

# Recursive Neural Network for Sentiment Analysis

ADL Homework 2

B02902039 資工四 李奕皓

## Regenerate the Answer

```
$ python3 answer.py --testing_data_file=/path/to/testing_data.txt.tree \
--output_file=/path/to/answer.txt
```

## Validation Set

我從 training data 中取句子最長的 **10%** 出來當作 validation set (positive, negative 各半) , 剩下的 90% 當作 training set, 目的是希望在 train 的時候, 樹的結構可以比較單純不會太大。

有了 validation set 就也可以知道什麼時候開始會 over-fitting

## Embeddings

這次作業的 embeddings 我是採用隨機初始化然後一起 train 的方式, 而沒有用預先 train 好的 embeddings dimension 是 32 維

## Matrix Representation of Tree

為了要讓 tensorflow 同一個 **graph** 可以適用全部的樹, 而不用根據樹的架構每次都重新建立新的 graph, 需要將樹轉換成 tensor 才可以處理。以 `a scary werewolf jumps` 為例：

```
((a (scary werewolf)) jumps)
```

假設每個字的 index 分別是 11, 22, 33, 44, 我們可以把這個樹以 2 個 column 的 matrix 表示：

Left child	Right child	
-23	-34	scary werewolf
-12	0	a scary werewolf
1	-45	a scary werewolf jumps

Matrix 中的數字一定是整數, 小於 0 代表是 embeddings 的 index (加負號再減一可轉換回來), 大於等於 0 代表是上面第幾個 row 做完的結果。

我們可以用這個方法把每一棵樹都轉換成 2 個 column 的 matrix, 在 graph 中用 `tf.while_loop` 以及 `tf.TensorArray` 就可以計算整個句子的 word vector

## Initialization

我把要組合 2 個 word vector 的 weights 初始化成這種形式（以 8x4 為例，實際上是 64x32）：

```
[[ 0.5  0    0    0   ],
 [ 0    0.5  0    0   ],
 [ 0    0    0.5  0   ],
 [ 0    0    0    0.5 ],
 [ 0.5  0    0    0   ],
 [ 0    0.5  0    0   ],
 [ 0    0    0.5  0   ],
 [ 0    0    0    0.5 ]]
```

每個 entry 再加上 Uniform(-0.001, 0.001)。這樣非常接近 2 個 word vector 的平均，直覺上應該是不錯的出發點。