Graph Convolution over Pruned Dependency Trees Improves Relation Extraction

1. Introduction

1.1 出发点

- 依存结构对于关系抽取很重要.
- 依存树结构的句子没有一个有效的可以用mini-batch模拟的好方法.

1.2 本文贡献

- 利用 GCN 去编码依存结构的句子, 在此之上做关系抽出的任务.
- 设计了一个 path-centric pruning 方法去移除树中, 与关系抽取无关的path.
- 对模型进行了分析, 以及提出了一中 pruning 的方法(就是第二步).
- 揭示了 dependency-based 模型与 sequence models 有互补的作用.

2. Models

2.1 Graph Convolutional Networks over Dependency Trees

1) 大概框架介绍

初始版本, 和上一篇论文(SRL)中介绍的一样:

模型共有 L 层, 每一层的中间向量的计算方法为:

$$h_i^{(l)} = \sigma\big(\sum_{j=1}^{n} A_{ij} W^{(l)} h_j^{(l-1)} + b^{(l)}\big)$$

> 其中 A_{ij} 是依存树的邻接矩阵

这里有个问题是, 每个 $h_i^{(l)}$ 的大小差的很多, 因此改进版将其进行归一化:

$$h_i^{(l)} = \sigma\big(\sum_{j=1}^{n} \tilde{A}_{ij} W^{(l)} h_j^{(l-1)} / d_i + b^{(l)}\big),$$

> 其中 $\tilde{A}_{ij} = A + I$. $d_i = \Sigma_{j=1}^{n} \tilde{A}_{ij}$

2) 关于边的方式:

- 这里的边设定的全部是一样的 W

对照实验:

- 设置了 top-down, bottom-up, self-loop 三种形式的边, 结果没什么改进
- 设置了 dependency relation-specific parameters , 反而会对结果有不好的影响.

> 至于会产生这样的结果的原因, 我自己的看法是,
>
> 1. 用依存分析工具标注的句子的依存关系本身就存在错误, 只有方向的标注应该要比什么包含边类型的标注错误率更高一些.
> 2. 依存语法本身就是人为特征, 不一定具有百分之百的合理性. -> 这个需要验证.
>
> 论文中自己的分析是:
>
> 1. 提出的 GCN 模型已经有能力去学得依存关系的类型
> 2. 而增加这一部分信息将会导致overfitting

2.2 Encoding Relations with GCN

1) 模型细节

1. 设句子的标记为 : $\mathcal{X} = [x_1, \ldots, x_n]$
2. 设两个 entities 的 span 分别是 $\mathcal{X}_s = [x_{s1}, \ldots, x_{s2}]$, $\mathcal{X}_o = [x_{o1}, \ldots, x_{o2}]$
3. 将整个句子输入 GCN, 输出 $h^{(L)} = \mathrm{GCN}(h^{(0)})$, 一个矩阵, 每行对应着一个词汇的 embedding.
4. 利用两个 entities 中每个词汇的 embeddings 求 entities 的 embedding, 利用的方法是 max pooling , 例如对第一个entity:

$$h_s = f\left(\mathbf{h}^{(L)}_{s_1:s_2}\right)$$
$$f : \mathbb{R}^{d \times n} \rightarrow \mathbb{R}^d$$

5. 再把句子的向量 也经过 max pooling
6. 通过一个分类器进行分类:

$$h_{\text{final}} = \mathrm{FFNN}\left([h_{\text{sent}}; h_s; h_o]\right)$$

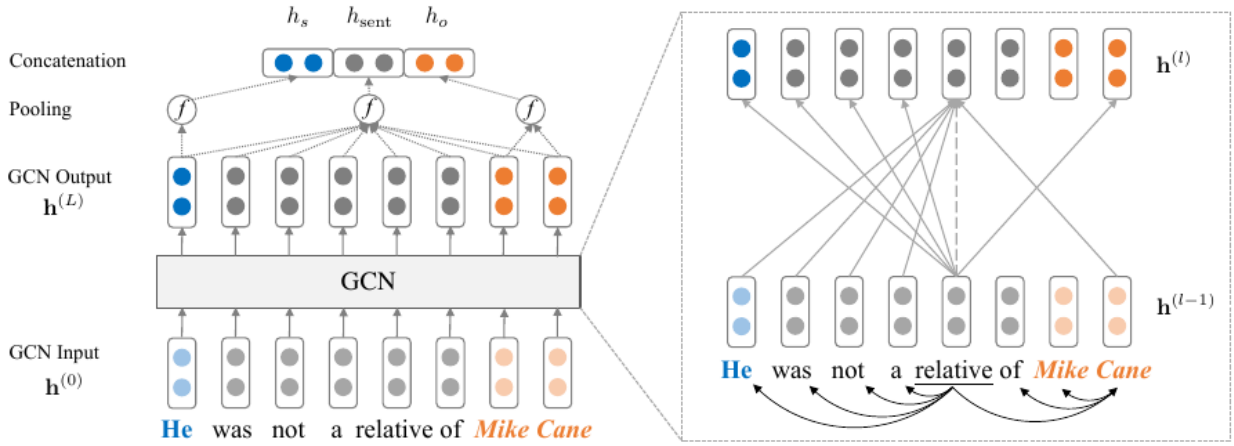


Figure 2: Relation extraction with a graph convolutional network. The left side shows the overall architecture, while on the right side, we only show the detailed graph convolution computation for the word "relative" for clarity. A full unlabeled dependency parse of the sentence is also provided for reference.

2.3 Contextualized GCN

这里还是很上一篇论文一样, 在 GCN 的前面 concat 了一层 Bi-LSTM 层. 因为两个原因

- GNN 无法编码长距离信息
- 依存结构的解析会出现很多错误 (这个是前面没有提到的, 不知道为什么LSTM能解决这个问题)

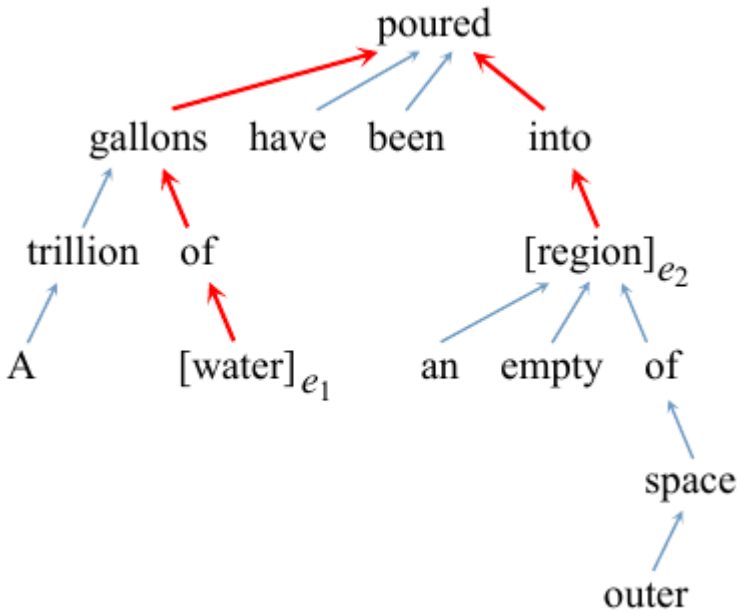
并且, 这个相比于一般的 Tree-LSTM 训练速度提升了上百倍.

3. Path-centric Pruning

3.1 理论基础

1) LCA Tree

在下面的论文中已经被指出：一个关系的大部分信息都是可以通过, 从两者共有的父节点下的子树包含的. 称这个子树为 lowest common ancestor tree (LCA subtree) 如下图:



其中, 两个实体是 water 和 region , 这两个节点的公共父节点就是 poured. 因此, 这个例子下的 LCA subtree 就是整个树.

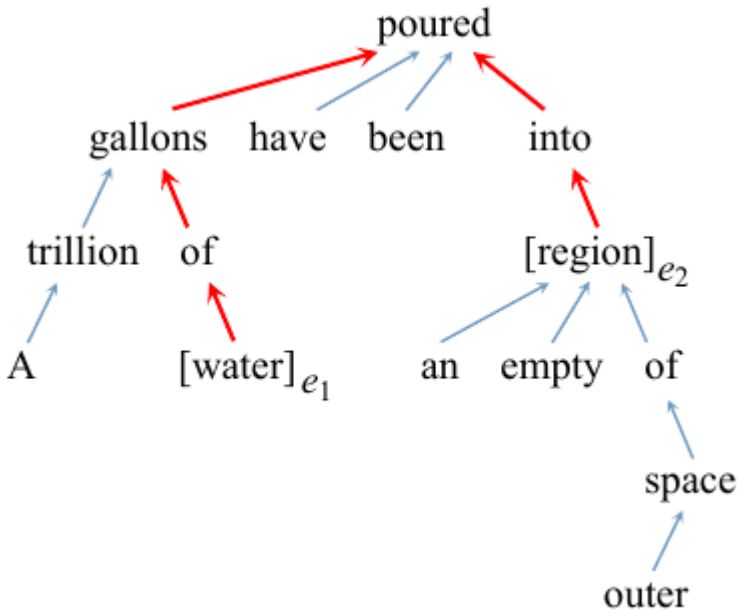
2) 冗余信息

一个句子对完整的树对于关系抽取来说不是必须的, 反而过多的path以及依存关系会影响最后的结果, 因此需要除去多余的path. (同样在下面的论文中提到)

> Yan Xu, 2015b. Classifying relations via long short term memory networks along shortest dependency paths.
>
> Makoto. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures.

3.2 Path-centric Pruning

1) The Shortest Dependency Path



这个里面, 红色的就是最短路径

2) distance K

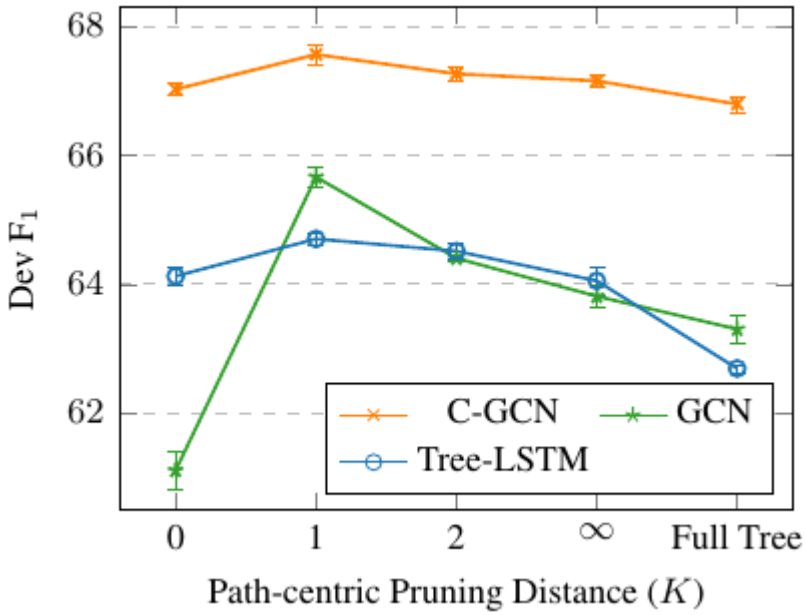
这里引入一个概念 K, 是指, 距离最短路径距离为 K 的图. 例如:

- K=0, 也就是最短路径本身
- K=1, 也就是最短路径上的点, 加上距离最短路径为K的点的全部点的子图.
    - K=无穷, 是指 LCA subtree 全部.

这里测试最佳 K, 答案是 K=1 是最佳答案.

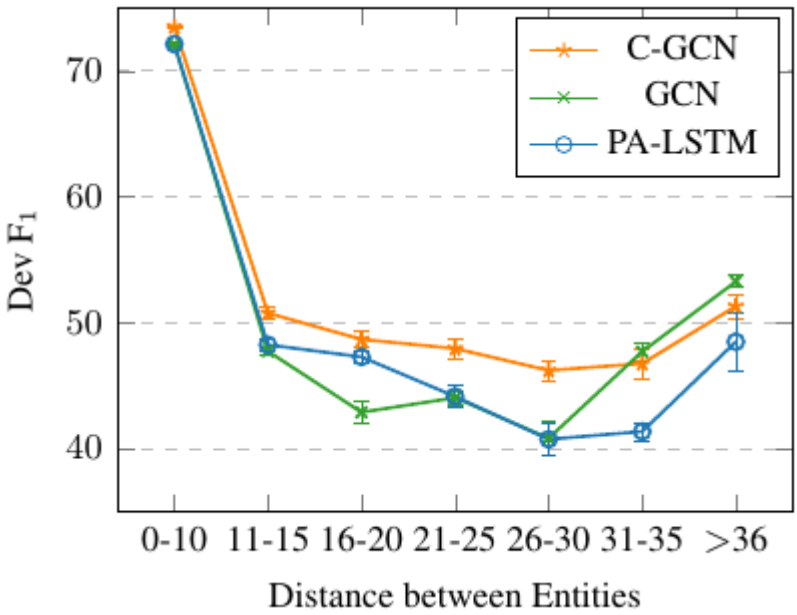
4. Experiments&Analysis

4.1 关于Path-centric Pruning



可以看出, K=1时候效果最好, 最后太多反而是不好, 也就是说 完整的tree<LCA subtree.

4.2 短距离与长距离关系



这里的 PA-LSTM 是一个 sequential 模型, 没有利用到依存信息

> PA-LSTM : position-aware attention mechanism over LSTM outputs

- 看前面部分 : 说明与利用依存信息的方法相比, 在短距离的关系上, 序列模型效果也很好.
- 看后面部分 : 说明在长的距离上, 利用了GCN的模型更好. 但是由于 GCN 本身无法处理长距离, 因此加上一层LSTM会更好.

4.3 Ablation Study

Model	Dev F_1
Best C-GCN	67.4
– h_s, h_o, and Feedforward (FF)	66.4
– LSTM Layer	65.5
– Dependency tree structure	64.2
– FF, LSTM, and Tree	57.1
– FF, LSTM, Tree, and Pruning	47.4

Table 3: An ablation study of the best C-GCN model. Scores are median of 5 models.

> 这里的英语的用法很有意思, 可以借鉴

4.4 Complementary Strengths of GCNs and PA-LSTMs

这里还对 C-GCNs 和 PA-LSTMs 的集成模型进行实验:

System	P	R	F_1
LR† (Zhang+2017)	**73.5**	49.9	59.4
SDP-LSTM† (Xu+2015b)	66.3	52.7	58.7
Tree-LSTM‡ (Tai+2015)	66.0	59.2	62.4
PA-LSTM† (Zhang+2017)	65.7	64.5	65.1
GCN	69.8	59.0	64.0
C-GCN	69.9	63.3	66.4*
GCN + PA-LSTM	71.7	63.0	67.1*
C-GCN + PA-LSTM	71.3	**65.4**	**68.2***

对集成的结果的分析显示, 在847个dev examples上, C-GANs好, 在629个dev examples上, PA-LSTMs好. 这个差异的原因如下(在4.2中也分析过了):

- LSTMs系列更容易处理local feature
- GCNs系列更容易处理entities farther apart