Recurrent Neural Network (RNN) for Slot Filling and Intent Prediction

B02902039　資工四　李奕皓

Word Embeddings

這次作業的 word embeddings 我是用隨機初始化，然後跟整個 model 一起 train，維度是 128 維。

其中 training data 以及 testing data 有數字的部分，一律都會當作特殊字 `<num>` 來處理，目的是避免沒看過的數字會被當作 `<unk>` 。

Model

RNN 的部分我是使用 tensorflow 的 `dynamic_rnn` 、 `GRUCell` 以及 `DropoutWrapper` 來實作，裡面的 hidden layer 是使用 128 維的向量，dropout 的 keep probability 是 0.5。

`dynamic_rnn` 的每一個字的 output （128 維向量）都會乘上一個矩陣 `W_tag` 再加上 bias `b_tag` ，最後經過 softmax 之後看哪一維最大來決定每個字是哪個 slot。Loss 是（batch 內）每個字的 cross entropy 平均起來。

最後一個 output 則會乘上矩陣 `W_intent` 再加上 bias `b_intent` ，同樣經過 softmax 後看哪一維最大來決定這句話的 intent。Loss 是（batch 內）每句話的 cross entropy 平均起來。

實際上在 train 的時候是 2 個 task 一起做，所以 loss function 是 2 個 task 的 loss 直接相加，並且用 `AdamOptimizer` 做 minimize。

Training Time

這次作業 train 的時間相當快，第一次 train 只要 10 分鐘 validation set accuracy 就收斂了，上傳後也有過 baseline（>0.9）。

後來稍微 train 了久一些，花了 30 分鐘，上傳後分數也有往上提升。

Implementation Details

這次作業比較困難的部分是，因為每個句子長度不一樣，所以 (slot) loss 的計算比較麻煩。

例如，假設 batch 裡面有 3 個 sentences，長度分別為 4、2、3，那表示每個字的 loss 的 2D tensor 可能會長這樣：

```
[[ 1,  2,  3,  4 ],
 [ 5,  6,  ?,  ? ],
 [ 7,  8,  9,  ? ]]
```

可是今天我們只要把需要的 entry 平均起來，而不要 `?` 的 entry，我的作法是做一個 mask，而 mask 的作法是，先用 `tf.range` 產生一個 1D tensor

```
[ 0,  1,  2,  3 ]
```

把它用 `tf.reshape` 轉成 2D tensor 之後再用 `tf.tile` 讓它變成 3x4

```
[[ 0,  1,  2,  3 ],
 [ 0,  1,  2,  3 ],
 [ 0,  1,  2,  3 ]]
```

再用 `tf.less` 讓每個 column 都跟 length vector `[[4], [2], [3]]` 做運算，變成一個 boolean 2D tensor，最後再用 `tf.to_float` 轉成 float

```
[[ 1.,  1.,  1.,  1. ],
 [ 1.,  1.,  0.,  0. ],
 [ 1.,  1.,  1.,  0. ]]
```

Mask 做好後，把它跟 loss tensor 做 element-wise 的相乘就可以把不需要的 entry 變成 0。整個 loss tensor 全部的值相加再除以長度總和就是最終的 loss。