深度學習與實務 Lab 4

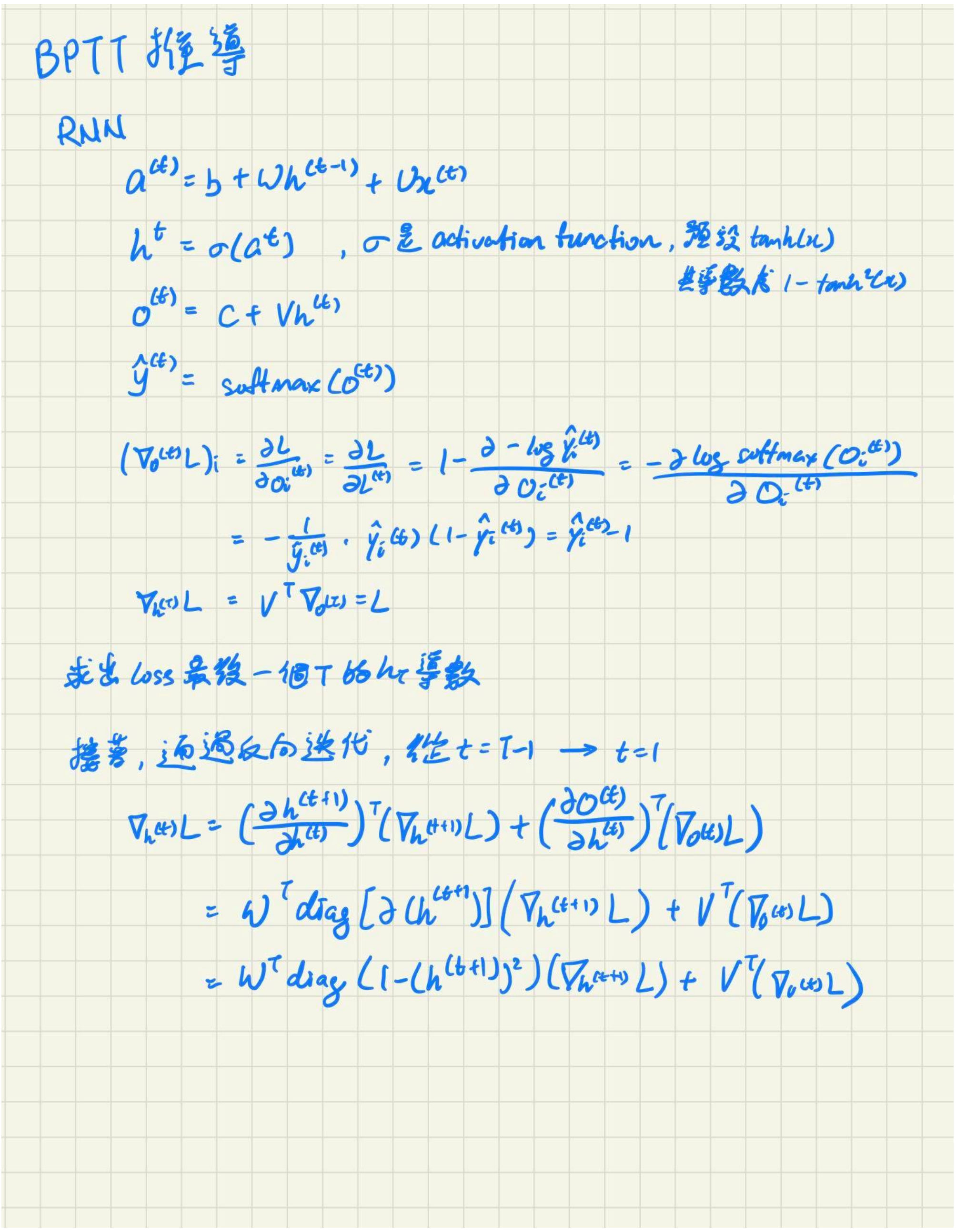
關薦勇 0856702

Introduction

使用python與pytorch框架搭建一個能夠糾正英文單字的sequence-to-sequence 遞迴神經網路模型。輸入模型的inputs為一筆收錄了一系列錯別單字的數據集，output則是對應錯別字數據集的正確單字。

Derivation of BPTT (Back Propagation Through Time)

BPTT 梯度沿时间通道传播的反向傳播，為了獲得參數節點上的梯度，必須先評估其即時子節點（下游）節點上的梯度



BPTT 推導

RNN

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$

$h^{t} = \sigma(a^{t})$, σ 是 activation function, 預設 $\tanh(x)$ 其導數為 $1-\tanh^2(x)$

$$o^{(t)} = c + Vh^{(t)}$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)})$$

$$(\nabla_{o^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} = 1 - \frac{\partial - \log \hat{y}_i^{(t)}}{\partial o_i^{(t)}} = \frac{-\partial \log \text{softmax}(o_i^{(t)})}{\partial o_i^{(t)}}$$

$$= -\frac{1}{\hat{y}_i^{(t)}} \cdot \hat{y}_i^{(t)}(1-\hat{y}_i^{(t)}) = \hat{y}_i^{(t)} - 1$$

$$\nabla_{h^{(\tau)}} L = V^T \nabla_{o^{(\tau)}} = L$$

求出 loss 最後一個 T 的 h_τ 導數

接著，通過反向迭代，從 $t = T-1 \longrightarrow t=1$

$$\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right)^T (\nabla_{h^{(t+1)}} L) + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right)^T (\nabla_{o^{(t)}} L)$$

$$= W^T \text{diag}[\partial (h^{(t+1)})] (\nabla_{h^{(t+1)}} L) + V^T (\nabla_{o^{(t)}} L)$$

$$= W^T \text{diag}(1-(h^{(t+1)})^2)(\nabla_{h^{(t+1)}} L) + V^T (\nabla_{o^{(t)}} L)$$

Implementation details

- dataloader
  - 我設立一個可將英文單字依照英文字母的排序，將數據集中單字裡的每個字母轉換成 1～26的數字來表示。下圖為示例圖

$$\begin{tikzcd}\boxed{\text{data}} \arrow[r] & \boxed{[d, a, t, a]} \arrow[r] & \boxed{[4, 1, 20, 1]}\end{tikzcd}$$

  - alphabet_to_category() : 收集資料集中所有單字字母的set，並用dict的方式收集所有對應的category值。
  - word_to_indices() : 將單字中的每個字母轉換成對應的category值。
  - indices_to_word() : 將category值復原到對應的英文字母，並合併成單字

- encoder

```
EncoderRNN(
  (embedding): Embedding(256, 29)
  (lstm): LSTM(29, 29)
)
```

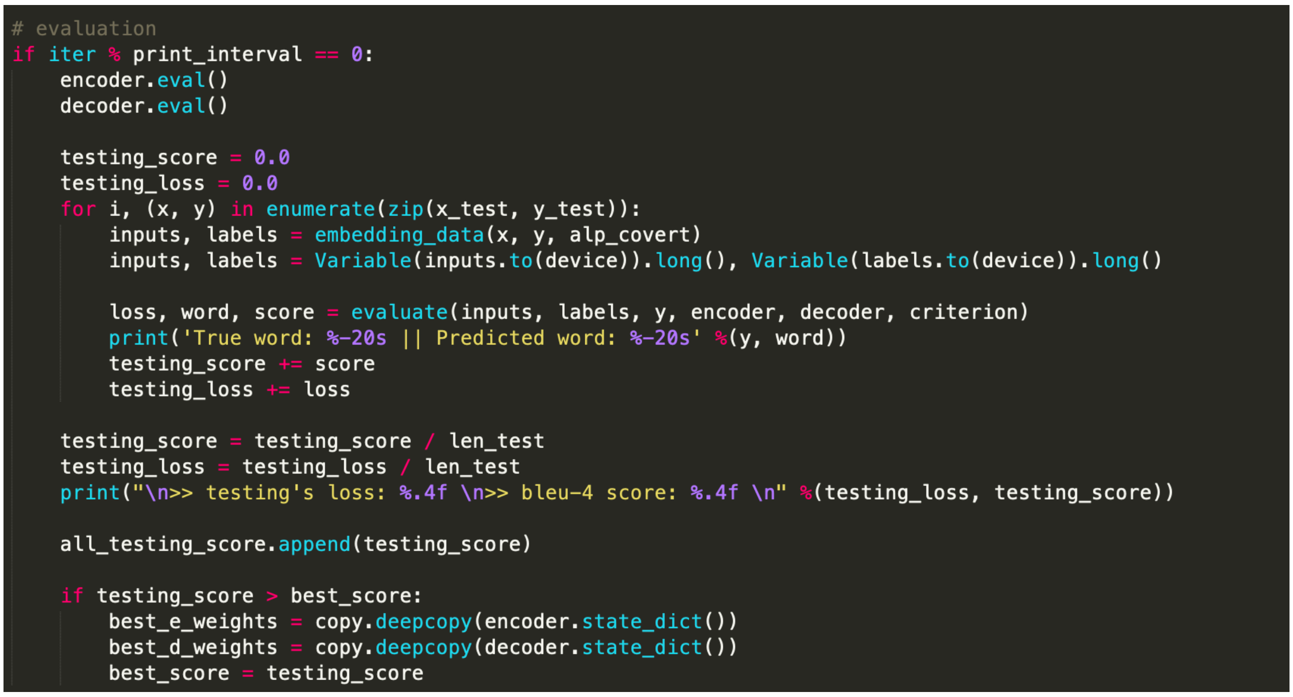
- decoder

```
DecoderRNN(
  (embedding): Embedding(256, 29)
  (lstm): LSTM(29, 29)
  (out): Linear(in_features=29, out_features=256, bias=True)
  (softmax): LogSoftmax()
)
```

encoder 和 decoder 的架構則基本依照sample.py的設置

- optimizer 兩者皆使用 SGD， learning rate 為0.01
- lr_scheduler 兩者皆設置 step size 為100，gamma為0.5
- decoder 的activation fucntion 是ReLU
- LSTM 的 hidden size 為 256
- teacher forcing ratio 為 0.8

下圖為evalution時候的code，程式執行時是以test.json的50筆單字資料集進行驗證，並無涉及到train.json

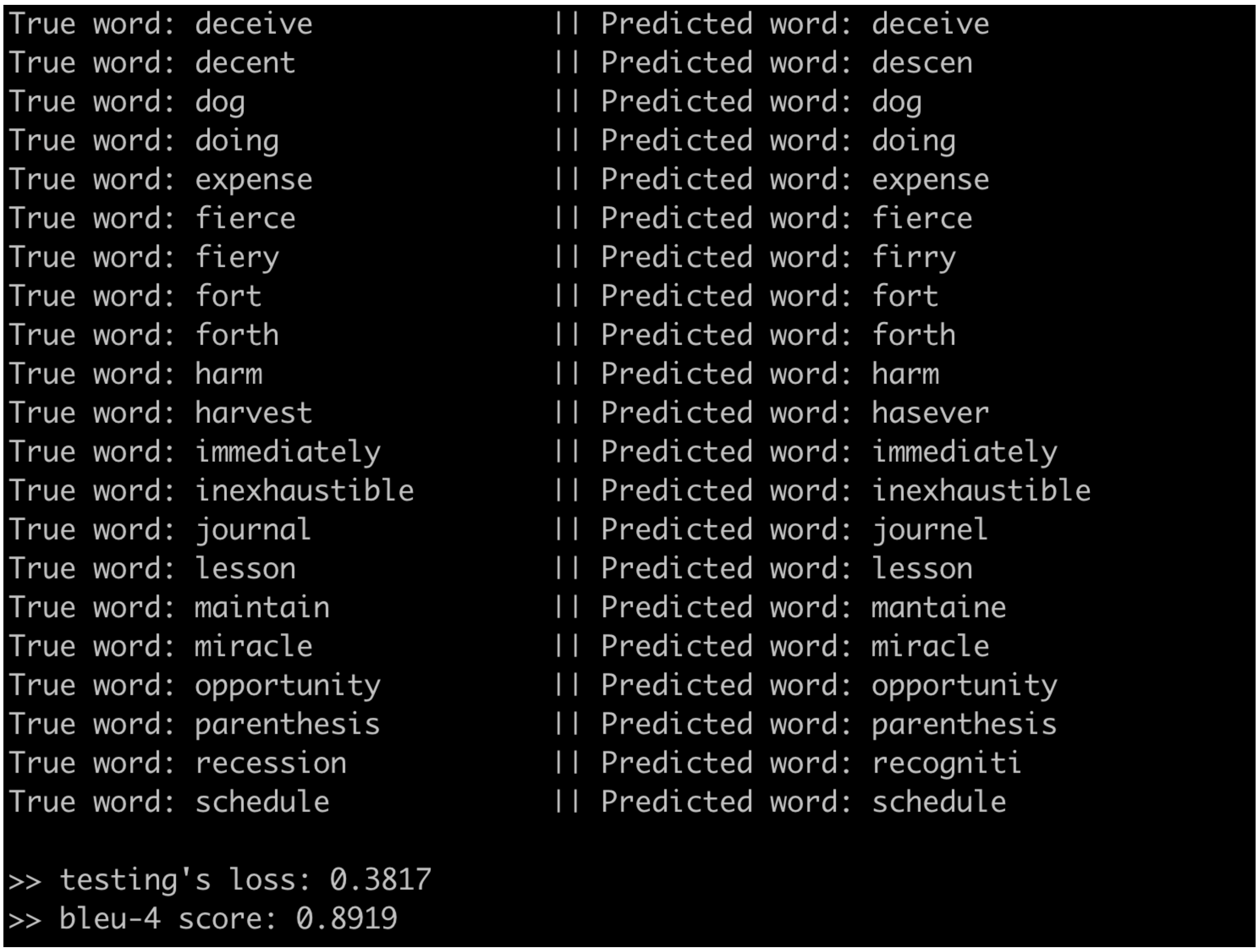


```python
# evaluation
if iter % print_interval == 0:
    encoder.eval()
    decoder.eval()

    testing_score = 0.0
    testing_loss = 0.0
    for i, (x, y) in enumerate(zip(x_test, y_test)):
        inputs, labels = embedding_data(x, y, alp_covert)
        inputs, labels = Variable(inputs.to(device)).long(), Variable(labels.to(device)).long()

        loss, word, score = evaluate(inputs, labels, y, encoder, decoder, criterion)
        print('True word: %-20s || Predicted word: %-20s' %(y, word))
        testing_score += score
        testing_loss += loss

    testing_score = testing_score / len_test
    testing_loss = testing_loss / len_test
    print("\n>> testing's loss: %.4f \n>> bleu-4 score: %.4f \n" %(testing_loss, testing_score))

    all_testing_score.append(testing_score)

    if testing_score > best_score:
        best_e_weights = copy.deepcopy(encoder.state_dict())
        best_d_weights = copy.deepcopy(decoder.state_dict())
        best_score = testing_score
```

Result and Discussion

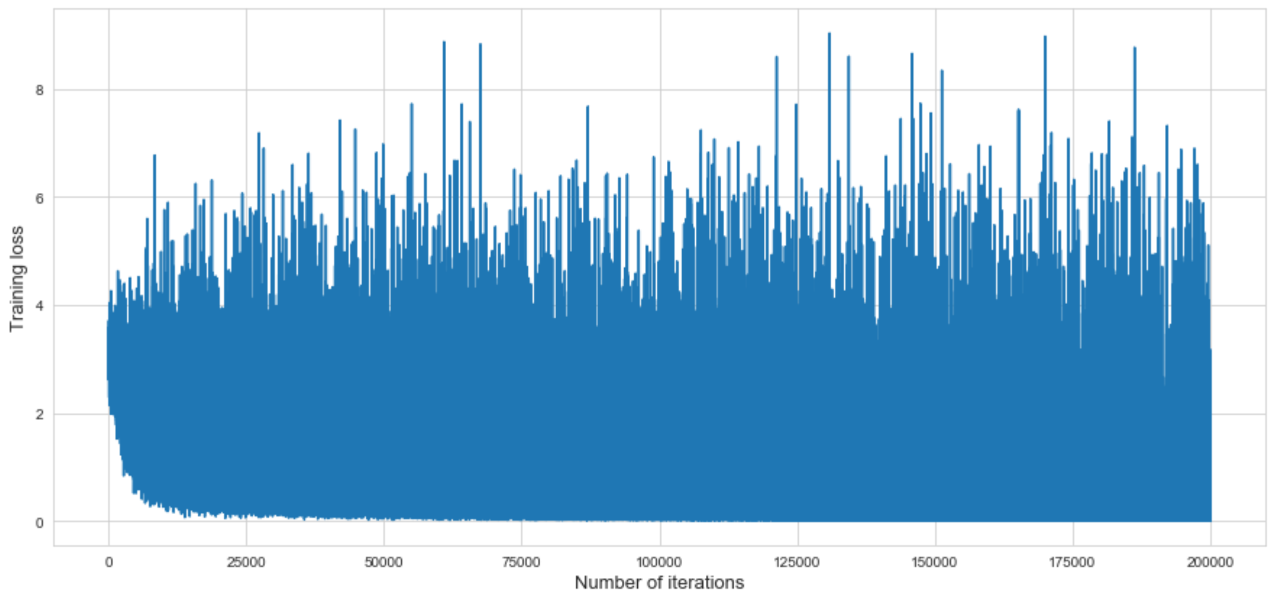
result of spelling correction

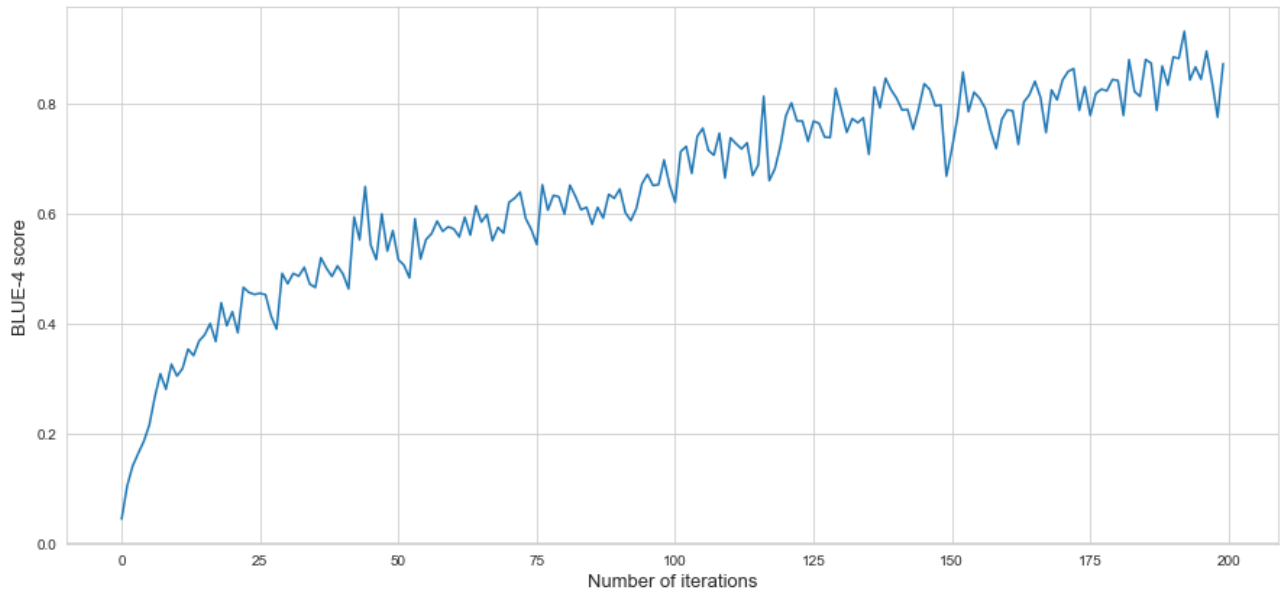
(左邊為正確單字；右邊為模型預測的單字)



```
True word: deceive              || Predicted word: deceive
True word: decent               || Predicted word: descen
True word: dog                  || Predicted word: dog
True word: doing                || Predicted word: doing
True word: expense              || Predicted word: expense
True word: fierce               || Predicted word: fierce
True word: fiery                || Predicted word: firry
True word: fort                 || Predicted word: fort
True word: forth                || Predicted word: forth
True word: harm                 || Predicted word: harm
True word: harvest              || Predicted word: hasever
True word: immediately          || Predicted word: immediately
True word: inexhaustible        || Predicted word: inexhaustible
True word: journal              || Predicted word: journel
True word: lesson               || Predicted word: lesson
True word: maintain             || Predicted word: mantaine
True word: miracle              || Predicted word: miracle
True word: opportunity          || Predicted word: opportunity
True word: parenthesis          || Predicted word: parenthesis
True word: recession            || Predicted word: recogniti
True word: schedule             || Predicted word: schedule

>> testing's loss: 0.3817
>> bleu-4 score: 0.8919
```

plot of training loss curve & BLUE-4 score testing curve during training





(上圖) plot of training loss curve

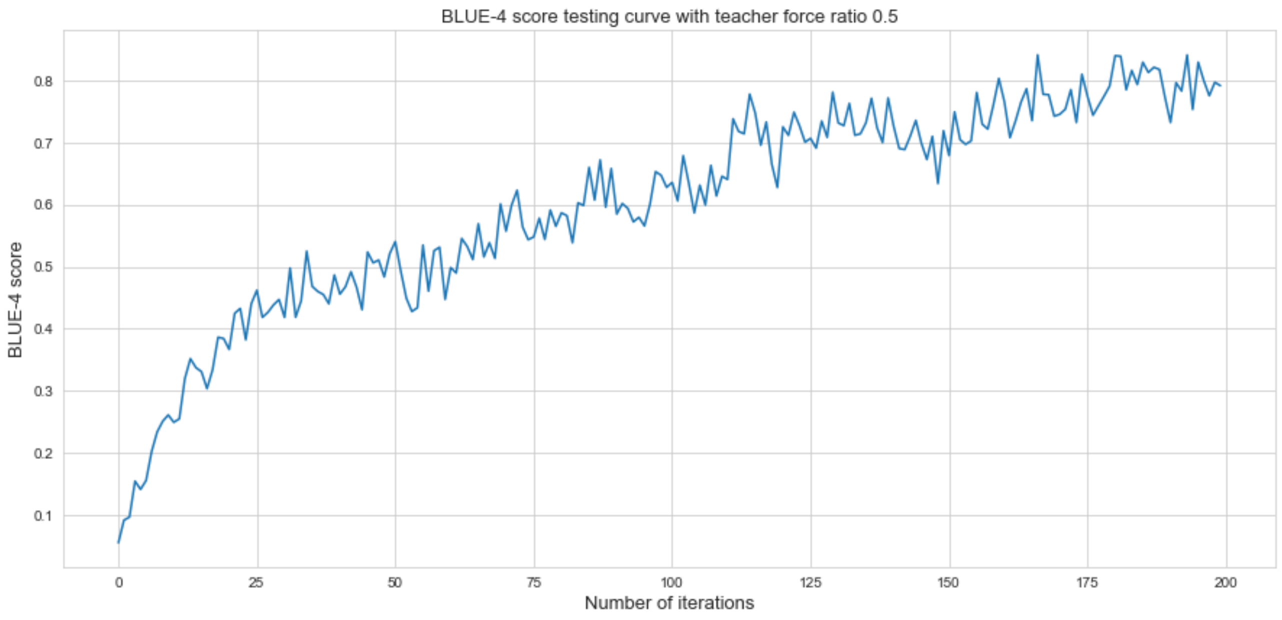
(下圖) plot of BLUE-4 score testing curve during training

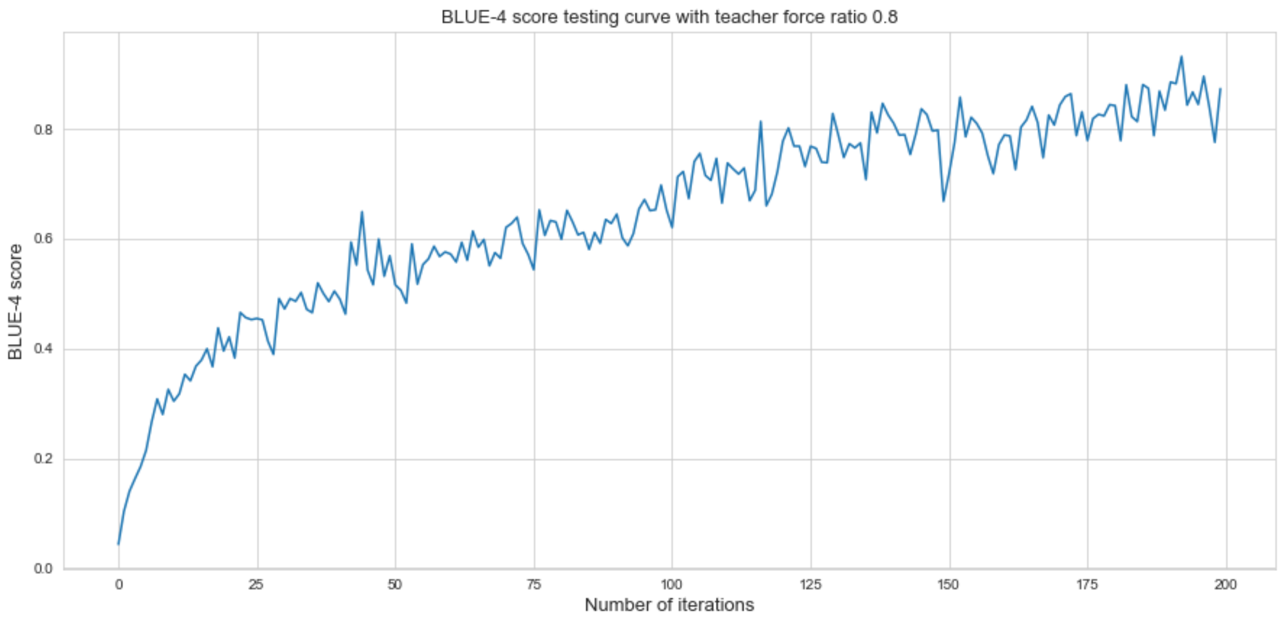
從 (下圖) 可觀察到在步入約第120000次迭代時，測試集的BLUE-4 score就已經可達到0.8的水準了, 而最高的一次則是達到了0.93。

Discussion of the results

- 模型在持續迭代次數時，training loss會呈現上下起伏的現象；但把test.json作為驗證集時的BLUE-4則似乎會因迭代次數的增加而持續上升。在約第175000次迭代時，BLUE-4達到了0.93。本人最高的迭代上限設為200000次，因此本人不排除假設繼續往上迭代的話，會不會得到比0.93更好的分數。但也有可能是因為test.json的樣本數只有50筆，作為這筆資料的驗證集可能有點太小（train.json有將近12000個樣本數）

- 我也嘗試了將teacher force ratio 分別設為0.5 和0.8，在其他參數不變的情況下，各訓練一次。試圖藉此觀察這筆數據集在不同的teacher force ratio設置下的表現





從兩者的BLUE-4 score curve 中可初步看出，似乎沒有區別。再者由於模型的迭代次數非常多次，所以其實也有隨機性的問題。因此本人認為單純將teacher force ratio分別設為0.5和0.8 (其他參數不變的情況下) 對模型的整體性能並不會有影響。