Lecture 14: Tree Recursive Neural Network Constituency Parsing

2018 年 7 月 31 日

1 Introduction

递归关系在语言中非常常见

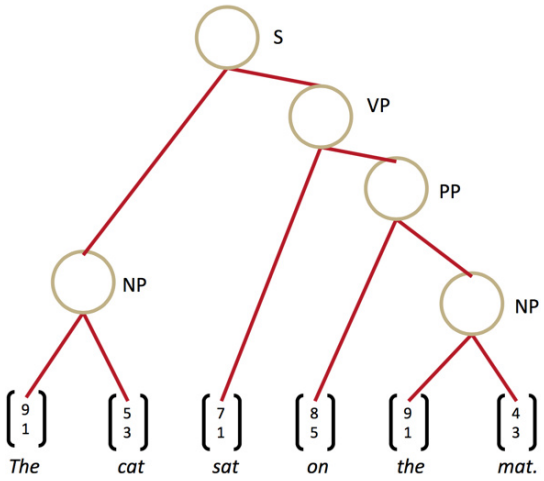
The man from [the company that you spoke with about [the project] yesterday]

Principle of Compositionality The meaning of a sentence is determined by the meanings of its words and the rules that combine them.

2 Constituency Sentence Parsing

目标是用 compositionality 方式解析句子。

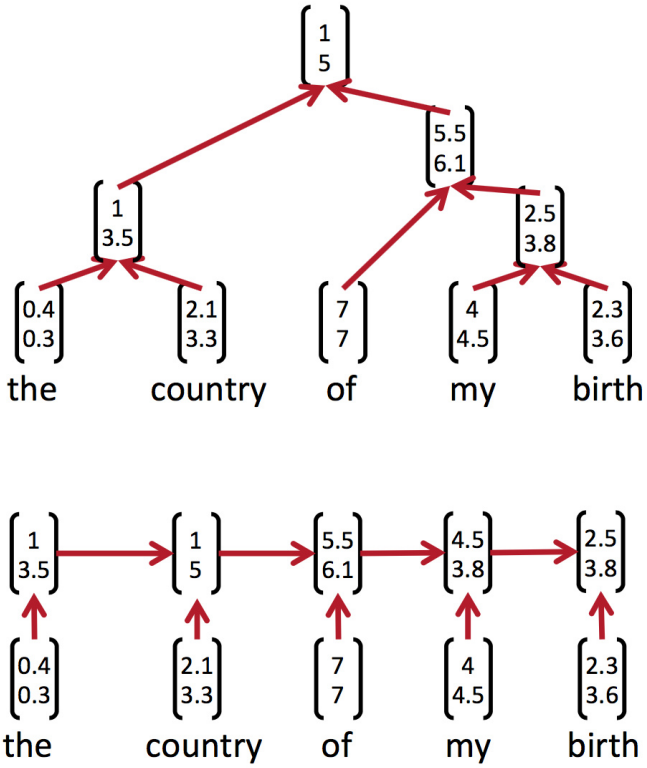
- Meanings: 复合型短语向量
- Rule: 语法树



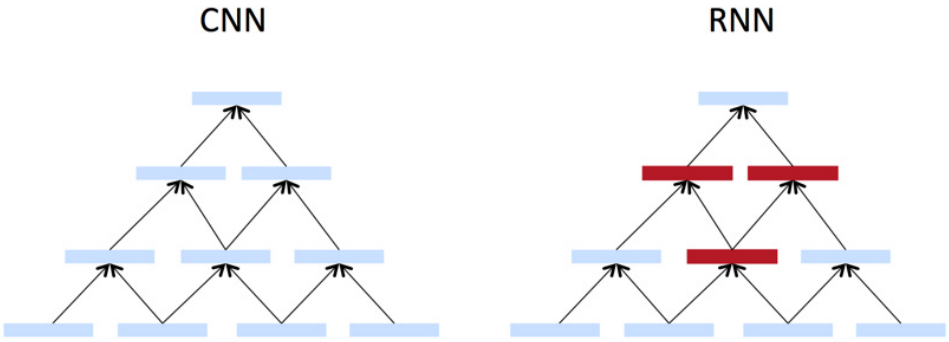
3 Recursive NN v.s. Recurrent NN CNN

Recursive NN 主要是自下而上地提取句子的结构信息。

Recurrent NN 则是通过上下文语境获取信息 (序列信息)。



CNN 则是暴力提取所有 n-grams 组合，not linguistically or cognitively plausible.....

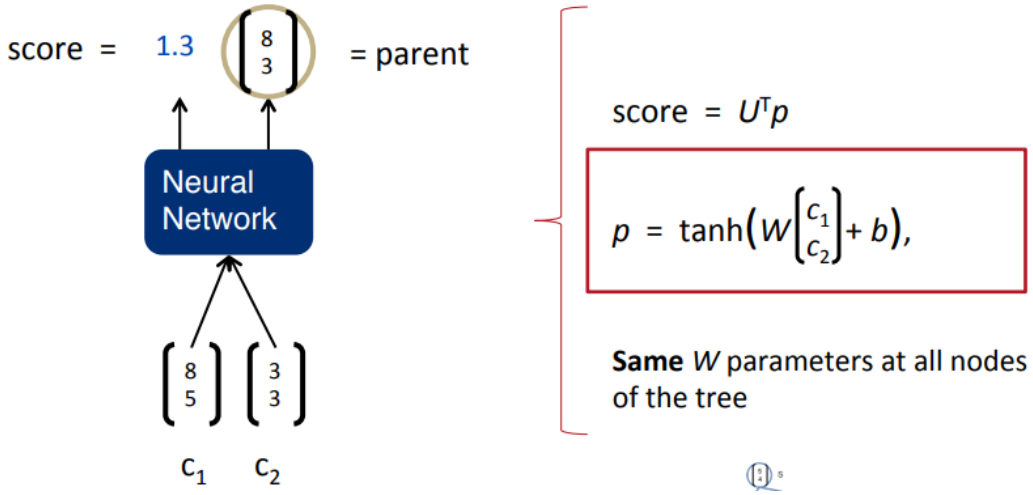


他们各有优缺点，Recursive NN 的训练需要得到语法树，且更新是离散的，not GPU-friendly；Recurrent NN 则只能捕捉 prefix context；CNN 则计算了大量多余的词语组合。

4 Details of Recursive Neural Network

Aim ：同时的到语法树以及 phrase 向量表示。

贪心算法 ：首先面对所有当前 Nodes，两两通过 NN 计算 semantic representation 以及评分，选择评分最大的一组进行合并，成为新的节点，重复上述过程。(Huffman 树思想很接近)



4.1 Objective Function

Max-Margin Framework

$$s(x,y) = \sum_{n \in nodes(y)} s_n$$

x is sentence, y is parse tree.

$$J = \sum_i s(x_i, y_i) - \max_{y \in A(x_i)} (s(x_i, y) + \Delta(y, y_i))$$

$\Delta(y, y_i)$ 是对于错误 parsing 的 penalty，$A(x)$ 是 sentence x_i 的所有可能解析树。

4.2 Backpropagation Through Structure

对于树形结构误差反向传播的三个特点：

1. 将所有 Node 中 W 的导数加和（Like RNN）。
2. 计算导数时对子节点分别计算。
3. 误差反向传播时，将父节点与自己的 error 相加后向下传播。

计算导数时，将共享的权重 W 看做两个不同的权重

$$
\begin{aligned}
& \frac{\partial}{\partial W_2} f(W_2(f(W_1 x)) + \frac{\partial}{\partial W_1} f(W_2(f(W_1 x)) \\
= \quad & f'(W_2(f(W_1 x))\,(f(W_1 x)) + f'(W_2(f(W_1 x))\,(W_2 f'(W_1 x) x) \\
= \quad & f'(W_2(f(W_1 x))\,(f(W_1 x) + W_2 f'(W_1 x) x) \\
= \quad & f'(W(f(W x))\,(f(W x) + W f'(W x) x)
\end{aligned}
$$

反向传播

```
def backprop(self, node, error = None):
    deltas = node.probs
    #特点1
    self.dWs += np.outer(deltas, node.h)
    self.dbs += deltas
    deltas = np.dot(self.Ws.T, deltas)

    #特点3
    if error is not None:
        deltas += error

    deltas *= (node.h != 0) #derivatives of ReLU

    #若为叶子节点更新词向量
    if node.isLeaf:
        self.dL[node.word] += deltas
        return

    #Recursively bp
    if not node.isLeaf:
        self.dW += np.outer(deltas, np.hstack([node.left.h, node.right.h]))
        self.db += deltas
        deltas = np.dot(self.W.T, deltas)
        self.backprop(node.left, deltas[:self.hiddenDim])
        self.backprop(node.right, deltas[self,hiddenDim,:])
```

Single Matrix TreeRNN 存在的问题

1. 共享单一权重矩阵的表达能力有限，只能捕捉到一些简单的关系
2. 词与词之间没有实质性的 interaction，有的只是 concate 后过线性层
3. 对所有的语法结构（verb phrase，prepositional phrase，noun phrase）都使用相同的权重，显然不合理

5 问题和讨论

1. 现有的 Rule-based parser 是否足够可靠?
2. 递归神经网络的目的是什么，是通过构建语法树进而获取副产物——sentence vector 吗?
3. 不同句子的 TreeNet 的结构都不同，无法 train on batch?
4. 同一个句子的每一轮训练 TreeNet 结构也不同，如何构建计算图?