學號：**B06902125** 系級：資工三 姓名：黃柏瑋

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)

首先，先說明word embedding的方法：

- 對於所有出現在training_label.txt、trainind_nolabel.txt和 testing_data.txt中的字，利用genism中word2vec套件產生出一個256維的embedding
- 如果字的頻率少於五的話，就將他移除並在往後把它視為<UNK>

接著，先說明RNN架構：

- 每個句子的長度為24，每個字的embedding是256
- 由於LSTM蠻容易overfitting，所以我RNN cell選用GRU
- hidden size為160
- 2-layer與uni-directional、並加上dropout(0.5)
- RNN跑完結果後會進入一個classifier，架構如下：

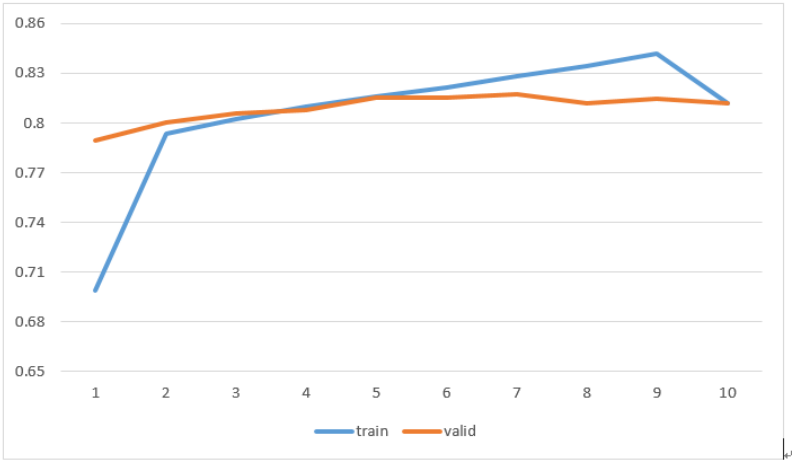
```
nn.Sequential(nn.Dropout(0.5),
              nn.Linear(in_features=160, out_features=512),
              nn.PReLU(),
              nn.Dropout(0.5),
              nn.Linear(in_features=512, out_features=1),
              nn.Sigmoid())
```

以下是training時的參數：

- batch size為128
- learning rate為1e-3

最後，看看這個model的成績

- learning curve
  由於此次作業是分類問題，著重準確率，於是以準確率作圖：



- accuracy

	VAL ACC	PUBLIC TEST ACC
basic	0.8171	0.8219

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。

首先，先說明DNN模型：

- 架構：

```
nn.Sequential(nn.Linear(in_features=input_size, out_features=1024),
              nn.PReLU(),
              nn.Dropout(0.5),
              nn.Linear(in_features=1024, out_features=2048),
              nn.PReLU(),
              nn.Dropout(0.5),
              nn.Linear(in_features=2048, out_features=1),
              nn.Sigmoid())
```

- 參數量：39519235

接著，比較兩個模型對於這兩句話的分數：

	TODAY IS A GOOD DAY, BUT IT IS HOT	TODAY IS HOT, BUT IT IS A GOOD DAY
BOW+DNN	0.8356	0.8356
RNN	0.0077	0.9994

由於BOW無法考慮到文字的先後，因此會給予這兩個句子相同的feature，餵進去model所得到的結果也是一樣的；此外，因為BOW+DNN無法感受到句子中的轉折語氣，所以看到文章中有"good"就會認為這個句子偏正面。

然而，RNN就不一樣，RNN對於文字的時序性相當敏感，可以分辨出第一句是想在抱怨"今天太熱"，而第二句真正想說"今天是個好日子"。

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）

首先，先對preprocessing下手：

- 長度小於24的句子只包含80%，小於32的約有98%，因此將句子長度擴增到32
- 將gensim求出的embedding做一下standardization，讓模型訓練是更快速、穩定一些

接著說明對模型架構的改良：

- 將uni-directional升級為bi-directional，讓模型在學習語意時能根據字的前後文，而不只是根據前文

並調整訓練模型時的初始化及參數：

- 利用orthogonal initialization初始化RNN，能夠增加模型的收斂效果
- 由於RNN對於參數特別敏感，為了讓他學習到更細緻的資訊，減少batch size至96並調降learning rate至2e-4可以達到相對好的效果，overfitting的速度也不會太快。

最後，來比較他們的準確率：

	VAL ACC	PUBLIC TEST ACC
basic	0.8171	0.8219
improved	0.8241	0.8226

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）。

首先先從題目的建議下手，將有label的training data調降至2萬筆，做完一輪的training後，對沒有label的data進行預測，結果高於0.8者加入下一輪成為positive label，小於0.2者則為negative label，其餘保持unlabeled。

這樣的self training的效果為下，ITER已經進行了多少輪self-training：

ITER	DATA NUMBER	TRAIN ACC	VAL ACC
0	20000	0.8171	0.7957
1	1094854	0.9887	0.8279

不難發現，有做semi-supervised的效果相當顯著。這是因為原本的模型擁有近八成的準確率，這也代表新被加入的資料中，有相當多是被預測正確的，因而成功增加了模型的訓練資料庫。當資料越多，模型的效能就有可能會更好。

接著就是想辦法加以改良，應用在原本的20萬筆資料上：

由於經過一輪後，新加入的資料量太大，導致相當程度的雜訊，且訓練速度過慢。而若只是簡單地將positive threshold調高至0.9或0.95(negative threshold調降至0.1或0.05)，雖然能使雜訊減少，但會使模型在下一輪的學習效果不彰，畢竟那是他已經認定為極端正向或負向的data了。

於是我將theshold一樣維持在0.8和0.2，但從中最多sample出各80000筆資料，加入原先的資料中進行下一輪的訓練，重複五個iteration，想辦法將模型逐步推向更好的境界。結果如下：

ITER	DATA NUMBER	TRAIN ACC	VAL ACC
0	200000	0.8412	0.8241
1	360000	0.9249	0.8267

ITER	DATA NUMBER	TRAIN ACC	VAL ACC
2	520000	0.9564	0.8289
3	680000	0.9640	0.8295
4	840000	0.9721	0.8293
5	1000000	0.9778	0.8295