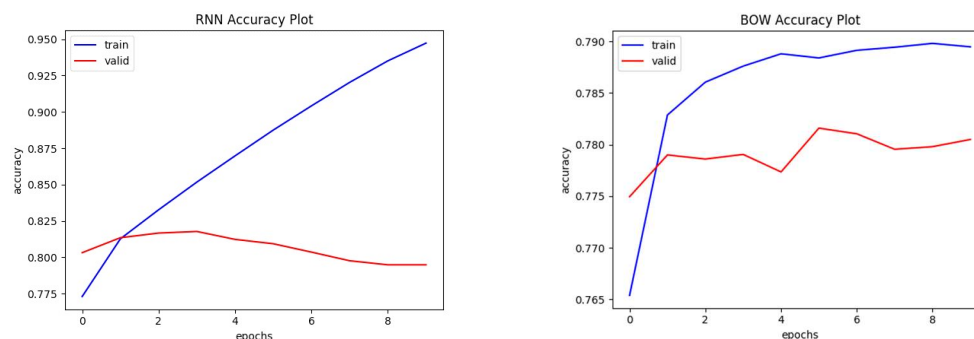


1. (1%) 請說明你實作的 RNN model，其模型架構、訓練過程和準確率為何？
2. (1%) 請說明你實作的 BOW model，其模型架構、訓練過程和準確率為何？

兩題的詳細內容列成表格，如下：

模型	1. RNN	2. BOW
訓練過程	我首先將 label、no label、testing 的語句都先經過 gensim 的 word2vec 處理，訓練成 size=100 的 vector，再把他放進 model 之中，繼續隨著 training data 繼續 train，以下的結果是有去除標點符號的。	同樣採用三份資料，一樣去除標點後，使用 keras.preprocessing.text 中 Tokenizer 的 texts_to_matrix，將 BOW 所需要的個數統計好，並只記前 4000 高頻率出現的單詞，再丟進 DNN 中訓練。
架構	 <pre> graph TD A[embedding_1_input: InputLayer] --> B[embedding_1: Embedding] B --> C[bidirectional_1(gru_1): Bidirectional(GRU)] C --> D[dropout_1: Dropout] D --> E[dense_1: Dense] E --> F[dropout_2: Dropout] F --> G[dense_2: Dense] </pre>	 <pre> graph TD A[dense_1_input: InputLayer] --> B[dense_1: Dense] B --> C[dropout_1: Dropout] C --> D[dense_2: Dense] D --> E[dropout_2: Dropout] E --> F[dense_3: Dense] </pre>
參數量	Total params: 5,996,137 Trainable params: 5,996,137 Non-trainable params: 0	Total params: 2,311,681 Trainable params: 2,311,681 Non-trainable params: 0

準確率



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。

	RNN	BOW
today is a good day, but it is hot	0.637509	0.601447
today is hot, but it is a good day	0.971831	0.601447

可以看見表現上仍是 RNN 較佳，並且我的模型對於先說明 hot 再說明 good 的評分較高。而在 BOW 的模型當中，因為都是統計句子中重複單字的數量，所以兩個句子評分相同，也因為缺少了字與字之間位置的關係，train 出來的 model 效果較差。

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。

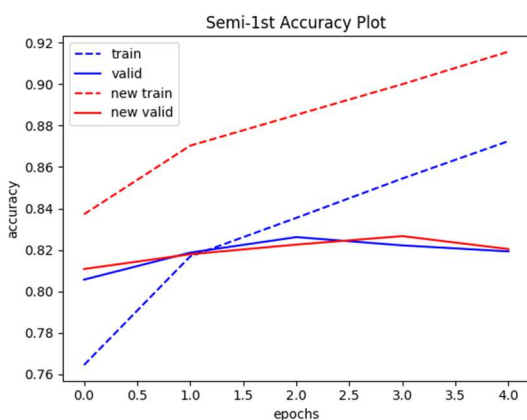
	無標點	有標點																																																							
accuracy	<div><p>Punctuation Accuracy Compare</p><table border="1"><thead><tr><th>epochs</th><th>no punct train</th><th>no punct valid</th><th>punct train</th><th>punct valid</th></tr></thead><tbody><tr><td>0</td><td>0.775</td><td>0.780</td><td>0.765</td><td>0.805</td></tr><tr><td>1</td><td>0.815</td><td>0.810</td><td>0.815</td><td>0.820</td></tr><tr><td>2</td><td>0.845</td><td>0.815</td><td>0.845</td><td>0.825</td></tr><tr><td>3</td><td>0.875</td><td>0.815</td><td>0.875</td><td>0.820</td></tr><tr><td>4</td><td>0.900</td><td>0.810</td><td>0.900</td><td>0.815</td></tr><tr><td>5</td><td>0.920</td><td>0.805</td><td>0.920</td><td>0.810</td></tr><tr><td>6</td><td>0.935</td><td>0.800</td><td>0.935</td><td>0.805</td></tr><tr><td>7</td><td>0.945</td><td>0.795</td><td>0.945</td><td>0.800</td></tr><tr><td>8</td><td>0.950</td><td>0.795</td><td>0.950</td><td>0.800</td></tr><tr><td>9</td><td>0.955</td><td>0.795</td><td>0.955</td><td>0.795</td></tr></tbody></table></div>		epochs	no punct train	no punct valid	punct train	punct valid	0	0.775	0.780	0.765	0.805	1	0.815	0.810	0.815	0.820	2	0.845	0.815	0.845	0.825	3	0.875	0.815	0.875	0.820	4	0.900	0.810	0.900	0.815	5	0.920	0.805	0.920	0.810	6	0.935	0.800	0.935	0.805	7	0.945	0.795	0.945	0.800	8	0.950	0.795	0.950	0.800	9	0.955	0.795	0.955	0.795
epochs	no punct train	no punct valid	punct train	punct valid																																																					
0	0.775	0.780	0.765	0.805																																																					
1	0.815	0.810	0.815	0.820																																																					
2	0.845	0.815	0.845	0.825																																																					
3	0.875	0.815	0.875	0.820																																																					
4	0.900	0.810	0.900	0.815																																																					
5	0.920	0.805	0.920	0.810																																																					
6	0.935	0.800	0.935	0.805																																																					
7	0.945	0.795	0.945	0.800																																																					
8	0.950	0.795	0.950	0.800																																																					
9	0.955	0.795	0.955	0.795																																																					

Kaggle public	0.81836	0.82297
實作方式	使用 gensim.utils 的 tokenize	僅使用 python list split(` `)
觀察	1. 在圖中可以發現，這組採用的 10% validation data，有標點符號的 model 訓練全程的準確率都較無標點來的高。 2. 推測可能是 twitter 推文中情緒的正負面與標點符號出現的關聯性較大。	

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

我將 no_label data 放入 predict 之後，把分數 0.95 以上與 0.05 以下的句子標上 1 與 0，再將它們重新 train 過，得到新的 model。重複兩次後得到下面的結果。

	Original	Semi-1 st time	Semi-2 nd time
New labeled-data added	0	63665	435957
Validation Loss	0.38635	0.40365	0.39944
Validation Accuracy	0.82615	0.82663	0.83164



而 training accuracy 的變化如左圖所示，在第二輪甚至是到了 98%，可見 vector 在後面 semi 的步驟相當於將原本的 model 的 epoch 增加，更加符合 label data 的情緒分數。

在 validation accuracy 上，因為 training data 基數的增加，有了一定程度的提升。