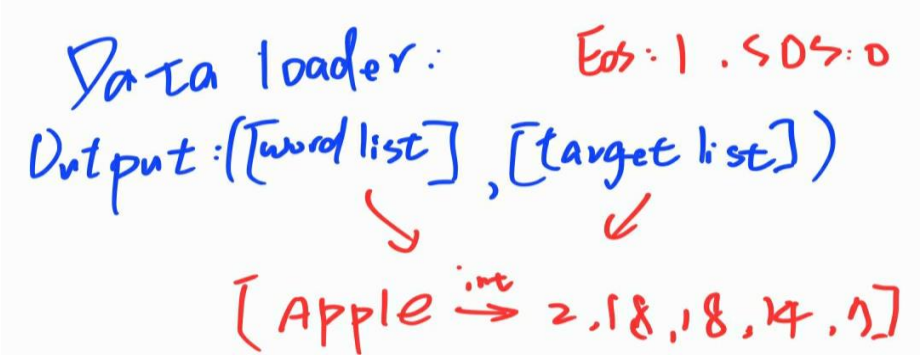
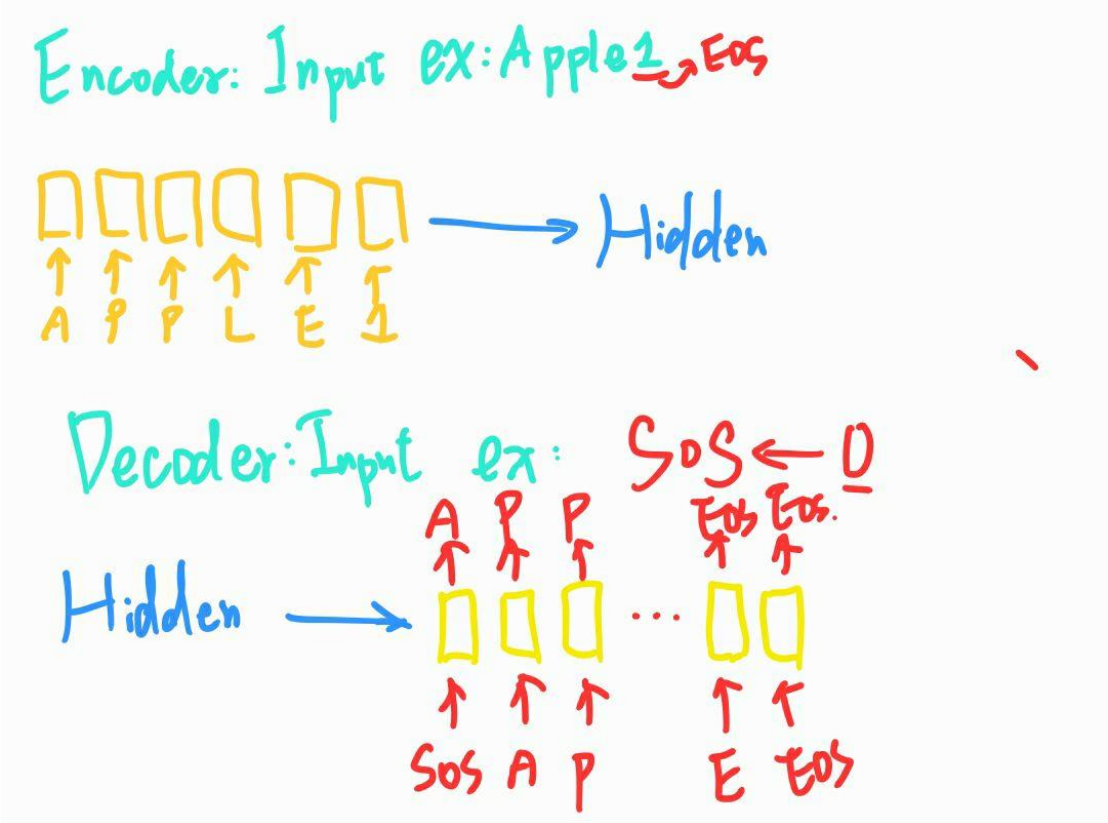
Evalutioan : bleu_score, word_acc(整個單字全對才算分)

Simple LSTM :

Data structure:



LSTM model:

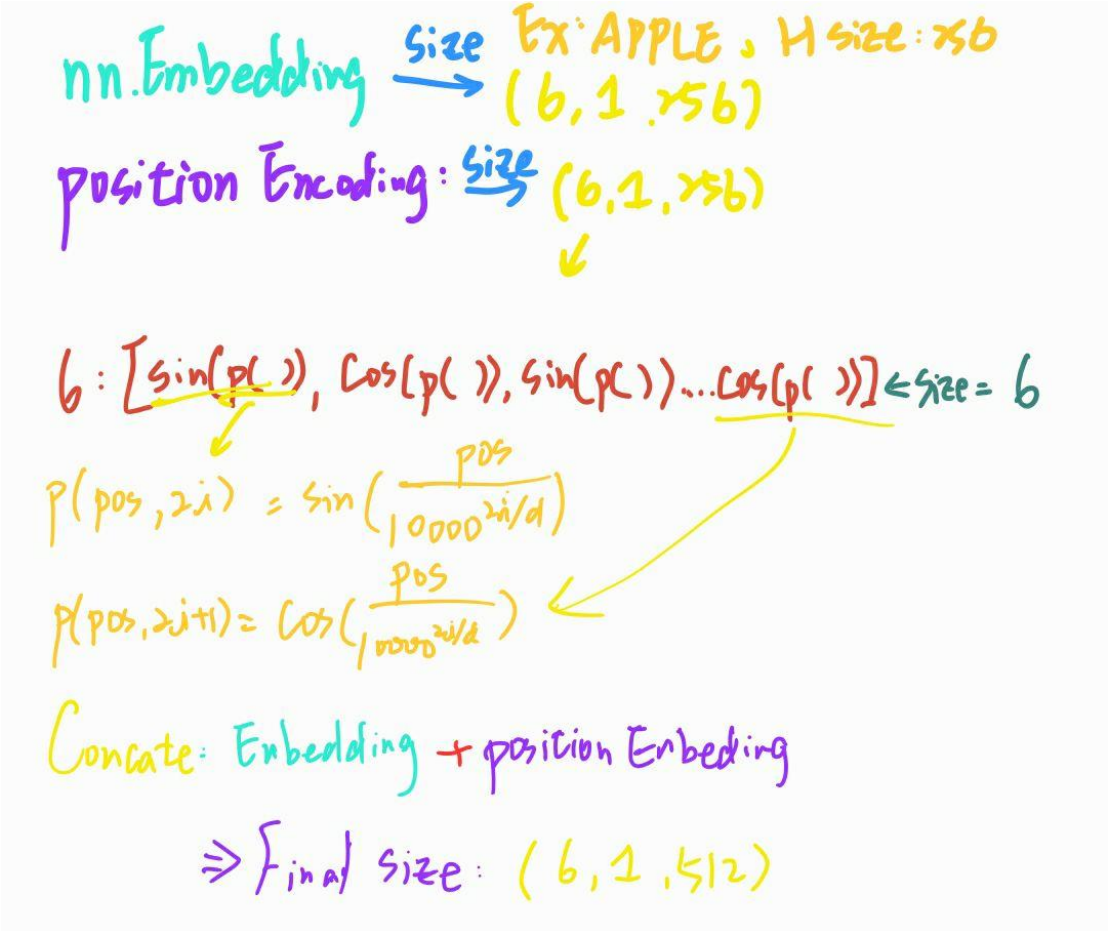


本次功課我測試了最陽春的 LSTM(hidden layer : 256 ,num layer: 1)來訓練。發現只要訓練 6,70 epoches，在 test, new_test 上分數的表現分別可以達到 0.86,0.63

```
test:
    score : 0.8679614201481775
    acc : 0.82

new_test:
    score : 0.6335110200087941
    acc : 0.46
```

以上面的 simple LSTM 為基準，我另外加上了 position encoding。
實際上為 implement 為下圖這個樣子。



實際上程式碼:

```python
class PositionEncoding(nn.Module):
    def __init__(self, max_seq_len, d_model):
        super(PositionEncoding, self).__init__()
        #公式
        self.max_seq_len = max_seq_len
        self.d_model = d_model

        position_encoding = np.array([
            [pos / np.power(10000, 2.0 * (j//2) / self.d_model) for j in range(self.d_model)] for pos in range(self.max_seq_len)])
        #偶數 sin, 奇數 cos
        position_encoding[:, 0::2] = np.sin(position_encoding[:, 0::2])
        position_encoding[:, 1::2] = np.cos(position_encoding[:, 1::2])
        position_encoding = torch.from_numpy(position_encoding)

        self.position_encoding = nn.Embedding(max_seq_len, d_model)
        self.position_encoding.weight = nn.Parameter(position_encoding, requires_grad = False)
```

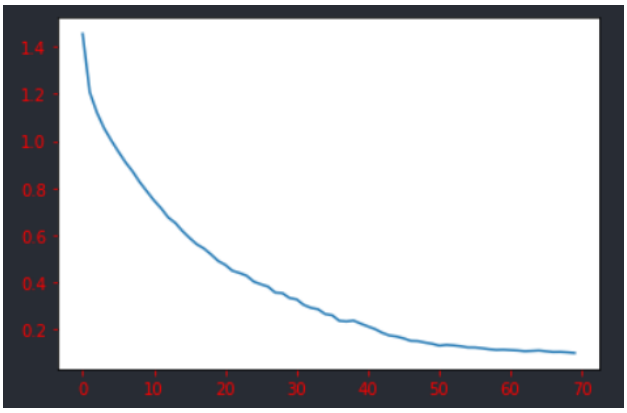
最後發現 ： 加了 position encoding 後，

1. 模型收斂速度變快。(simple : 60epochs, position encoding : 40epochs)
2. 在 test 上 score 的表現差不多。(simple : 90, position encoding : 93)
3. 在 new test 上 score 的表現也差不多(simple : 0.75, position encoding : 0.73)

Plot :

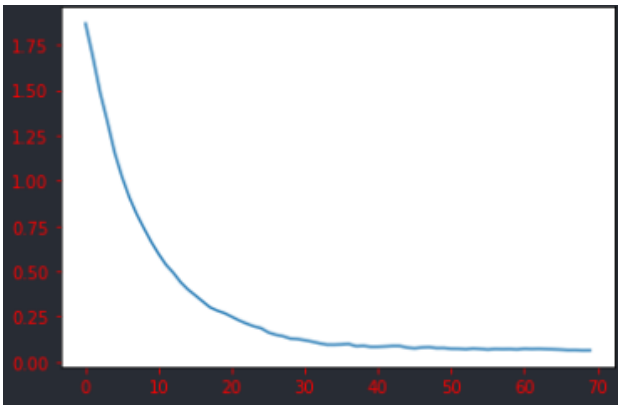
simple

Loss

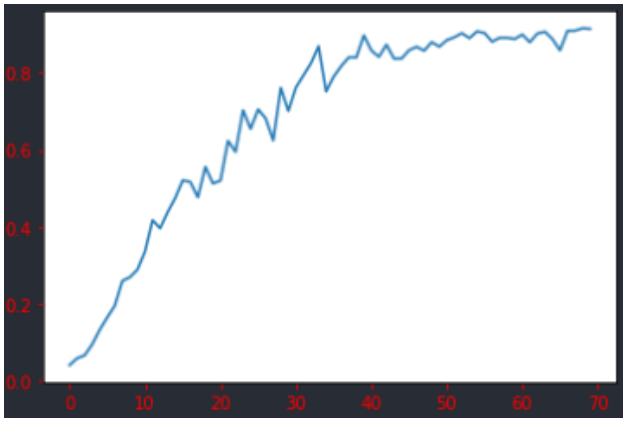


position encoding

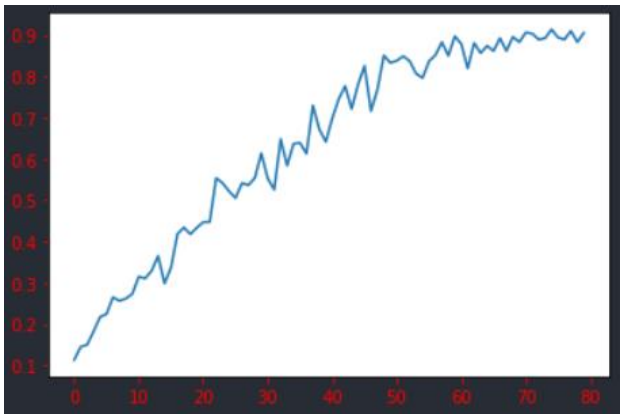
loss



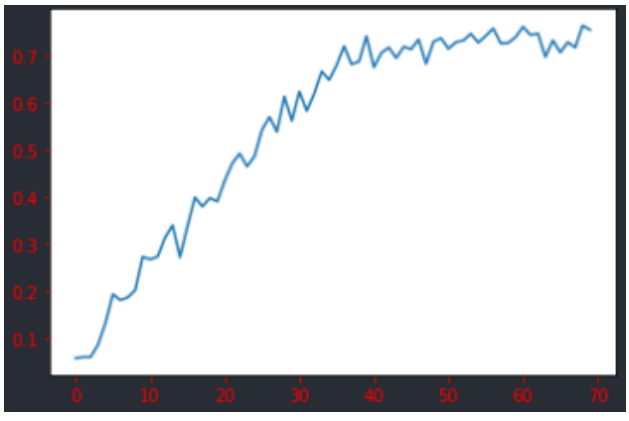
test score



test score



New_test score



New_test score