Q1: Data Processing

使用助教的sample code進行data preprocessing:

- 讀取intent classification、slot tagging的train, eval, 將tokens及labels(intent, tags)轉成可以餵進model的index, tokens的部分存成一個(0=padding, 1=unknown, 2以後=tokens)的vocab, 最後將其輸出為vocab.pkl, 而labels的部分則存成一個index以0開始的dictionary, 最後輸出為json檔(i.e. intent2idx.json, tag2idx.json)
- 在function build_vocab()內, 將之前存成vocab.pkl的dictionary中的words和一開始利用wget抓取的`glove.840B.300d.txt`進行比對, 若有比對到就直接用`glove`裡面的`float`, 沒比對到的話就透過`random()`隨機生成一個, 最後輸出為`embedding.pt`

Q2: intent classification model

descriptions:

將資料經過preprocessing embeddings轉換成float tensor, 接著利用layer norm標準化後再將embeddings丟進雙層、雙向的GRU model, 抓取雙向final layer並將其連接起來, 最後input進分類器依序經過標準化→drop out→linear layer算出predicting probability, 最後再透過loss function(CrossEntropy)算出loss rate評估模型

model: GRU

parameters

- performance:
  validation: 0.939(local)
  test: 0.92444(public score on Kaggle)
- learning rate: 1e-3
- loss function: CrossEntropyLoss
- batch size: 128
- max len: 35
- dropout rate: 0.4
- optimization algorithm: Adam
- number of epoch: 20
- hidden size: 128

Q3: slot tagging model

descriptions:

將資料經過preprocessing embeddings轉換成float tensor, 接著利用layer norm標準化後再將embeddings丟進LSTM model並抓取feature, 最後將輸出的每一層都input進分類器, 經過標準化→drop out→linear layer算出predicting probability(因為是tagging問題, 轉換成n個輸出), 最後再透過loss function(CrossEntropy)算出所有output的loss rate評估模型

model: LSTM

parameters

- performance:
  validation: 0.814(local)
  test: 0.79356(public score on Kaggle)
- learning rate: 1e-3
- loss function: CrossEntropyLoss
- batch size: 128
- max len: 40
- dropout rate: 0.4
- optimization algorithm: Adam
- number of epoch: 40
- hidden size: 512

Q4: Sequence Tagging Evaluation

```
Joint Accuracy = 814 / 1000 (0.814)
Token Accuracy = 7647 / 7891 (0.9690786972500317)
              precision    recall  f1-score   support

        date       0.81      0.78      0.79       206
  first_name       0.95      0.89      0.92       102
   last_name       0.83      0.76      0.79        78
      people       0.75      0.79      0.77       238
        time       0.85      0.87      0.86       218

   micro avg       0.82      0.82      0.82       842
   macro avg       0.84      0.82      0.83       842
weighted avg       0.82      0.82      0.82       842
```

由上圖可以看出, joint accuracy=0.814 < token accuracy=0.96, 這是因為token accuracy是透過一個個text進行計算, 相對地joint accuracy為只要一句話其中一個text mismatch就算錯, 所以準確率明顯較低, 而簡單觀察可發現name和time的準確率偏高、date和people的準確率則較低, 另外, first_name和last_name的precision, recall相較其他類別差異較大, 可推測前者在prediction上有相對較突出的表現, 且在actual的match上有較大的落差, 整體而言sequence的mismatch幾乎都來自於小部分的錯誤, 若針對這些錯誤對model進行微調應該可以有不錯的進步。

Q5:Compare with different configurations

Slot tagging model:

一開始是使用GRU model, 結果表現不佳, validation rate平均落在不到0.78, 接著先嘗試提高dropout rate(0.2→0.4), 結果表現有些微上升, 落在0.78附近, 後來放棄GRU改使用LSTM, 發現其他條件不變下LSTM整體的表現會比GRU來得好一些些, 且調高dropout rate的效果也同樣適用在LSTM上(validation rate從0.781→0.789), 最後開始變動hidden size和number of epoch, 在LSTM下, 我將hidden size從原本的128調到最後的512, 每次放大兩倍去觀察performance的變化, 可以得到validation rate從0.789→0.794→0.804, 而number of epoch則是從20調整到40, 最後模型表現的變動由0.808→0.814。

Final Result:

validation rate = 0.814
testing performance = 0.79356(on Kaggle)