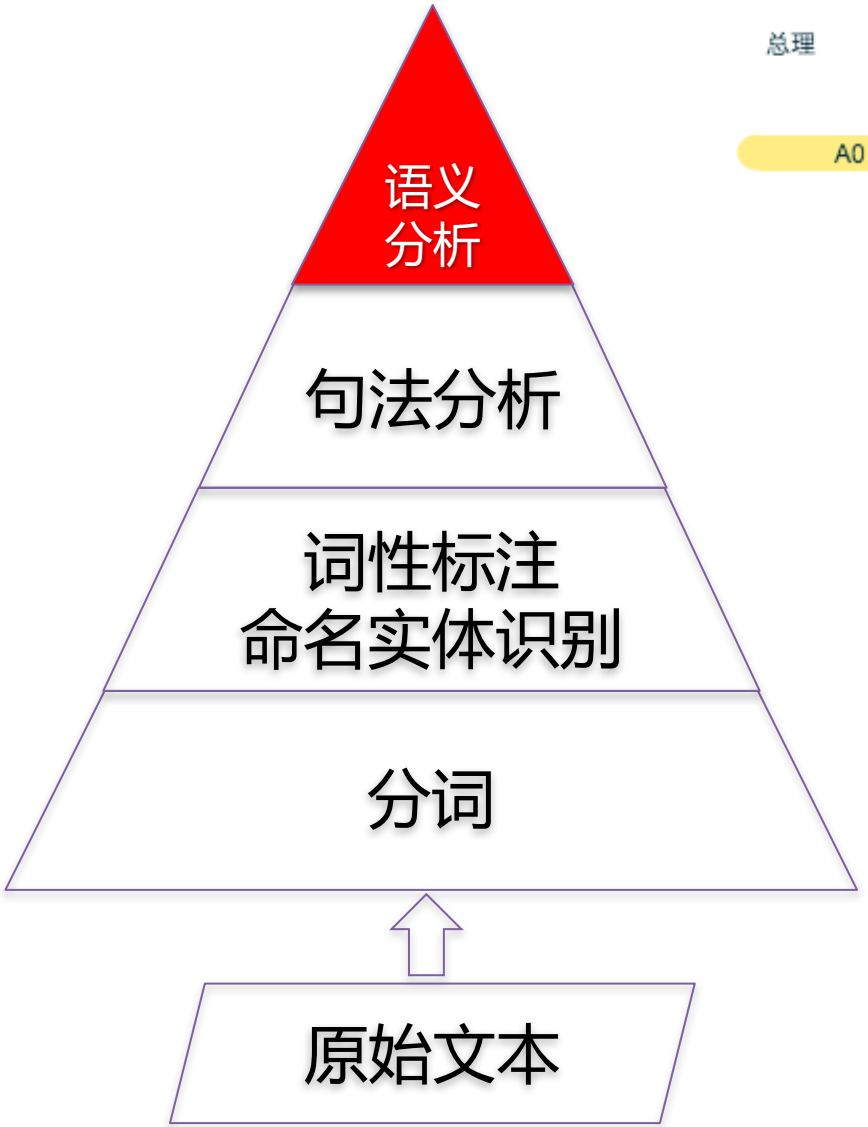
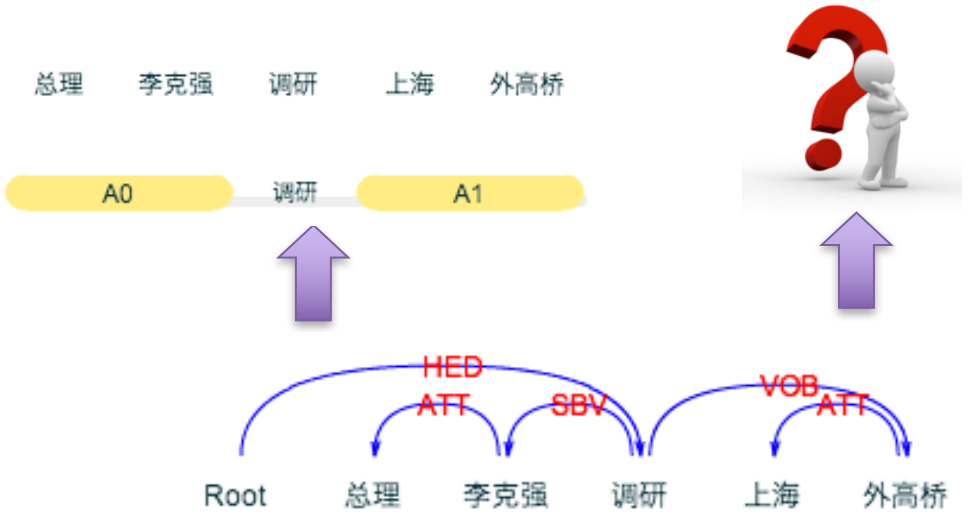
语义依存分析

车万翔
社会计算与信息检索研究中心
2017年春季学期

语言分析任务



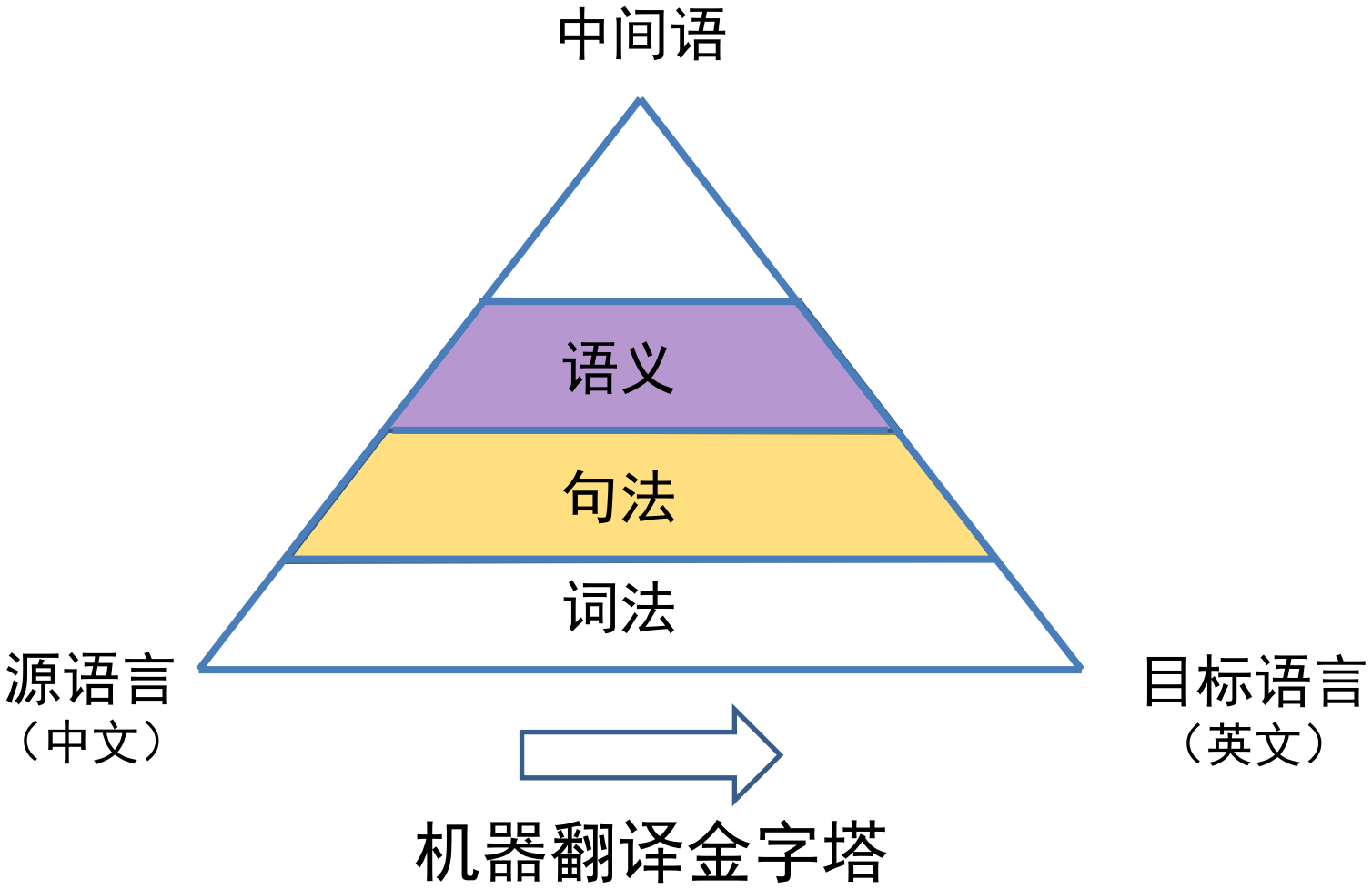


总理/n 李克强/nh 调研/v 上海/ns 外高桥/ns

总理 李克强 调研 上海 外高桥

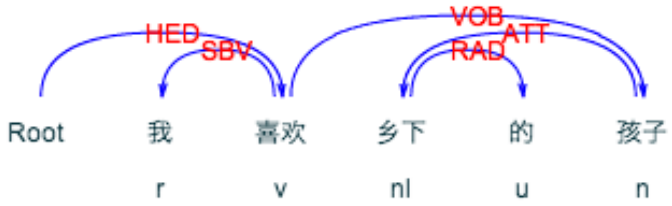
总理李克强调研上海外高桥

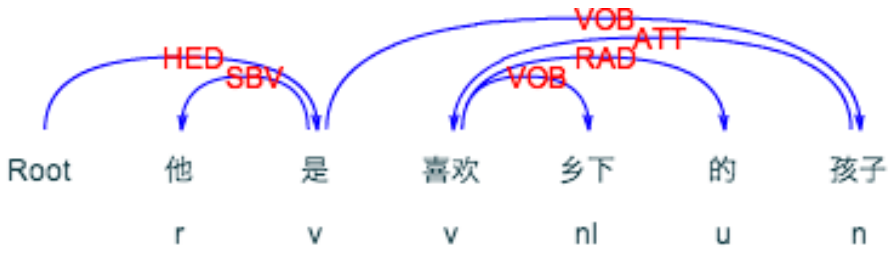
分析得越深，转换的代价越小



句法分析

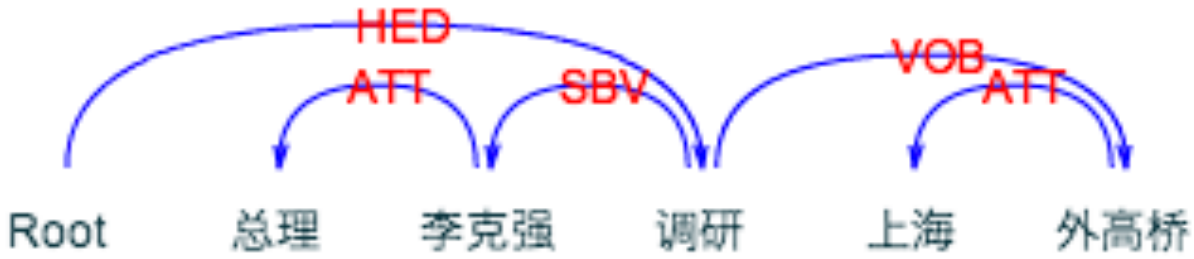
- 将输入文本从序列形式转化为树状结构，刻画句子内部的句法关系，是自然语言处理的核心问题
- 常见的句法关系
  - 主、谓、宾、定、状、补、...
- 句法歧义
  - 如：“喜欢乡下的孩子”





依存句法分析

- 形式简洁，易于理解和应用，是句法分析最主要的形式
- 词对之间由不对称的依存关系连接，形成树状语法结构
  - 箭头从核心词指向修饰词，弧上标记为依存关系类型
  - 约束条件
    - 唯一根节点
    - 单一父节点
    - 不对称
    - 无交叉



句法和语义

- 结构关系和构成成分不同

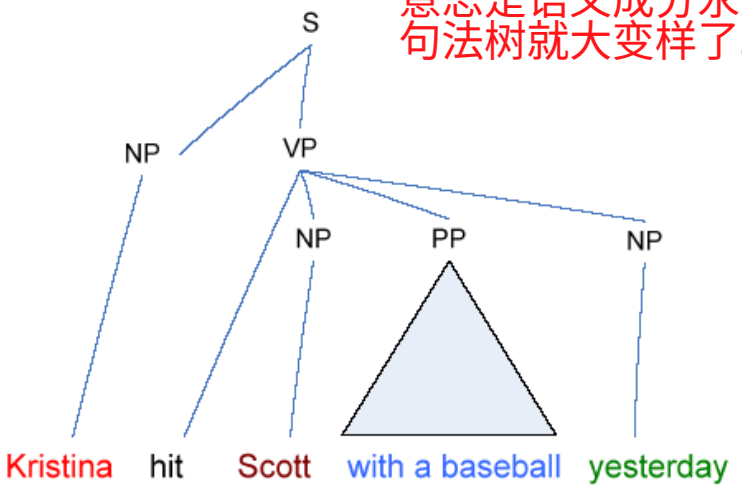
句子： **昨天** **我** 在**教室**里 给**老师** **写**了 **一封** **信**

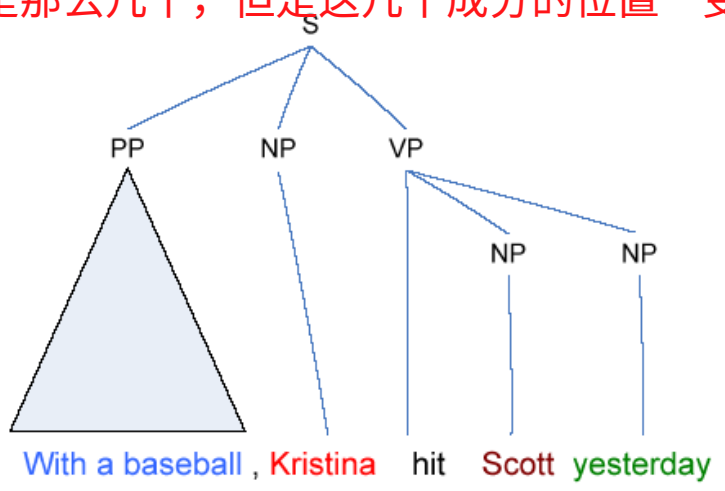
句法成分： **状语** **主语** **状语** **状语** **谓语** **定语** **宾语**

语义成分： **时间** **施事** **处所** **与事** **动作** **数量** **受事**

- 语义结构稳定，而句法结构随字面变化而变化

意思是语义成分永远是那么几个，但是这几个成分的位置一变，句法树就大变样了。





句法结构和语义关系不对应

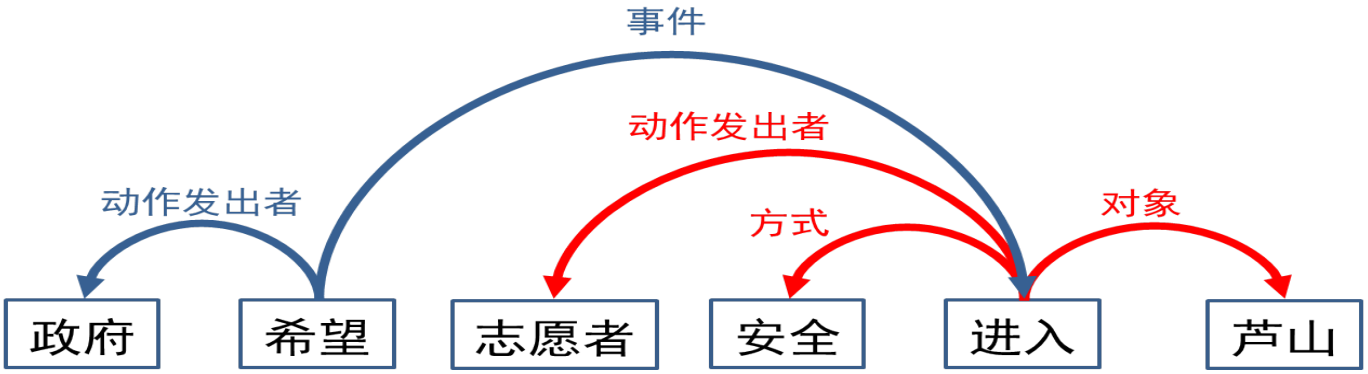
- 句法结构相同，语义关系不同
  - 吃食堂 vs 买衣服
  - 我[施事]去过了 vs 北京[地点]去过了。
- 句法结构不同，语义结构相同
  - 打败敌人 vs 被敌人打败
  - 汤姆吃了苹果 vs 汤姆把苹果吃了

语义角色标注

- 论元（Argument）：和动词搭配的名词性成分
- 语义角色标注(Semantic Role Labeling，SRL)
  - 给定一个句子，分析该句中特定动词的论元结构，找出和特定动词搭配的论元并分析其语义角色
    - 例如：[施事张平]给了[与事李华][受事一本书]。
  - 核心角色：施事，受事，与事……
  - 附属角色：地点，时间，行为方式，原因……
- 语义角色标注是浅层语义分析的一种主要实现方式，是深层语义分析的简化和基础

语义依存分析

- 中文句子语义依存树
  - 融合了句子的依存结构和语义关系



- 语义依存关系不因句子表层形式而变化

张三吃了苹果

张三把苹果吃了

苹果被张三吃了

吃(张三，苹果)

语义依存分析与语义角色标注的区别

	语义角色标注	语义依存分析
理论	谓词理论	依存理论
结构	谓词-角色	依存结构
焦点	主要谓词	句子中每个词
对象	主要谓词及其论元	所有核心词及其修饰词
标签	含义随不同谓词而变化 （Arg0-5，ArgM-）	直接刻画具体含义
标签数量	少	多
标注结果	主要谓词的语义角色树	整个句子的完整依存图
关系转化	难	易
包含关系	--	包含语义角色标注

相关语义体系与语料库

- 语义体系
  - 董振东的“知网”、鲁川的“意合网络”
  - 冯志伟的“论元结构”、袁毓林的语义体系、林杏光的“汉语基本格”
- 语料库

研究者	语义理论	标注集	语料库	语言
[袁, 2008]	论元结构	23种论元	新闻语料	简体中文
[Xue and Palmer, 2003]	语义角色	语义角色(argN, N:1-5)和18种secondary tags	Penn Chinese Treebank	简体中文
[You and Chen, 2004]	语义角色	74 种不同的语义角色	Sinica Treebank	繁体中文
[Li et al., 2003]	语义依存结构	59 种语义关系, 9 种句法关系, 2种特殊关系部分	Tsinghua语料	简体中文
[Yan, 2007]	语义依存结构	70种语义关系，20种句法关系，2种特殊标记	Penn Chinese Treebank	简体中文

汉语与英语的基本特点：存在显著差异

- 英语重形合，汉语重意合，语法灵活，仅依靠句法信息而不充分利用语义关联，难以有效判断结构关系
- 英语句子理解的主流方法是“先句法后语义”，汉语呢？

	中文	英文	例子
词语间隔	无	有	中国的经济发展 The development of China’s economy
形态变化	无	有	总理李克强调研（动词还是名词?） 很深入 Premier Keqiang Li made a very in-depth investigation 机器翻译，翻译人员，翻译小说 Machine Translation, Translator, Translated Novel
时态变化	无	有	我回家了. I will go home. / I went home.
句法形式	灵活	规范	你买票了么？/ 票你买了么？（Have you bought the ticket?）美国反恐为何越反越恐？
成分省略	多	少	（救治）很及时，（伤者）情况比较好
复杂名词短语	多	少	中国北京红十字芦山抢险救援队“五一”节期间工作掠影

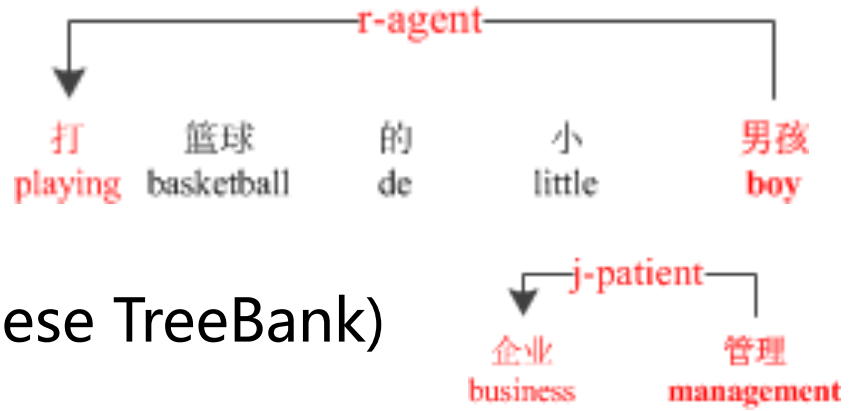
句法依存与语义依存的异同

- 相似之处
  - 都是以依存语法为理论基础
  - 为句子中的每个词标注依存关系
- 不同之处

句法依存	语义依存
① 在依存弧中隐含谓词-论元结构 ② 关系标签少（十个左右） ③ 从汉语语法角度描述主谓宾等结构 ④ 依存树结构足够描述句法关系	① 明确标出依存弧中谓词-论元之间的语义关系 ② 关系标签多（几十或上百） ③ 以谓词论元指派为理论基础，以词对为单位刻画依存词对间的语义 ④ 依存图才能完整刻画语义

BH-SDP对语义关系的定义

- 北语-哈工大语义依存分析体系
  - BLCU-HIT Semantic Dependency Parsing (BH-SDP)
- 语义关系更精细
  - 123 种语义关系（借鉴HowNet 84 种关系） vs. 20 种句法关系
- 有语义特色的依存结构
  - 反关系：动词作为名词的修饰语
  - 间接关系：核心词是动词化的名词
- 10,068 句子（来源：CTB，Chinese TreeBank)

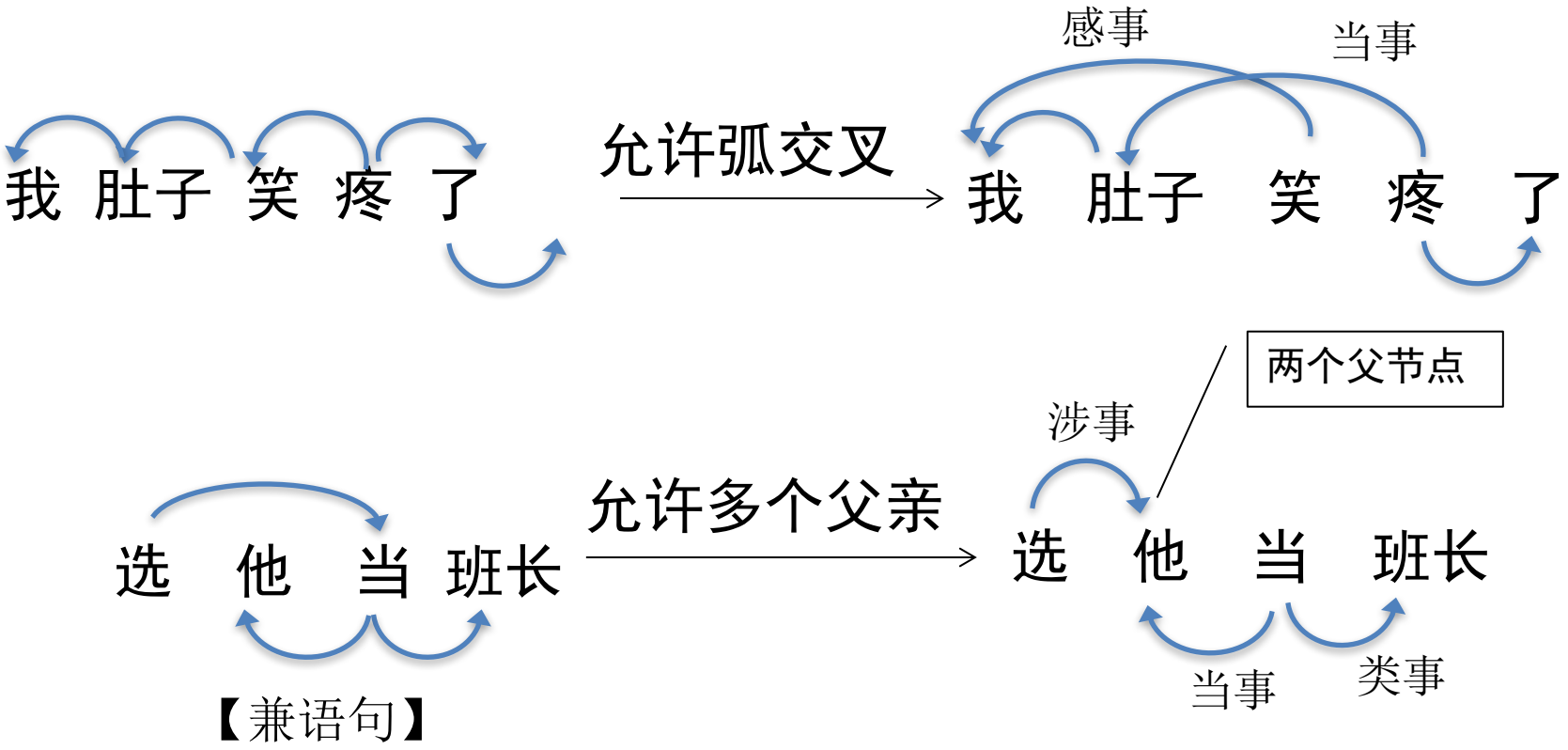


BH-SDP-v1(2011)语料库的不足

- 有些语义关系彼此易混淆
- 语义关系数量太大，有些关系在标注语料中出现次数很少
- 句子全部来自新闻，涵盖的语言现象有限
- 依存树结构，刻画语义不全面

语义依存（图）分析

- 形式上类似于依存语法，但必要时突破树形结构
- 弧上标注的是“语义”关系



新版语义依存语料库

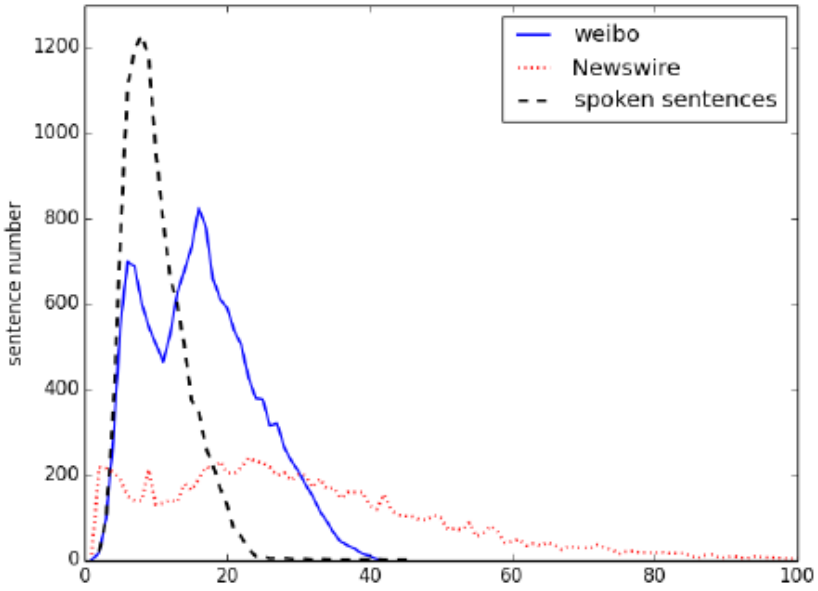
- 在BH-SDP-v1基础上，提出了一个新的语义依存体系BH-SDP-v2，压缩了语义关系类型的数量
- 从“语义依存树”到“语义依存图”，语

	BH-SDP-v1	BH-SDP-v2
关系总数	123	98
主要区别	v2重新组织了v1的语义关系，将关系分为主要语义角色，事件关系，关系标记，减少不必要的类间关系混淆	

依存图库构建

- 语料描述
  - 口语语料，新闻语料，微博语料
  - 新闻语料来自Penn Chinese Treebank 6.01

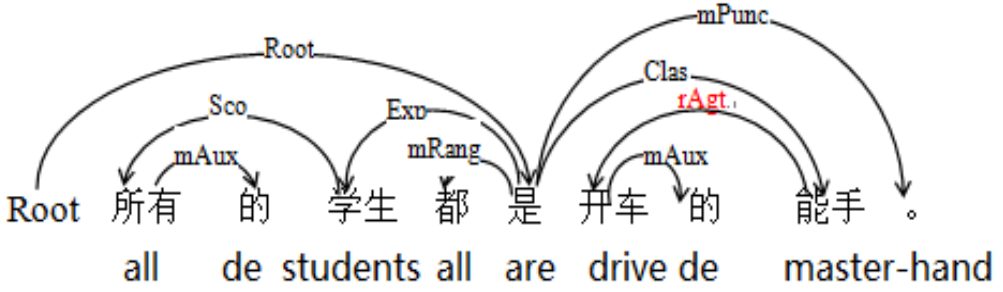
	句子数	词数	句子平均长度
口语	10038	101140	10.08
新闻	10068	308383	30.63
微博	10755	172080	16



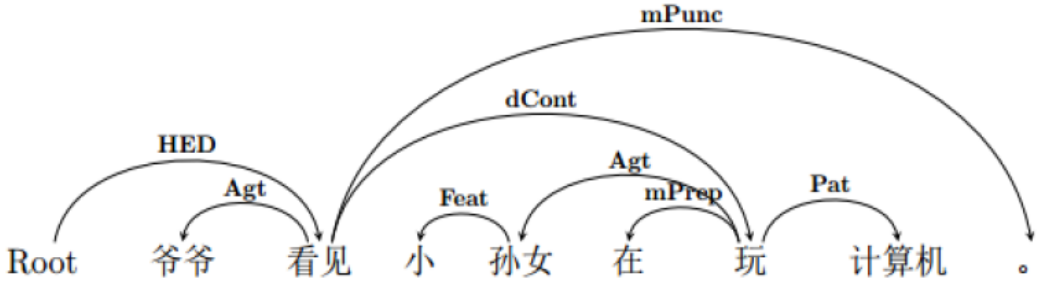
句子数目在不同句子长度下的分布

语义标签

- 语义角色（32种）
  - 一般关系
  - 反关系（r+Role）
  - 嵌套关系（d+Role）
- 事件关系（19种）
- 语义标记（16种）



反关系（**r+Role**）



嵌套关系（**d+Role**）

一般关系标签集合

周边论元	
主体角色	施事 Ag、当事 Exp、感事 Aft、领事 Poss、
客体角色	受事 Pat、客事 Cont、成事 Prod, 源事 Orig、与事 Datv
系体角色	属事 Belg、类事 Clas、依据 Accd
情由角色	缘故 Reas、意图 Int、结局 Cons
状况角色	方式 Mann、工具 Tool、材 Matl
时空角色	时间 Time、空间 Loc, 趋向 Dir、历程 Proc、范围 Sco
度量角色	数量 Quan、数量数组 Qf、频率 Scop、顺序 Seq
其他角色	特征 Feat、宿主 Host、名字修饰 Nmod、时间修饰 Tmod、比较 Comp

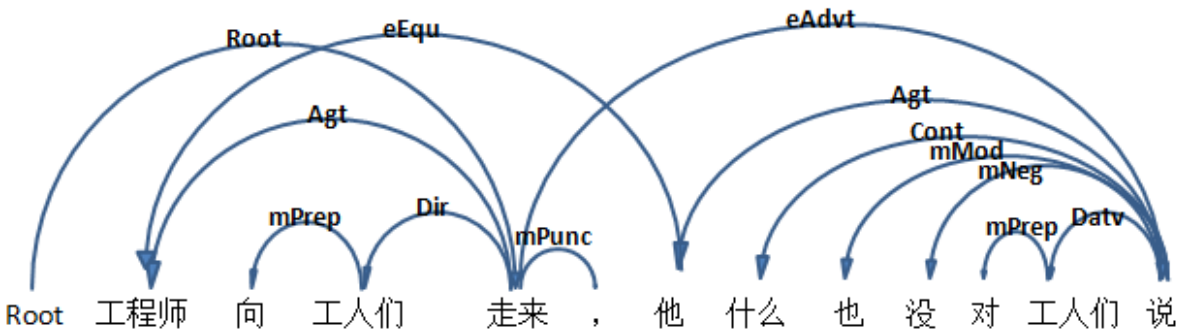
事件关系和语义标记

对称关系	并列关系 eCoo、选择关系 eSelt、等同关系 eEquv
接续关系	先行 EPrec、后继 ESucc、递进 EProg、转折 eAdvt、原因 eCau、结果 eRes、推论 eInf、条件 eCond、假设 eSupp、让步 eConc、手段 eMetd、 目的 ePurp、割舍 eAban、选取 ePref、总括 eSum、分叙 eRect

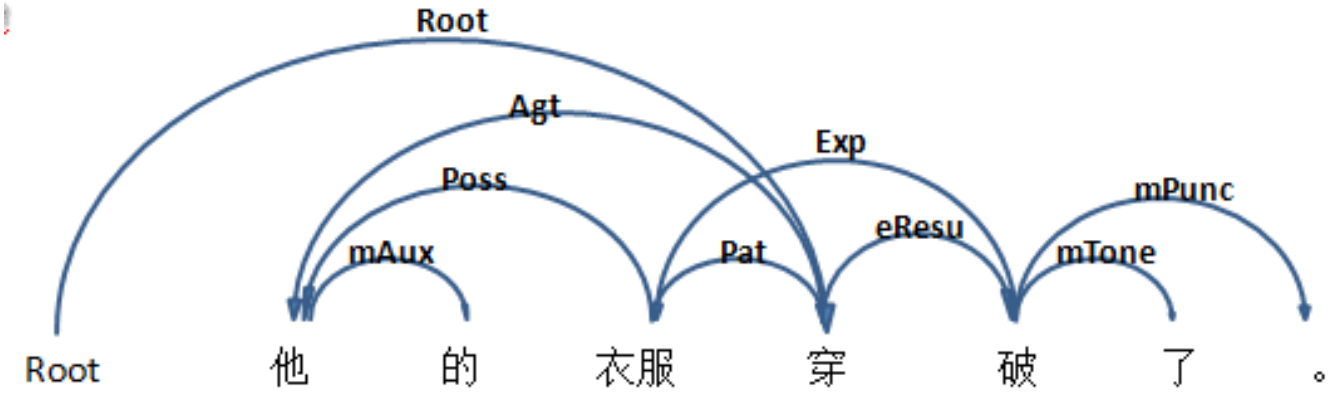
语义标记	
关系标记	连词标记 mConj、的字标记 mAux、介词标记 mPrep
依附标记	语气标记 mTone、时间标记 mTime、范围标记 mRang、程度标记 mDegr、频率标记 mFreq、趋向标记 mDir、插入语标记 mPars、否定标记 mNeg、 情态标记 mMod
其他	标点标记 mPunc、多数标记 mMaj、无实意 mVain、离合标记 mSepa

“交叉弧”典型句子举例

- A:代词指称

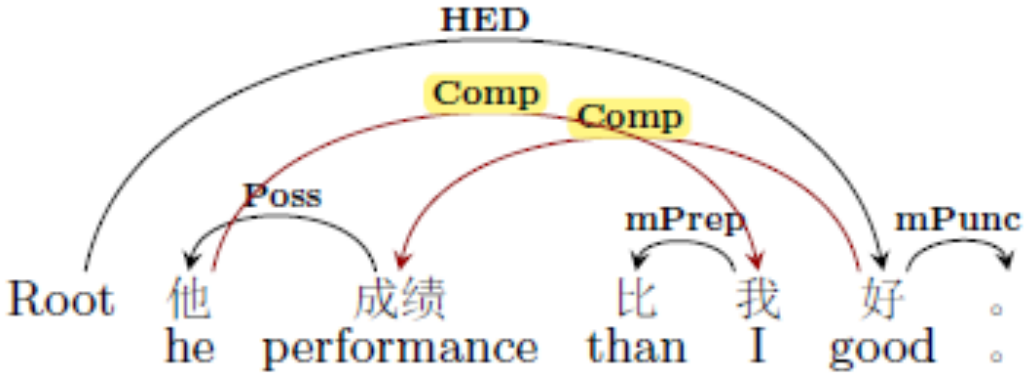


- B:动补谓语句

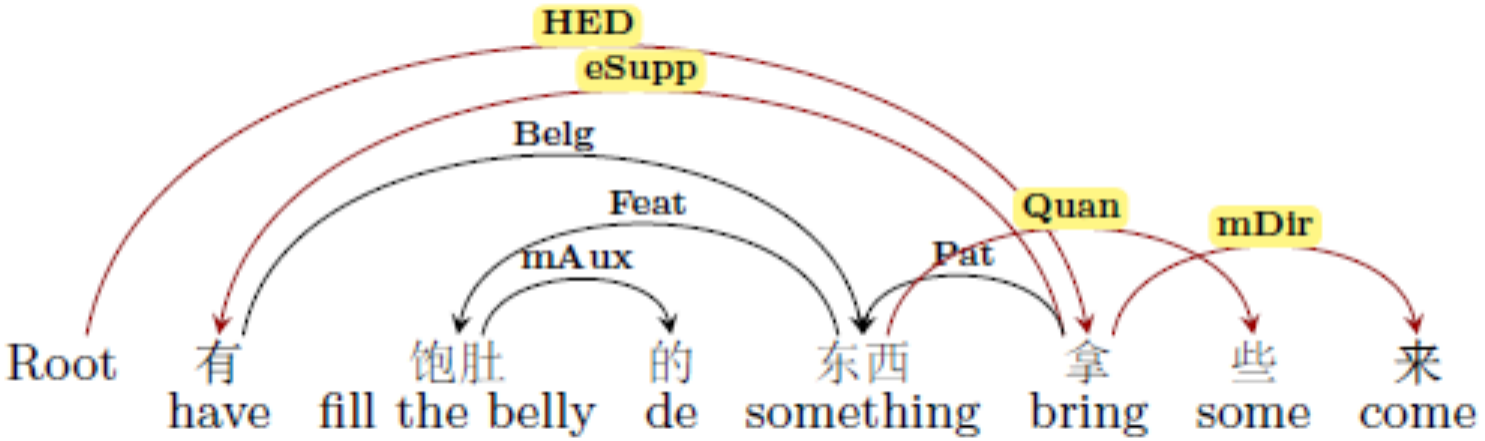


“交叉弧”典型句子举例

- C:比较句

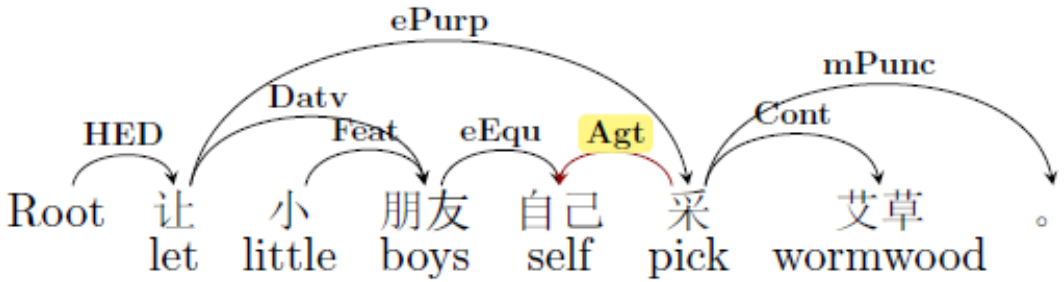


- D:紧缩句：

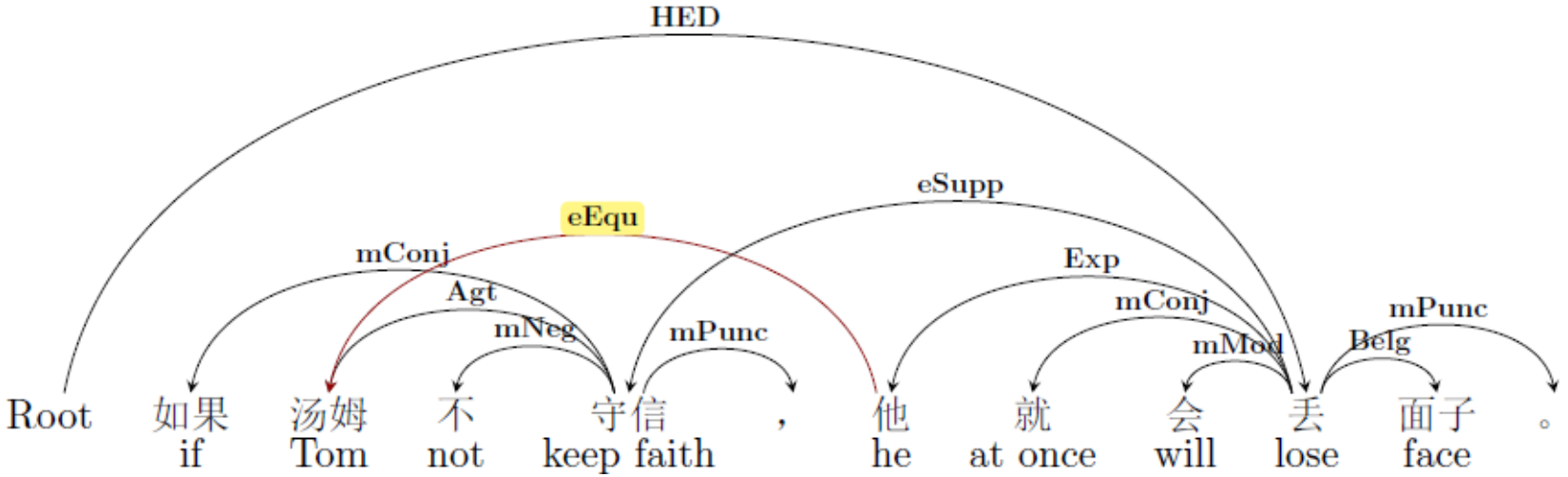


“多父节点”典型句子举例

- A:使动句：多父节点

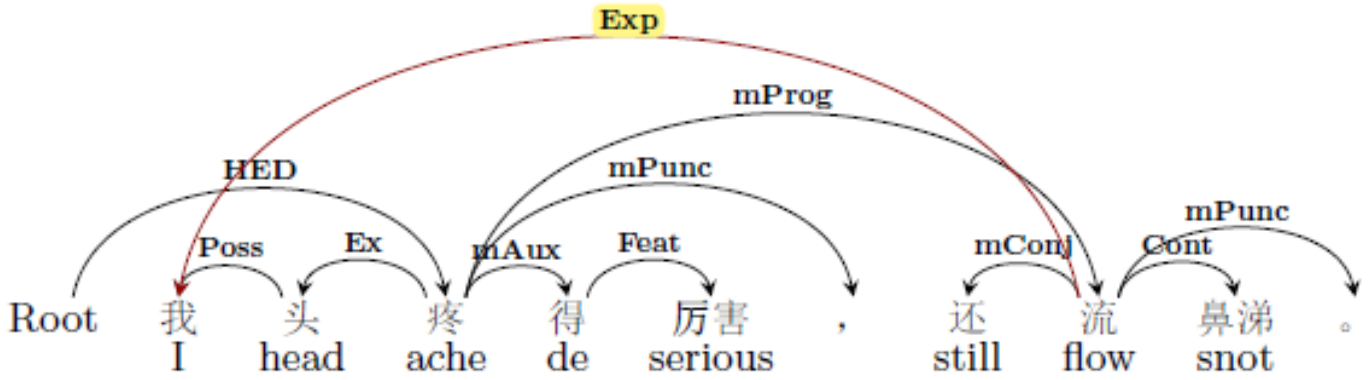


- B:多代词：多父节点

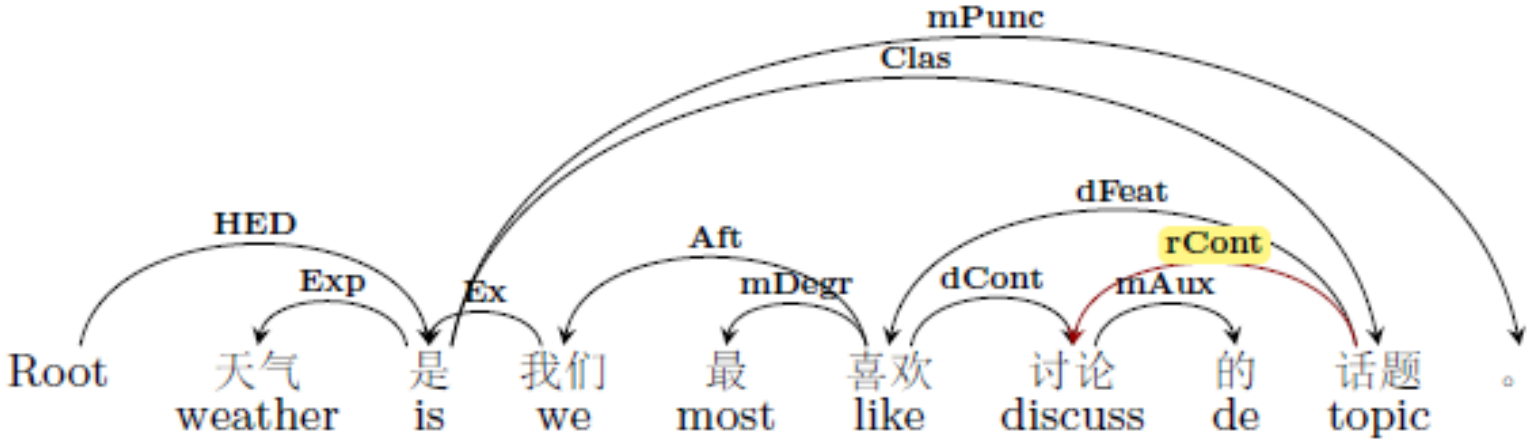


“多父节点”典型句子举例

- C:带省略结构的复句：多父节点



- D:动词短语做修饰语：多父节点



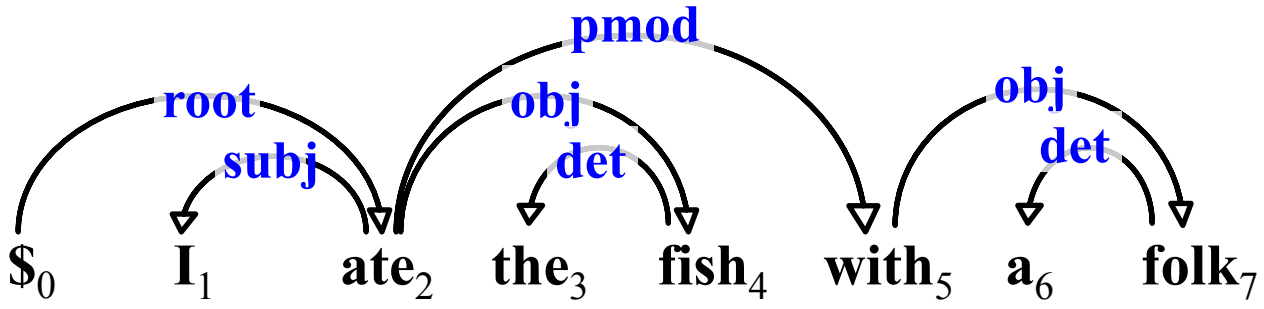
依存树与依存图的对比

依存树	依存图
一个句子中只有一个成分是独立的	一个句子中只有一个成分是独立的
其他成分直接依存于某一个成分	其他成分直接依存于某一个成分
任何一个成分都不能依存于两个或两个以上的成分	任何一个成分可以同时依存于两个或两个以上的成分
如果A成分直接依存于B成分，而C成分在句中位于A和B之间，那么C或者直接依存于B，或者直接依存处于A和B之间的某一成分	如果A成分直接依存于B成分，而C成分在句中位于A和B之间，那么C可以依存于A左边的成分或B右边的成分，即允许依存弧之间交叉
中心成分左右两边的其他成分互相不发生关系	只要中心成分左右两边的成分有语义关系就可以有依存弧连接

依存分析算法

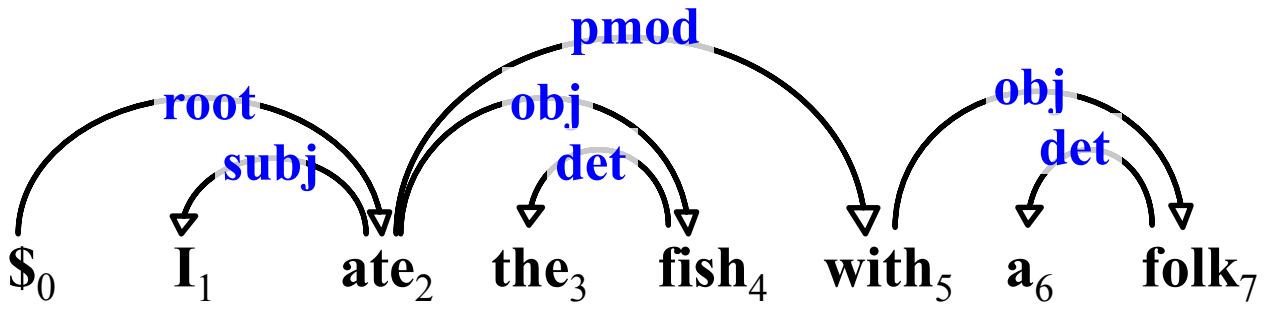
Dependency tree

- A dependency tree is a tree structure composed of the input words and meets a few constraints:
  - Single-head
  - Connected
  - Acyclic



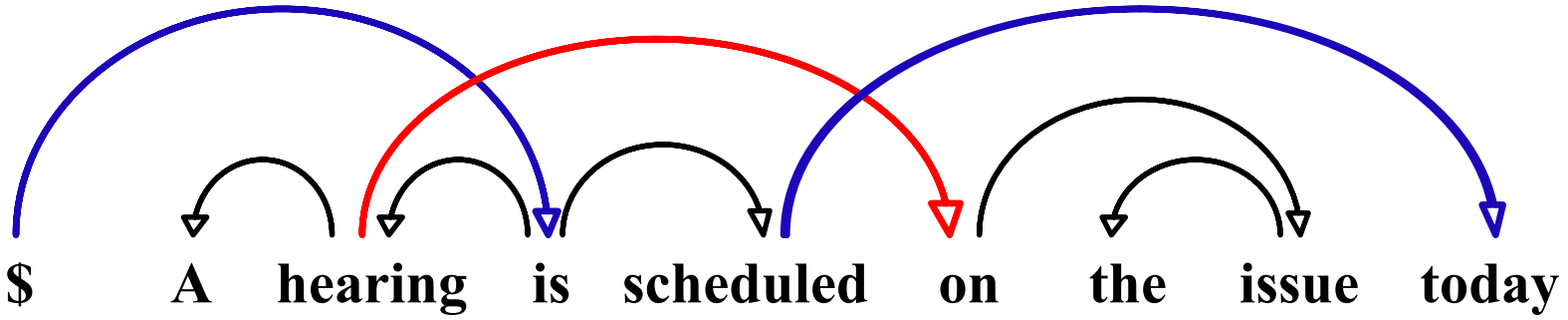
Projective dependency trees

- Informally, "projective" means the tree does not contain any crossing arcs.



Non-projective dependency trees

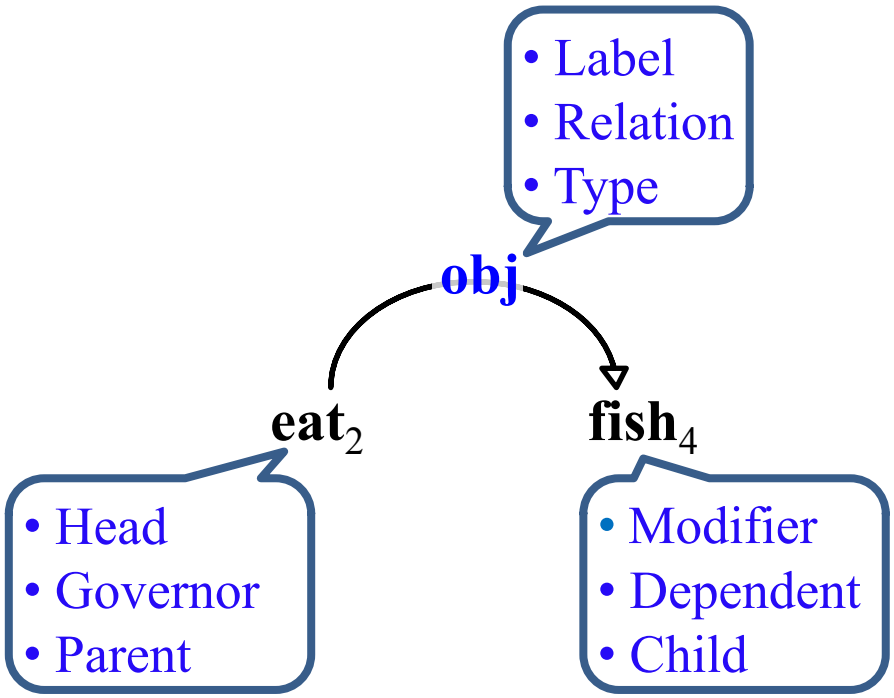
- A non-projective dependency tree contains crossing arcs.



Example from “Dependency Parsing” by Kübler, Nivre, and McDonald, 2009

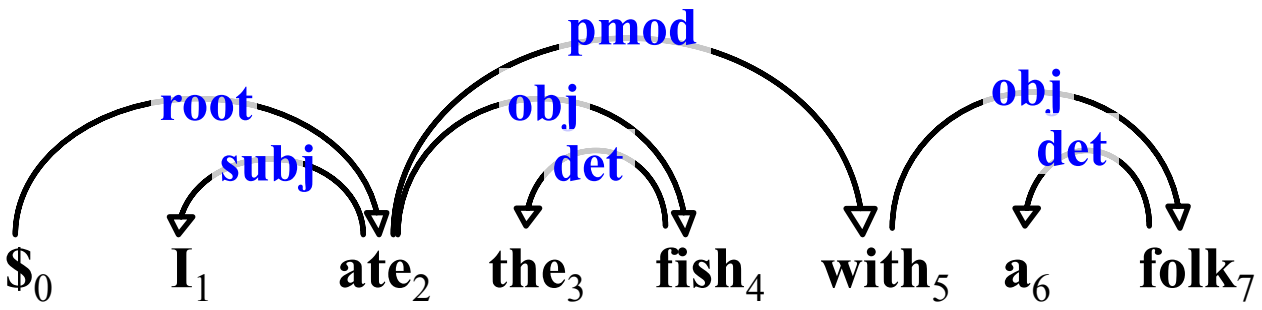
Dependency Tree

- The basic unit is a link (dependency, arc) from the head to the modifier.

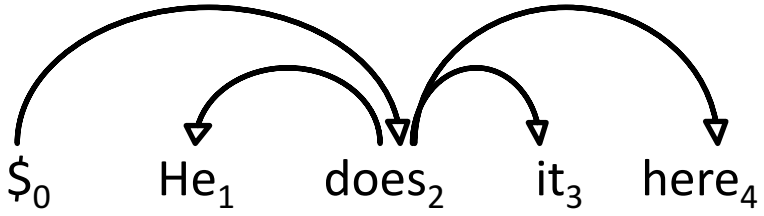


Evaluation Metrics

- Unlabeled attachment score (UAS)
  - The percent of words that have the correct heads
- Labeled attachment score (LAS)
  - The percent of words that have the correct heads and labels.
- Root Accuracy (RA)
- Complete Match rate (CM)



Formalism of Dependency Parsing



$$Y^* = \arg\max_{Y \in \Phi(X)} score(X, Y)$$

$X = x_1 x_2 ... x_n$ → the input sentence

(h, m) → a link from the head x_h to the modifier x_m

$Y = \{(h, m) : 0 \le h \le n, 0 < m \le n\}$ → a candidate tree

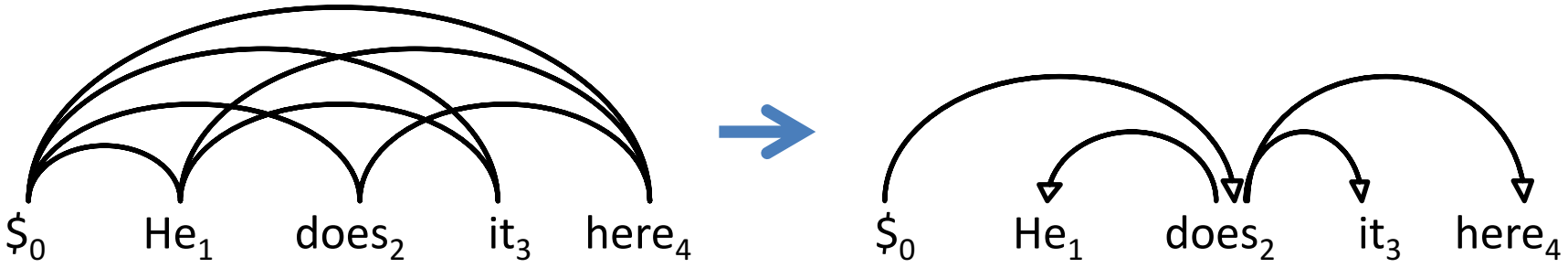
$\Phi(X)$ → The set of all possible dependency trees over X

Supervised Approaches for Dependency Parsing

- Graph-based
- Transition-based
- Hybrid (ensemble)
- Other methods

Graph-based Dependency Parsing

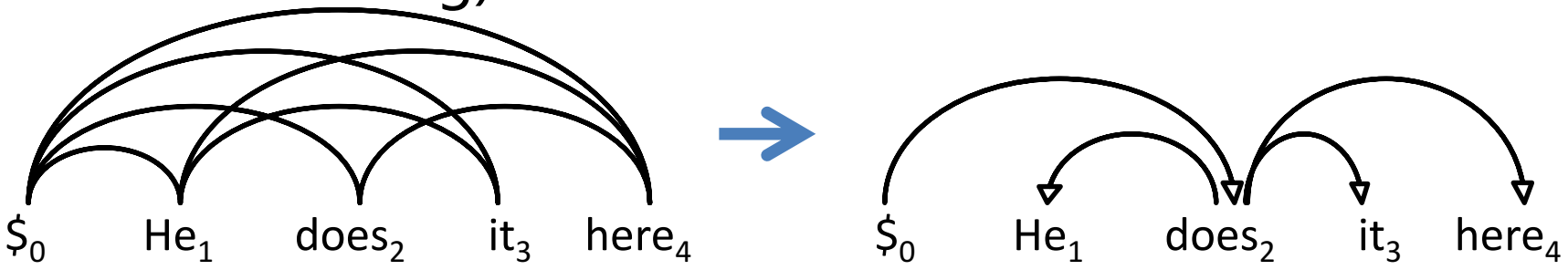
- Find the highest scoring tree from a complete dependency graph.



$$Y^* = \arg\max_{Y \in \Phi(X)} score(X, Y)$$

Two problems

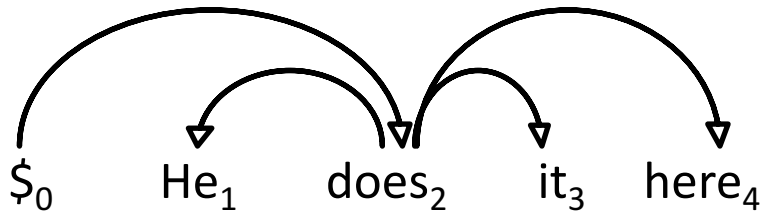
- The search problem
  - Given the score of each link, how to find Y^*?
- The learning problem
  - Given an input sentence, how to determine the score of each link? $\mathbf{w} \cdot \mathbf{f}$
  - How to learn **w** using a treebank (supervised learning)?



$$Y^* = \arg\max_{Y \in \Phi(X)} score(X, Y)$$

First-order as an example

- The first-order graph-based method assumes that the dependencies in a tree are independent from each other. (arc-factorization)



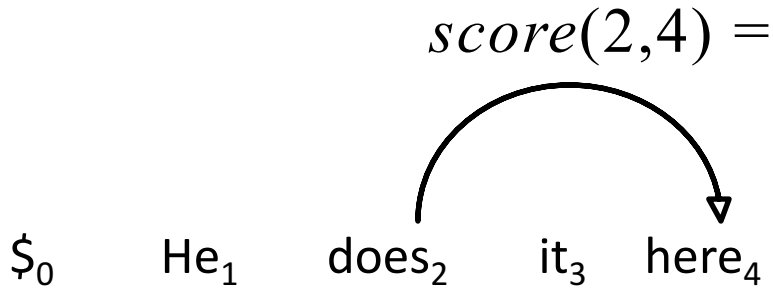
$$score(X, Y) = \sum_{(h,m) \in Y} score(X, h, m)$$

Search problem for first-order model

- Eisner (2000) described a dynamic programming based decoding algorithm for bilexical grammar.
- McDonald+ (2005) applied this algorithm to the search problem of the first-order model.
- Time complexity $O(n^3)$

The learning problem

- Given an input sentence, how to determine the score of each link?



- Feature based representation: a link is represented as a feature vector **f** (2,4)

$$score(2,4) = \mathbf{w} \cdot \mathbf{f}(2,4)$$

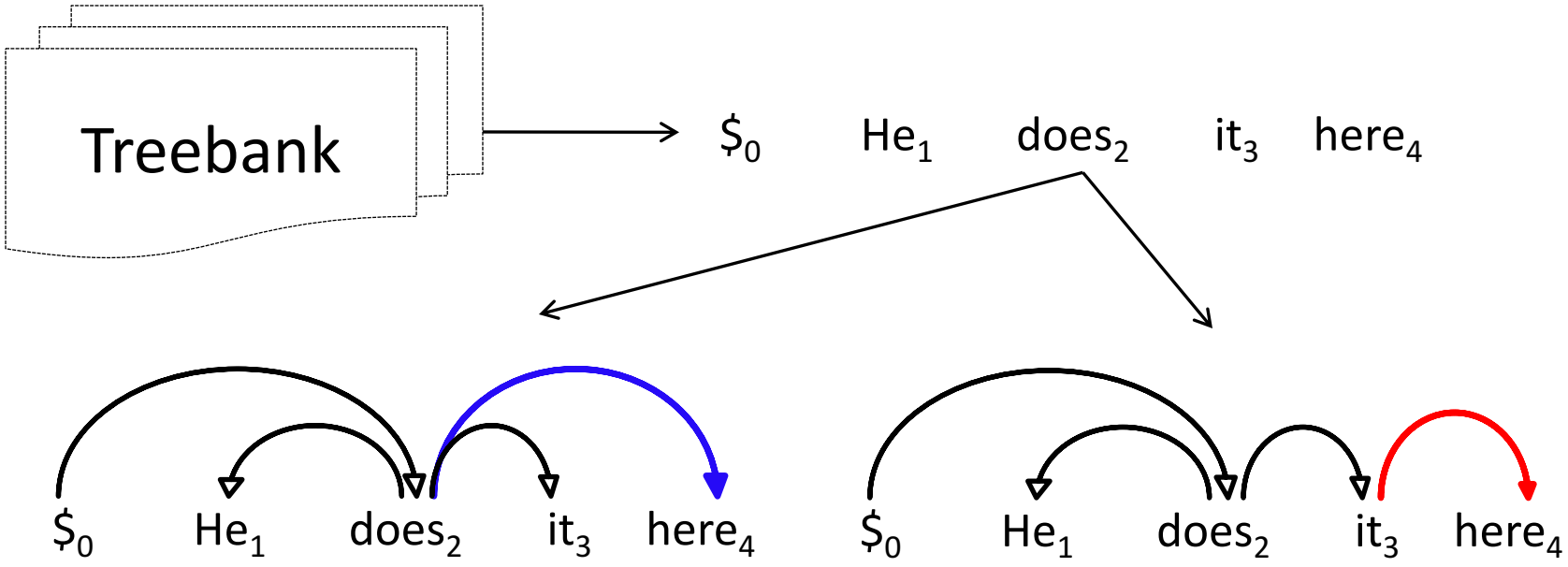
Features for one dependency

[went]	[VBD]	[As]	[ADP]	[went]
[VERB]	[As]	[IN]	[went, VBD]	[As, ADP]
[went, As]	[VBD, ADP]	[went, VERB]	[As, IN]	[went, As]
[VERB, IN]	[VBD, As, ADP]	[went, As, ADP]	[went, VBD, ADP]	[went, VBD, As]
[ADJ, *, ADP]	[VBD, *, ADP]	[VBD, ADJ, ADP]	[VBD, ADJ, *]	[NNS, *, ADP]
[NNS, VBD, ADP]	[NNS, VBD, *]	[ADJ, ADP, NNP]	[VBD, ADP, NNP]	[VBD, ADJ, NNP]
[NNS, ADP, NNP]	[NNS, VBD, NNP]	[went, left, 5]	[VBD, left, 5]	[As, left, 5]
[ADP, left, 5]	[VERB, As, IN]	[went, As, IN]	[went, VERB, IN]	[went, VERB, As]
[JJ, *, IN]	[VERB, *, IN]	[VERB, JJ, IN]	[VERB, JJ, *]	[NOUN, *, IN]
[NOUN, VERB, IN]	[NOUN, VERB, *]	[JJ, IN, NOUN]	[VERB, IN, NOUN]	[VERB, JJ, NOUN]
[NOUN, IN, NOUN]	[NOUN, VERB, NOUN]	[went, left, 5]	[VERB, left, 5]	[As, left, 5]
[IN, left, 5]	[went, VBD, As, ADP]	[VBD, ADJ, *, ADP]	[NNS, VBD, *, ADP]	[VBD, ADJ, ADP, NNP]
[NNS, VBD, ADP, NNP]	[went, VBD, left, 5]	[As, ADP, left, 5]	[went, As, left, 5]	[VBD, ADP, left, 5]
[went, VERB, As, IN]	[VERB, JJ, *, IN]	[NOUN, VERB, *, IN]	[VERB, JJ, IN, NOUN]	[NOUN, VERB, IN, NOUN]
[went, VERB, left, 5]	[As, IN, left, 5]	[went, As, left, 5]	[VERB, IN, left, 5]	[VBD, As, ADP, left, 5]
[went, As, ADP, left, 5]	[went, VBD, ADP, left, 5]	[went, VBD, As, left, 5]	[ADJ, *, ADP, left, 5]	[VBD, *, ADP, left, 5]
[VBD, ADJ, ADP, left, 5]	[VBD, ADJ, *, left, 5]	[NNS, *, ADP, left, 5]	[NNS, VBD, ADP, left, 5]	[NNS, VBD, *, left, 5]
[ADJ, ADP, NNP, left, 5]	[VBD, ADP, NNP, left, 5]	[VBD, ADJ, NNP, left, 5]	[NNS, ADP, NNP, left, 5]	[NNS, VBD, NNP, left, 5]
[VERB, As, IN, left, 5]	[went, As, IN, left, 5]	[went, VERB, IN, left, 5]	[went, VERB, As, left, 5]	[JJ, *, IN, left, 5]
[VERB, *, IN, left, 5]	[VERB, JJ, IN, left, 5]	[VERB, JJ, *, left, 5]	[NOUN, *, IN, left, 5]	[NOUN, VERB, IN, left, 5]

How to learn w? (supervised)

- Use a treebank
  - Each sentence has a manually annotated dependency tree.
- Online training (Collins, 2002; Crammer and Singer, 2001; Crammer+, 2003)
  - Initialize **w** = **0**
  - Go though the treebank for a few (10) iterations.
    - Use one instance to update the weight vector.

Online learning w



Gold-standard parse Y^+

1-best parse Y^- with $\mathbf{w}^{(k)}$

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{f}(X,Y^+) - \mathbf{f}(X,Y^-)$$

Transition-based Dependency Parsing

- Gradually build a tree by applying a sequence of transition actions. (Yamada and Matsumoto, 2003; Nivre, 2003)
- The score of the tree is equal to the summation of the scores of the actions.

$$score(X,Y) = \sum_{i=0}^{m} score(X, h_i, a_i)$$

$a_i \rightarrow$ the action adopted in step i

$h_i \rightarrow$ the partial results built so far by $a_0 \ldots a_{i-1}$

$Y \rightarrow$ the tree built by the action sequence $a_0 \ldots a_m$

Transition-based Dependency Parsing

- The goal of a transition-based dependency parser is to find the highest scoring action sequence that builds a legal tree.

$$Y^* = \arg\max_{Y \in \Phi(X)} score(X, Y)$$

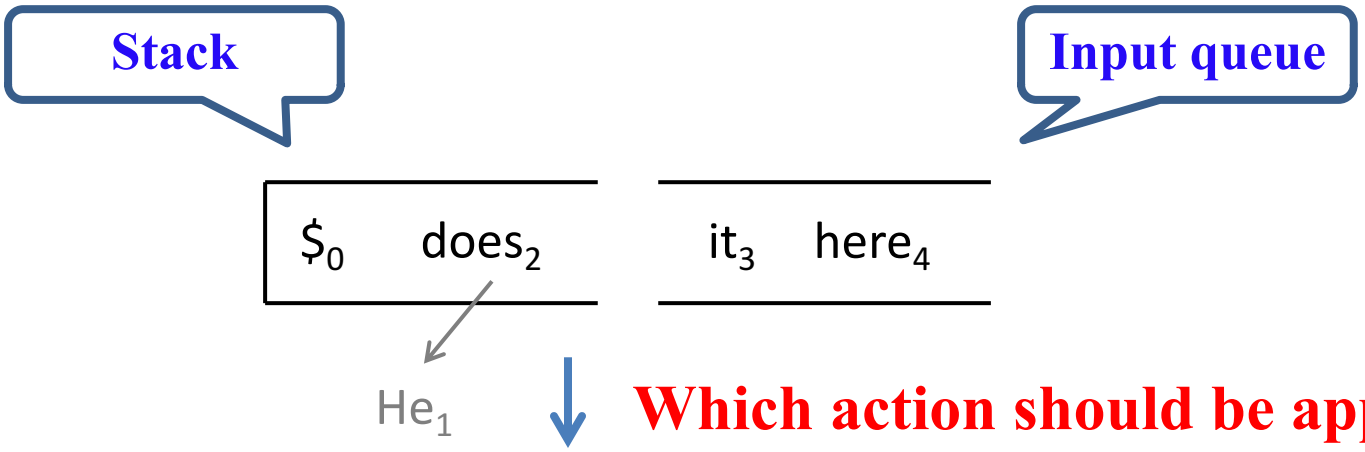
$$= \arg\max_{a_0 \ldots a_m \rightarrow Y} \sum_{i=0}^{m} score(X, h_i, a_i)$$

Two problems for Transition-based Parsing

- The search problem
  - Assuming the model parameters (feature weights) are known, how to find the optimal action sequence that leads to a legal tree for an input sentence?
  - Greedy search as an example
- The learning problem
  - How to learn the feature weights?
  - Online training

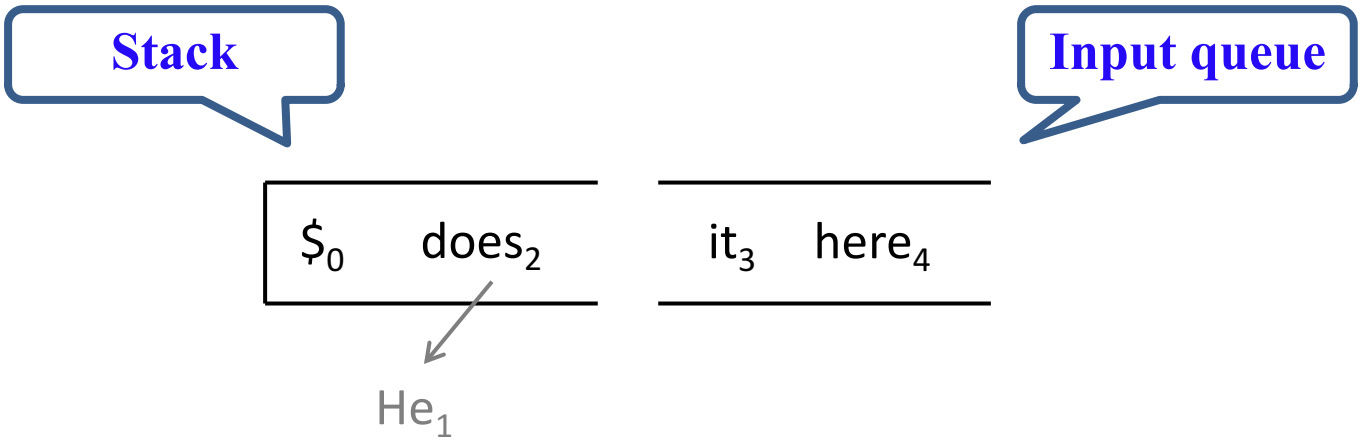
Components of Transition-based Parsing

- A stack to store the processed words and the partial trees
- A queue to store the unseen input words
- Transition actions
  - Gradually build a dependency tree according to the contexts (history) in the stack and the queue.

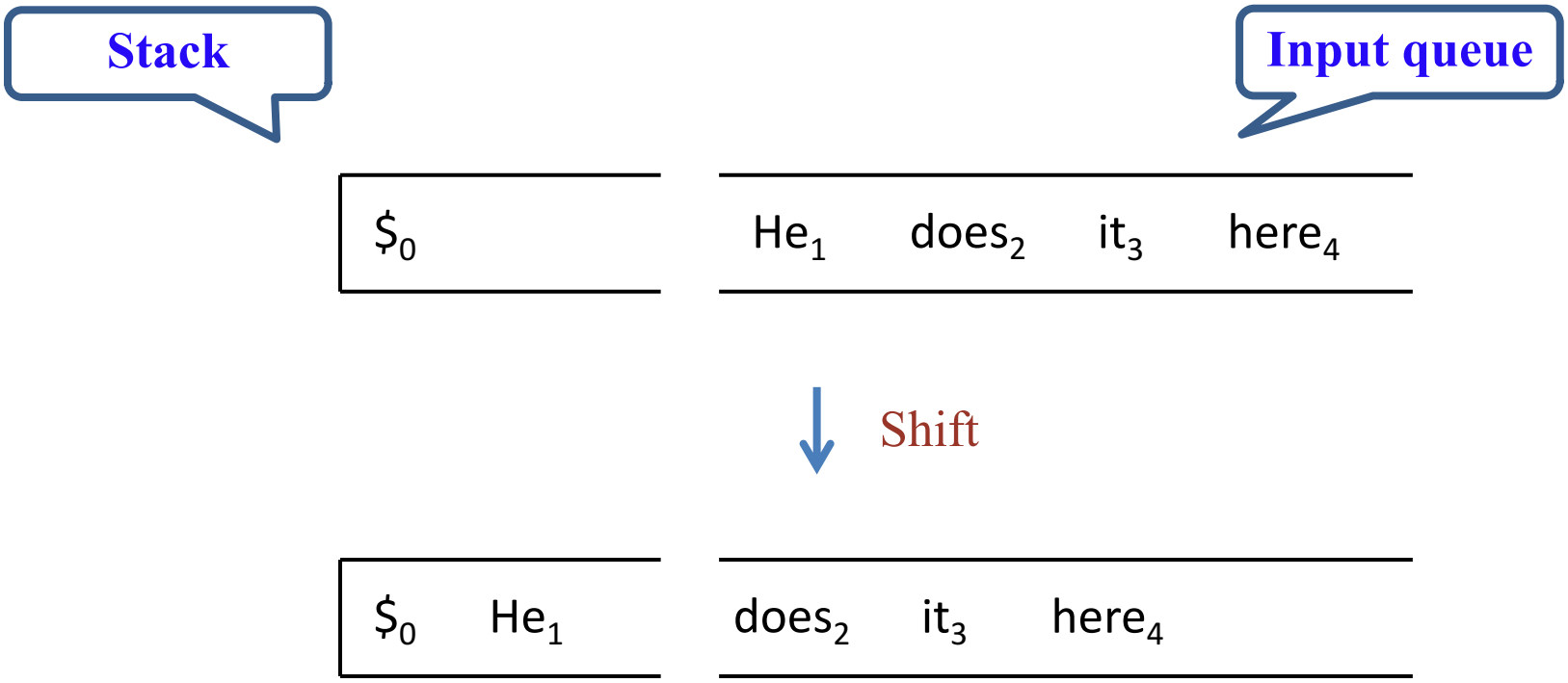


An arc-eager transition-based parser

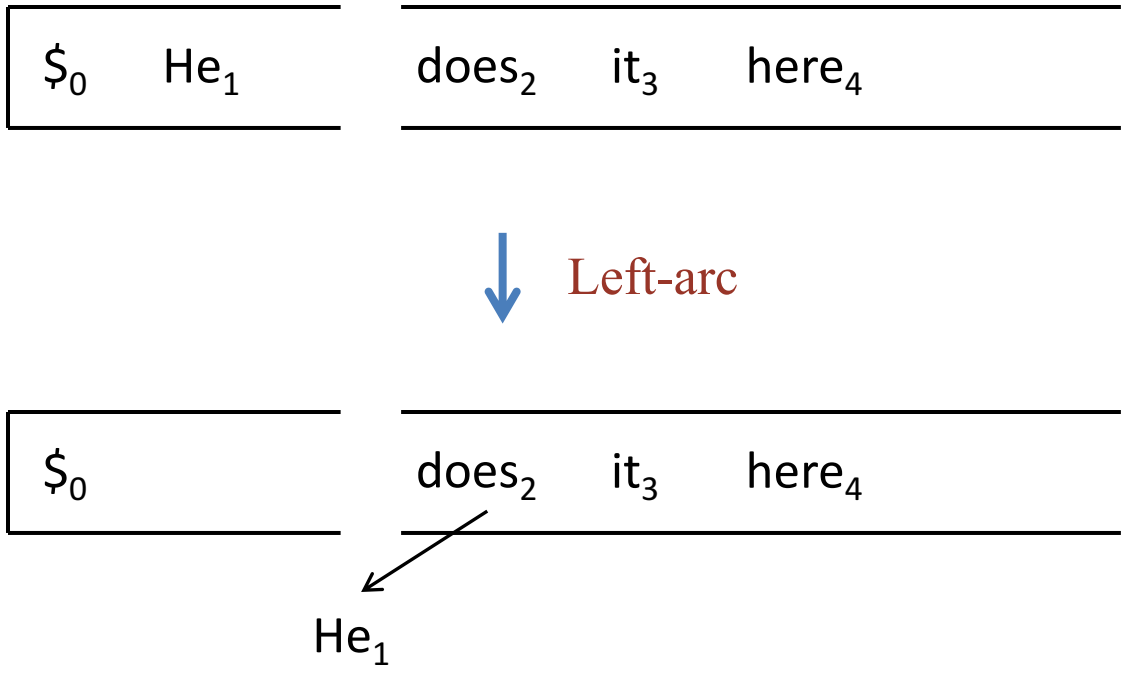
- Four actions
  - Shift
  - Left-arc
  - Right-arc
  - Reduce



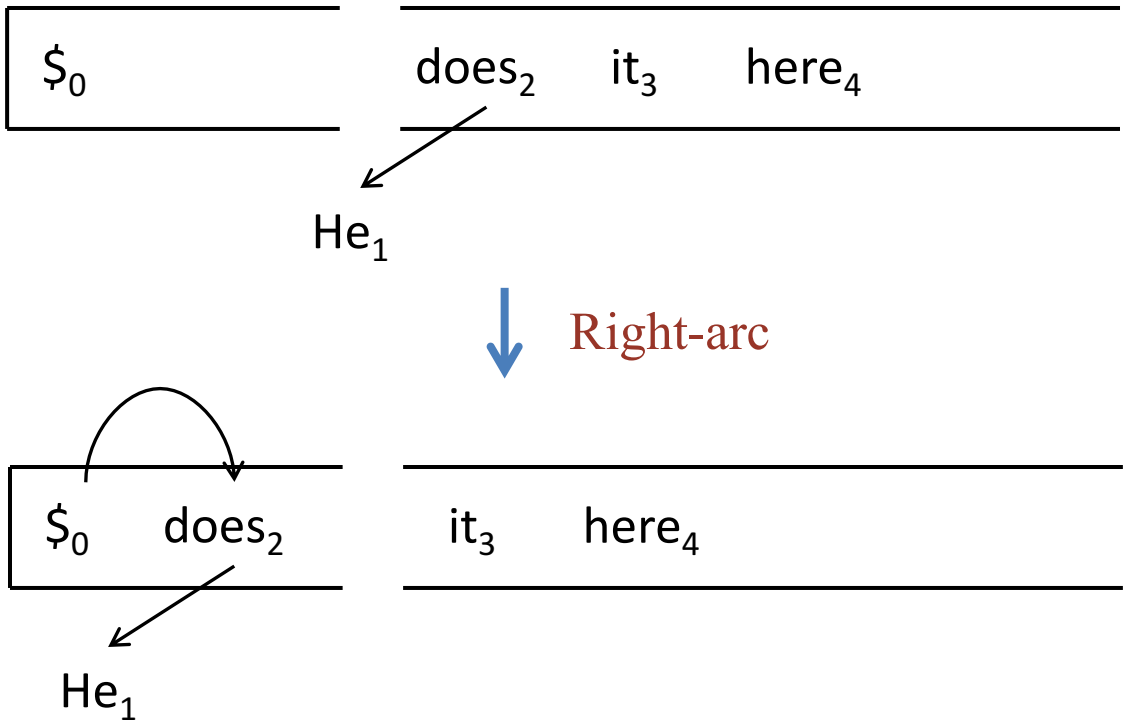
An arc-eager transition-based parser



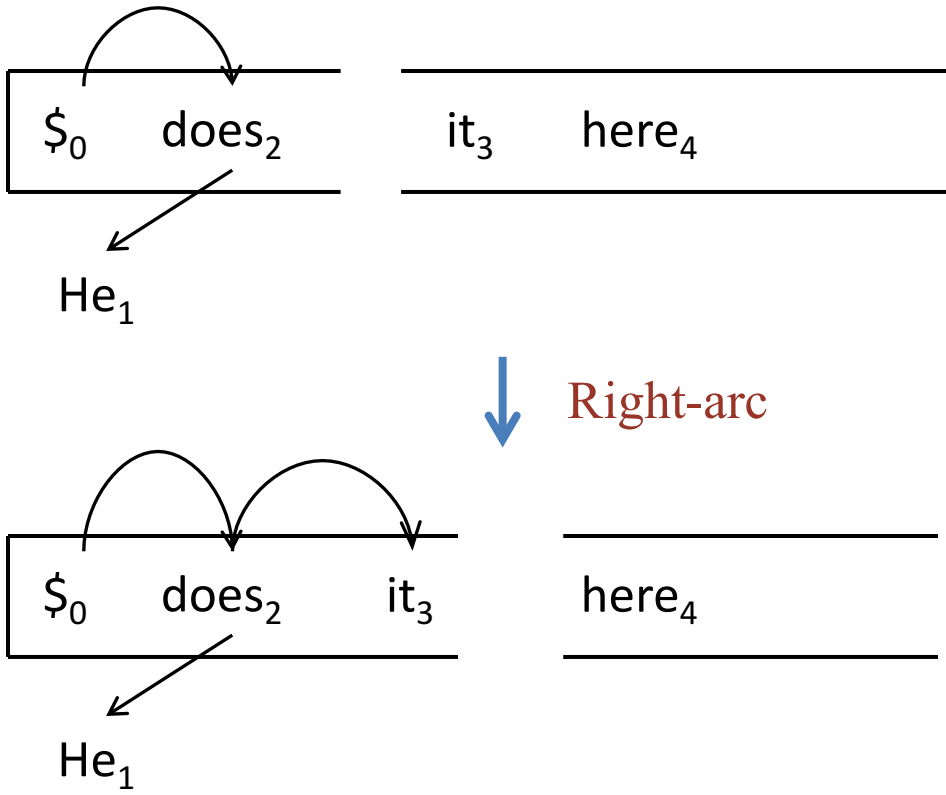
An arc-eager transition-based parser



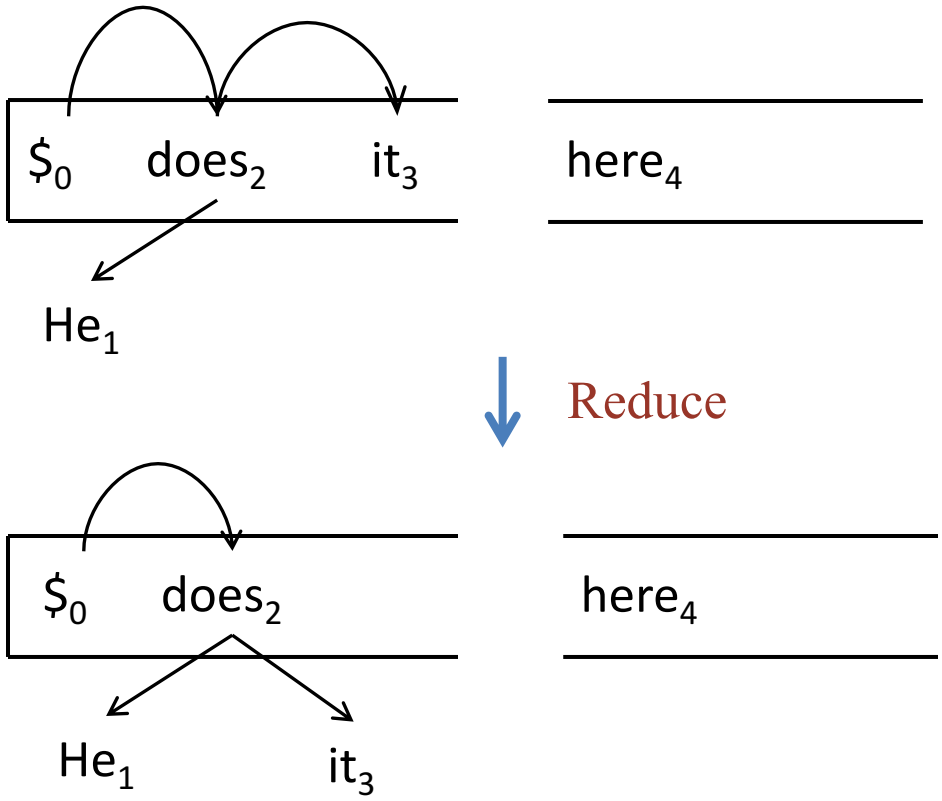
An arc-eager transition-based parser



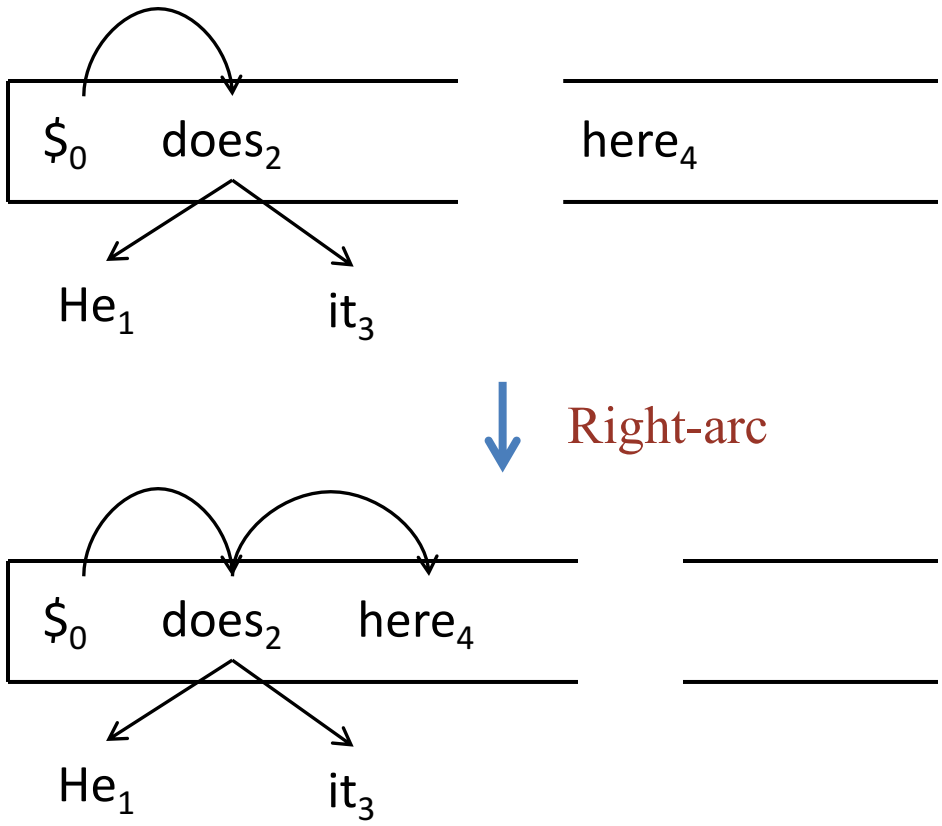
An arc-eager transition-based parser



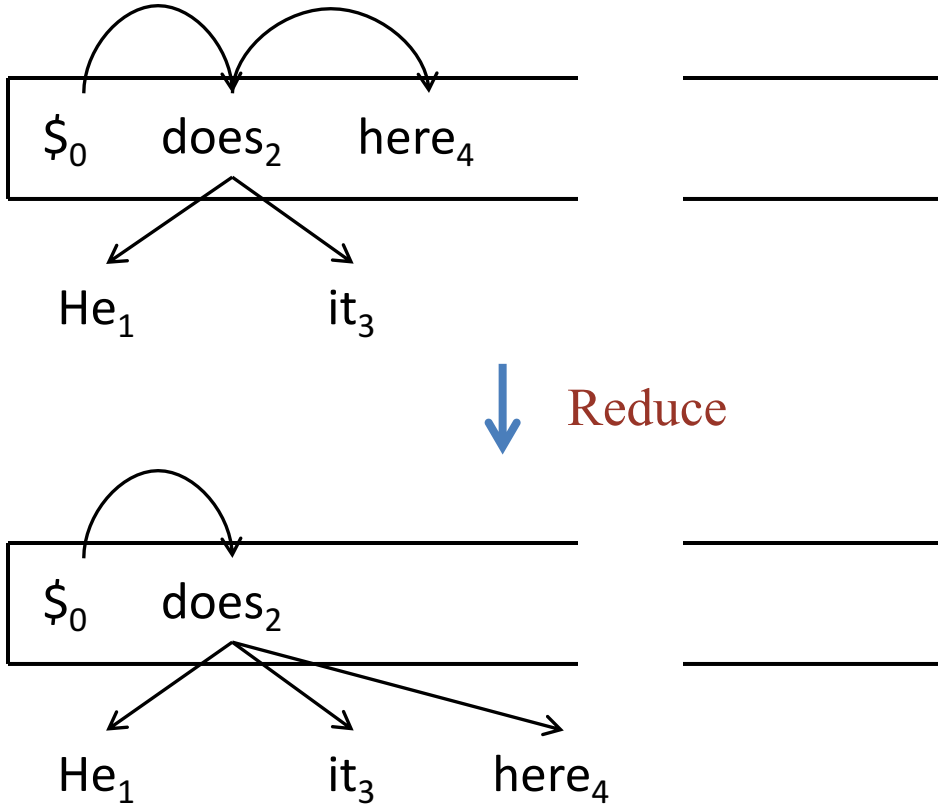
An arc-eager transition-based parser



An arc-eager transition-based parser



An arc-eager transition-based parser



An arc-eager transition-based parser