Lecture Notes: Part IV

Dependency Parsing

CS224n 是顶级院校斯坦福出品的深度学习与自然语言处理方向专业课程，核心内容覆盖 RNN、LSTM、CNN、transformer、bert、问答、摘要、文本生成、语言模型、阅读理解等前沿内容。

笔记核心词：

Dependency Grammar, Dependency Structure, Neural Dependency Parsing, 依存解析, 依存句法, 语法依赖

课程**全部资料和信息**已整理发布，扫描下方**任意**二维码，均可获取！！

微信公众号 · 全套资料
回复 CS224n
底部菜单栏

Bilibili · 课程视频
视频简介
置顶评论

GitHub · 项目代码
阅读 ReadMe
点击超链接

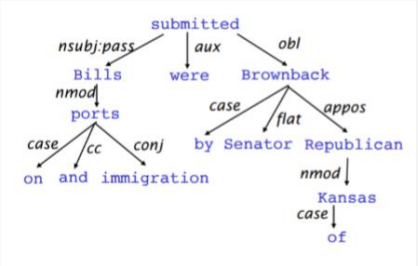
1. Dependency Grammar and Dependency Structure

I Notes info.	
课件/Slides	Lecture 5, P16
视频/Video	Lecture 5, 37:00
GitHub · 代码	实时在线查阅文档
Bilibili · 视频	中英字幕课程视频
Stanford University X ShowMeAI	

与编译器中的解析树类似， NLP 中的解析树是用于分析句子的句法结构。使用的结构主要有两种类型—短语结构和依存结构。

短语结构文法使用短语结构语法将词组织成嵌套成分。以下章节将对此进行更详细的说明。我们现在关注依存语法。

句子的依存结构展示了单词依赖于另外一个单词（修饰或者是参数）。词与词之间的二元非对称关系称为依存关系，描述为从 head（被修饰的主题）用箭头指向 dependent（修饰语）。一般这些依存关系形成树结构。他们通常用语法关系的名称（主体，介词宾语，同位语等）。“*Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas.*”依存树的例子如右图所示：



❐ Figure 1: Dependency tree for this sentence【图 1：这句话的依存树】

有时，在树的头部增加一个假的 ROOT 节点，这样每个单词都依存于唯一一个节点。

1.1 Dependency Parsing

1.1 Notes info.	
课件/Slides	Lecture 5, P30
视频/Video	Lecture 5, 41:00
GitHub · 代码	实时在线查阅文档
Bilibili · 视频	中英字幕课程视频
Stanford University X ShowMeAI	

依存语法是给定一个输入句子 S，分析句子的句法依存结构的任务。依存句法的输出是一棵依存语法树，其中输入句子的单词是通过依存关系的方式连接。正式地，依存语法问题是创建一个输入句子的单词 $S = w_0 w_1 \ldots w_n$（其中 w_0 是 ROOT）到它的依存语法树的映射图 G。最近几年提出了很多以依存句法为基础的的变体，包括基于神经网络的方法，我们将会在后面介绍。

确切地说，在依存语法中有两个子问题：

- 学习：给定用依赖语法图标注的句子的训练集 D，创建一个可以用于解析新句子的解析模型 M
- 解析：给定解析模型 M 和句子 S，根据 M 得到 S 的最优依存语法图

1.2 Transition-Based Dependency Parsing

1.2 Notes info.	
课件/Slides	Lecture 5, P32
视频/Video	Lecture 5, 53:00
GitHub · 代码	实时在线查阅文档
Bilibili · 视频	中英字幕课程视频
Stanford University X ShowMeAI	

Transition-based 依存语法依赖于定义可能转换的状态机，以创建从输入句到依存句法树的映射。学习问题是创建一个可以根据转移历史来预测状态机中的下一个转换的模型。分析问题是使用在学习问题中得到的模型对输入句子构建一个最优的转移序列。大多数 Transition-based 系统不会使用正式的语法。

1.3 Greedy Deterministic Transition-Based Parsing

这个系统是由 Nivre 在 2003 年提出，与当时的常用方法截然不同。

这个转换系统是一个状态机，它由状态和这些状态之间的转换组成。该模型导出了从初始状态到几种终端状态之一的一系列转换。

状态：对任意句子 $S = w_0 w_1 \dots w_n$ ，一个状态可以描述为一个三元组 $c = (\sigma, \beta, A)$：

- 来自 S 的单词 w_i 的堆 σ
- 来自 S 的单词 w_i 的缓冲区 β
- 一组形式为 (w_i, r, w_j) 的依存弧，其中 w_i, w_j 是来自 S，和 r 描述依存关系。

因此，对于任意句子 $S = w_0 w_1 \dots w_n$

- 一个形式为 $([w_0]_\sigma, [w_1 \dots w_n]_\beta, \varnothing)$ 的初始状态 c_0（现在只有 ROOT 在堆 σ 中，没有被选择的单词都在缓冲区 β 中。
- 一个形式为 $(\sigma, []_\beta, A)$ 的终点状态。

转移：在状态之间有三种不同类型的转移：

- SHIFT：移除在缓冲区的第一个单词，然后将其放在堆的顶部（前提条件：缓冲区不能为空）。
- $\text{Left} - \text{Arc}_r$：向依存弧集合 A 中加入一个依存弧 (w_j, r, w_i)，其中 w_i 是堆顶的第二个单词，w_j 是堆顶部的单词。从栈中移除 w_i（前提条件：堆必须包含两个单词以及 w_i 不是 ROOT ）。
- $\text{Right} - \text{Arc}_r$：向依存弧集合 A 中加入一个依存弧 (w_i, r, w_j)，其中 w_i 是堆顶的第二个单词，w_j 是堆顶部的单词。从栈中移除 w_j（前提条件：堆必须包含两个单词）。

下图给出了这三个转换的更正式的定义

1. Shift $\quad \sigma, w_i \mid \beta, A \rightarrow \sigma \mid w_i, \beta, A$
2. Left-Arc$_r$ $\quad \sigma \mid w_i \mid w_j, \beta, A \rightarrow \sigma \mid w_j, \beta, A \cup \{r(w_j, w_i)\}$
3. Right-Arc$_r$ $\quad \sigma \mid w_i \mid w_j, \beta, A \rightarrow \sigma \mid w_i, \beta, A \cup \{r(w_i, w_j)\}$

❐ ← Figure 2: Transitions for Dependency Parsing.【图 2：依赖解析的转换】

1.3 Notes info.

课件/Slides	Lecture 5, P33
视频/Video	Lecture 5, 54:30
GitHub • 代码	实时在线查阅文档
Bilibili • 视频	中英字幕课程视频

Stanford University X ShowMeAI

1.4 Neural Dependency Parsing

虽然依赖项解析有很多深层模型，这部分特别侧重于贪心，基于转移的神经网络依存语法解析器。与传统的基于特征的判别依存语法解析器相比，神经网络依存语法解析器性能和效果更好。与以前模型的主要区别在于这类模型依赖稠密而不是稀疏的特征表示。

1.4 Notes info.	
课件/Slides	Lecture 5, P38
视频/Video	Lecture 5, 62:40
GitHub · 代码	实时在线查阅文档
Bilibili · 视频	中英字幕课程视频
Stanford University X ShowMeAI	

我们将要描述的模型采用上一部分中讲述的标准依存弧转换系统。最终，模型的目标是预测从一些初始状态 c 到一个终点状态的转换序列，对模型中的依存语法树进行编码的。由于模型是贪心的，它基于从当前的状态 $c = (\sigma, \beta, A)$ 提取特征，然后尝试一次正确地预测一次转移 $T \in \{SHIFT, Left - Arc_r, Right - Arc_r\}$ 。回想一下， σ 是栈，β 是缓存，A 是对于一个给定的句子的依赖弧的集合

特征选择：根据该模型所需的复杂性，定义神经网络的输入是有灵活性的。对给定句子 S 的特征包含一些子集：

1. S_{word} ：在堆 σ 的顶部和缓冲区 β 的 S 中一些单词的词向量（和它们的依存）。
2. S_{tag} ：在 S 中一些单词的词性标注（ POS ）。词性标注是由一个离散集合组成： $\mathcal{P} = \{NN, NNP, NNS, DT, JJ, \ldots\}$ 。
3. S_{label} ：在 S 中一些单词的依存标签。依存标签是由一个依存关系的离散集合组成： $\mathcal{L} = \{amod, tmod, nsubj, csubj, dobj, \ldots\}$。

对每种特征类型，我们都有一个对应的将特征的 one-hot 编码映射到一个 d 维的稠密的向量表示的嵌入矩阵。S_{word} 的完全嵌入矩阵是 $E^w \in \mathbb{R}^{d \times N_w}$ ，其中 N_w 是字典/词汇表的大小。相应地， POS 和依存标签的嵌入矩阵分别为 $E^t \in \mathbb{R}^{d \times N_t}$ 和 $E^l \in \mathbb{R}^{d \times N_l}$ ，其中 N_t 和 N_l 分别为不同词性标注和依存标签的个数。最后，定义从每组特征中选出的元素的数量分别为 $n_{word}, n_{tag}, n_{label}$ 。

特征选择的例子：作为一个例子，考虑一下对 S_{word} ， S_{tag} 和 S_{label} 的选择：

1. S_{word} ：在堆和缓冲区的前三个单词： $s_1, s_2, s_3, b_1, b_2, b_3$ 。栈顶部两个单词的第一个和第二个的 leftmost / rightmost 的子单词： $lc_1(s_i), rc_1(s_i), lc_2(s_i), rc_2(s_i), i = 1,2$ 。栈顶部两个单词的第一个和第二个的 leftmost of leftmost / rightmost of rightmost 的子单词：

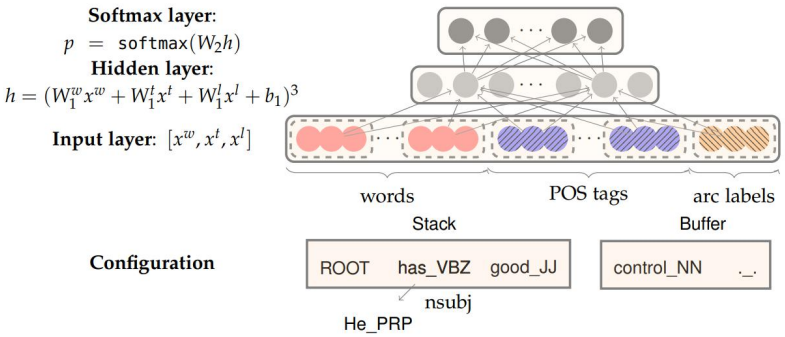
2. $lc_1(lc_1(s_i)), rc_1(rc_1(s_i)), i = 1,2$ 。S_{word} 总共含有 $n_{word} = 18$ 个元素。

3. S_{tag}：相应的词性标注，则 S_{tag} 含有 $n_{tag} = 18$ 个元素。

4. S_{label}：单词的对应的依存标签，不包括堆/缓冲区上的 6 个单词，因此 S_{label} 含有 $n_{label} = 12$ 个元素。

注意我们使用一个特殊的 NULL 表示不存在的元素：当堆和缓冲区为空或者还没有指定依存关系时。对一个给定句子例子，我们按照上述的方法选择单词，词性标注和依存标签，从嵌入矩阵 E^w, E^t, E^l 中提取它们对应的稠密的特征的表示，然后将这些向量连接起来作为输入 $[x^w, x^t, x^l]$ 。在训练时间，我们反向传播到稠密的向量表示，以及后面各层的参数。

前馈神经网络模型：这个神经网络包含一个输入层 $[x^w, x^t, x^l]$，一个隐藏层，以及具有交叉熵损失函数的最终 softmax 层。我们可以在隐藏层中定义单个权值矩阵，与 $[x^w, x^t, x^l]$ 进行运算，我们可以使用三个权值矩阵 $[W_1^w, W_1^t, W_1^l]$，每个矩阵对应着相应的输入类型，如下图所示。然后我们应用一个非线性函数并使用一个额外的仿射层 $[W_2]$，使得对于可能的转移次数（输出维度），有相同数量的 softmax 概率。



❐ Figure 3: The neural network architecture for greedy, transition-base dependency parsing.【图 3：贪婪的神经网络架构，转换基依赖解析】

注意在上图中，使用的非线性函数是 $f(x) = x^3$ 。

有关 greedy transition-based 神经网络依存语法解析器的更完整的解释，请参考论文：A Fast and Accurate Dependency Parser using Neural Networks。

机器学习	深度学习	自然语言处理	计算机视觉	知识图谱
Machine Learning	Deep Learning	Natural Language Processing	Computer Vision	Knowledge Graphs
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n	Stanford · CS520

 # 系列内容 Awesome AI Courses Notes Cheatsheets

图机器学习	深度强化学习	自动驾驶
Machine Learning with Graphs	Deep Reinforcement Learning	Deep Learning for Self-Driving Cars
Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094
...	...	...

是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 TOP20+门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

斯坦福大学(Stanford University) Natural Language Processing with Deep Learning (CS224n) 课程，是本系列的第三门产出。

课程版本为 2019 Winter，核心深度内容(transformer、bert、问答、摘要、文本生成等)在当前(2021 年)工业界和研究界依旧是前沿的方法。最新版课程的笔记生产已在规划中，也敬请期待。

笔记内容经由深度加工整合，以 5 个部分构建起完整的“CS224n 内容世界”，并依托 GitHub 创建了汇总页。快扫描二维码，跳转进入吧！有任何建议和反馈，也欢迎通过下方渠道和我们联络 (*￣3￣)~