ADL HW1 Report

R10546017 朱瑋民

Q1 Data processing

a. 針對前處理的部分，我是沒有參考助教的 code 為自己重新寫的，tokenize 的部分，因為 slot 的 json 已經有做 tokenize 了，所以沒有特別去做，而 intent 的 json 是用空格來切，也就是像 tell me what to call you 會變成 ” tell”, ” me”, “what” , “to” , “call” , “you” 這六個字，而因為有些句子或是有些詞是會有標點符號像是 ” ,” “!” “]” 等等，我自己是直接把這些標點符號給刪除，然後像是 I’ ll , I’ ve , he’ s 類似這種表示詞態的，我直接把’ ll , ‘s 等後面的部分給拿掉，而有些表示時間的，像是 12:00pm 這種的，是直接把這些詞替換成一個時間像是 time 或是 pm 等等

b. 而自己使用的 pretrained embedding 是史丹佛 Glove 提供的多種預先訓練的詞向量，是先將載好的 glove.840B.300d.txt 變成 word2vector 的 gensim_glove.840B.300d.txt 形式後，在對前面處理完的資料進行轉換，而在轉換的過程中有些字詞是 glove 裡面所沒有的，針對這些字詞在 intent 裡面是直接把它給刪除，而在 slot 裡面則

是替換成一個 name 後再丟入 word2vector 進行詞向量的 embedding，因此轉換完的資料以 tell me what to call you 為例，會變成（6 , 300）的維度，6 為多少個字而 300 為這個字 300 維的向量，並且會做 padding 將每個 sentence 的長度變成一樣長

Q2 Describe your intent classification model

a. intent classification 的 model 是用一個 Bi-GRU，input 的 features 為詞向量的維度 300，hidden size 的數量為 300，layers 的數量為兩層，並且有設定 drop out 為 0.3 來避免 overfitting ，Bi GRU output 的結果為 (4 , batch_size , 300) 其中 4 為 bi 的兩個 output 乘上兩個 layers ，300 為 hidden size 的大小，並將 4 做平均之後丟入一個 linear 層來變成 150 個 output，代表 intent 的 150 個 class 的分別機率

b. Public score 為 0.92755

c. 自己的 loss function 是使用 Crossentropy 的 loss function，為一個蠻適合針對 multi class 使用的 loss function

d. 自己的 optimizer 是使用 RAdam，全名是 Rectified Adam，是一種改進 Adam 的方法，利用自動化的 warmup 來改善 Adam 的 adaptive learning rate 變異非常大的問題，而 batch size 是使用 100，learning rate 是使用 0.001

Q3: Describe your slot tagging model

a. Slot tagging 的 model 是用一個 Bi-GRU，input 的 features 為詞向量的維度 300，hidden size 的數量為 300，layers 的數量為三層，並且有設定 drop out 為 0.3 來避免 overfitting ，Bi GRU output 的結果為（6, batch_size , 300) 其中 6 為 bi 的兩個 output 乘上三個 layers ，300 為 hidden size 的大小，並將 6 做平均之後丟入一個 linear 層來變成 35 個 output，代表這個句子的 35 個詞的大小，因為自己是用 regression 的方式，所以在不同的區間代表不同的類別，像是> 0.5 或 <1.5 為 date 之類的，

b. Public score 為 0.77694

c. 自己的 loss function 是使用 Huber loss 的 loss function，因為我是用迴歸的方式來試試看，來看每個字出來的值與實際類別的差距，並把這個類別的文字像是 date 轉成 id 之後來看神經元跑出來的值與類別的 id 值的大小差距，所以用 regression 的 loss function，而 huber loss 為 MAE 與 MSE 的改良版，取 MAE 跟 MSE 的優點並針對他們的缺點來做改良

d. 自己的 optimizer 是使用 RAdam，全名是 Rectified Adam，是一種改進 Adam 的方法，利用自動化的 warmup 來改善 Adam 的 adaptive learning rate 變異非常大的問題，而 batch size 是使用 100，learning rate 是使用 0.001

Q4: Sequence Tagging Evaluation

由 from seqeval.metrics import f1_score 可以看到 F1 score 為 0.7679465776293821

而由 from seqeval.metrics import classification_report

的 classification_report 結果如下 ：

	precision	recall	f1-score	support
last_name	0.61	0.88	0.72	65
date	0.66	0.78	0.72	211
people	0.76	0.73	0.75	267
time	0.73	0.86	0.79	221
first_name	0.98	0.93	0.95	88
micro avg	0.73	0.81	0.77	852
macro avg	0.75	0.84	0.78	852
weighted avg	0.74	0.81	0.77	852

而 Joint accuracy 的值為 ：0.8611279563371741

Token accuracy 的值為 ： 0.9755653961727029

可以發現，the evaluation method in seqeval 較為貼近現實想得到的 Recall 跟 Precision，因為它針對不同的類別來分別計算 precision 跟 recall，可以看到每個類別的結果，並且比較沒有去考慮沒有類別的 O 所造成的結果，而相比

Joint accuracy 是看一整個句子的對錯與 Token accuracy 是看每個字詞的對錯，較具有鑑別性並且可以知道不同類別的預測情形，而 Token accuracy 跟 Joint accuracy 可能會因為 O 的影響，模型 O 的部分幾乎都答對而看似結果很好，其實不然，其中 Token accuracy 應該又會比 Joint accuracy 好，因為 Joint accuracy 要整句答對但 Token 只要裡面的一些字詞答對即可，可見在以真實想預測的情形來說，the evaluation method in seqeval > Joint accuracy > Token accuracy 的 。

Q5: Compare with different configurations

自己當初是把 train 跟 eval 的 data 合併再一起之後，再用 validation ratio = 0.2 下去做切割，而最初的模型架構是使用 LSTM，固定層數為 2，hidden size 為 300， 在 intent 的 val loss 為 0.36，slot 的 val loss 為 0.024，接著便更改模型架構為 GRU，在 intent 的 val loss 來到了 0.31，slot 的 val loss 來到了 0.022，因為在固定層數跟 hidden size 的情形下 GRU 的結果相對比較好，因此最後選擇用 GRU 來做 tuning，接著更改了 GRU 的從單向變成雙向的，intent 的 Val loss 也來到了 0.28，slot 的 val loss 來到了 0.019，很明顯將 GRU 變成雙向了之後效果提升了很多，接著因為發現 train

loss 有持續下降，而 val loss 卻沒有下降，因此猜測可能有 overfitting 的問題，於是加了 drop out 為 0.3，intent 的 Val loss 變成了 0.25，slot 的 val loss 來到了 0.016，最後便是調整 learning rate 與 batch size，batch size 從兩百多變成一百多甚至幾十，使得泛化能力變好，learning rate 也下降一點，在實驗結果後選擇了 batch size 為 100 learning rate 為 0.001，模型的 val loss 在 intent 與 slot 分別為 0.243 與 0.0148 ，最後便是調整 epoch 的訓練時間來到最後的成績，其實蠻意外 LSTM 的成績會比 GRU 差的，但也有可能是因為模型其他參數剛好在 GRU 上面會有比較好的成果，然後將 optimizer 從原本的 SGD 換成 RAdam 之後成績也會有明顯的提升，尤其在 intent 上面從 public leaderboard 0.87 直接跳到了 0.92，可見一個好的 optimizer 也是可以讓準確率提升的改良