Applied Deep Learning, Spring 2022 - Homework 1

郭柏志 R09521205

Q1: Data processing

Intent classification:

step1: Tokenization

蒐集 intent data 中 train.json, eval.json 的所有出現過的 intent tokens，給定每個 intent token 一個 label 並將 label mapping dictionary 存成 intent2idx.json。接著將 train.json, eval.json 中所有的 text 斷詞，蒐集不重複的 token 並計算個別 token 出現的次數。

step2: Build vocabulary dictionary

留下最常出現的前 10000 個 tokens，給定每個 token 一個 index，建立 token to index 關係，再用助教提供的 sample code 中的 Vocab Class 建立 vocab object 存成 vocab.pkl。

Step3: Encoding and Align sequences

原始資料輸入模型之前，會先將每個句子 tokenize 成 token list，個別資料的句子長度不同會導致每一筆資料的 token sequence 長度也不同，因此透過 vocab object 將一個 mini batch 中的每個 token sequence padding 到相同的長度，並且將 token sequence 轉換成 index sequence。

Step4: Word embedding

使用 pre-train embedding glove(300d)，將 index sequence 中的每個 token index 透過 embedding layer 轉換成 dimension 為 300 的 word vector。

Slot tagging:

Data processing 流程與 Intent classification 相同，不同處在於原始 data 已經做過 tokenize，因此不需要自行斷句。處理 data IOB tagging label 時，除了給定所有出現過的 tagging 一個對應的 index label 外，另外額外手動加一個”PAD”對應的 label 到 label mapping dictionary 再存成 tag2idx.json。

Q2: Describe your intent classification model

a. Model

Encoder 為 2 layers Bidirectional LSTM 提取句子的特徵，Classifier 為 1 Linear layer + 1 Dropout Layer with 10% dropout rate，輸出 feature dimension = intent 總數的 logits 向量。

$$index\ vector\ =\ [id_1, id_2, \ldots id_n],\ \ n =\ sequence\ length$$
$$[X_1, X_2, \ldots, X_n]\ =\ embedding(index\ vector)$$
$$seq\_output\ , (h_n, c_n)\ =\ LSTM(X_n, (h_{n-1}, c_{n-1}))$$
$$h_n\ =\ concat(\ h_n^1, h_n^2\ )$$
$$logit\ output\ =\ \text{Classifier}(h_n)$$

b. Performance

Kaggle public score: 0.91022

c. Loss function

Cross Entropy Loss,
Loss = CrossEntropy($logit\ output,\ y_{label}$)

d. Optimization algorithm, Learning rate, Batch size

Adam with leaning rate = 0.001, batch size = 128

Q3: Describe your slot tagging model

a. Model

Encoder 為 2 layers Bidirectional LSTM，Classifier 為 2 Linear layer + 1 Dropout Layer with 20% dropout rate 輸出 output sequence length = input sequence length, feature dimension = intent 總數的 2D logits 向量。

$$index\ vector\ =\ [id_1, id_2, \ldots id_n],\ \ n =\ sequence\ length$$
$$word\ vector\ sequence\ =\ [X_1, X_2, \ldots, X_n]\ =\ embedding(index\ vector)$$
$$seq\_output\ , (h_n, c_n)\ =\ LSTM(word\ vector\ sequence, (h_0, c_0))$$
$$seq\_output\ =\ batchNorm1d(seq\_output)$$
$$logit\ output\ =\ \text{Classifier}(seq\_output\ )$$

b. Performance

Kaggle public score: 0.802

c. Loss function

Cross Entropy Loss
Loss = CrossEntropy($logit\ output,\ y_{label}$)

d. Optimization algorithm, Learning rate, Batch size

Adam with leaning rate = 0.001, batch size = 128

Q4: Sequence Tagging Evaluation

	precision	recall	f1-score	support
date	0.80	0.78	0.79	206
first_name	0.96	0.92	0.94	102
last_name	0.87	0.83	0.85	78
people	0.79	0.77	0.78	238
time	0.90	0.89	0.89	218
micro avg	0.85	0.83	0.84	842
macro avg	0.86	0.84	0.85	842
weighted avg	0.85	0.83	0.84	842

$$\text{token accuracy} = \frac{number\ of\ correctly\ predicted\ tokens}{number\ of\ predicted\ tokens}$$

$$\text{Joint accuracy} = \frac{number\ of\ correctly\ predicted\ sentences}{number\ of\ predicted\ sentences}$$

Seqeval 將各個 sequence 以 tag 種類分類，分成 date, first_name, last_name, people, time 這五類。接著會對每個 tag 計算 True Positives (TP), True Negatives (TP), False Positives(FP), False Negatives (FN)

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2TP}{2TP + FP + FN}$$

$$support = number\ of\ this\ tag$$

Q5: Compare with different configurations

在做 intent classification 與 slot tagging 時都有用到 LSTM，但由於輸入的句子長度不等長，在前處理時會先將 mini batch 中不等長的句子先 pad 到相同的長度，但這會導致一個 sequence 中有很多無意義的資訊，因此兩個任務我都有使用 pytorch 提供的 pack_padded_sequence function，可以讓註記 batch 中每個 sequence 的長度，當 time step 超過該列長度，padding 的值就不參與計算 loss 更新梯度，在其餘模型架構相同下，intent classification Dev Acc 從 0.92 進步到 0.94，slot tagging Dev Acc 從 0.81 進步到 0.83。

slot tagging 任務在訓練時很容易出現 overfitting，原因可能在於訓練集中有些單詞出現次數遠多於其他單詞，模型因此難以學習出現頻率較小的單詞的特徵，因此我嘗試在 embedding 後面接一個 dropout layer，隨機捨棄 word vector 資訊後有非常明顯的改善，Kaggle public score 從 0.711 進步到 0.802。