學號：R06921038 系級： 電機所碩一 姓名：謝宗宏

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:)

答：

模型架構：

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 39, 256)	64161536
gru_1 (GRU)	(None, 256)	393984
dense_1 (Dense)	(None, 256)	65792
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 256)	65792
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 2)	514

Total params: 64,687,618
Trainable params: 526,082
Non-trainable params: 64,161,536

我先用 gensim pre-train 了 wordvector，label data 跟 unlabel data 都有使用。再直接將 wordvector 設定成 embedding layer 的 weights。因此，上圖的 embedding layer 是設定成 non-trainable 的。Wordvector 是由 gensim word2vec 函數得到。Vector dimension = 256, window size = 7, min count = 3, iteration = 30, 使用 skip gram，其餘參數使用 gensim default。在 tokenizing training data 時，有將標點符號保留。

訓練細節：

epochs: 50。

optimizer: Nadam, 使用 keras default 參數。

loss function: categorical cross-entropy。

learning rate: Nadam default 0.002。

early stopping: 當 accuracy 連續 10 個 epochs 都沒有進步時停止。

model checkpoint: 當 accuracy 有提升的時候才儲存 model。

validation split: 0.1

batch size: 256

dropout rate: 0.2

準確率：

training (loss, accuracy): (0.3476, 0.8474)

validation (loss, accuracy): (0.3835, 0.8321)

kaggle (public, private): (0.83406, 0.83264)

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:)

答：

模型架構：

Layer (type)	Output Shape	Param #
reshape_1 (Reshape)	(None, 5000)	0
dense_1 (Dense)	(None, 256)	1280256
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8256
dropout_3 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 128)	8320
dropout_4 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 256)	33024
dropout_5 (Dropout)	(None, 256)	0
dense_6 (Dense)	(None, 2)	514

Total params: 1,363,266
Trainable params: 1,363,266
Non-trainable params: 0

先用 keras 的 Tokenizer 在 label data 和 unlabel data 上建立字典，並去除掉標點符號，再使用 tokenizer.texts_to_matrix 將 label data 轉換成以 tfidf 表示的向量矩陣。為避免矩陣過大導致 out of memory ，因此只使用前 5000 個最常出現的字。

訓練細節：

epochs: 50。

optimizer: Nadam, 使用 keras default 參數。

loss function: categorical cross-entropy。

learning rate: Nadam default 0.002。

early stopping: 當 accuracy 連續 10 個 epochs 都沒有進步時停止。

model checkpoint: 當 accuracy 有提升的時候才儲存 model。

validation split: 0.1

batch size: 256

dropout rate: 0.2

準確率：

training (loss, accuracy): (0.4471, 0.7897)

validation (loss, accuracy): (0.4894, 0.7822)

kaggle (public, private): (0.78274,0.78523)

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：

Bag of word model:
"today is a good day, but it is hot" : [0.42326856, 0.57673138]
"today is hot, but it is a good day" : [0.42326856, 0.57673138]

RNN model:
"today is a good day, but it is hot" : [0.66669476, 0.33330527]
"today is hot, but it is a good day" : [0.01860088, 0.98139912]

數組的第一個維度是該句話為 label 0 的機率，第二個維度是該句話為 label 1 的機率。對於 bag of word model 兩句話結果是一樣的，因為這個模型並不考慮句子中每個字和符號出現的順序，所以對它而言這兩句話沒有差別。RNN model 有考慮句子中單字和符號出現的順序，所以可能有能力知道第一個句子的語氣由好轉壞，第二個句子由壞轉好，從而將兩者區分開來。
另外可以發現，兩個不同的模型對於預測這兩句話的信心程度也有不同。RNN model 似乎有能力區分出語氣前後的轉折，機率的分佈明顯有被拉開，然而 bag of word model 由於不能區分句中文字出現的順序，機率分佈也相對比較靠近。

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：

使用 keras 的函數 text_to_word_sequence，用 filters 參數決定是否留下標點符號。
無包含標點符號 (filters=' !"#$%&()*+,-./:;<=>?@[\]^_`{|}~')
training (loss, accuracy): (0.3612, 0.8392)
validation (loss, accuracy): (0.3882, 0.8264)
kaggle (public, private): (0.82517,0.82461)
有包含標點符號： (filters='\t\n')
training (loss, accuracy): (0.3476, 0.8474)
validation (loss, accuracy): (0.3835, 0.8321)
kaggle (public, private): (0.83406, 0.83264)
可以看出保留標點符號對於準確率是有明顯提升的。推測如驚嘆號、問號等表點符號確實可能帶有更多情緒上的提示，因此留下這些符號可能有助提升 model 的表現。此外也可能有些將標點符號作為表情符號的使用，能夠更直接顯示出情緒。

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:)

答：如何標記 label: 首先只以 labeled data 訓練一個 RNN model，接著用訓練好的 model 對每一筆 unlabeled data 做 evaluation ，若是屬於 label 0 的機率大於 threshold 則將該筆 unlabeled data 當作 label 為 0，反之，若屬於 label 1 的機率大於 threshold 則將該筆 unlabeled data 當作 label 為 1。(我使用 categorical cross-entropy 而非 binary cross-entropy, output layer 使用 softmax 而非 sigmoid)。Threshold 我設為 0.9。

準確率比較：

1. without semi-supervised

training (loss, accuracy): (0.3612, 0.8392)

validation (loss, accuracy): (0.3882, 0.8264)

kaggle (public, private): (0.82517,0.82461)

2. semi-supervised

training (loss, accuracy): (0.1155, 0.9616)

validation (loss, accuracy): (0.5119, 0.8249)

kaggle (public, private): (0.82548,0.82433)

從 kaggle score 來看，兩者其實差不多，semi-supervised 的方法在 public 上略好，在 private 上略差。但是從 training accuracy 和 validation accuracy 上看起來，semi-supervised 的方法似乎更有 overfitting 的趨勢。畢竟 threshold = 0.9 表示 model 會更往前一次 predict 的方向去 update 參數。

References:

[1] https://docs.google.com/presentation/d/1zZ-Kh-aeXqATgwmpyyifkV3oWGifz7LDpPfKKmTcCvs/edit#slide=id.p35

[2] https://keras.io/preprocessing/text/

[3] https://radimrehurek.com/gensim/models/word2vec.html

[4] https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html