句法分析

郝舒迪 路畅 陈敬歌 戴灼 康姝元

什么是句法分析

- 简单来说，句法分析就是对一个句子中的词语语法功能进行分析。
- 从句子成分上讲，有主谓宾等多种句子成分，不同的词语在句子中充当不同的成分会使得句子有不同的理解。
- 句法分析在自然语言处理中起着承上启下的作用，位于词法分析之后语义分析之前。

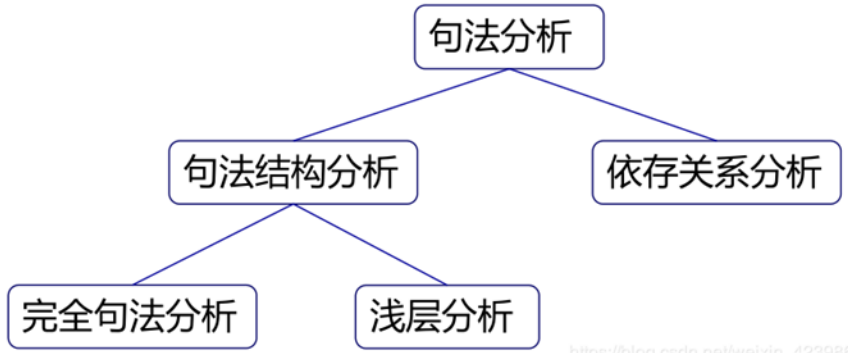
句法分析主要任务

- 判断字符串是否符合某种语言；
- 消除词法和结构方面的歧义；
- 分析句子的内部结构，如成分构成，上下文关系。

句法分析的分类

句法分析分为句法结构分析和依存关系分析两种。

- 以获取整个句子的句法结构为目的的句法分析，被称为完全句法结构分析，以获取局部成分为目的的句法分析为浅层分析
- 分析语言单位内成分之间的依存关系的句法分析，被称为依存关系分析



句法分析的分类

- 现代的句法分析逐渐分为两个方向，成分句法分析和依存句法分析。根据hankcs的说法;这并不是随机选择，而是由于前者的优势。90年代的句法分析论文99%都是短语结构树，但后来人们发现依存句法树标注简单，parser准确率高，所以后来（特别是最近十年）基本上就是依存句法树的天下了（至少80%）。

短语结构句法分析

短语句法分析的发展过程：

- 第一阶段（1950年初-1990年左右）

使用语法规则进行短语结构句法分析，基于语言学家提供的语法规则形成规则库后，根据已有的规则进行句法分析，速度慢，准确率较低。

- 第二阶段（1990年-2010年左右）

将统计的方法引入句法分析过程，利用大量标注好短语成分的语料作为基础进行概率的计算，找到概率最大的短语标注，速度与准确率都大有提升。

- 第三阶段（2010年-至今）

在以上两种的基础上将神经网络引入句法分析，将RNN、LSTM等神经网络应用在句法分析中，使得模型更加符合语言规则。

基于规则的方法

- CFG上下文无关文法，表示为一个四元组

$$N = \{\text{S, NP, VP, PP, DT, Vi, Vt, NN, IN}\}$$
$$S = \text{S}$$
$$\Sigma = \{\text{sleeps, saw, man, woman, telescope, the, with, in}\}$$

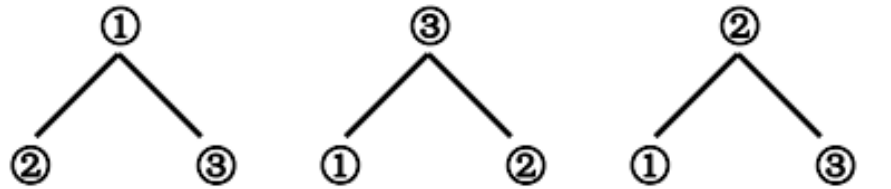
$R =$

S	→	NP	VP
VP	→	Vi	
VP	→	Vt	NP
VP	→	VP	PP
NP	→	DT	NN
NP	→	NP	PP
PP	→	IN	NP

Vi	→	sleeps
Vt	→	saw
NN	→	man
NN	→	woman
NN	→	telescope
DT	→	the
IN	→	with
IN	→	in

- 基于这样的规则不断搜索，找到一棵满足规则的语法结构树

- 依据语法树形成方向的区别可以分为三种：自顶向下的方法，自底向上的方法以及二者相结合的方法

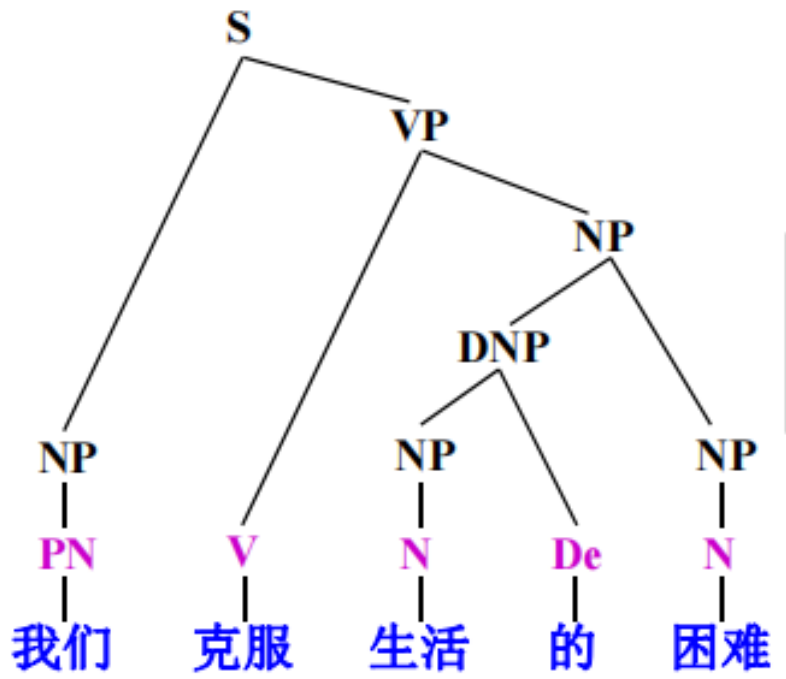


- 例如

S → NP VP　　S → NP VP　　S → NP VP

Rules: **Lexicon:**

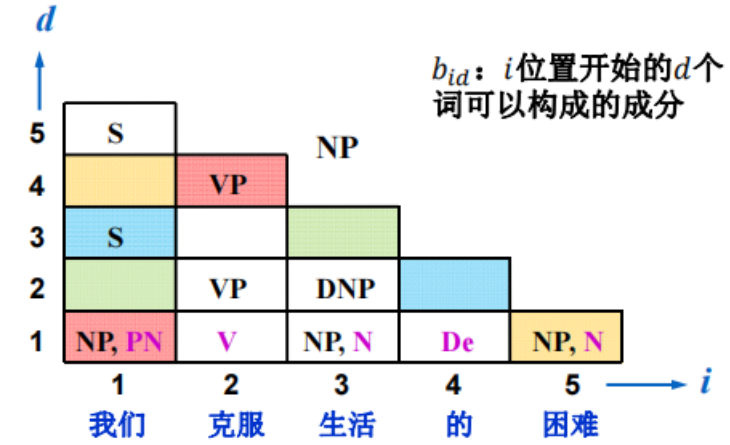
S → NP VP	我们：PN
VP → V NP	克服：V
NP → PN	生活：N
NP → N	的：De
NP → DNP NP	困难：N
DNP → NP De	



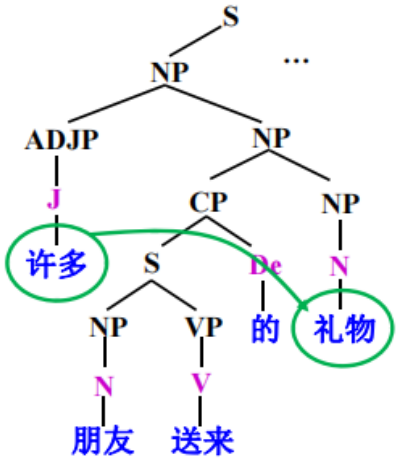
基于规则的算法

- CYK算法
  - 是一种自底向上基于Chomsky范式的算法，本质上是动态规划的算法。
  - 构造一个二维矩阵，b_{id}表示以第 个词为第一个词开始的d个词(包 含)，所有可能形成的短语的非终结符的集合

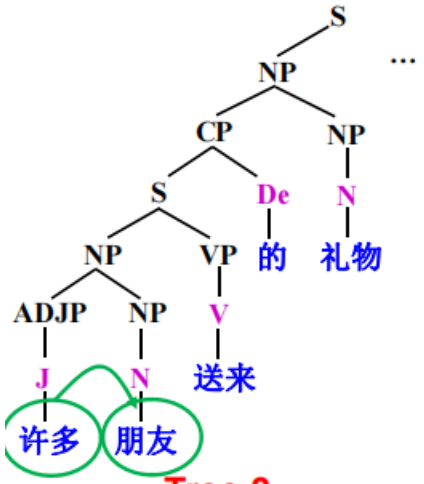
S → NP VP	NP → PN
VP → V NP	NP → N
DNP → NP De	NP → DNP NP



- 缺点
  - 对于中等长度的输入句子来说，基于规则的方法分析出所有可能的句子结构比较困难
  - 句子成分复杂，不同词语可以充当不同的句子成分，容易产生歧义



Tree 1



Tree 2

  - 手工编写规则工作量大，而且编写的规则领域有密切的相关性，不利于句法分析系统向其他领域移植。

基于统计的算法

- PCFG法

在CFG的基础上，在制定规则时加入概率，表示出每个词语充当不同句子成分的概率。

- 输入: I saw a girl with a telescope

PCFG: $G=(V_n, V_t, S, P)$

$V_n = \{S, VP, NP, PP\}$

$V_t = \{I, saw, a, girl, with, telescope\}$

$S=S$

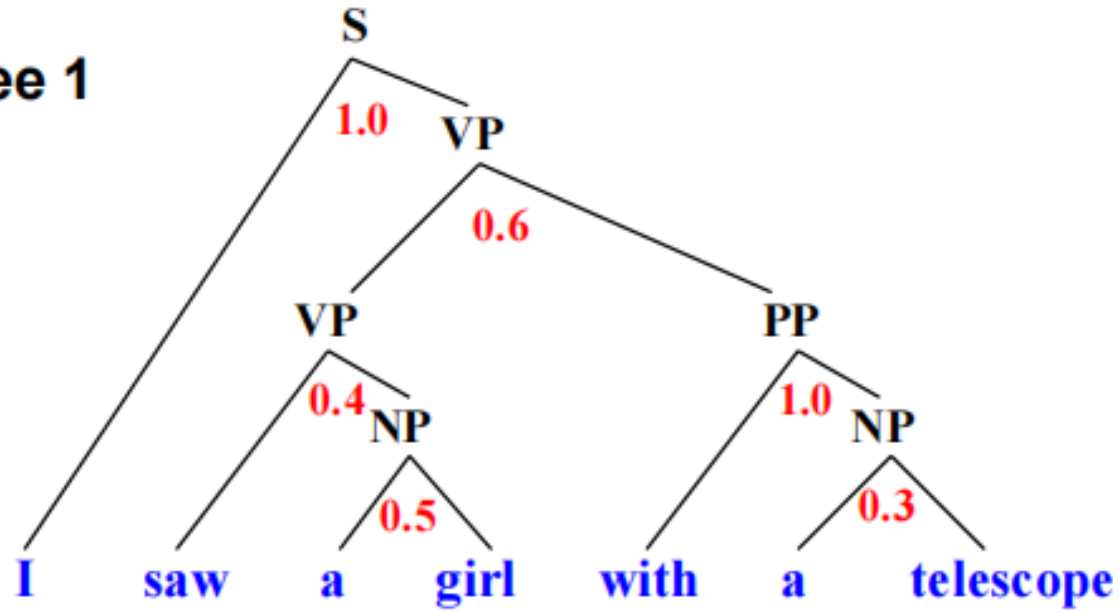
P: S → I VP, 1.0; NP → NP PP, 0.2;

VP → saw NP, 0.4; NP → a girl, 0.5;

VP → VP PP, 0.6; NP → a telescope, 0.3.

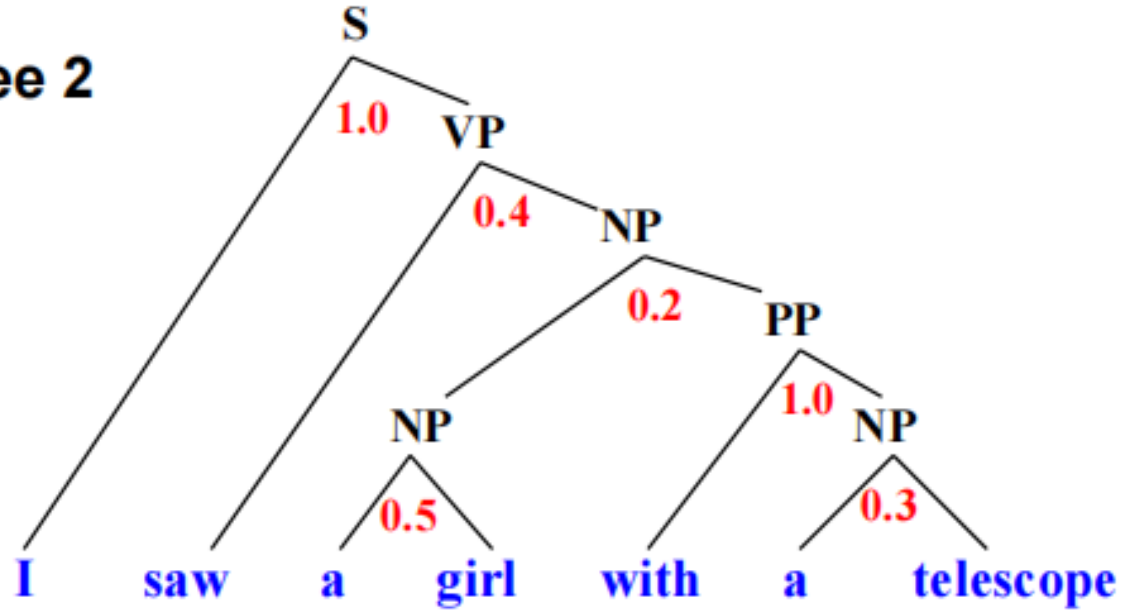
PP → with NP, 1.0;

Tree 1



$1.0 \times 0.6 \times 0.4 \times 0.5 \times 1.0 \times 0.3 = 0.036$

Tree 2

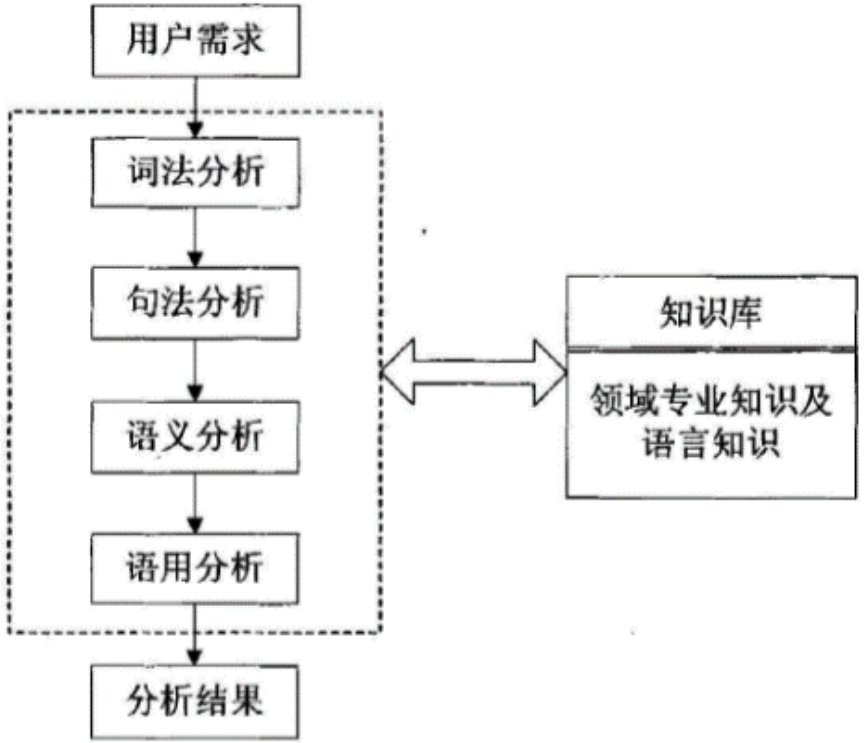


$1.0 \times 0.4 \times 0.2 \times 0.5 \times 1.0 \times 0.3 = 0.012$

- 基于PCFG的句法分析模型，满足以下三个条件：
  - 位置不变性：子树的概率不依赖于该子树所管辖的单词在句子中的位置
  - 上下文无关性：子树的概率不依赖于子树控制范围以外的单词
  - 祖先无关性：子树的概率不依赖于推导出子树的祖先节点

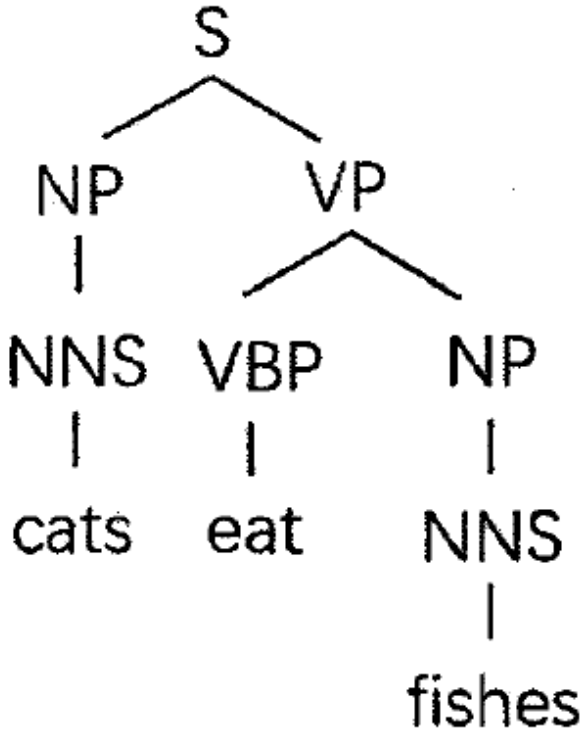
应用

- 基于规则和统计的短语结构分析可以用于设计领域来进行设计需要的分析。
- 建立静态知识库→分析和词性标注→句法分析，完成对用户需求信息的理解



基于深度神经网络的成分句法分析

- 解析树



(a)

(S (NP (NNS cats)) (VP (VBP eat) (NP (NNS fishes))))

(b)

图 1.1　一颗解析树（a）和它的序列形式（b）

传统成分句法分析存在的问题

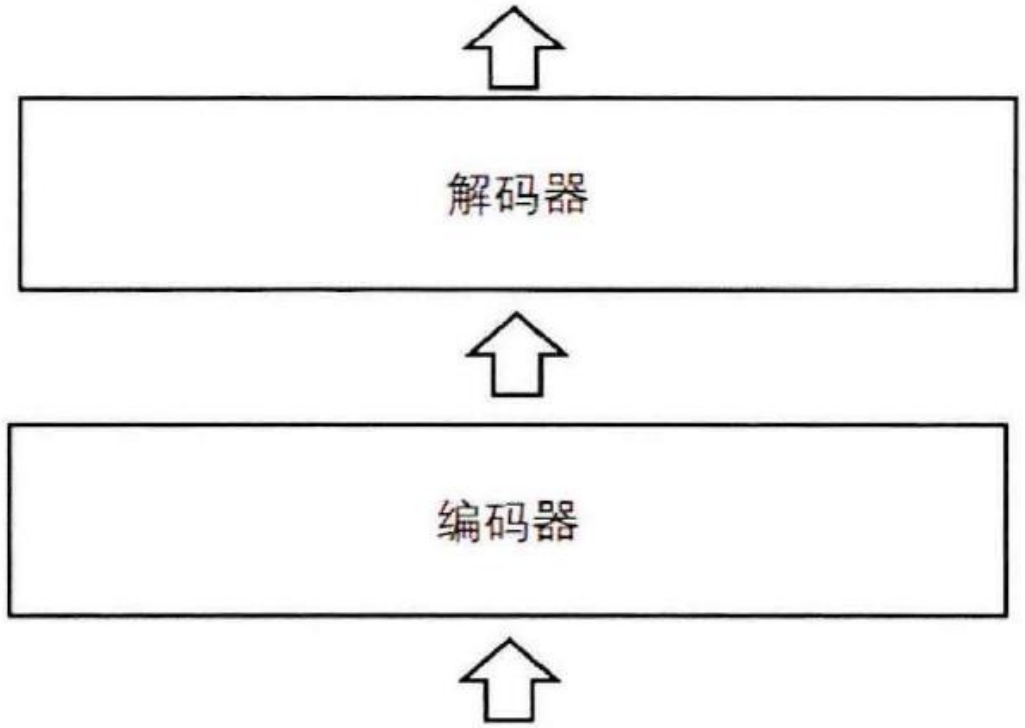
- 特征表示较为稀疏不完全
- 太过依赖于人工语法规则，计算代价大大增加
- 模式仍然依赖于上下文无关文法这一骨干

基于深度神经网络的成分句法分析

- 局域方法
  - 头词汇化
  - 结构诠释
  - 状态分割
  - ……
- 非局域方法
  - 重排序方法
  - 树替换解析器
  - 神经网络解析器
  - ……

神经网络编码器-解码器模型

输出 … (NP(PRP We))(VP(VBP 're)(VP(IN about)(S(VP(TO to)(VP(VB see)…



输入	…	We	're	about	to	see	…
	…	PRP	VBP	IN	TO	VB	…

神经网络编码器 – 解码器模型

$$s(T) = \sum_{(i,j,l) \in T} s(i, j, l)$$

$$\hat{T} = \arg\max_{T} s(T)$$

$$\max\left(0, \max_{T \neq T^*}[s(T) + \Delta(T, T^*)] - s(T^*)\right)$$

基于深度神经网络的成分句法分析

基于解码器的改进模型

- 基于转移系统的模型
- 自顶向下
- 自底向上
- 中序策略

基于图表的模型
解码算法是由传统的CKY算法改进而来的，它将一颗句法解析树看作是句子中所有带标签的短语跨度的集合，由神经网络计算出每个跨度的得分后，找到总得分最高的解析树。

基于编码器的改进模型

- （1）前馈神经网络
- 早期有着表面特征的成分句法分析系统通常使用围绕跨度左右两端固定窗口大小的前馈神经网络来进行编码，学习非线性特征，然后解码推理过程处理了大部分来自句子其余部分的信息传播。

基于编码器的改进模型

- （2）循环神经网络
- 双向LSTM

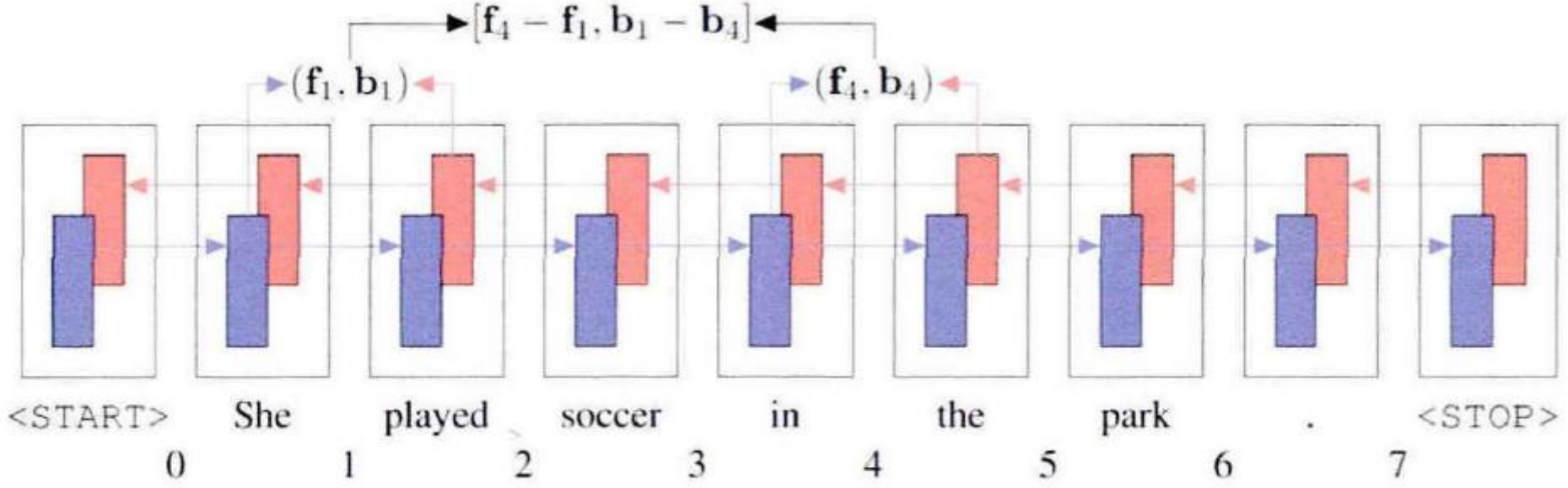


图 1.5　使用双向 LSTM 进行跨度表示[55]

基于编码器的改进模型

- （3）注意力机制
- 2017年，Vaswani等人提出了完全基于注意力机制的Transformer模型，在机器翻译任务和成分句法分析任务上都取得了优异的结果。
- 2018年，Kitaev等人改进了Transformer模型，使用了一个八层八头的自我注意力编码器，取得了当时最优的结果。

面临的挑战与研究动机

- 自我注意力编码器结构的冗余
- 局部成分句法分析器的鲁棒性

浅层句法分析

- **浅层句法分析**，也叫部分句法分析或语块分析，来自然语言处理领域出现的一种新的语言处理策略。它是与完全句法分析相对的，完全句法分析要求通过一系列分析过程，最终得到句子的完整的句法树。而浅层句法分析则不要求得到完全的句法分析树，它只要求识别其中的某些结构相对简单的成分，如非递归的名词短语、动词短语等。这些识别出来的结构通常被称作语块，语块和短语这两个概念通常可以换用。

浅层句法分析

- 基于SVM的base NP识别方法
- 基于WINNOW的base NP识别方法
- 基于CRF的base NP识别方法

依存结构句法分析

依存句法分析

- 百度百科定义：依存句法是由法国语言学家L.Tesniere最先提出，它将句子分析成一棵依存句法树，描述出各个词语之间的依存关系。
- 用词与词之间的支配和被支配关系来描述语言结构

产生和发展

- 严格来讲，法国语言学家吕西安·泰尼埃的遗作《结构句法》于 1959 年的发表，标志着依存语法的正式诞生。这个时间虽然略晚于乔姆斯基的《句法结构》（1954），但也引起了以德国学者为代表的语言学家的关注。从泰尼埃的论述中，学者们认识到了依存语法和短语结构语法的本质区别，这在欧洲尤其是德国引发了运用依存语法理论解决问题的热潮。值得注意的是，生成语法此时已经统治了美国语言学界，但仍有学者将关注的目光投向依存语法。Hays 正式提出了“依存”和“依存语法”两个术语，并且形成了一种完全基于依存关系的句子结构分析方法。
- 泰尼埃与 Hays 是今天公认的现代依存语法的先驱，在两位学者之后，依存语法理论的发展势头十分迅猛。其中，产生广泛影响的主要有四家，即理查德·哈德森的“词语法（Word Grammar）”理论、Mel’čuk的“意义—文本理论（Meaning-Text Theory）”、Petr Sgall 等人的“功能生成描述（Functional Generative Description）”理论、Stan Starosta 的“词格（Lexicase）”理论

汉语的依存分析研究

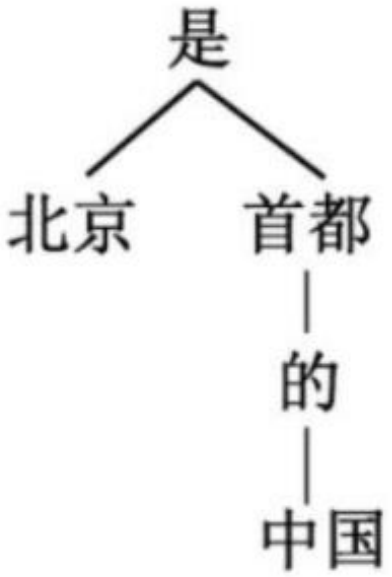
- 黄昌宁等介绍了一种基于语料库的依存分析；周明、黄昌宁提出了一种基于规则和统计的汉语依存分析模型；刘伟权等初步建立起汉语依存关系的层次体系；Zhou 结合浅层短语结构分析和深层依存分析所研制的分析方法，已应用于汉日机器翻译。值得注意的是，在以依存分析为主题的 2006 年、2007年 CoNLL（Conference on Computional NaturalLanguage Learning）中，汉语的依存分析精确度和英语、意大利语等印欧语言同属于高分区。

一些成熟的面向汉语的句法分析工具

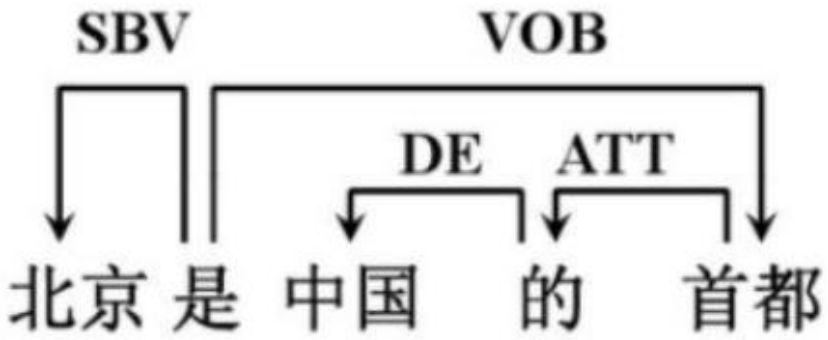
- NLTK（Natural Language Toolkit）是最经典的自然语言处理工具包，在 Python 上可以实现词性标注、依存分析等任务。
- LTP（Language Technology Platform）是哈尔滨工业大学研发的自然语言处理基础技术平台，加载训练后的模型，能够实现分词、词性标注、依存句法分析、语义角色分析等功能。
- ……

依存结构的表示方式

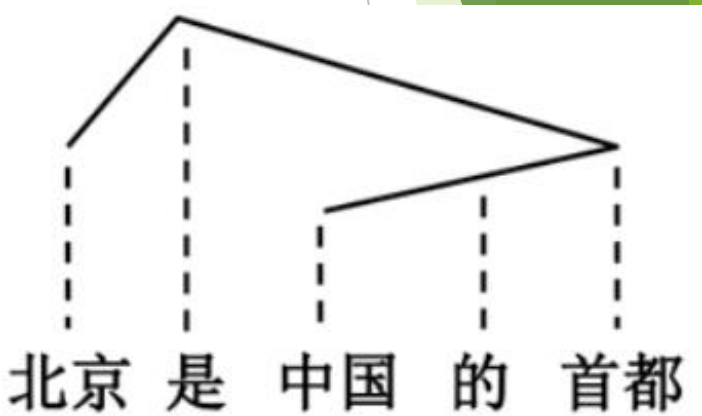
- 词语之间的层次关系
- 依存关系的类型，依存关系词语两端的地位



依存树



有向图



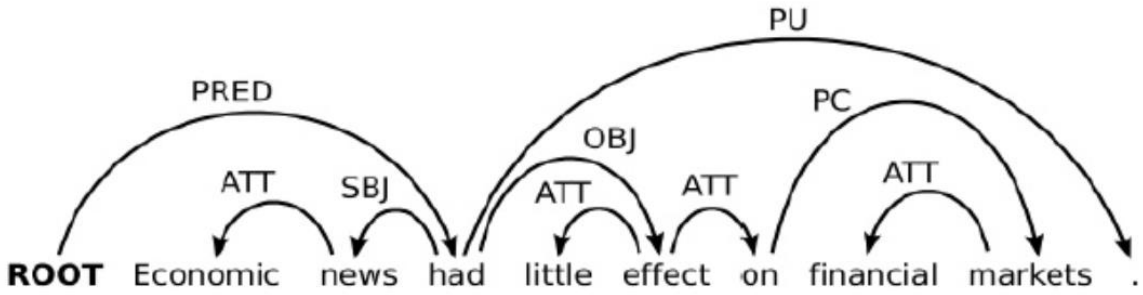
依存投射树

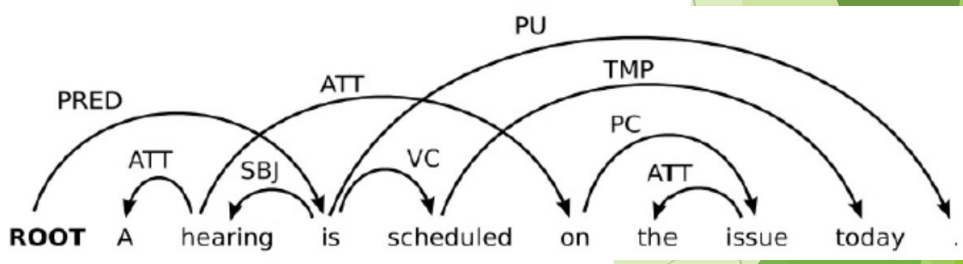
依存结构的优点

- 相比于短语结构，更适合 free order 语言
  - 短语结构难以实现交叉
  - 更符合依存结构自身的特点
- 对于语义更加“透明”
  - “透明”指整词与其组成词素的语义相关程度
  - 如“汉语”、“马上”

Tip: 关于投射性

- 如果词p依存于词q，那么p和q之间的任意词r就不能依存到p和q所构成的跨度之外
- 非投射就没有上述限制了，就会导致依存边有交叉





缺点

- 存在依附歧义

  很难确定如何把一个短语（介词短语、状语短语、分词短语、不定式）依附到其他成分上去。

基于图的模型（Graph-based）

- 最大生成树模型（MST model）

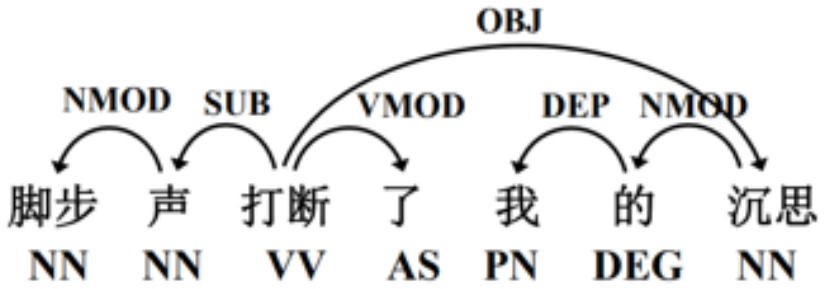
将依存树结构看作是生成树，概率最大的那个即为所求

生成树：包含连通图中的所有顶点，任意两顶点之间有且仅有一条通路

Graph-based

$$s(\boldsymbol{x}, y) = \sum_{(i,j)\in y} s(i,j) = \sum_{(i,j)\in y} \boldsymbol{w} \cdot \boldsymbol{f}(i,j)$$

$\boldsymbol{f}$为高维二元特征函数



$$f(i,j) = \begin{cases} 1 & \text{if } x_i = '\text{打断}' \text{and } x_j = '\text{沉思}' \\ 0 & \text{otherwise} \end{cases}$$

Graph-based 方法优缺点

- 判别式模型，直接为条件概率建模
- 全局寻优
- 将投射与非投射统一在同一框架下

- 不易使用动态特征
- 不能直接预测 relation type

Transition-based

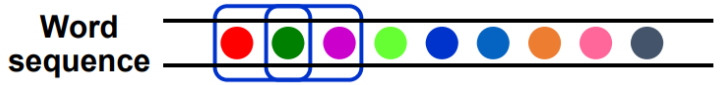
一种确定性的分析方法，将句法结构转化为一系列的动作，动作导致的状态转移最终规约出句法树

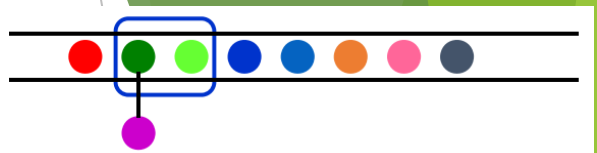
$$t^* = \text{argmax} \sum_{t \in T} \text{score}(c, t)$$

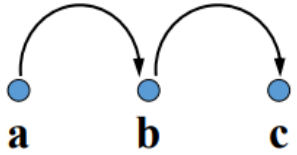
t: transition
c: configuration

- Yamada's parser
- Nivre's parser

Yamada's parser







Pass1: [a b]→Shift→Left→
Pass2: [a b]→Left

初始化: $[\,]_W$

结束: $[\mathrm{ROOT}]_W$

W: 窗口
i: 词的位置
A: 弧的集合

关注window中的两个token

Right

$$\frac{\cdots [w_i, w_{i+1}]_W \cdots}{\cdots [w_{i+1}]_W \cdots \quad A \cup \{w_i \leftarrow w_{i+1}\}}$$

Left

$$\frac{\cdots [w_i, w_{i+1}]_W \cdots}{\cdots [w_i]_W \cdots \quad A \cup \{w_i \rightarrow w_{i+1}\}}$$

Shift

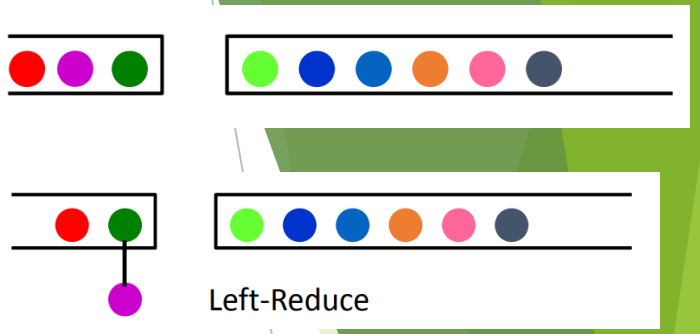
$$\frac{\text{动作执行之前}}{\text{动作执行之后}}$$

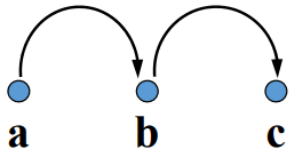
$$\frac{\cdots [w_i, w_{i+1}]_W \cdots}{\cdots [w_{i+1}, w_{i+2}]_W \cdots}$$

Actions in Yamada's parser

- 特征丰富
- 错误传递

Nivre's parser





[a b]→Shift→Right-Reduce→Right-Reduce

初始化: (nil, *I*, Ø)

结束: (*S*, nil, *A*)

关注栈顶的两个token

Left-Reduce$_l$

$$\frac{[\ldots,w_i,\ w_j]_S \quad [\ldots]_I}{[\ldots,w_j]_S \quad [\ldots]_I \quad A \cup \{w_i \xleftarrow{l} w_j\}}$$

Right-Reduce$_l$

$$\frac{[\ldots,w_i,\ w_j]_S \quad [\ldots]_I}{[\ldots,w_i]_S \quad [\ldots]_I \quad A \cup \{w_i \xrightarrow{l} w_j\}}$$

Shift

$$\frac{[\ldots]_S \quad [w_i,\ldots]_I}{[\ldots,w_i]_S \quad [\ldots]_I}$$

Actions in Arc-standard model

- 预测出最优的动作序列就可以得到依存树，通常可以看作序列分类问题
- 是一个确定性模型

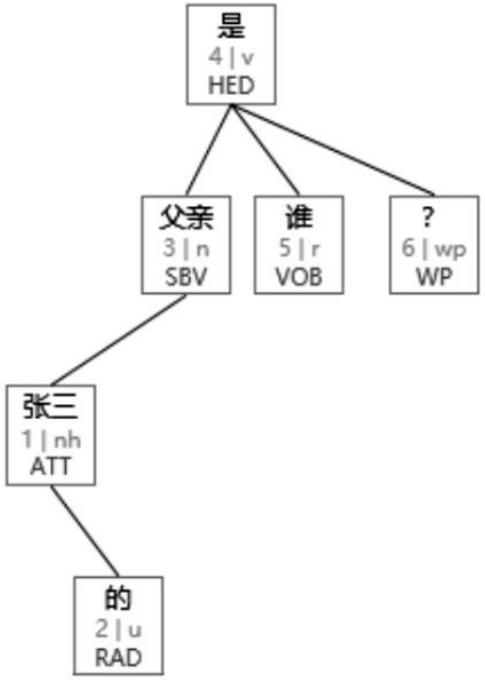
Transition-based 方法优缺点

- 易使用动态特征
- 可以方便预测 relation type

- 贪心，并不是全局寻优
- 存在错误传递问题
  - 可以通过每次保留多条动作路径得到缓解

应用

- 文本理解
  - “张三的父亲”是哪个人
  - “张三”是谁的父亲
  - “谁”和“的”“父亲”并不存在依存关系
- 事件抽取
- 情感分析
- 机器翻译
- 树库搭建
- ……

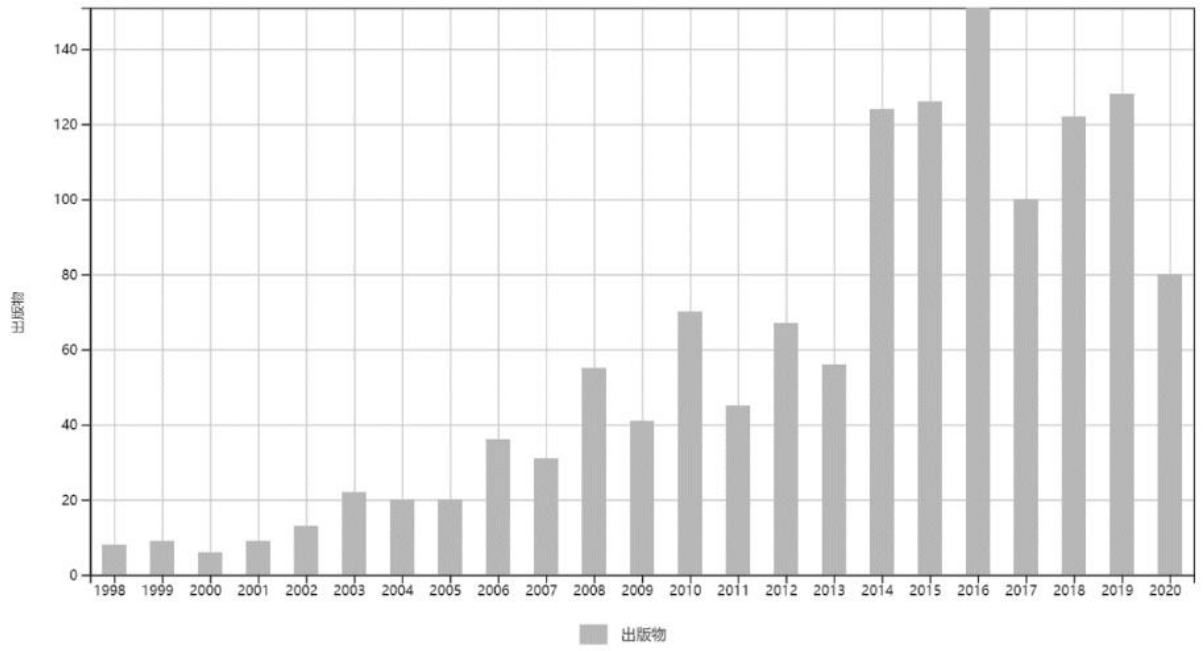


研究现状及趋势

杨牧,蔡言胜. 依存句法分析的回顾与发展[J]. 现代语文,2022(1):89-95.

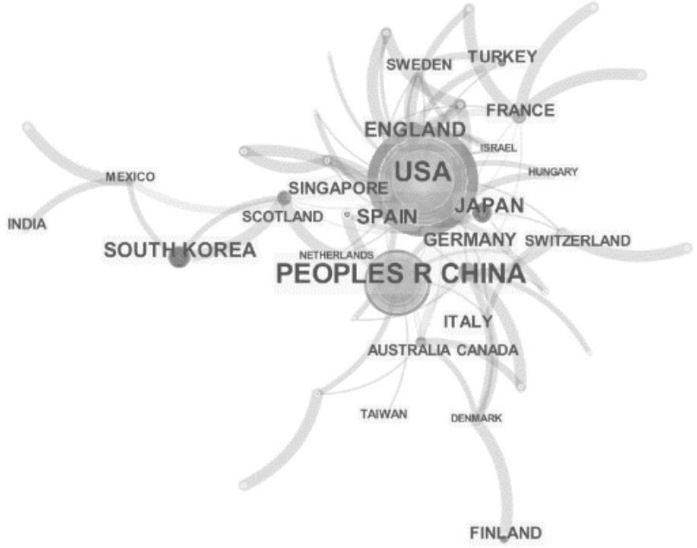
用 CiteSpace 软件，对 Web of Science 核心数据库所收录的文献进行检索，主题为“dependency parsing”，时间跨度为 1985 年 1 月 1 日到 2020 年 12月 31 日，共获得1339篇文献。

- 年度发文量
  - 1985—1997 年：依存分析尚处于起步阶段；
  - 1998—2002 年间，每年有 10 篇左右的相关文献被收录，这说明依存分析已引起学界注意，但研究成果相对匮乏，发展较为缓慢；
  - 2003—2013 年间，依存分析研究进入新阶段，每年发文量均在 20 篇以上；
  - 2014年开始，发文量迅速增多，表明依存分析已成为研究热点……



续

- 关键词共现知识图谱分析
- 收录文献的国家来源分布



基于NN的依存句法分析

基于NN的Graph-based方法

- 思路：

  找到概率最大（即评分最高的）生成树

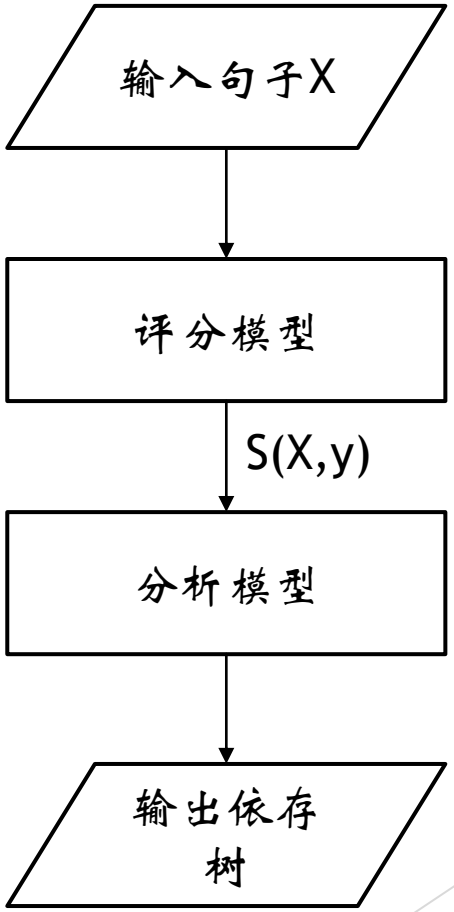
- 问题表述：

  对于由n个词$x_i, i = 1,2,\ldots,n$组成的句子

  $$X = w_{ROOT} w_1 w_2 \ldots w_n$$

  用$\tau(X)$表示生成树集，则问题表示为

  $$\hat{y} = \arg\max \mathrm{Score}(X, y), y \in \tau(X)$$



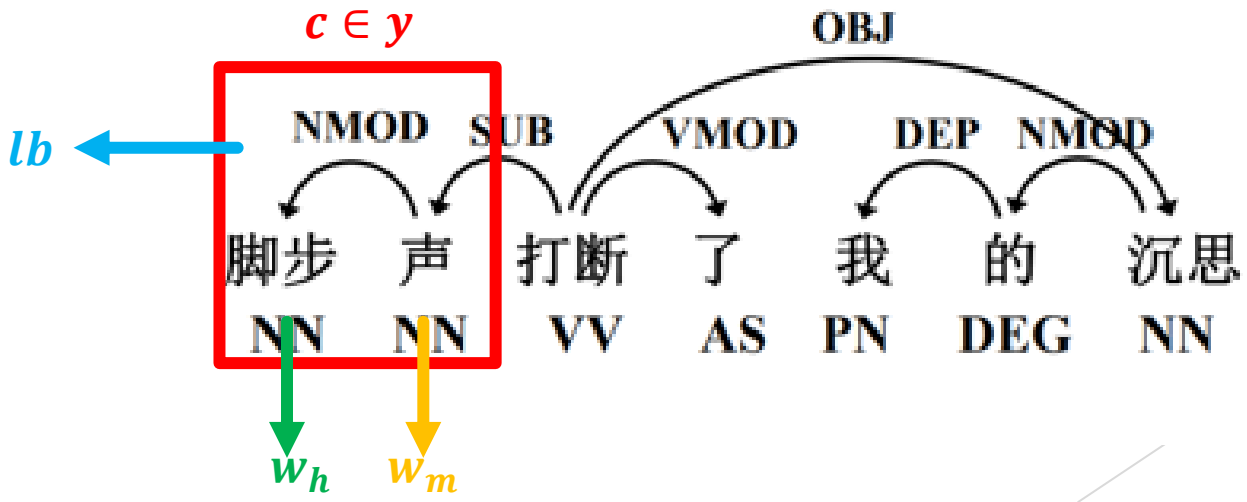
基于NN的Graph-based方法

- 评分模型：由研究人员自行选择

  常用的预处理：子图分解(受限于分析模型的效率)

  一阶子图分解：$Score(X,y)=\sum_{c\in y} Score(X,c)$

  $=\sum_{(w_h,w_m,lb)\in y} Score(\ w_h,w_m,lb)$



基于NN的Graph-based方法

- 特征提取：
  - 传统的特征工程方法（受特征选取影响较大）
  - 当前最常用的方法：基于LSTM的特征学习

对于句子 $X = w_{ROOT} w_1 w_2 \dots w_n$，设词类序列为 $T = t_1 t_2 \dots t_n$，

将 (w_i, t_i) 编码为Bi-LSTM模型的输入向量：$x_i = (w_i, t_i)$

Bi-LSTM模型的输出 $v_i = BiLSTM(x_i) = \varphi(h_i^F, h_i^B)$ 结合了词在句子中的语境信息，将中心词和依存词所对应的Bi-LSTM编码向量作为评分模型的输入

基于NN的Transition-based方法

- 思路：
  - 贪心解码：

    学习一个分类器，其输入为一个状态，输出为该状态下最可能的动作
  - 全局解码：

    综合考虑多个状态之间的依存关系
- 方法：

DNN → RNN

基于NN的Transition-based方法

- DNN：

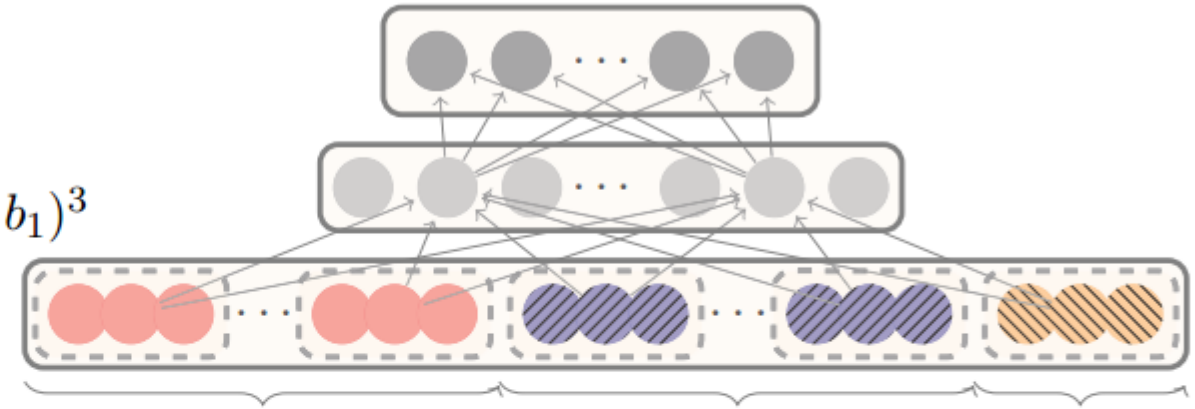
Softmax layer:

$$p = \texttt{softmax}(W_2 h)$$

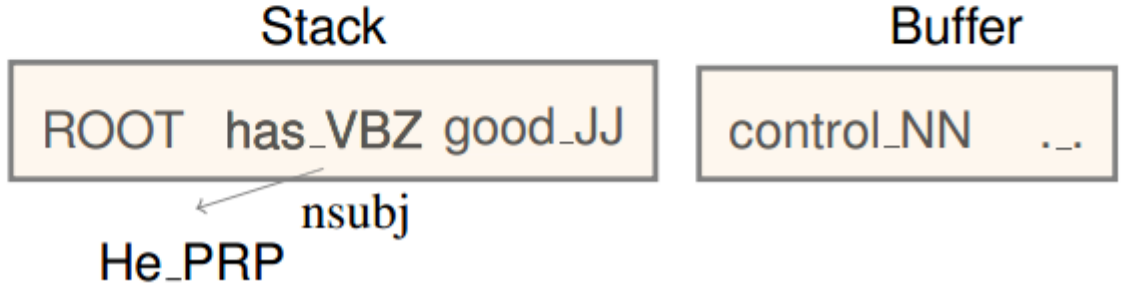
Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$

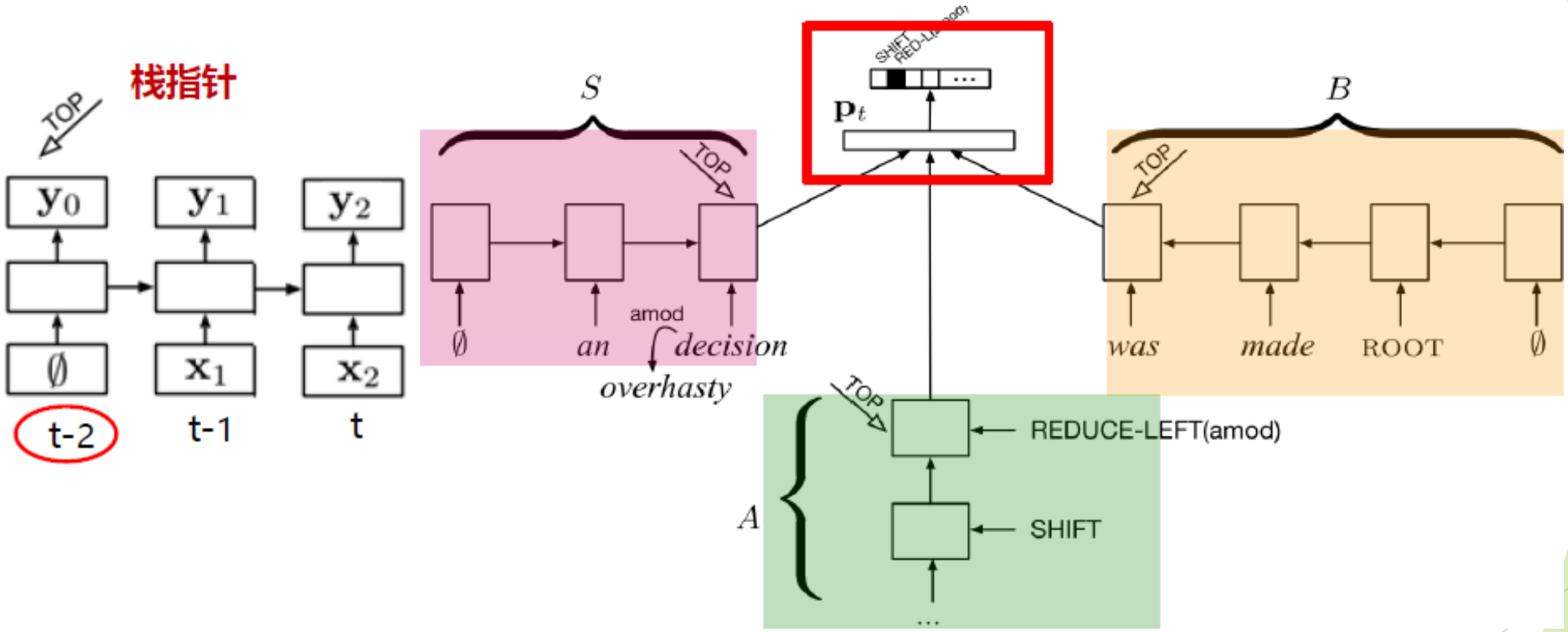


Configuration



基于NN的Transition-based方法

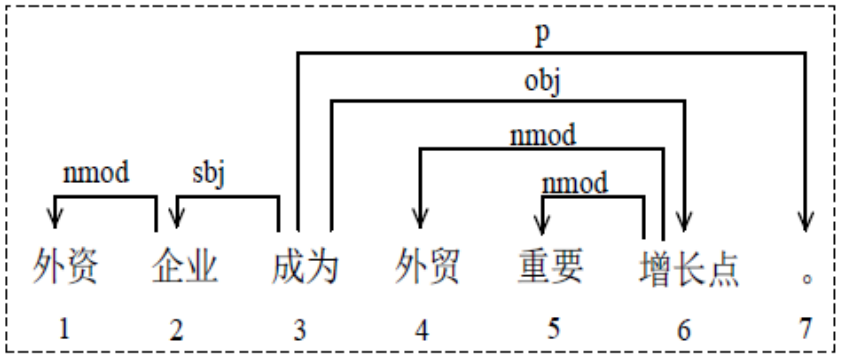
- RNN：



依存句法分析评价方法

- 无标记依存正确率：找到其正确支配词的词所占的百分比
- 带标记依存正确率：找到其正确支配词且依存关系类型也标注正确的词所占的百分比

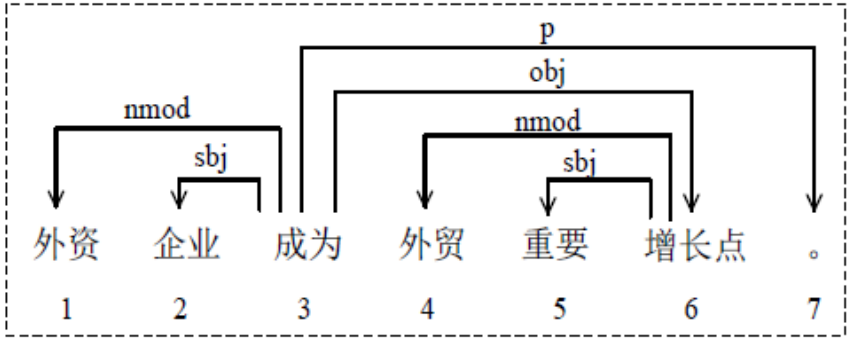
标准依存树



每个词对应的支配词及依存关系为：

外资(2,nmod)，企业(3,sbj)，成为(0,root),外贸(6,nmod)，重要(6,nmod)，增长点(3,obj)，。(3,p)

系统输出分析树



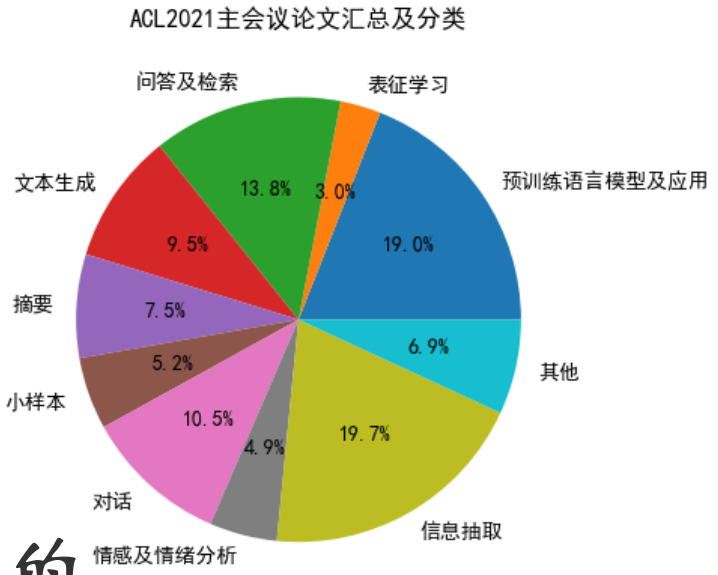
每个词对应的支配词及依存关系为：

外资(3,nmod)，企业(3,sbj)，成为(0,root)，外贸(6,nmod)，重要(6,sbj)，增长(3,obj)，。(3,p)

$$\text{无标记依存正确率（UA）} = \frac{6}{7} \times 100\% = 85.7\%$$

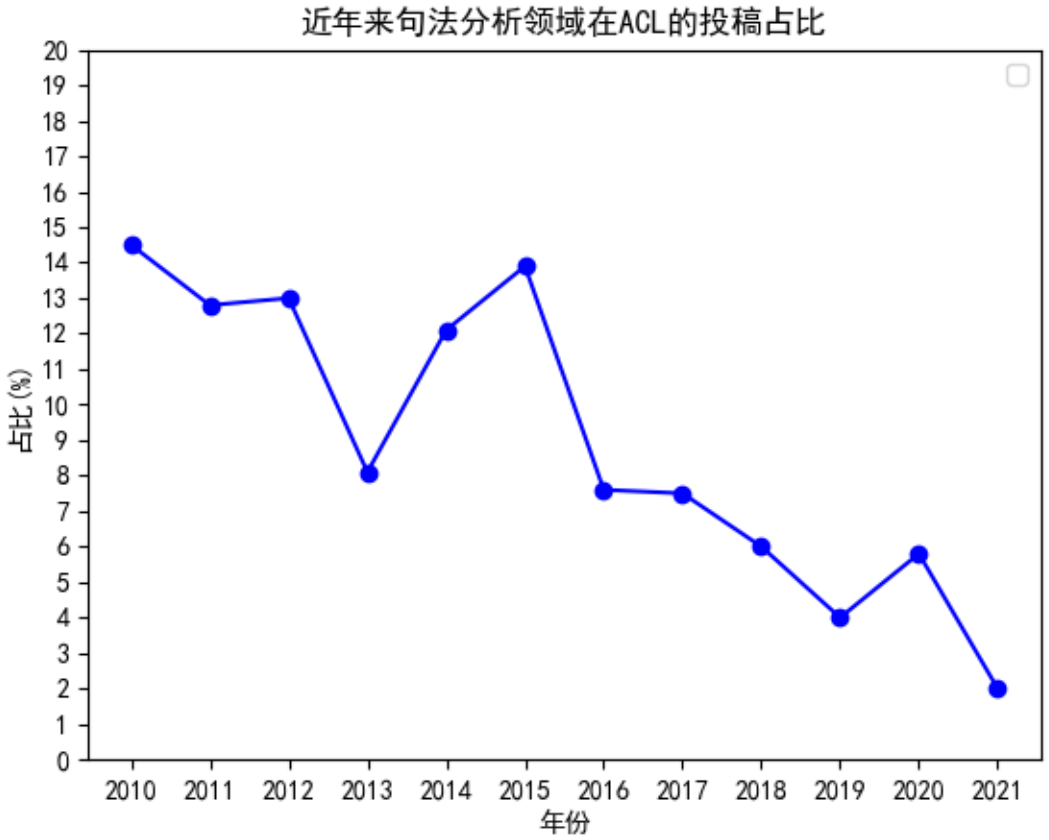
$$\text{带标记依存正确率（LA）} = \frac{5}{7} \times 100\% = 71.4\%$$

研究现状



- ACL大会：
- 自然语言处理领域影响力最大、最具活力的国际学术组织之一。
- 在机器翻译、搜索、信息流、输入法等领域有着广泛的应用。
- ACL2021 - The Asian Conference on Language (ACL) (iafor.org)

研究现状



- 趋势：句法分析逐渐式微
- 原因分析：
- 难度较高
- 提升空间不大
- 不是任何一个下游NLP任务的必需品
- 预训练模型的出现

Demo 展示

一些句法分析包

- 斯坦福： https://nlp.stanford.edu/software/
- MSTParser：https://sourceforge.net/projects/mstparser
- MaltPARSER： http://maltparser.org/index.html
- 哈工大LTP： http://ltp.ai/index.html

NTLK

- 基于java
- 高度优化的PCFG句法分析
- 使用 A* 算法。
- 除了提供英语句法分析外，还可以适配与其他语言。

NLTK+Stanford NLP

- NLTK：Python 自然语言处理工具包。
- Stanford NLP：由斯坦福大学的 NLP 小组开源的 Java 实现的 NLP 工具包。
- 在 2004 年 Steve Bird 在 NLTK 中加上了对 Stanford NLP 工具包的支持，通过调用外部的 jar 文件来使用 Stanford NLP 工具包的功能。

短语结构句法分析

```
import os
from nltk.parse import stanford
os.environ['STANFORD_PARSER'] = "F:/anaconda/envs/nlp/jar/stanford-parser.jar"
os.environ['STANFORD_MODELS'] = "F:/anaconda/envs/nlp/jar/stanford-parser-3.9.2-models.jar"
parser=stanford.StanfordParser(model_path="F:/anaconda/envs/nlp/jar/chinesePCFG.ser.gz")


res=list(parser.parse(u'依存句法 分析 通过 分析 单词 之间 的 依存 关系 揭示 其 句法 结构 。'.split()))
for row in res[0]:
    print(row)

sent = parser.raw_parse("依存句法 分析 通过 分析 单词 之间 的 依存 关系 揭示 其 句法 结构 。")
for line in sent:
    line.draw()
```

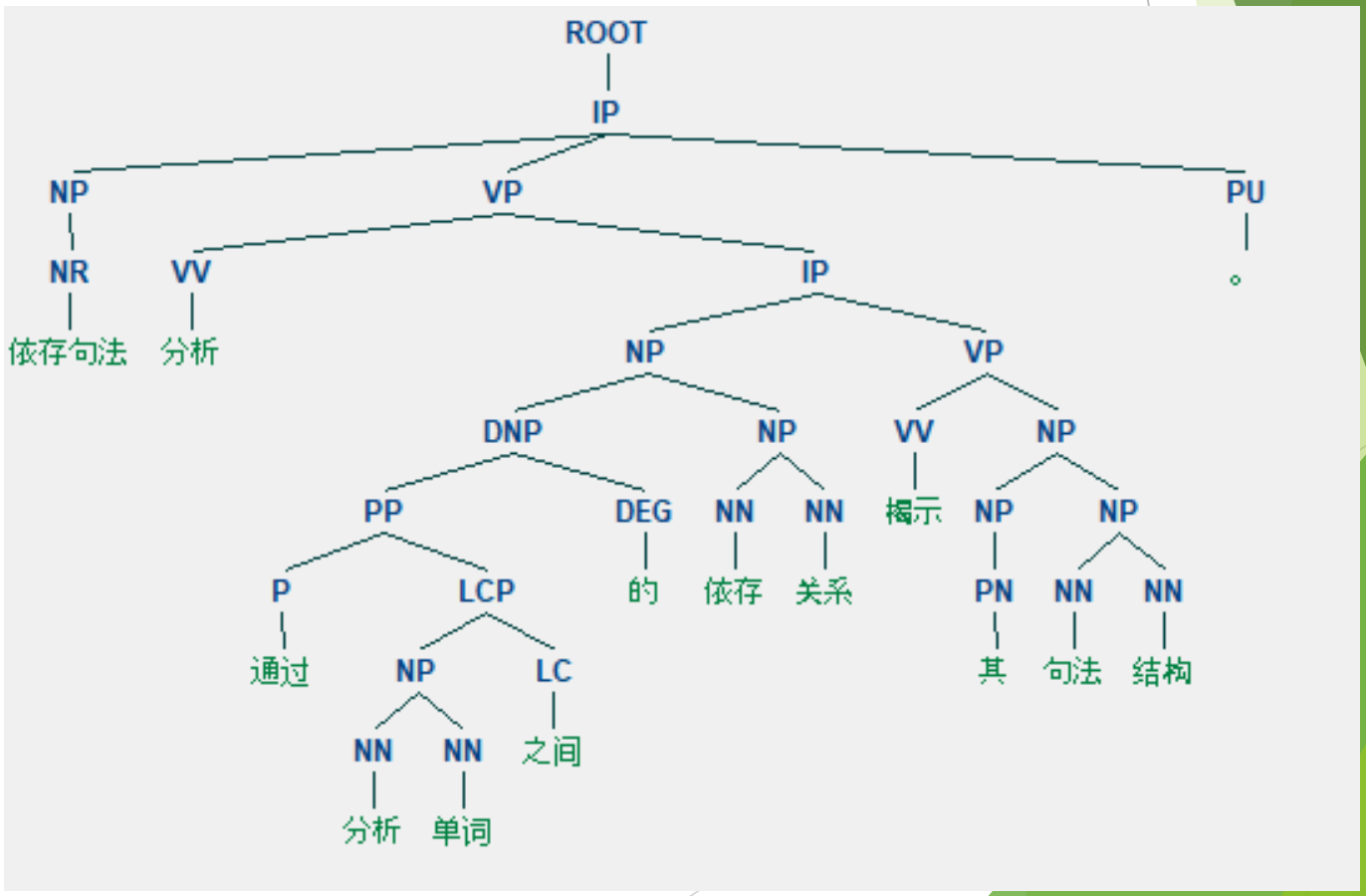
例句1

- 输入：
- “依存句法分析通过分析单词之间的依存关系揭示其句法结构。”
- 输出：

```
(IP
  (NP (NR 依存句法))
  (VP
    (VV 分析)
    (IP
      (NP
        (DNP (PP (P 通过) (LCP (NP (NN 分析) (NN 单词)) (LC 之间))) (DEG 的))
        (NP (NN 依存) (NN 关系)))
      (VP (VV 揭示) (NP (NP (PN 其)) (NP (NN 句法) (NN 结构))))))
  (PU 。))
```

例句1可视化

- “依存句法分析通过分析单词之间的依存关系揭示其句法结构。”



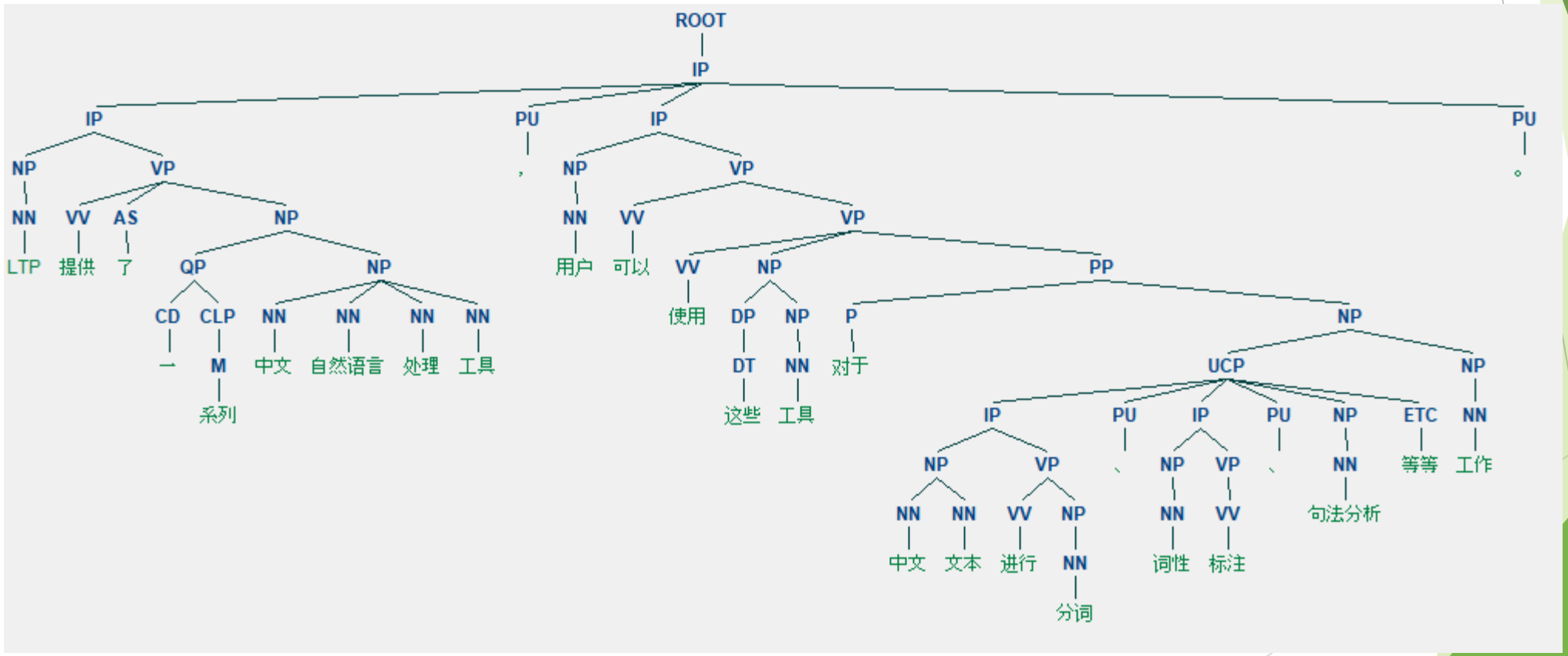
例句2

- 输入：
- ”LTP提供了一系列中文自然语言处理工具，用户可以使用这些工具对于中文文本进行分词、词性标注、句法分析等等工作。”
- 输出：

```
(IP
  (IP
    (NP (NN LTP))
    (VP
      (VV 提供)
      (AS 了)
      (NP
        (QP (CD 一) (CLP (M 系列)))
        (NP (NN 中文) (NN 自然语言) (NN 处理) (NN 工具)))))
  (PU ，)
  (IP
    (NP (NN 用户))
    (VP
      (VV 可以)
      (VP
        (VV 使用)
        (NP (DP (DT 这些)) (NP (NN 工具)))
        (PP
          (P 对于)
          (NP
            (UCP
              (IP (NP (NN 中文) (NN 文本)) (VP (VV 进行) (NP (NN 分词))))
              (PU 、)
              (IP (NP (NN 词性)) (VP (VV 标注)))
              (PU 、)
              (NP (NN 句法分析))
              (ETC 等等))
            (NP (NN 工作)))))))
  (PU 。))
```

例句1可视化

- ”LTP提供了一系列中文自然语言处理工具，用户可以使用这些工具对于中文文本进行分词、词性标注、句法分析等等工作。”



语言技术平台（ Language Technology Plantform | LTP ）

- LTP提供了一系列中文自然语言处理工具，用户可以使用这些工具对于中文文本进行分词、词性标注、句法分析等等工作。

谁在使用LTP?


Tencent 腾讯


Baidu 百度


HUAWEI


中国移动
China Mobile


科大讯飞
iFLYTEK


搜狗搜索


SONY
make.believe


SIEMENS


GRIDSUM 国双


万方数据 WANFANG DATA 知识服务平台


同花顺
知识 · 工具 · 信息


ANT'inno


UBIC


WISERS 慧科
Advantage Through Intelligence

特点

- 依存句法分析模型加入微博数据，使得在开放域上的句法分析性能更好
- 依存句法分析算法使用transition-based neural network parser，速度较快。同时加入了聚类特征，进一步优化了训练算法。

依存句法分析

```python
def analysis(str,name):
    seg, hidden = ltp.seg([str])
    dep = ltp.dep(hidden)
    print(dep)
    # [['他', '叫', '汤姆', '去', '拿', '外衣', '。']]
    # [
    #     [
    #         (1, 2, 'SBV'),
    #         (2, 0, 'HED'),     # 叫 --|HED|--> ROOT
    #         (3, 2, 'DBL'),
    #         (4, 2, 'VOB'),
    #         (5, 4, 'COO'),
    #         (6, 5, 'VOB'),
    #         (7, 2, 'WP')
    #     ]
    # ]
    #画图
    draw(seg,dep,name)
```

使用LTP库进行依存句法分析，
生成包含多个关系元组的列表

```python
def draw(seg,sdp,name):
    dot = Digraph('name', 'comment')
    dot.node(fontname="FangSong",name='Root')
    sdp=sdp[0]
    seg=seg[0]
    i=1
    for word in seg:
        dot.node(fontname="FangSong",name='('+str(i)+') '+word)
        i+=1
    for word in sdp:
        node1_num=word[0]
        node2_num=word[1]
        if node2_num==0:
            node2='Root'
        else:
            node2='('+str(node2_num)+') '+seg[node2_num-1]
        node1='('+str(node1_num)+') '+seg[node1_num-1]
        edge_name=word[2]
        dot.edge(node1,node2,edge_name)
    print(dot.source)
    dot.render(filename=name+'.gv', directory=None, view=True, cleanup=False)
    dot.view()
```

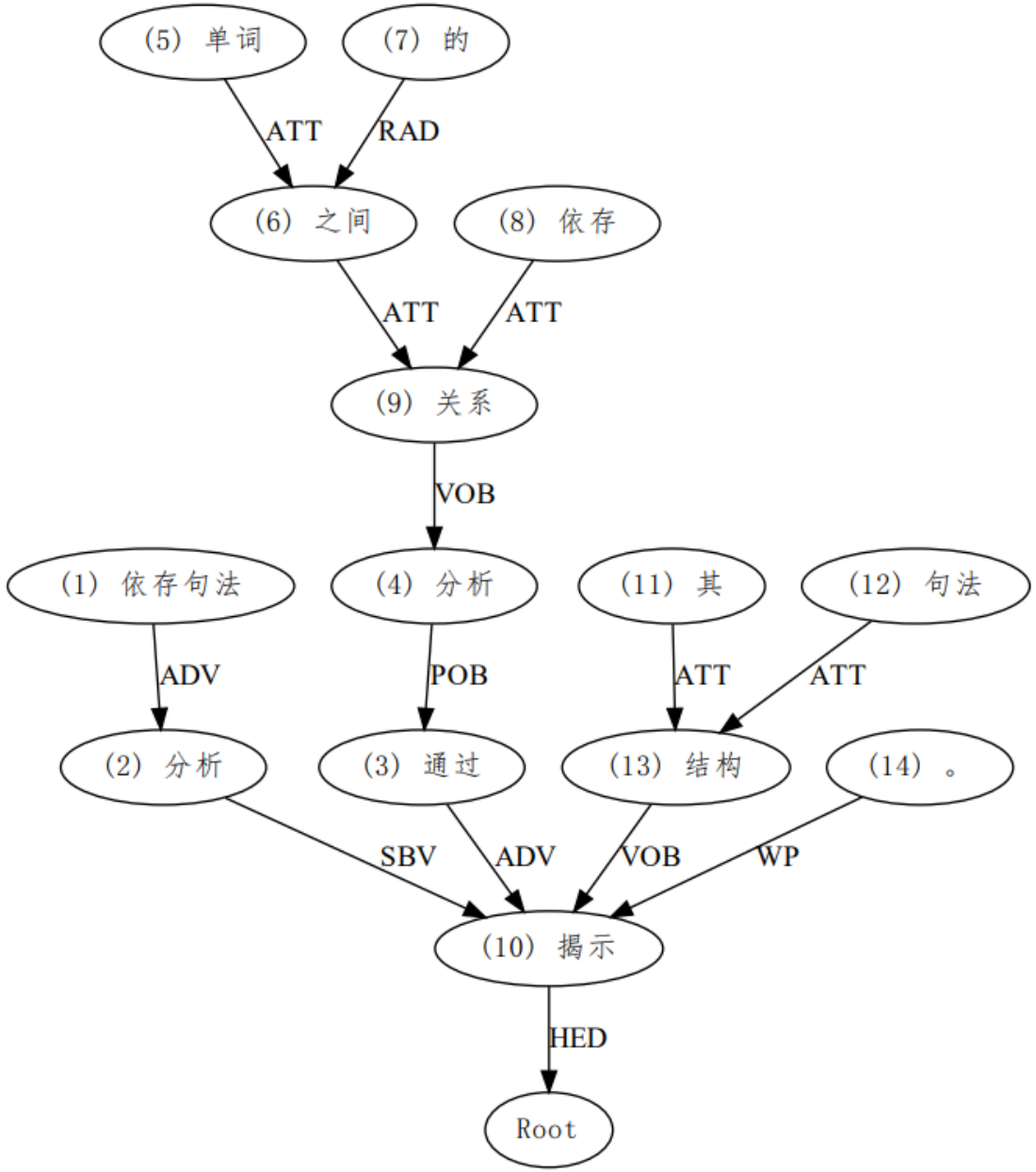
使用graphviz库进行依存句法树的可视化

例句1

- 输入：
- “依存句法分析通过分析单词之间的依存关系揭示其句法结构。”
- 输出：
- [(1, 2, 'ADV'), (2, 10, 'SBV'), (3, 10, 'ADV'), (4, 3, 'POB'), (5, 6, 'ATT'), (6, 9, 'ATT'), (7, 6, 'RAD'), (8, 9, 'ATT'), (9, 4, 'VOB'), (10, 0, 'HED'), (11, 13, 'ATT'), (12, 13, 'ATT'), (13, 10, 'VOB'), (14, 10, 'WP')]

例句1可视化

“依存句法分析通过分析单词之间的依存关系揭示其句法结构。”

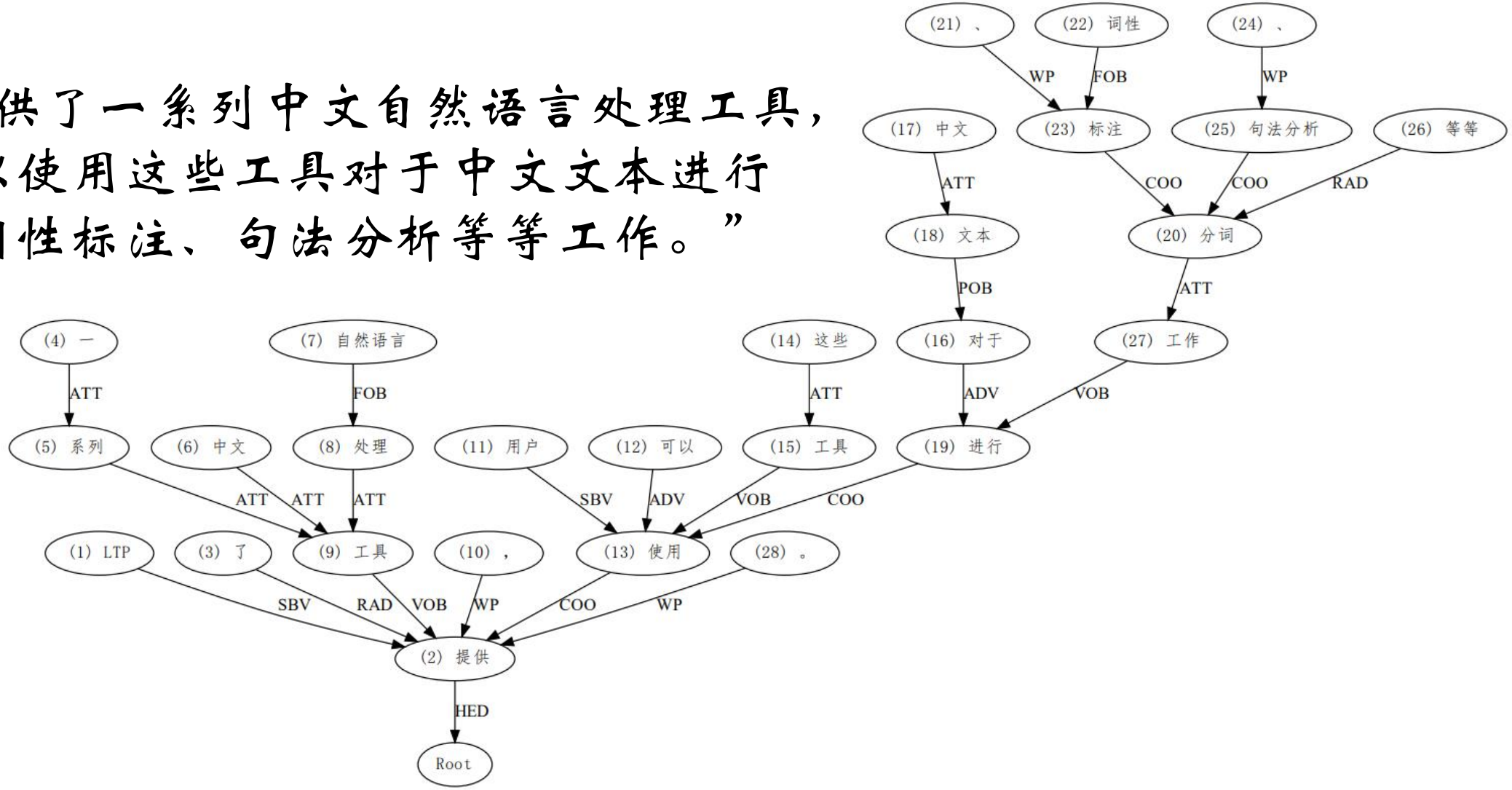


例句2

- 输入：
- ”LTP提供了一系列中文自然语言处理工具，用户可以使用这些工具对于中文文本进行分词、词性标注、句法分析等等工作。”
- 输出：
- [(1, 2, 'SBV'), (2, 0, 'HED'), (3, 2, 'RAD'), (4, 5, 'ATT'), (5, 9, 'ATT'), (6, 9, 'ATT'), (7, 8, 'FOB'), (8, 9, 'ATT'), (9, 2, 'VOB'), (10, 2, 'WP'), (11, 13, 'SBV'), (12, 13, 'ADV'), (13, 2, 'COO'), (14, 15, 'ATT'), (15, 13, 'VOB'), (16, 19, 'ADV'), (17, 18, 'ATT'), (18, 16, 'POB'), (19, 13, 'COO'), (20, 27, 'ATT'), (21, 23, 'WP'), (22, 23, 'FOB'), (23, 20, 'COO'), (24, 25, 'WP'), (25, 20, 'COO'), (26, 20, 'RAD'), (27, 19, 'VOB'), (28, 2, 'WP')]

例句2可视化

“LTP提供了一系列中文自然语言处理工具，用户可以使用这些工具对于中文文本进行分词、词性标注、句法分析等等工作。”



谢谢大家