lecture05 Linguistic Structure: Dependency Parsing

语言结构的两种观点

- 上下文无关文法CFGs
  短语结构将单词组织成嵌套的成分
  单词组合成不同类别的短语；短语可以递归组合成更大的短语
- 依存结构
  显示哪些单词依赖于其他哪些单词
  e.g. Look in the large crate in the kitchen by the door
  - 【look】是句子的根，【in the large crate】是【look】的一个依赖
  - 【the large】、【large】是【crate】的一个依赖
  - 【in the kitchen】、【the kitchen】是【crate】的修饰
  - 【by the door】、【the door】是【crate】的修饰

为什么需要句子结构

- 为了正确的解释语言
- 单词组合成更大的单元来传递更复杂的思想
- 需要知道什么与什么关联

介词短语依附歧义

e.g. San Jose cops kill man with knife

- 警察用刀杀了那个男人
  knife是kill的修饰
- 警察杀了那个有刀的男人
  knife是man的修饰

介词短语能修饰它前面的名词或是动词

面对复杂的句子结构，歧义就会增加
e.g. The board approved【its acquisition】
【by Royal Trustco Ltd.】
【of Toronto】
【for $27 a share】
【at its monthly meeting】

- 【by Royal Trustco Ltd.】修饰 【acquisition】

- 【of Toronto】修饰【Royal Trustco Ltd.】
- 【for $27 a share】修饰 【acquisition】
- 【at its monthly meeting】修饰【approved】

协调范围模糊

e.g. Shuttle veteran and longtime NASA executive Fred Gregory appainted to board.

依存语法与依存结构

依存语法假定句法结构由词汇项之间的关系构成，通常称为依赖关系的二元非对称关系（“箭头”）
依存关系组成了树（连接、无环、单根的）

依存语法/短语的历史：

- 依赖结构的概念可以追溯到
  - 帕尼尼(约公元前5世纪)，
  - 第一个千年阿拉伯语法学家的基本方法，
- 支持/上下文无关语法是20世纪的新发明(R.S.威尔斯，1947;乔姆斯基...）
- 现代的依赖工作通常来源于L. Tesniere(1959)。
- 对于NLP中最早的几种解析器 - 美国计算语言学的创始人之一David Hays在早期(第一次?)建立了依赖解析器(Hays 1962)。

依存关系箭头方向：均可。课上使用的是Tesniere的方法，即箭头指向被修饰词（dependent）

标注数据的兴起

使用人工制定语法规则很有效，但是使用treebank更好

- 劳动的可重用性
  - 可以在其上构建许多解析器、词性标记等
  - 语言学的宝贵资源
- 广泛的覆盖面，而不仅仅是一些直觉，
- 频率和分布信息
- 一种评估系统的方法

很多结构是模棱两可的，而在使用语法时，无法知道哪一个是正确的结构

依存条件偏好

依赖项解析的信息来源是什么?
e.g. ROOT Discussion of the outstanding issues was completed。

1. 双性亲和【discussion->issues】似是而非的关系
2. 依赖距离主要与相邻词有关
   3.插入词
   依赖关系很少跨越中间的动词或标点符号
3. 价的头一个人头通常靠哪边靠多少人?

依存项解析

通过为每个单词选择它所依赖的其他单词(包括根)来解析一个句子
通常一些限制:

- 只有一个词是由词根决定的。
- 不需要循环A→B, B> A

这使得依赖项成为树
最后一个问题是箭头是否可以交叉(非射影)

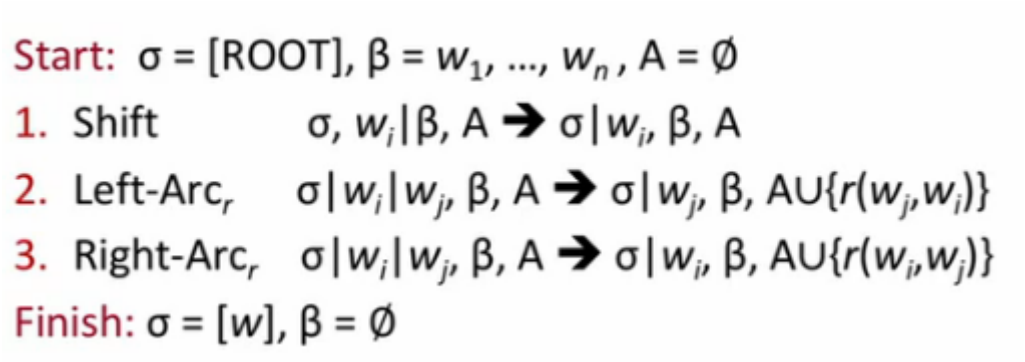
依存分析方法

1. 动态规划
   Eisner(1996)给出了一个具有复杂度O(n3)的算法，它生成的解析项的头部位于末端而不是中间
2. 图算法
   为一个句子创建一个最小生成树McDonald等人(2005)的MSTParser使用ML分类器独立地对依赖项进行评分(他使用MIRA进行在线学习，但它也可以是其他东西)
3. 约束满意度
   不满足硬约束的边被删除，Karlsson(1990)等。
4. **“基于转换的解析”或“确定性依赖解析”**
   由好的机器学习分类器引导的贪婪的附件选择MaltParser (Nivre et al. 2008)。已证明非常有效。

基于转换的解析

一种简单的贪婪判别依赖解析器

- 该解析器执行一系列自底向上的操作
  - 大致类似于shift-reduce解析器中的“shift”或“reduce”，但是“reduce”操作是专门用来创建依赖项的，其头部位于左侧或右侧
- 解析器有:
  - 堆栈a，从根符号开始，从上到右写。
  - 一个缓冲区Beta，写在左上角，它以输入语句开头，
  - 一组依赖弧A，一开始是空的
  - 一组动作



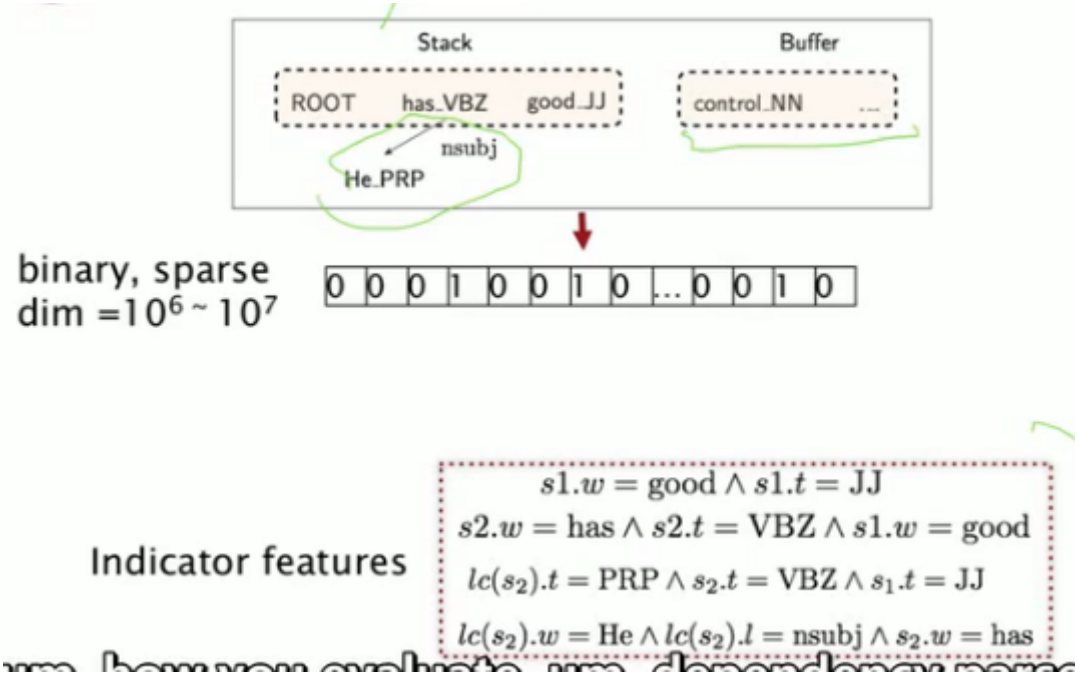
分析“I ate fish”

	stack		buffer
Start	【root】		I ate fish
Shift	【root】 I		ate fish
Shift	【root】 I ate		fish
Left Arc	【root】 ate		fish
Shift	【root】 ate fish		
Right Arc	【root】 ate		
Right Arc	【root】		

MaltParser

- 我们已经解释了我们是如何选择下一步行动的答案:退后
- 每一个动作的每个合法移动都由一个有区别的分类器(例如softmax分类器)来预测
- 没有搜索
- 在依赖项解析方面，模型的准确性略低于目前的水平，但是
- 提供快速的线性时间解析，性能好

传统的特征表示



表示为二进制稀疏的向量

依赖项解析的评估:(标记)依赖项准确性

ACC = #correct deps / #of deps
UAS：无标记依存正确率
LAS：有标记依存正确率

为什么要训练神经依存解析器

传统特征表示的问题：

1. 稀疏
2. 不完整
3. 计算复杂

神经依存解析器【Chen and Manning 2014】

斯坦福依赖的英语解析:
(UAS) =head
(LAS) =头和标签
效果好，速度快

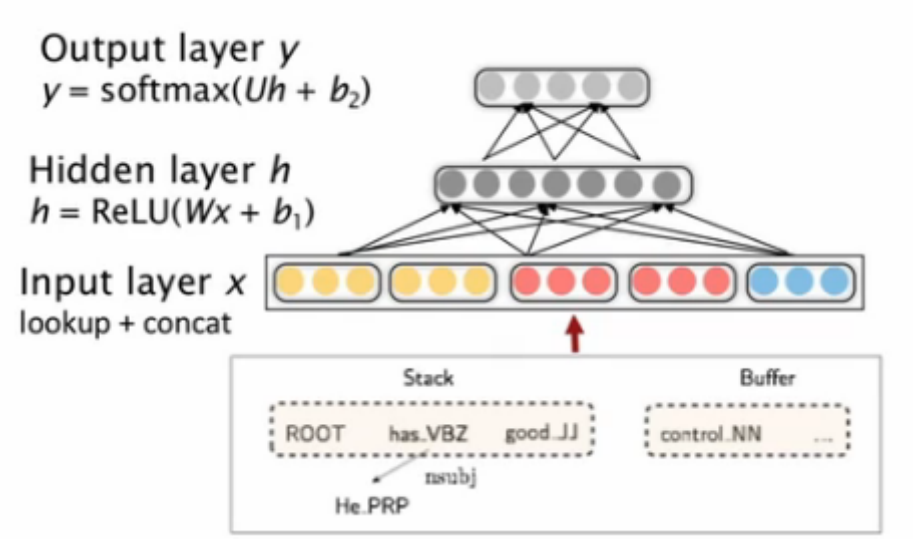
分布式表示：

- 我们将每个单词表示为一个密度向量(即密度向量)。,字嵌入)相似的单词应该有相近的向量。

- 同时，词性标签(POS)和依赖标签也被表示为d维向量。
  较小的离散集也表现出许多语义上的相似性。

我们根据堆栈/缓冲区位置提取一组tokens，转换成词向量，并作为输入层，再经过若干非线性的隐藏层，最后加入softmax得到shift-reduce解析器的动作

模型架构：



further developments

- 更大、更深的网络，具有更好的超参数
- 波束搜索
- 决策序列上的全局、条件随机域(CRF)式推理
- 这就引出了SyntaxNet和Parsey McParseFace模型