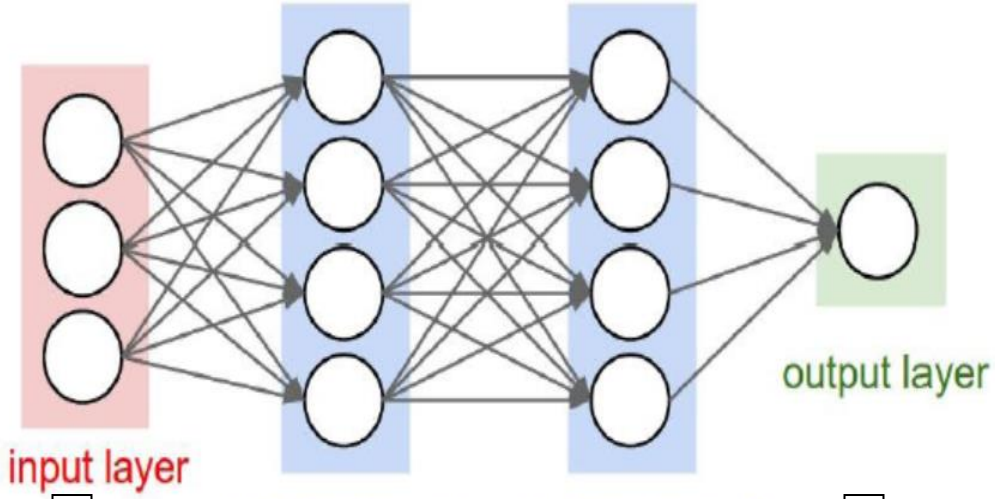
Outline

- 前馈神经模型
- 反向传播算法
- 案例分析

通用流程

- 前馈神经网络可以作为一个编码器(encoder)
- 编码(encoding): 把输入文本序列用一个固定向量来进行表示



文本 → 文本特征(向量)
e.g. 文本每一个词的向量
e.g. 文本长度、文本标点等特征组成的向量
...

可以看成文本最终的向量化表示

一般是sigmoid/ softmax单元
e.g. 二元分类，sigmoid<0.5代表第一类，否则第二类；
softmax=[0.6, 0.4]代表第一类
...

案例一：FNN for 依存分析

- 基于文章：A Fast and Accurate Dependency Parser using Neural Networks

- 发表于EMNLP 2014. PDF
- 第一项将神经网络用于**依存**句法分析的工作
- 传统的依存句法分析使用手工特征，人工总结特征难以覆盖全面，而且特征向量非常稀疏，计算速度慢
- 用神经网络分类器做基于转移(transition-based) 的贪心(greedy)模型来缓解上述问题.

单位：斯坦福NLP组

陈丹琦
普林斯顿NLP组
创始人

Christopher Manning

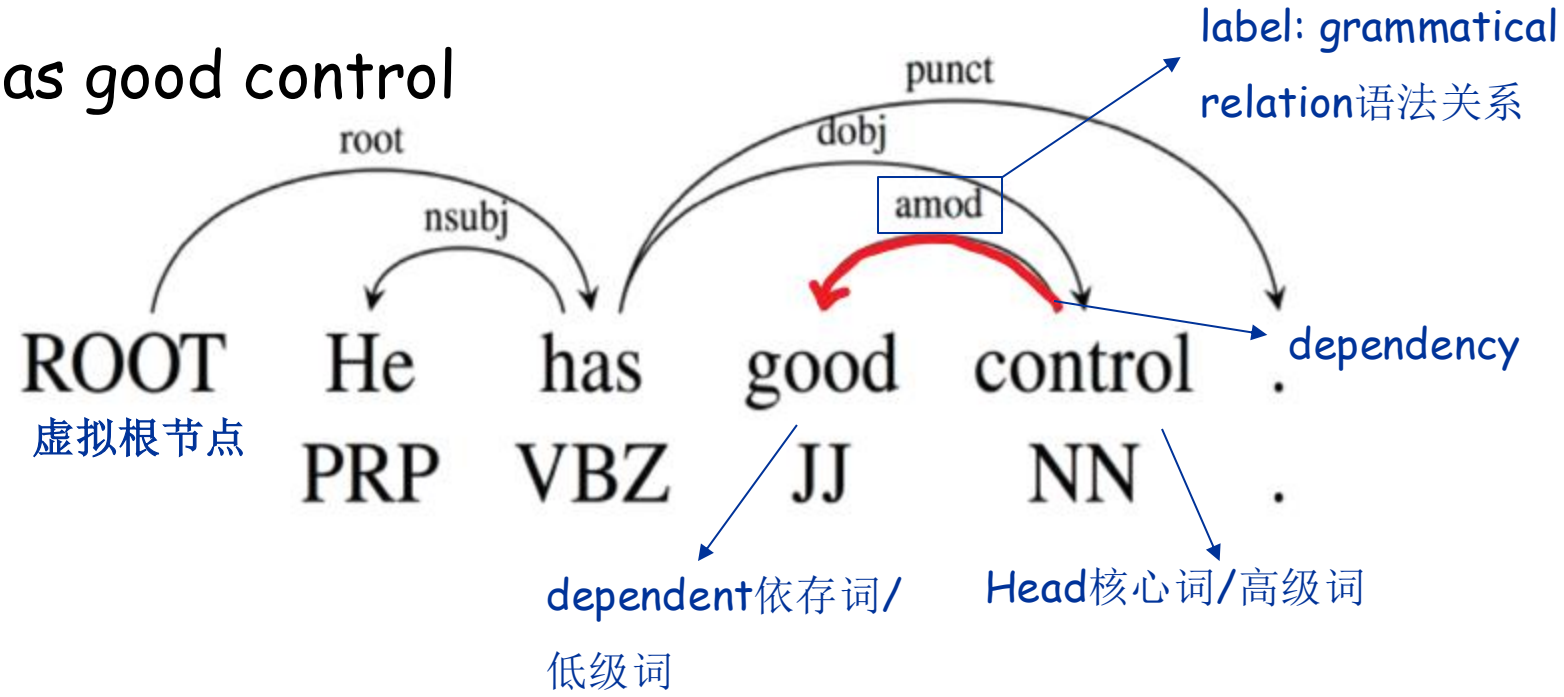
依存分析任务：回顾

Dependency Parsing

词之间的二元非对称依赖关系

形成一棵**连通，非循环，单根，无环，无交叉**的树结构

e.g. He has good control

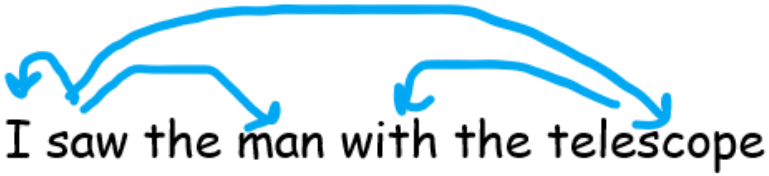


依存分析任务：回顾

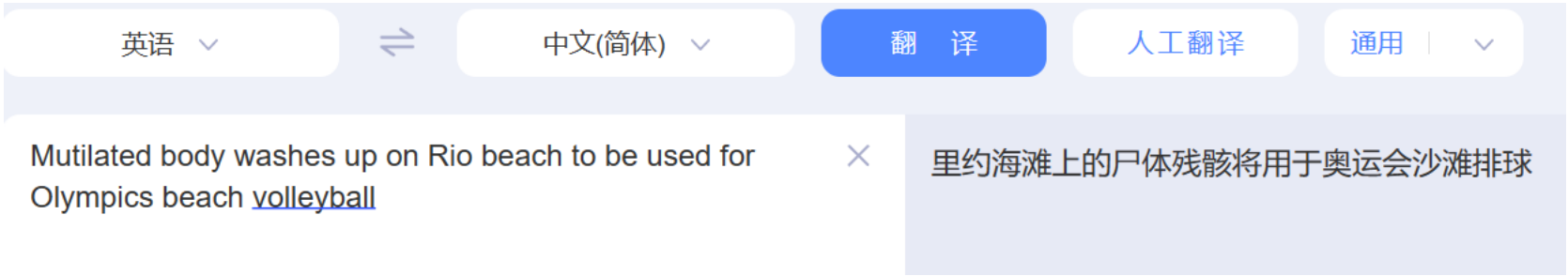
Dependency Parsing

帮助减少句法歧义，更好地理解句子语义，服务于后续应用

e.g. I saw the man with the telescope



e.g.

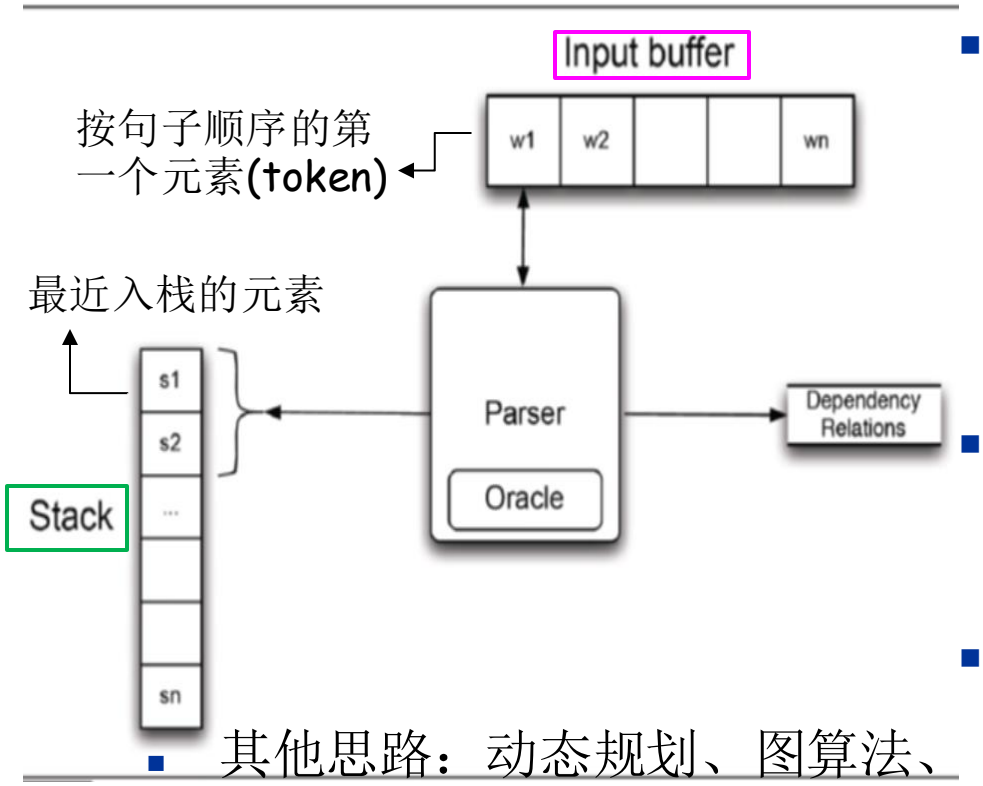


e.g.

???

经典思路：Transition-based (基于转移)

- 基于移进归约(shift-reduce):包含三个组成部分：一个上下文无关文法，一个堆栈，以及一个将要被分析的 token 列表。首先将待分析的 token 依次输入到堆栈中。栈顶两个元素去与文法规则的**右值**比较，如果匹配成功，则元素被语法规则的**左值**替换（规约）。被用来分析程序语言



- 其他思路：动态规划、图算法、约束满足性

- 状态Configuration: 记录不完整的预测结果
  - Stack栈(先进后出)
  - Input buffer of words缓存队列
  - 依存关系集合(存放依存边)
- 转移Transition: 控制每一步状态的变化
- **依存分析目标：**找到一个最终的状态，其中所有涉及关系的单词都会形成依存树

转移操作

- 每一步Transition做什么：根据当前状态（栈stack，缓冲区buffer，依存关系dependency），产生一个新的状态

- 开始状态
  - Stack使用根节点root初始化
  - Buffer使用句子中的词序列初始化
  - 依存关系集合为空

- 结束状态
  - Stack有一个根节点
  - Buffer为空

  此时，依存关系集合为最终的依存分析结果

- 此时依存分析也就是：找到一个转移序列，该转移序列实现了从开始状态到理想结束状态的过程

转移操作

- 转移操作：改变状态中的 stack, buffer, dependency。
- arc-standard transition-based parser 包含3类动作：

(1) LEFT-ARC:

- 添加一个依存边为s1→s2，s1是栈顶的词（最后入栈），s2是第二个词（要求stack中元素个数大于等于2）
- 将s2从stack中移除

(2) RIGHT-ARC:

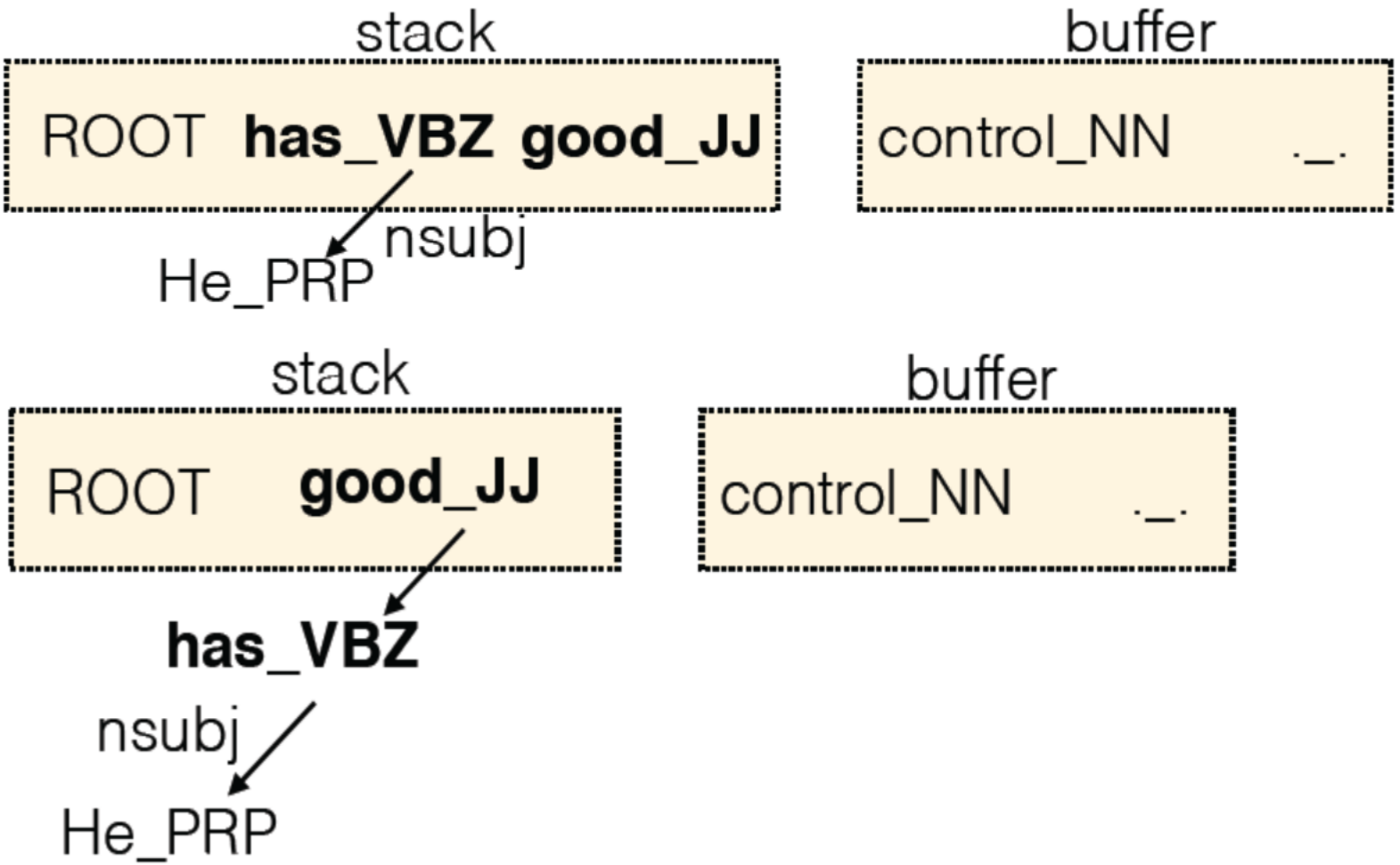
- 添加一个依存边为s2→s1 （要求stack中元素个数大于等于2）
- 将s1从stack中移除

(3) SHIFT

- 从buffer中移除第一个词b1（要求buffer中元素个数大于等于1）
- b1压入栈

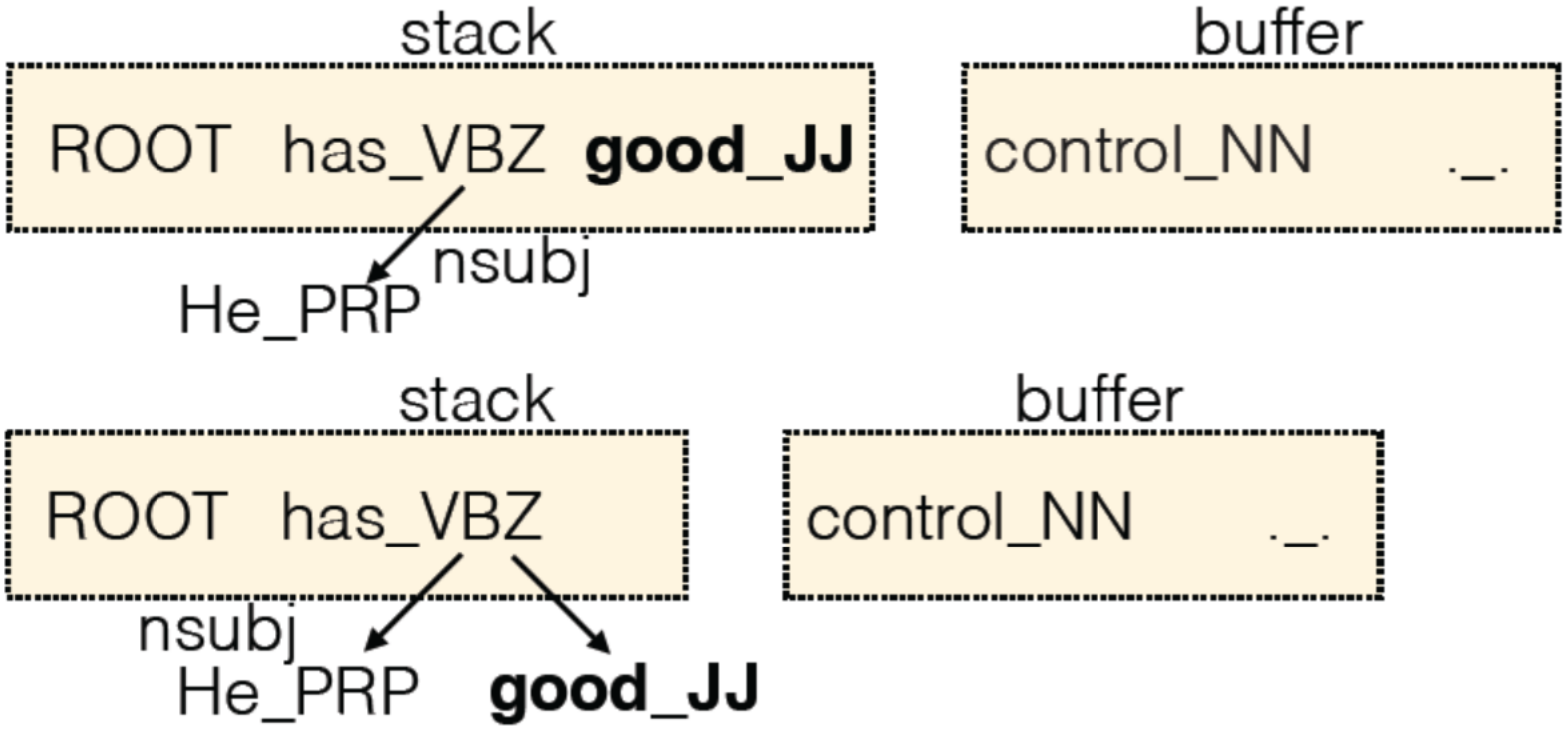
LEFT-ARC

添加依存边s1→s2，stack中移除s2



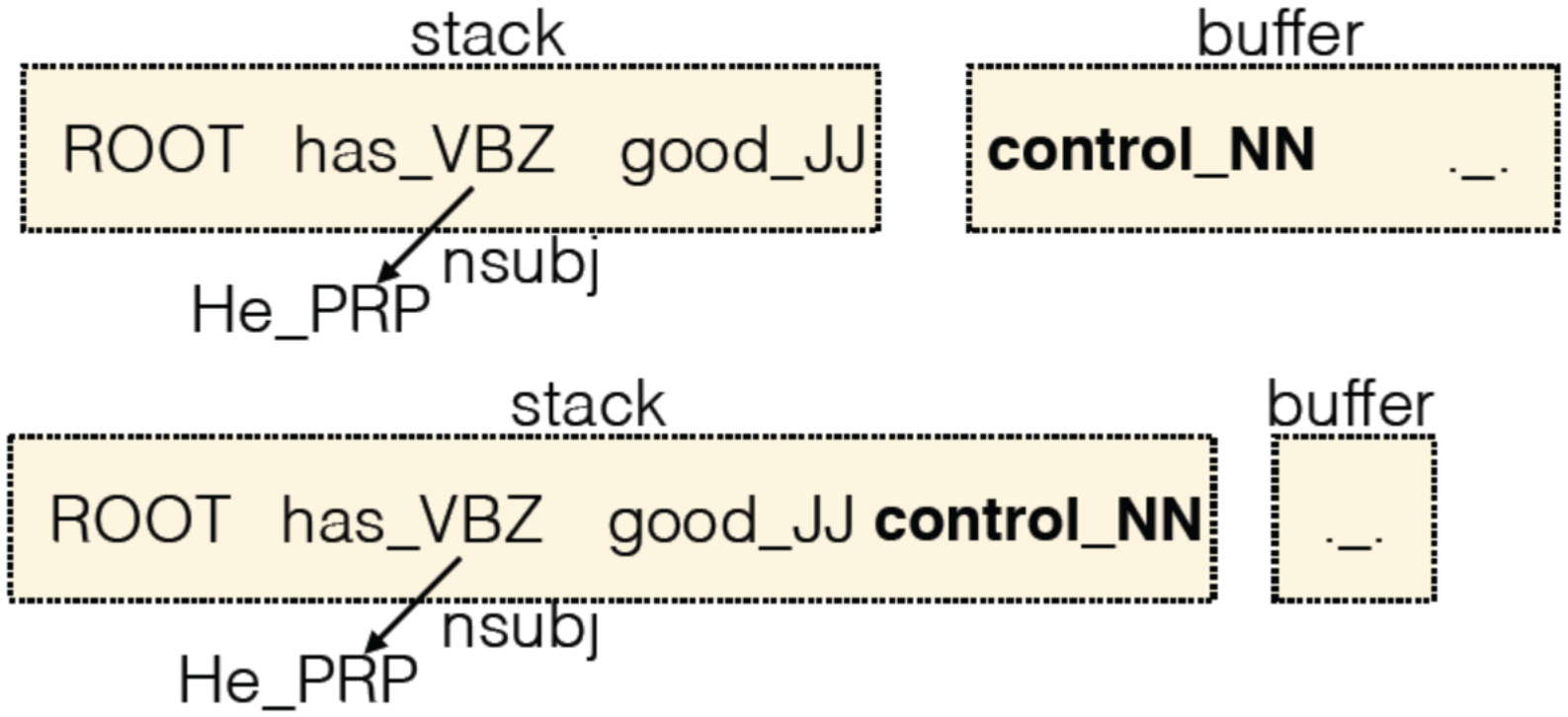
RIGHT-ARC

添加依存边s2→s1，stack中移除s1

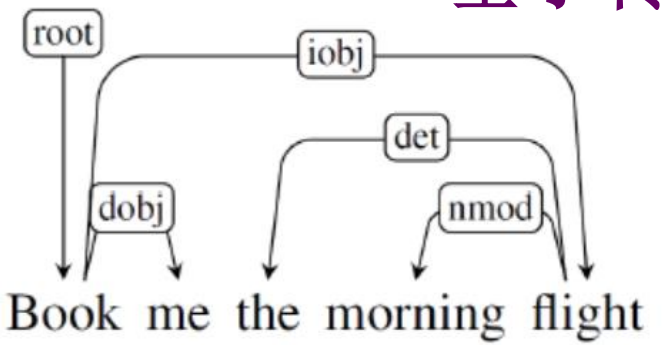


SHIFT

从buffer中移除第一个词b1，压入栈



基于转移的依存分析步骤演示

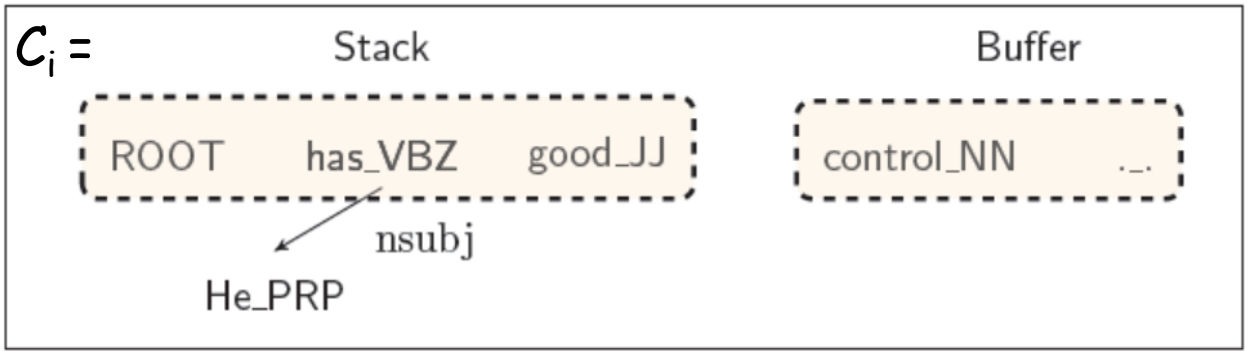


- LEFT-ARC: 创建s1→s2; 删除s2
- RIGHT-ARC: 创建s2→s1; 删除s1
- SHIFT: 删除b1; b1入栈
- 结束状态：stack只有root, buffer为空

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

基于转移的依存分析：贪心策略

- 每一个步骤贪心地预测下一个要采取的动作，只考虑当前状态下概率最大的动作，完成转移
- E.g. C_i =



动作是什么？
C_{i+1} =?

- 两种设置：

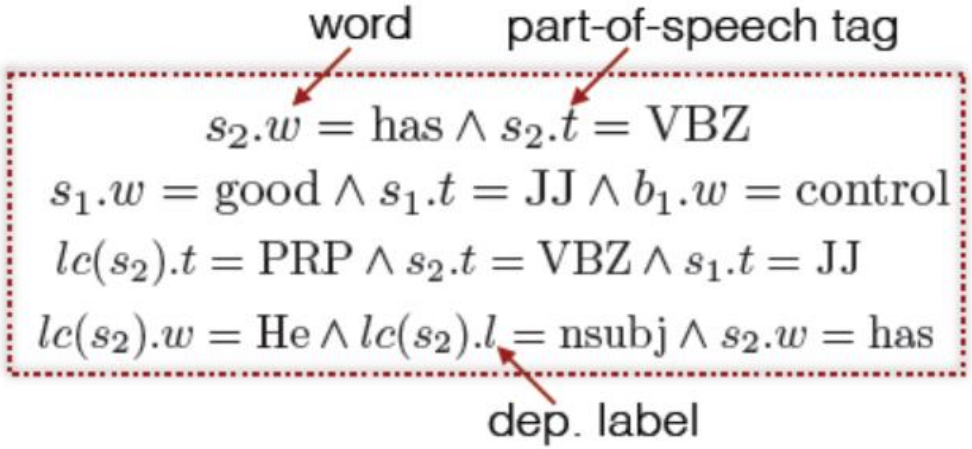
难点：如何分类？ 预言机(oracle)：提供操作符

(1) unlabeled：只预测哪一种动作(left-arc, right-arc, shift)。

(2) labeled：预测哪一种动作，以及left-arc或right-arc时两个词之间的依存关系。假设一共有n种依存关系，则进行(2n+1)类分类。

MaltParser(2005)：判别式分类器；s1的词和词性，b1的词和词性...

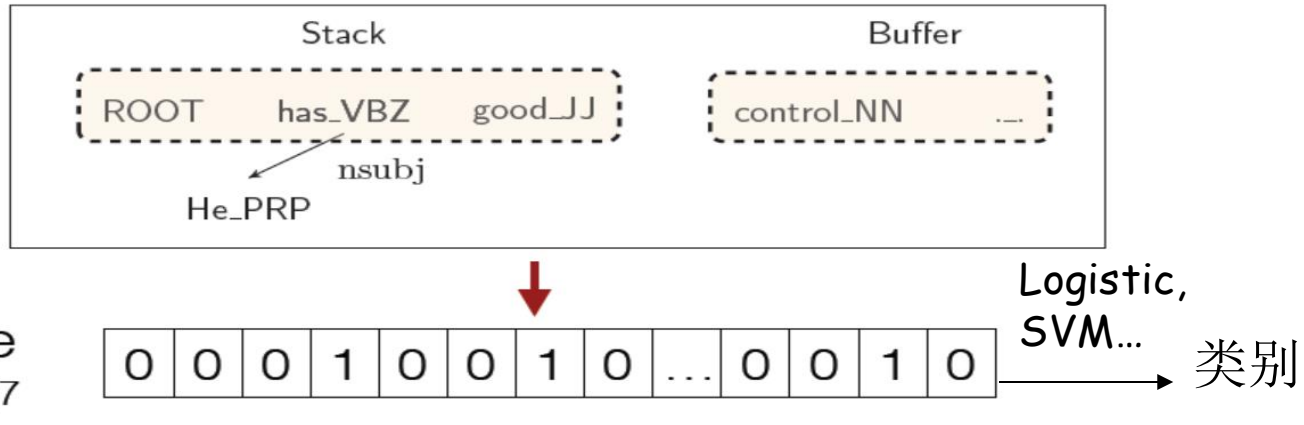
传统分类特征



Indicator features

查看是否满足？

传统的分类特征由人工总结。根据状态：词汇、词性、依存关系标
由这些indicator features通过拼接构成了一个很大的特征向量，该向量的值是0或1，且0占据非常大的比例，是一个稀疏向量，维度达10^6-10^7



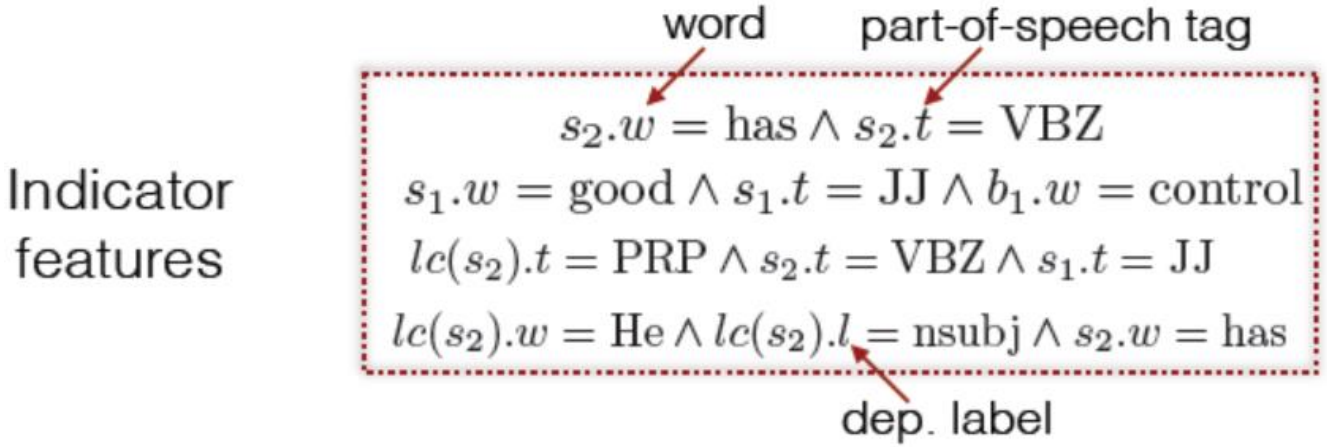
binary, sparse
dim = 10^6 ~ 10^7

传统分类特征

问题1: 向量稀疏。在indicator特征中匹配，本身很稀疏；难以表示向量的相互作用(乘法)

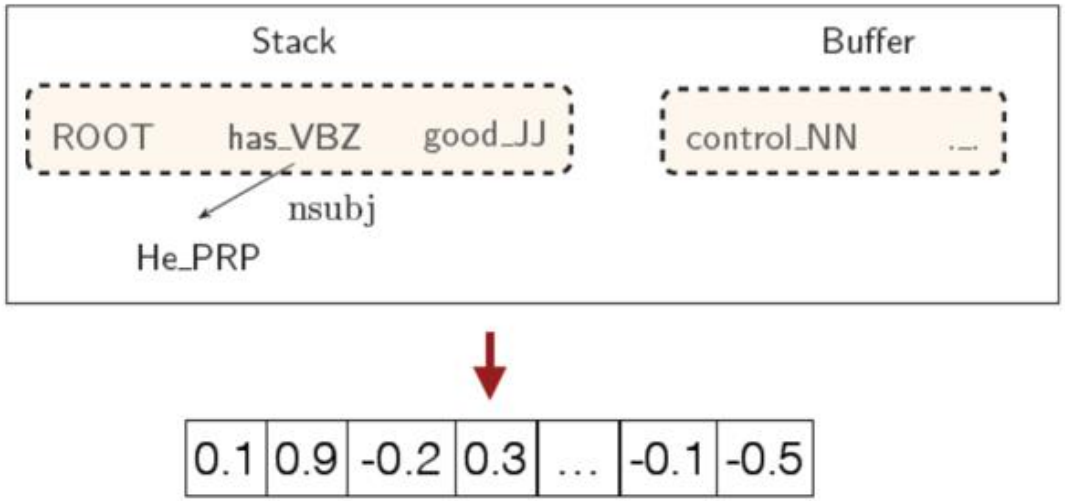
问题2: 特征不完整。难以总结所有特征模板

问题3: 计算代价昂贵。对词语、词性标签或语法关系标签进行拼接来生成特征字符串，并在包含数百万特征的巨大表格中进行查找。95%以上的解析时间用于特征计算。



使用神经网络的分类特征

- 3大问题的解决思路：使用神经网络
- 学习**稠密**（不会出现很多0）、**紧凑**（维数远小）的特征表示向量



dense
dim = 200

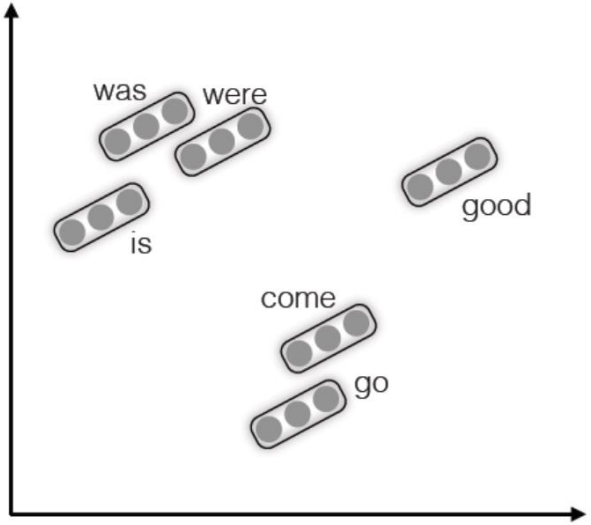
- 问题：

✓ 如何对所有可用信息进行编码？

✓ 如何对高阶特征建模？

使用神经网络的分类特征

- 首先考虑到这是一个针对文本数据的处理任务，采用词语的分布式表示：把每一个词语表示为一个d维的稠密向量 (即词向量Word embeddings).
  - 理想：相似的词，则词向量也相近
  - 映射到2维平面：



- 考虑到该任务是一个依存分析任务，词性和依存关系也是紧密相关的信息：使用向量表示词性和依存关系标签.

NNS (plural noun) should be close to NN (singular noun).

num (numerical modifier) should be close to amod (adjective modifier).

神经依存分析模型结构

在转移中每一个步骤都进行一次预测，每一次预测使用这样一个单隐藏层的神经网络：输入层、隐藏层、以softmax作为输出单元的输出层，实现对状态的更新。

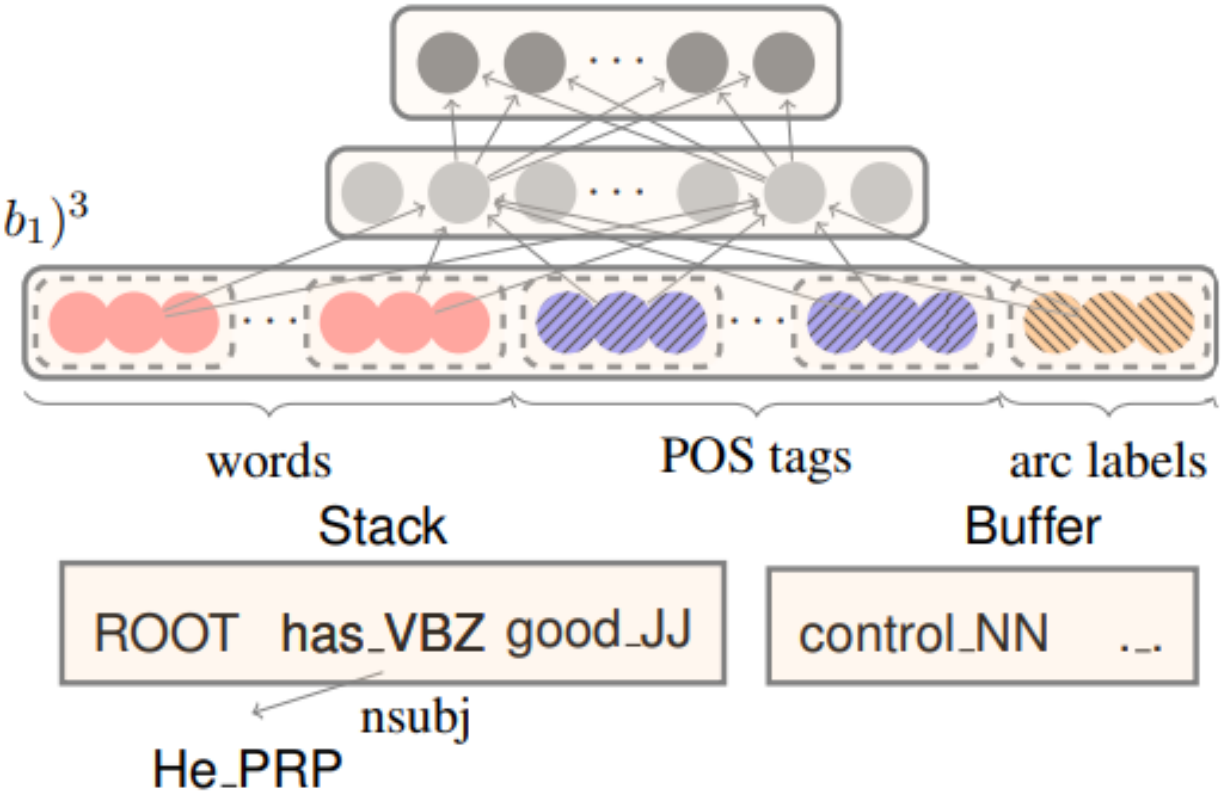
Softmax layer:

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

Hidden layer:

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

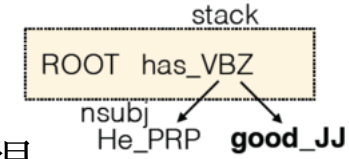
Input layer: $[x^w, x^t, x^l]$



神经网络实际上作为一个分类器组件，完成了文本分类

输入层特征

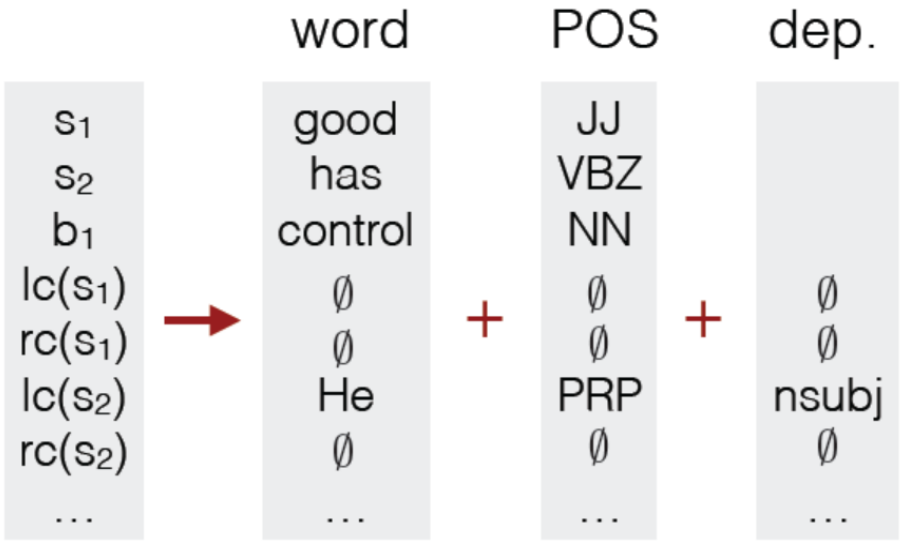
- 词语特征：取以下词的词向量
- stack和buffer前3个单词: $s_1, s_2, s_3, b_1, b_2, b_3$（不够补null_token）
- stack前两个单词的左、右孩子中距离最近的两个孩子:
  $lc_1(s_1), rc_1(s_1), lc_2(s_1), rc_2(s_1), lc_1(s_2), rc_1(s_2), lc_2(s_2), rc_2(s_2)$
- stack前两个单词距离最近左孩子的最近左孩子， 最近右孩子的最近右孩子: $lc_1(lc_1(s_1)), rc_1(rc_1(s_1)), lc_1(lc_1(s_2)), rc_1(rc_1(s_2))$



- 词性特征：
- 以上18个词的词性标记：Stanford POS tagger获得
- 依存边特征
- 以上 后12个词的依存边的标签
- 英文工具: CoNLL Syntactic Dependencies, Stanford Basic Dependencies；中文工具：Penn2Malt
- **比传统的手工特征工作量小**

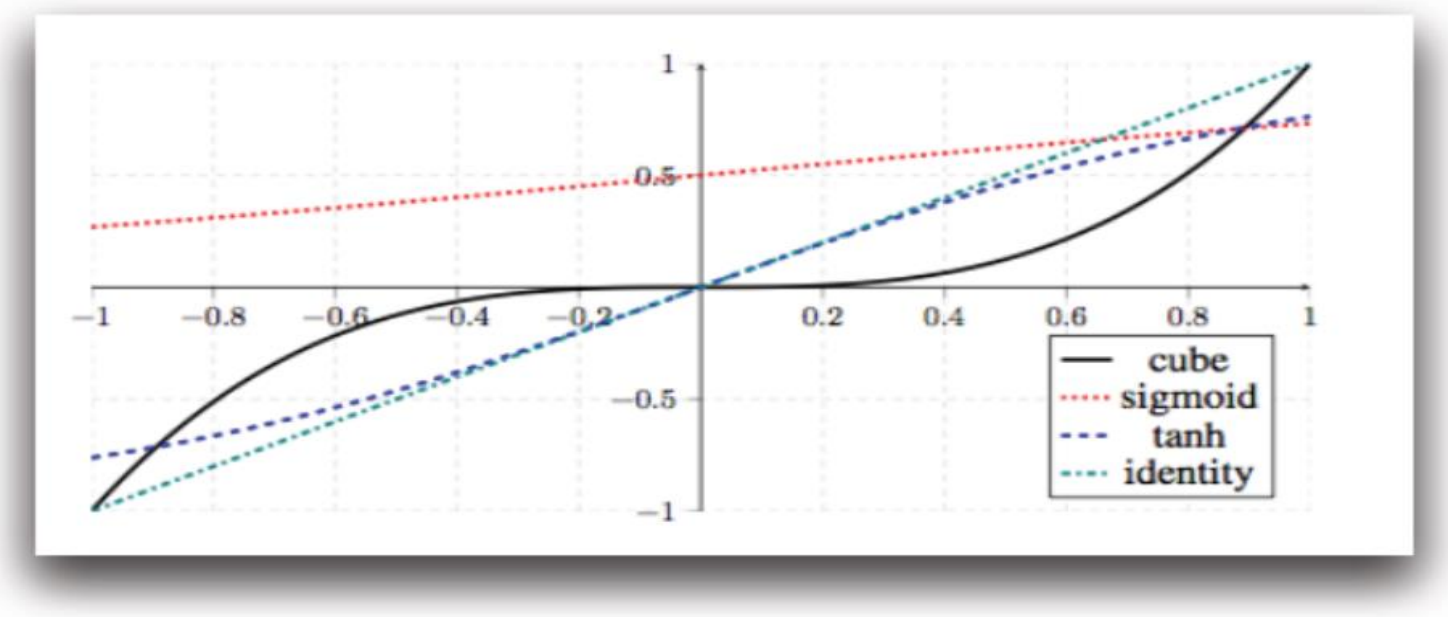
输入层特征

- 根据当前状态确定输入特征



3种向量维数相等
总共48个向量拼接

隐藏层激活函数：立方激活函数



$$g(w_1x_1 + \ldots + w_mx_m + b) =$$
$$\sum_{i,j,k}(w_iw_jw_k)x_ix_jx_k + \sum_{i,j} b(w_iw_j)x_ix_j \ldots$$

x_i, x_j, x_k可以来自三类特征中的任意一种，使用立方激活函数可以对它们进行建模，更好地捕捉不同种特征之间的相互作用

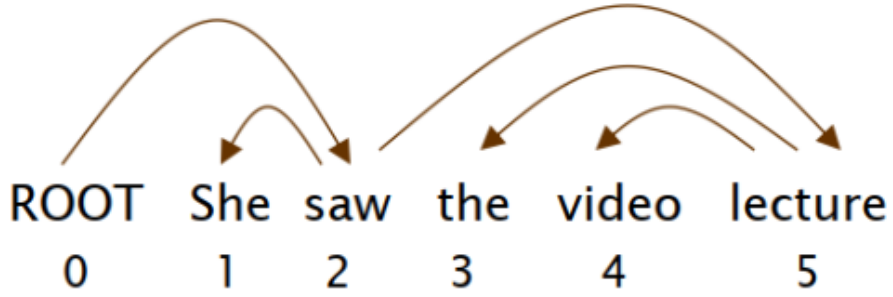
神经模型训练

- 训练样本选择：本项工作从训练文本和真实(gold)语法解析树中生成训练样本 {(c, t)}, 其中c是configuration，t是transition.并且遵循一种规则， 要让stack尽量短.
- **训练目标/损失函数: 交叉熵Cross entropy loss**
- 所有向量均使用**反向传播**计算梯度
- **优化器**: 小批量(mini-batch)的AdaGrad
- **初始化**:
  - 词语向量采用预训练的w2v向量. 50维
  - 其他向量(词性、依存关系标签) 随机初始化 (首次使用)

依存分析的评估

UAS: unlabeled 无标记依存准确率

LAS: labeled 有标记依存准确率



$$\text{Acc} = \frac{\text{\# correct deps}}{\text{\# of deps}}$$

UAS = 4 / 5 = 80%

LAS = 2 / 5 = 40%

Gold			
1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

Parsed			
1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

其他：

（1）依存准确率：中心词预测正确的非根节点词语个数 /总非根节点词数

（2）根准确率：正确根节点的个数/句子个数

（3）完全匹配率：无标记依存结构完全正确的句子/句子总数

传统特征与稠密特征的比较

- 问题 #1: 向量稀疏性

  indicator特征是稀疏向量，而本工作使用的是稠密的分布式向量，能更好地表达词语的语义相似度

- 问题 #2: 特征完整性

  indicator特征是手工整理的，特征之间可能需要进行组合，而人工枚举特征非常可能不完整。神经网络方法使用一个立方激活函数，可以自动地对不同类别、类别内的特征进行组合。

- 问题 #3: 计算代价

  人工整理特征费时费力，必须对词语、词性标签或语法关系标签进行拼接来生成特征字符串，并在包含数百万特征的巨大表格中查找它们。神经网络方法，只需要做一些矩阵操作。

实验1要求

- 基本要求

  完成依存分析模型的构建，包括整体架构、每一层、特别是48个特征的获取

- 进阶要求

  完成训练数据的构造、损失函数、模型训练过程、测试过程

  可以参考http://fancyerii.github.io/books/nndepparser/

后续工作

- A Neural Probabilistic Structured-Prediction Model for Transition-Based Dependency Parsing
- ACL 2015 (NLP创新的一个方式就是对已有工作的不足进行改进)
- 为什么需要改进? 贪心策略的缺点：每步选择得分最高的操作，得到一个局部最优，不能保证全局最优；无法修正，错误传递。

⇩

提出结构化的神经概率依存分析框架，最大化**整个动作序列**的概率

⇩

修改 解码算法 和 训练目标

还有其他优化思路吗？

解码算法

- 编码encoding: 把输入文本序列用一个固定向量来进行表示
- 解码decoding: 把固定向量转化为输出序列。本任务中解码的目标是给定输入x，找到全局上分数最高的动作序列

- 贪心策略修改为beam search束搜索策略，每一步骤保留k个分数最高的预测，取得的总体分数能够更好，效果接近于exact inference精确推断

精确推理是概率图模型的一类推断算法，希望能计算出条件分布的准确值。

精确推理的计算复杂度会随着图的规模增加发生指数性的增长；且只能用于无环的简单图中，使用范围有限。

句子级概率计算

- 避免局部最优：直接对整个动作序列的概率分布进行建模.
- 给定一个句子x和神经网络参数，第i个动作序列y_i的概率由softmax函数根据所有动作序列的分数给出:

序列y_i的概率：$$p(y_i \mid x, \theta) = \frac{e^{f(x,\theta)_i}}{\sum_{y_j \in \mathrm{GEN}(x)} e^{f(x,\theta)_j}}$$

GEN(x): 所有可能的序列

序列y_i的分数f：包含的所有动作a_k的分数之和

$$f(x, \theta)_i = \sum_{a_k \in y_i} o(x, y_i, k, a_k)$$

a_k：第k步的转移动作

o: 神经网络输出

模型结构和前一个工作相同

训练目标/损失

- 句子级的负对数似然损失:

$$L(\theta) = -\sum_{(x_i, y_i) \in (X,Y)} \log p(y_i \mid x_i, \theta)$$

$$= -\sum_{(x_i, y_i) \in (X,Y)} \log \frac{e^{f(x_i,\theta)_i}}{Z(x_i, \theta)}$$

$$= \sum_{(x_i, y_i) \in (X,Y)} \log Z(x_i, \theta) - f(x_i, \theta)_i$$

$$Z(x, \theta) = \sum_{y_j \in \mathrm{GEN}(x)} e^{f(x,\,\theta)_j}$$

Z包含了一个样本所有可能的预测序列
减少搜索范围：beam search

Z(x, θ)的计算

- 对比学习：给观测到的数据分配一个较大的概率值，给噪声数据分配一个较小的概率值。
- 本任务中，给gold动作序列更大的概率值，给束(beam)中的错误序列更小的概率值。本任务中的对比学习属于监督对比学习。

- 这样，我们只需要对比gold动作序列和beam中的错误序列(噪声序列)，而不需要对比全部的序列。beam选中的序列都是分数比较高的，所以我们希望模型可以区分正确答案，和得分高的错误答案。

新的训练目标

- 采用了对比学习的训练目标:

$$L'(\theta) = -\sum_{(x_i, y_i) \in (X,Y)} \log p'(y_i \mid x_i, \theta)$$

$$= -\sum_{(x_i, y_i) \in (X,Y)} \log \frac{e^{f(x_i,\theta)_i}}{Z'(x_i, \theta)}$$

$$= \sum_{(x_i, y_i) \in (X,Y)} \log Z'(x_i, \theta) - f(x_i, \theta)_i$$

$$Z'(x, \theta) = \sum_{y_j \in \mathrm{BEAM}(x)} e^{f(x,\,\theta)_j}$$

Z'选择的范围是BEAM 一个范围有限的束

```
Input: training examples (X, Y)
Output: θ
θ ← pretrained embedding
for i ← 1 to N do
    x, y = RANDOMSAMPLE(X, Y)
    δ = 0
    foreach x_j, y_j ∈ x, y do
        beam = φ
        goldState = null
        terminate = false
        beamGold = true
        while beamGold and not terminate
        do
            beam = DECODE(beam, x_j, y_j)
            goldState =
            GOLDMOVE(goldState, x_j, y_j)
            if not ISGOLD(beam) then
                beamGold = false
            if ITEMSCOMPLETE(beam) then
                terminate = true;
        δ = δ + UPDATE(goldState, beam)
    θ = θ + δ
```

搜索和学习集成在一个统一的框架中

随机采样一些训练集数据

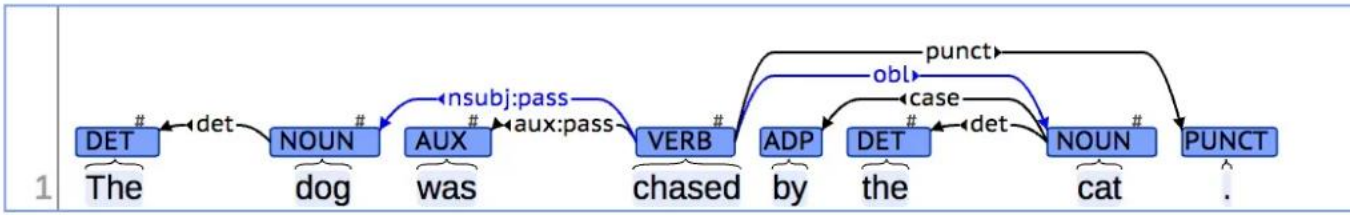
每个样本都有一个gold序列

循环里decode得到k个预测序列。如果beam里的序列有gold，就把其他预测当成负样本，内部循环终止，进行参数更新；否则重新解码，直到满足内部循环的终止条件

Mini-batched AdaGrad

依存分析数据资源

- 句法分析语料库也称为句法树库，包含大规模句子以及其对应句法树的集合。
- Universal Dependencies treebanks
- 一个在100多种不同人类语言中对语法（词性、形态特征和句法依赖性）进行一致注释的框架



依存分析数据资源

- 句法分析语料库也称为句法树库，包含大规模句子以及其对应句法树的集合。

语料库名称	单词数量	语法类型	语言
英语宾州树库（PTB）	117 万	成分语法	英文
通用依存树库（UD V2.0 CoNLL 2017）	281 万	依存语法	多语言
通用依存树库（UD V2.2 CoNLL 2018）	1714 万	依存语法	多语言
组合范畴语法树库（CCGBank）	116 万	组合范畴语法	英文
中文宾州树库 6.0（CTB 6.0）	78 万	成分语法	中文
中文宾州树库 7.0（CTB 7.0）	120 万	成分语法	中文
中文宾州树库 8.0（CTB 8.0）	162 万	成分语法	中文
中文宾州树库 9.0（CTB 9.0）	208 万	成分语法	中文
中文语义依存树库（SDP）	52 万	语义依存	中文

依存分析数据资源

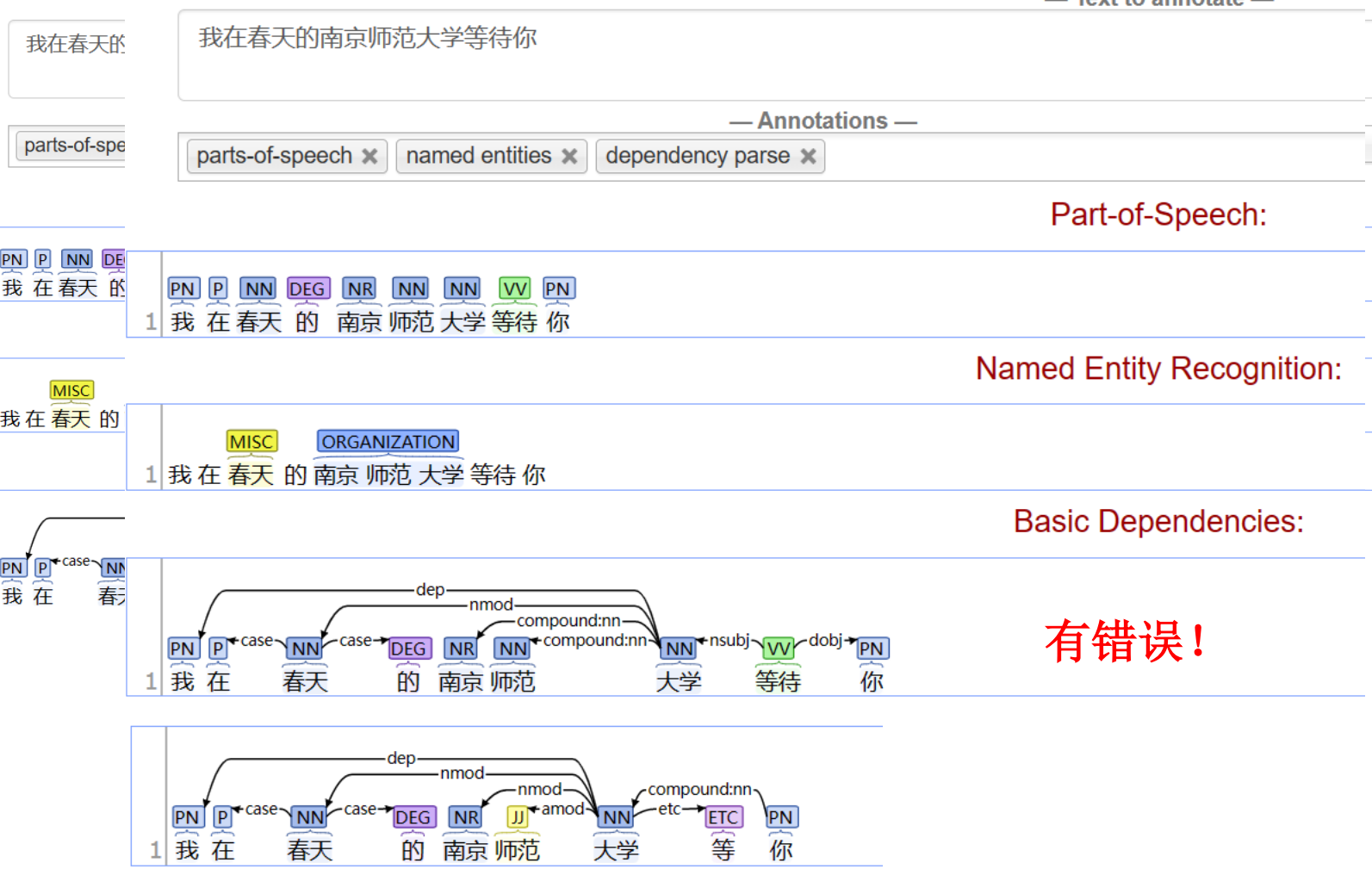
宾州树库的数据组织结构（中英文相同）

```
( (IP (NP-SBJ (DNP (NP-PN (NR 北海市))
                   (DEG 的))
              (NP (NN 崛起)))
      (PU ，)
      (VP (VC 是)
          (NP-PRD (CP-APP (IP (IP-SBJ (LCP-TMP (NP (NT 近年))
                                               (LC 来))
                                      (NP-PN-SBJ (NR 广西)
                                                 (NN 壮族)
                                                 (NN 自治区))
                                      (VP (PP-DIR (P 对)
                                                  (NP (NN 外)))
                                          (VP (VV 开放))))
                              (VP (VV 取得)
                                  (NP-OBJ (ADJP (JJ 卓著))
                                          (NP (NN 成就)))))
                          (DEC 的))
                  (ADJP (JJ 重要))
                  (NP (NN 标志)
                      (NN 之一))))
      (PU 。)) )
```

依存分析工具

- Stanford Parser

网页版：https://corenlp.run　　Python库：stanfordcorenlp



相关测评任务

- SemEval
- 子任务-中文语义依存评测

数据格式：conll　　dependent　　head

示例：

1	城建	城建	NN	NN	_	2	Exp	_	_
2	成为	成为	VV	VV	_	0	Root	_	_
3	外商	外商	NN	NN	_	4	Agt	_	_
4	投资	投资	VV	VV	_	7	dDesc	_	_
5	青海	青海	NR	NR	_	4	Datv	_	_
6	新	新	JJ	JJ	_	7	Desc	_	_
7	热点	热点	NN	NN	_	2	Clas	_	_

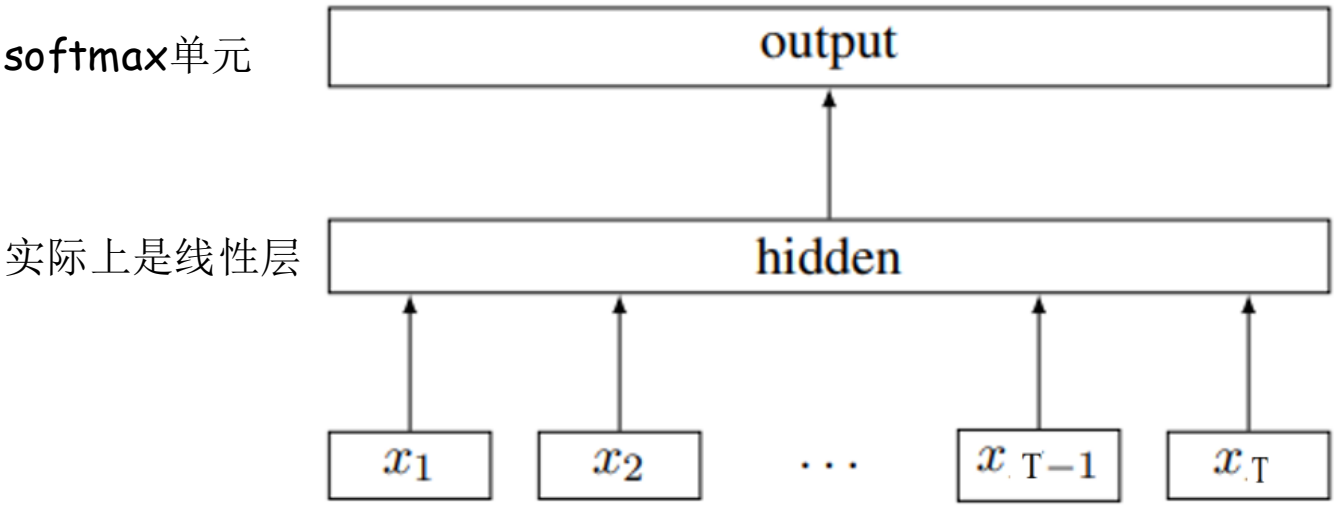
结构：中文语义依存树库。从语义角度构建依存关系，定义了 45 个标签用来描述论元 (Argument) 之间的语义关系，19 个标签用来描述谓词 (Predicate) 之间的关系，以及 17 个标签用来提供谓词描述。

案例二：FNN for 文本分类

- Bag of Tricks for Efficient Text Classification
- 2016. PDF
- 命名fastText

单位：Facebook, 现在的Meta

fast: 在使用标准多核CPU的情况下10分钟内处理超过10亿个词汇

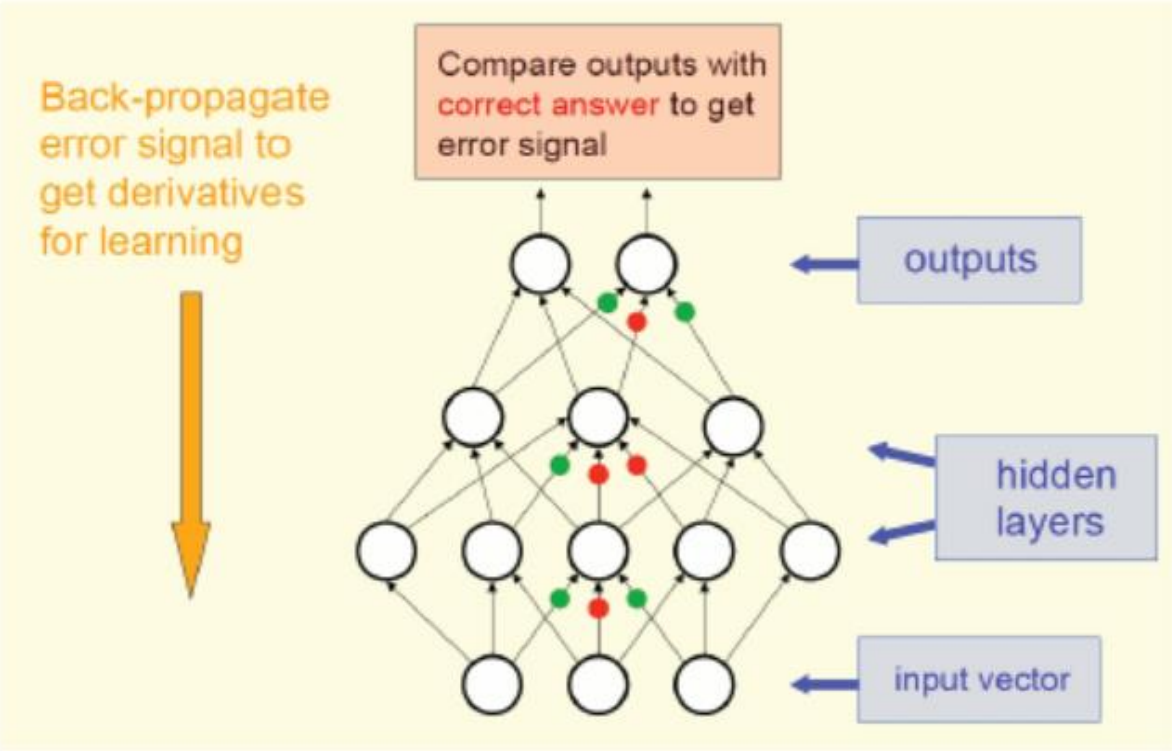


输入层：N元语法特征。
实际上是一个句子所有词的词向量累加
不考虑词序：词袋(bag of words, bow)

损失：$-\frac{1}{N}\sum_{i=1}^{N} y_i \log(f(BAx_i))$

更好的特征？ 更好的模型设计？

FNN复习



(1) 前馈计算每一层的净输入和激活值， 直到最后一层；

(2) 反向传播计算每一层的误差项$\delta^{(l)}$，计算每一层参数的偏导数（链式法则+动态规划）；

(3) 根据优化器算法更新参数