Review of constituent parsing by Qiao

Monday, August 31, 2020 10:11 AM

句法分析——成分分析：

By Qiao for NLP7 2020-09-06

Core reference: Speech and Language Processing 3rd (Chapter 12 - 14)

CS224N: Constituency Parsing and Tree Recursive Neural Networks

1. CFG基本认识
2. CKY
3. PCFG和Probabilistic CKY
4. Neural Parsing

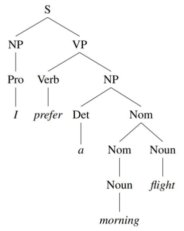
基本认识

成分(Constituency):

句子中相对固定的单词集合，它们组织在一起实现特定的语法功能

上下文无关语法 Context-Free Grammar (CFG)

也被称作术语结构语法 Phrase-Structure Grammars , 等价于 Backus-Naur Form (BNF)。它是一种生成语法(generative grammar)，因为它能定义的语言就是根据它的规则所能生成的所有的句子的集合。



一个CFG包括：

1. 符号：具体的词汇（terminals）和抽象的成分符号（non-terminals）
2. 一套产生规则，规定了符号是如何组合的

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid the \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Figure 13.1 The $\mathscr{L}_1$ miniature English grammar and lexicon.

CFG的形式化定义：

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$, where A is a non-terminal, β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$
S	a designated **start symbol** and a member of N

CFG两种用途：

1. 生成句子，也就是从开始符号S开始，根据产生规则不断改写符号最终至只剩下词汇的过程。
2. 给定句子，解析出句子的语法结构。

定义完一个CFG之后，例如Figure13.1中的L1，所有可以由这个语法生成的句子是和语法的句子，其他的句子则对于L1来说是不符合语法(ungrammatical)的句子。

两个CFG可生成的合法句子集合若相等，则它们是弱等价的。在此之上，若对于每一个句子，它们的句法树都相同，则它们是强等价的。

Parsing with CKY 算法

前置操作：

将CFG转换为 Chomsky Normal Form (CNF), 一种弱等价的语法形式，不会损失原语法的表达能力。

CNF中Production Rule的特点，非终止符号只能被转化为两个非终止符号或一个终止符号：

A → B C or A → w

转化规则：

情形一：

Non-terminal和terminal符号混合的情形。

$$INF\text{-}VP \rightarrow to\ VP \quad\Rightarrow\quad INF\text{-}VP \rightarrow TO\ VP,\quad TO \rightarrow to$$

情形二：

Non-terminal链。

$$A \overset{*}{\Rightarrow} B,\quad B \rightarrow \gamma \quad\Rightarrow\quad A \rightarrow \gamma$$

情形三：

右边大于两个symbol, 重复执行下述转化直到每个rule右边只剩下两个symbol.

$$A \rightarrow B\ C\ \gamma \quad\Rightarrow\quad A \rightarrow X1\ \gamma,\quad X1 \rightarrow B\ C$$

$\mathscr{L}_1$ Grammar	$\mathscr{L}_1$ in CNF
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

Figure 13.3 $\mathscr{L}_1$ Grammar and its conversion to CNF. Note that although they aren't shown here, all the original lexical entries from $\mathscr{L}_1$ carry over unchanged as well.

CKY算法思路（动态规划）

- DP: $(n+1) \times (n+1)$的矩阵，DP[i, j]储存在原句半开半闭区间[i, j)内部分的所有成分结构的组合，于是DP[0, n]储存的就是原句的所有可能的解析树。(DP实际只使用对角线以上的上三角区域)
- 转移方程：D[i, j] -> DP[i, k] DP[k, j]
  - 对于一个non-terminal成分来说，它要么转换为一个单词（初始化时处理），要么转换为两个子non-terminal成分，所以用k来遍历查找所有的左右子成分的分割点。D[i, k]在第j行上，D[k, j]在第i列上。
  - 如果DP[i, k] 和 DP[k, j] 都非空，则对两者做笛卡尔积。同时搜索语法，得到DP[i, j]给定k时的成分集合 $\{x \mid x \rightarrow DP[i,k]DP[k,j]\}$。将所有k的搜索结果求并集便得到DP[i, j]。
- 初始化： DP[i-1, i]的位置初始化为可以生成该位置单词的所有non-terminal集合
- 计算方向：bottom-up, 与对角线平行的方向由长向短计算（确保计算DP[i, j]时，所有DP[i, k], DP[k, j]已经计算完毕）。或从左到右从下到上；从下到上，从左到右；

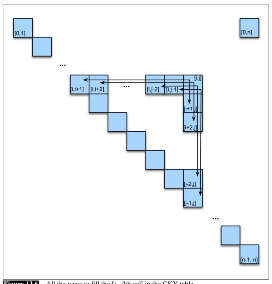
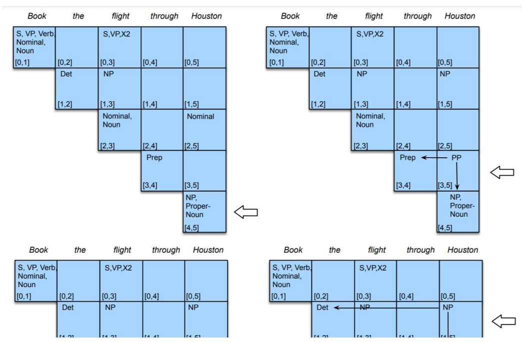


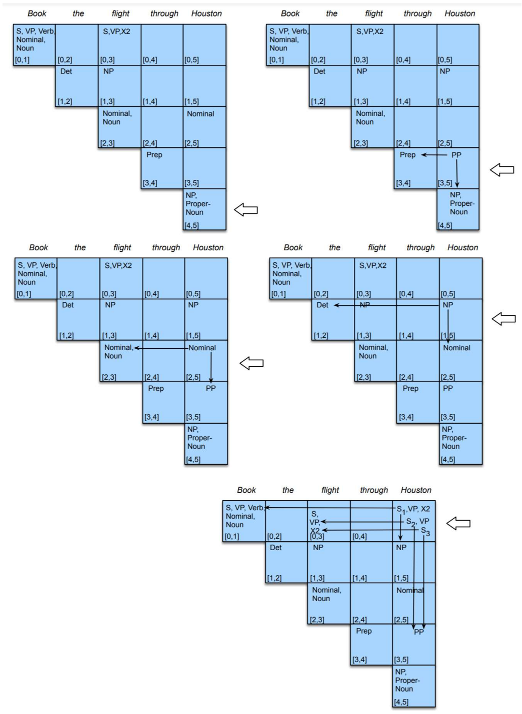
Figure 13.6 All the ways to fill the [i, j]th cell in the CKY table.




```
function CKY-PARSE(words, grammar) returns table

  for j ← from 1 to LENGTH(words) do
    for all {A | A → words[j] ∈ grammar}
        table[j−1, j] ← table[j−1, j] ∪ A
    for i ← from j−2 downto 0 do
      for k ← i+1 to j−1 do
        for all {A | A → BC ∈ grammar and B ∈ table[i,k] and C ∈ table[k, j]}
            table[i,j] ← table[i,j] ∪ A
```

Figure 13.5 The CKY algorithm.



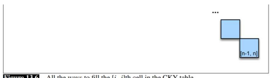


Figure 13.6 All the ways to fill the $[i, j]$th cell in the CKY table.

function CKY-PARSE(*words, grammar*) **returns** *table*

  for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
    for all $\{A \mid A \rightarrow words[j] \in grammar\}$
      $table[j-1, j] \leftarrow table[j-1, j] \cup A$
    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
      for $k \leftarrow i+1$ **to** $j-1$ **do**
        for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
          $table[i,j] \leftarrow table[i,j] \cup A$

Figure 13.5 The CKY algorithm.

浅层语法分析:

Chunking
- 可看作一个序列标注任务(IOB序列)
- Chunked texts 无成分句法分析的层级结构

$[_{NP}$ The morning flight$]$ $[_{PP}$ from$]$ $[_{NP}$ Denver$]$ $[_{VP}$ has arrived

(13.7) *The morning flight from Denver has arrived*
B_NP I_NP I_NP B_PP B_NP B_VP I_VP

(13.8) *The morning flight from Denver has arrived.*
B_NP I_NP I_NP O B_NP O O

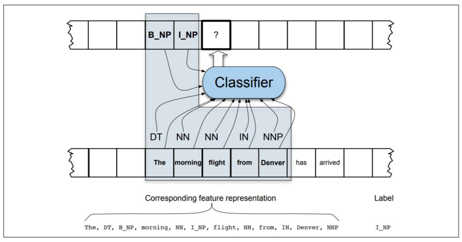


Figure 13.8 A sequence model for chunking. The chunker slides a context window over the sentence, classifying words as it proceeds. At this point, the classifier is attempting to label *flight*, using features like words, embeddings, part-of-speech tags and previously assigned chunk tags.

消歧: Probabilistic CFG (PCFG)

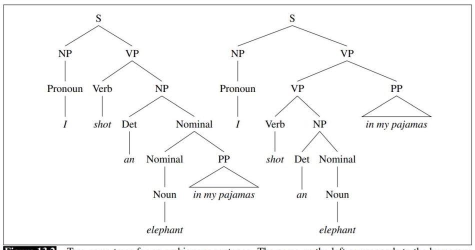


Figure 13.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the humorous reading in which the elephant is in the pajamas, the parse on the right corresponds to the reading in which Captain Spaulding did the shooting in his pajamas.

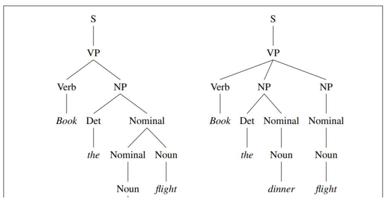
形式化定义

N a set of **non-terminal symbols** (or **variables**)
Σ a set of **terminal symbols** (disjoint from N)
R a set of **rules** or productions, each of the form $A \rightarrow \beta$ $[p]$,
where A is a non-terminal,
β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$,
and p is a number between 0 and 1 expressing $P(\beta|A)$
S a designated **start symbol**

一棵解析树的概率是从s符号开始，到生成完整句子所使用到的所有规则的概率的乘积。这里我们求的实际上是P(T, S)，但由于给定S的一棵固定的解

例子

Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.10] | a [.30] | the [.60]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$ [.10] | $flight$ [.30]
$S \rightarrow VP$	[.05]	| $meal$ [.05] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flight$ [.40] | $dinner$ [.10]
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow book$ [.30] | $include$ [.30]
$NP \rightarrow Det\ Nominal$	[.20]	| $prefer$ [.40]
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I$ [.40] | she [.05]
$Nominal \rightarrow Noun$	[.75]	| me [.15] | you [.40]
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow Houston$ [.60]
$Nominal \rightarrow Nominal\ PP$	[.05]	| NWA [.40]
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does$ [.60] | can [.40]
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow from$ [.30] | to [.30]



	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$, and p is a number between 0 and 1 expressing $P(\beta\|A)$
S	a designated **start symbol**

一棵解析树的概率是从s符号开始，到生成完整句子所使用到的所有规则的概率的乘积。这里我们求的实际上是P(T, S)，但由于给定S的一棵固定的解析树，我们用树中的规则只能确定地生成一个句子S, 所以P(S|T) = 1, 所以P(T) = P(T, S)/ P(S |T) = P(T, S)

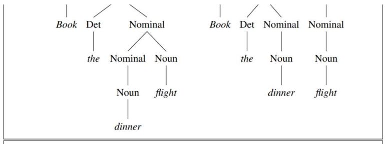
$$P(T,S) = \prod_{i=1}^{n} P(RHS_i|LHS_i)$$

$$\hat{T}(S) = \underset{T s.t. S=\text{yield}(T)}{\text{argmax}} P(T,S)$$

$$\hat{T}(S) = \underset{T s.t. S=\text{yield}(T)}{\text{argmax}} P(T)$$

$NP \rightarrow Proper\text{-}Noun$	[.30]		$Verb \rightarrow book$ [.30] | $include$ [.30]
$NP \rightarrow Det\ Nominal$	[.20]		| $prefer$ [.40]
$NP \rightarrow Nominal$	[.15]		$Pronoun \rightarrow I$ [.40] | she [.05]
$Nominal \rightarrow Noun$	[.75]		| me [.15] | you [.40]
$Nominal \rightarrow Nominal\ Noun$	[.20]		$Proper\text{-}Noun \rightarrow Houston$ [.60]
$Nominal \rightarrow Nominal\ PP$	[.05]		| NWA [.40]
$VP \rightarrow Verb$	[.35]		$Aux \rightarrow does$ [.60] | can [.40]
$VP \rightarrow Verb\ NP$	[.20]		$Preposition \rightarrow from$ [.30] | to [.30]
$VP \rightarrow Verb\ NP\ PP$	[.10]		| on [.20] | $near$ [.15]
$VP \rightarrow Verb\ PP$	[.15]		| $through$ [.05]
$VP \rightarrow Verb\ NP\ NP$	[.05]		
$VP \rightarrow VP\ PP$	[.15]		
$PP \rightarrow Preposition\ NP$	[1.0]		

$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$
$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$



Rules			P	Rules			P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flight	.40	Noun	→	flight	.40

Figure 14.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the sensible meaning "Book a flight that serves dinner", while the parse on the right corresponds to the nonsensical meaning "Book a flight on behalf of 'the dinner' ".

Probabilistic CKY算法：找到概率最大的解析树

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
                                                   and its probability
  for j ← from 1 to LENGTH(words) do
     for all { A | A → words[j] ∈ grammar}
        table[j−1, j,A] ← P(A → words[j])
     for i ← from j−2 downto 0 do
        for k ← i+1 to j−1 do
           for all { A | A → BC ∈ grammar,
                      and table[i,k,B] > 0 and table[k, j,C] > 0 }
              if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
                 table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
                 back[i,j,A] ← {k,B,C}
     return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Figure 14.3 The probabilistic CKY algorithm for finding the maximum probability parse of a string of *num_words* words given a PCFG grammar with *num_rules* rules in Chomsky normal form. *back* is an array of backpointers used to recover the best parse. The *build_tree* function is left as an exercise to the reader.

S	$\rightarrow$	$NP\ VP$	.80	Det	$\rightarrow$	the	.40
NP	$\rightarrow$	$Det\ N$	.30	Det	$\rightarrow$	a	.40
VP	$\rightarrow$	$V\ NP$	.20	N	$\rightarrow$	$meal$	.01
V	$\rightarrow$	$includes$	.05	N	$\rightarrow$	$flight$	.02

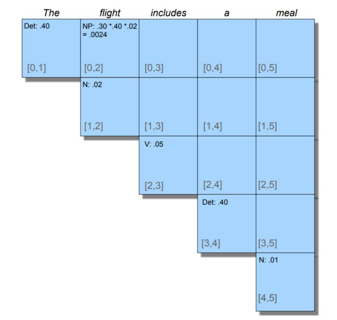


Figure 14.4 The beginning of the probabilistic CKY matrix. Filling out the rest of the chart is left as Exercise 14.4 for the reader.

一些讨论

- Rule概率的获取
  - 基于树库的统计
  - 利用inside-outside算法估计得到
    - EM算法
- PCFG的缺陷
  - 上下文无关/条件独立性的假设
    - 一条规则的转换X-> AB，X->a不考虑上下文/句法树其他部分的结构(所以我们可以用连乘的方式得到树的概率)。
    - 以英语为例，实际上其他部分的结构会影响当前成分的扩展。
      例如 a. NP->PRP，b. NP->DT NN，当NP为句子主语时，a. 更常见；当NP为句子宾语时，b. 更常见。

	Pronoun	Non-Pronoun
Subject	91%	9%
Object	34%	66%

      然而，PCFG无法利用这种依赖关系。 假如我们基于树库统计得到两条规则的概率，在Switchboard corpus,中，它们的概率几乎相等NP → DT NN (.28), NP → PRP (.25)。所以PCFG中我们无法利用主语/宾语的上下文信息来扩展NP，而是几乎等可能地将NP转换为DT NN 或 PRP。
  - 成分的构建没有利用词汇信息
    - CFG无法刻画基于词的语法信息，例如介词短语的附着(PP)、动词的subcategorization（某些动词可接特定的语法形式，而其他动词不可，例如及物与不及物动词，think vs visualize），以及并列结构。
    - PCFG无法很好处理这些问题。例如对于PP问题来说，图14.5给出了两个句法分析结果。左边时正确的树。两者用到的规则除了 a. VP → VBD NP PP和 b. VP → VBD NP, NP → NP PP 之外都相同。一个PCFG在概率设定完之后必定会给a.或 b.其中一个更高的概率，如果给a.更高的概率，那么在图14.5当中我们可以得到正确的结果。但解析如下的句子则会出错：*(14.19) fishermen caught tons of herring* （这个句子应当用b.解析）。反过来，我们得不到图14.5中的正确结果。

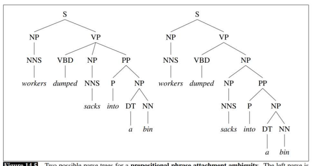


Figure 14.5 Two possible parse trees for a **prepositional phrase attachment ambiguity**. The left parse is the sensible one, in which "into a bin" describes the resulting location of the sacks. In the right incorrect parse, the sacks to be dumped are the ones which are already "into a bin", whatever that might mean.

- 一些改进方案
  - 上下文依赖的non-terminal符号
    - 如图14.8所示，主语的信息现在可以传递到NP的生成过程
  - Probabilistic Lexicalized CFGs
    - 如图14.10所示，动词短语VP（dumped, VBD）可以利用动词的信息。

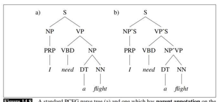
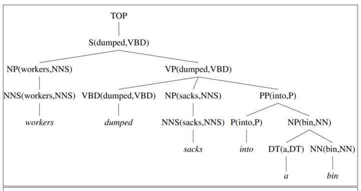


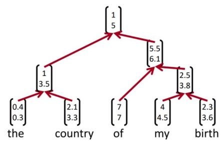
Figure 14.8 A standard PCFG parse tree (a) and one which has **parent annotation** on the nodes which aren't pre-terminal (b). All the non-terminal nodes (except the pre-terminal part-of-speech nodes) in parse (b) have been annotated with the identity of their parent.

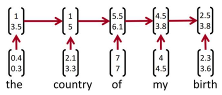


Internal Rules					Lexical Rules		
TOP	→	S(dumped,VBD)			NNS(workers,NNS)	→	workers
S(dumped,VBD)	→	NP(workers,NNS)	VP(dumped,VBD)		VBD(dumped,VBD)	→	dumped
NP(workers,NNS)	→	NNS(workers,NNS)			NNS(sacks,NNS)	→	sacks
VP(dumped,VBD)	→	VBD(dumped, VBD)	NP(sacks,NNS)	PP(into,P)	P(into,P)	→	into
PP(into,P)	→	P(into,P)	NP(bin,NN)		DT(a,DT)	→	a
NP(bin,NN)	→	DT(a,DT)	NN(bin,NN)		NN(bin,NN)	→	bin

Figure 14.10 A lexicalized tree, including head tags, for a WSJ sentence, adapted from Collins (1999). Below we show the PCFG rules needed for this parse tree, internal rules on the left, and lexical rules on the right.

成分分析的经典神经网络模型:
Recursive Neural Networks



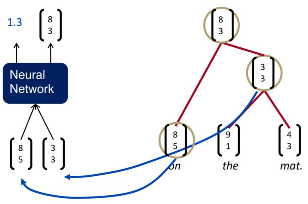


组合（Composition）

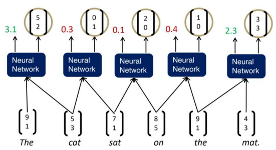
score $= U^{\mathsf{T}}p$

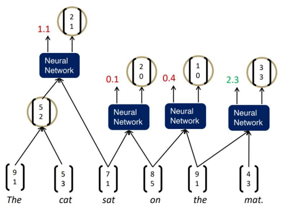
$$\boxed{p = \tanh\left(W\begin{bmatrix}c_1\\c_2\end{bmatrix} + b\right),}$$

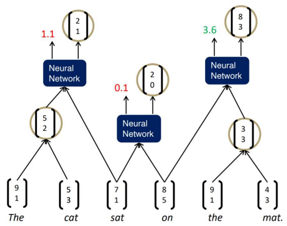
Same *W* parameters at all nodes of the tree



贪心解析(greedily Parsing)







训练:

数据：句子+解析树

解析树的分数计算：

$$s(x,y) = \sum_{n \in nodes(y)} s_n$$

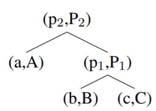
$$J = \sum_i s(x_i, y_i) - \max_{y \in A(x_i)} \left(s(x_i, y) + \Delta(y, y_i)\right)$$

$$\Delta(y, y_i) = \sum_{d \in T(y)} \lambda \mathbf{1}\{d \notin T(y_i)\}$$

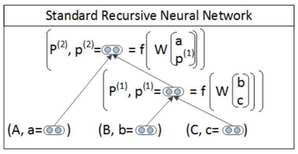
Notes:

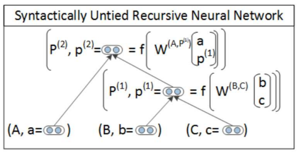
- 结构损失的计算方法：
  - 以span为单位。span是句子中由树上一个节点支配的最左到最右叶子节点的片段，
  - 与正确树的每个节点（单词）的span比较，统计不相同的span数
- 结构损失 Δ 用于惩罚与正确树的结构差异，结构相差越大代价越高。
- A(x)的结构搜索可以使用贪心的策略，或者可用Beam Search方法
- 对整体目标函数J的优化可以最大化正确树的得分并且最小化错误树的得分（通过直接最小化错误树中分数最大的那棵树实现）。

组合方法的演进

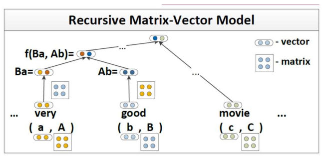


RNTN: Recursive Neural Tensor Network





$$h^{(1)} = \tanh(W^{(1)} \begin{bmatrix} h^{(1)}_{Left} \\ h^{(1)}_{Right} \end{bmatrix} + b^{(1)})$$



$$p = f\left(W \begin{bmatrix} Ba \\ Ab \end{bmatrix}\right)$$

$$P = g(A, B) = W_M \begin{bmatrix} A \\ B \end{bmatrix}$$

$$W_M \in \mathbb{R}^{n \times 2n}$$

MV-RNN (Socher et al., 2012) 的主要思想是将词和成分表示为向量+矩阵的形式。在组合时，两个子成分首先利用各自的矩阵对彼此的向量做投影，这么做可以使得子成分参与到组合函数f()的运算中（对比第一个组合函数的运算）。

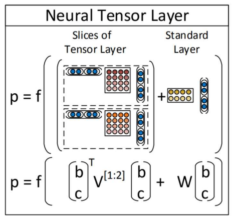


Figure 5: A single layer of the Recursive Neural Tensor Network. Each dashed box represents one of d-many slices and can capture a type of influence a child can have on its parent.

$$h^{(1)} = tanh(x^T V x + W x)$$

使得组合运算时向量间既能有加法也能有乘法交互（前三种方法只有加法交互）