Probabilistic Context-Free Grammars

Michael Collins, Columbia University

上下文无关的语法可以很容易的推导出一个句子的语法结构，但是缺点是推导出的结构可能存在二义性。

Overview

- Probabilistic Context-Free Grammars (PCFGs)
- The CKY Algorithm for parsing with PCFGs

A Probabilistic Context-Free Grammar (PCFG)

S	$\Rightarrow$	NP	VP	1.0
VP	$\Rightarrow$	Vi		0.4
VP	$\Rightarrow$	Vt	NP	0.4
VP	$\Rightarrow$	VP	PP	0.2
NP	$\Rightarrow$	DT	NN	0.3
NP	$\Rightarrow$	NP	PP	0.7
PP	$\Rightarrow$	P	NP	1.0

Vi	$\Rightarrow$	sleeps	1.0
Vt	$\Rightarrow$	saw	1.0
NN	$\Rightarrow$	man	0.7
NN	$\Rightarrow$	woman	0.2
NN	$\Rightarrow$	telescope	0.1
DT	$\Rightarrow$	the	1.0
IN	$\Rightarrow$	with	0.5
IN	$\Rightarrow$	in	0.5

- Probability of a tree t with rules

$$\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n$$

is $p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$ where $q(\alpha \rightarrow \beta)$ is the probability for rule $\alpha \rightarrow \beta$.

DERIVATION	RULES USED	PROBABILITY
S		

DERIVATION	RULES USED	PROBABILITY
S		1.0
	S $\rightarrow$ NP VP	
NP VP		

DERIVATION	RULES USED	PROBABILITY
S		1.0
	S $\rightarrow$ NP VP	
NP VP		0.3
	NP $\rightarrow$ DT NN	
DT NN VP		

DERIVATION	RULES USED	PROBABILITY
S		
NP VP	$S \rightarrow NP\ VP$	1.0
DT NN VP	$NP \rightarrow DT\ NN$	0.3
the NN VP	$DT \rightarrow$ the	1.0

DERIVATION	RULES USED	PROBABILITY
S		1.0
	$S \rightarrow NP\ VP$	
NP VP		0.3
	$NP \rightarrow DT\ NN$	
DT NN VP		1.0
	$DT \rightarrow$ the	
the NN VP		0.1
	$NN \rightarrow$ dog	
the dog VP		

DERIVATION	RULES USED	PROBABILITY
S	S $\rightarrow$ NP VP	1.0
NP VP	NP $\rightarrow$ DT NN	0.3
DT NN VP	DT $\rightarrow$ the	1.0
the NN VP	NN $\rightarrow$ dog	0.1
the dog VP	VP $\rightarrow$ Vi	0.4
the dog Vi		

DERIVATION	RULES USED	PROBABILITY
S	S $\rightarrow$ NP VP	1.0
NP VP	NP $\rightarrow$ DT NN	0.3
DT NN VP	DT $\rightarrow$ the	1.0
the NN VP	NN $\rightarrow$ dog	0.1
the dog VP	VP $\rightarrow$ Vi	0.4
the dog Vi	Vi $\rightarrow$ laughs	0.5
the dog laughs		

Properties of PCFGs

- Assigns a probability to each *left-most derivation*, or parse-tree, allowed by the underlying CFG

Properties of PCFGs

- Assigns a probability to each *left-most derivation*, or parse-tree, allowed by the underlying CFG
- Say we have a sentence s, set of derivations for that sentence is $\mathcal{T}(s)$. Then a PCFG assigns a probability $p(t)$ to each member of $\mathcal{T}(s)$. i.e., *we now have a ranking in order of probability*.

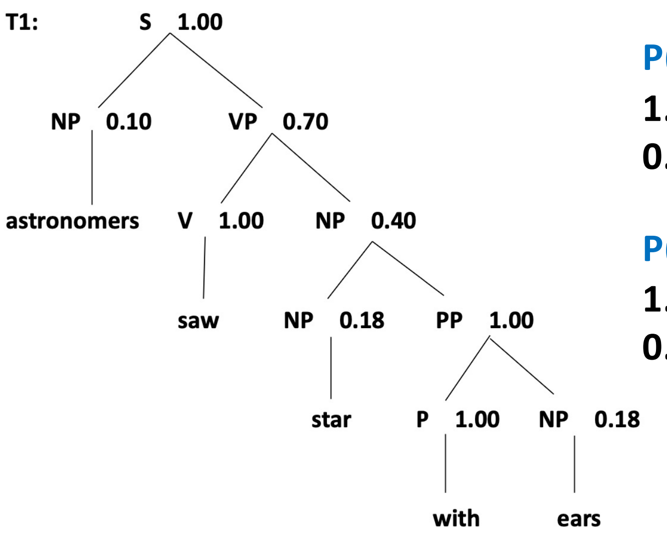
Properties of PCFGs

- Assigns a probability to each *left-most derivation*, or parse-tree, allowed by the underlying CFG
- Say we have a sentence s, set of derivations for that sentence is $\mathcal{T}(s)$. Then a PCFG assigns a probability $p(t)$ to each member of $\mathcal{T}(s)$. i.e., *we now have a ranking in order of probability*.
- The most likely parse tree for a sentence s is

$$\arg\max_{t \in \mathcal{T}(s)} p(t)$$

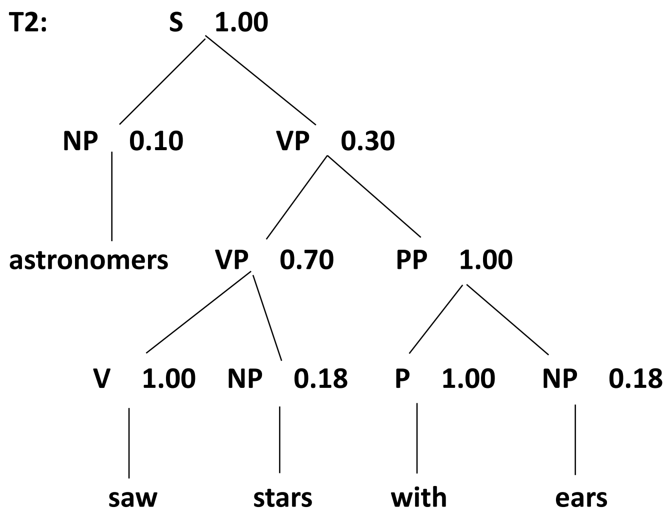
S→NP VP	1.0	NP→NP PP	0.4
PP→P NP	1.0	NP→astronomers	0.1
VP→V NP	0.7	NP→ears	0.18
VP→VP PP	0.3	NP→saw	0.04
P→with	1.0	NP→stars	0.18
V→saw	1.0	NP→telescope	0.1

给定句子 S : astronomers saw stars with ears ，得到两个句法树,



$P(T_1)$=S*NP*VP*V*NP*NP*PP*P*NP=
1.0*0.1*0.7*1.0*0.4*0.18*1.0*1.0*0.18=
0.0009072

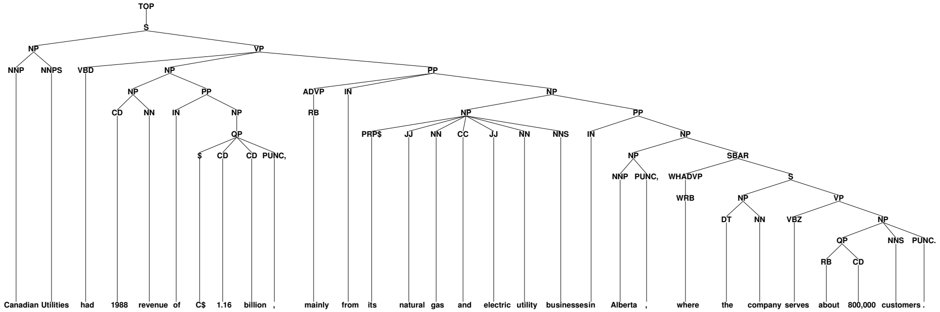
$P(T_2)$=S*NP*VP*VP*V*NP*PP*P*NP=
1.0*0.1*0.3*0.7*1.0*0.18*1.0*1.0*0.18=
0.0006804



Data for Parsing Experiments: Treebanks

- Penn WSJ Treebank = 50,000 sentences with associated trees
- Usual set-up: 40,000 training sentences, 2400 test sentences

An example tree:



Deriving a PCFG from a Treebank

- Given a set of example trees (a treebank), the underlying CFG can simply be **all rules seen in the corpus**
- Maximum Likelihood estimates:

$$q_{ML}(\alpha \to \beta) = \frac{\text{Count}(\alpha \to \beta)}{\text{Count}(\alpha)}$$

  where the counts are taken from a training set of example trees.
- **If the training data is generated by a PCFG**, then as the training data size goes to infinity, the maximum-likelihood PCFG will converge to the same distribution as the "true" PCFG.

PCFGs

Booth and Thompson (1973) showed that a CFG with rule probabilities correctly defines a distribution over the set of derivations provided that:

1. The rule probabilities define conditional distributions over the different ways of rewriting each non-terminal.
2. A technical condition on the rule probabilities ensuring that the probability of the derivation terminating in a finite number of steps is 1. (This condition is not really a practical concern.)

Parsing with a PCFG

- Given a PCFG and a sentence s, define $\mathcal{T}(s)$ to be the set of trees with s as the yield.

- Given a PCFG and a sentence s, how do we find

$$\arg\max_{t \in \mathcal{T}(s)} p(t)$$

Chomsky Normal Form

乔姆斯基范式

A context free grammar $G = (N, \Sigma, R, S)$ in Chomsky Normal Form is as follows

- N is a set of non-terminal symbols
- Σ is a set of terminal symbols
- R is a set of rules which take one of two forms:
  - $X \to Y_1 Y_2$ for $X \in N$, and $Y_1, Y_2 \in N$
  - $X \to Y$ for $X \in N$, and $Y \in \Sigma$
- $S \in N$ is a distinguished start symbol

推导树均可简化为二元形式，
这样可以采用二分法来分析自然语言，
采用二叉树来表示自然语言的句子结构。

一般CFL(上下文无关文法)都可转化为CNF

A Dynamic Programming Algorithm

- Given a PCFG and a sentence s, how do we find

$$\max_{t \in \mathcal{T}(s)} p(t)$$

- Notation:

$$n = \text{number of words in the sentence}$$
$$w_i = i\text{'th word in the sentence}$$
$$N = \text{the set of non-terminals in the grammar}$$
$$S = \text{the start symbol in the grammar}$$

- Define a dynamic programming table

$$\pi[i, j, X] = \text{maximum probability of a constituent with non-terminal } X \text{ spanning words } i \ldots j \text{ inclusive}$$

- Our goal is to calculate $\max_{t \in \mathcal{T}(s)} p(t) = \pi[1, n, S]$

An Example

the dog saw the man with the telescope

A Dynamic Programming Algorithm

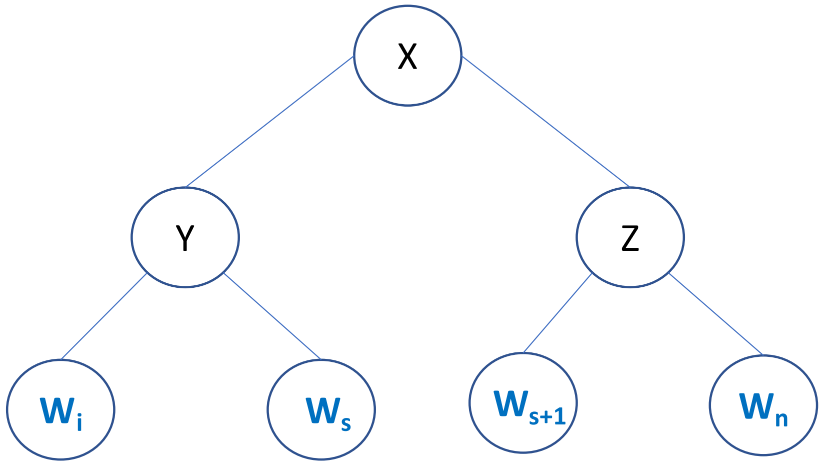
- Base case definition: for all $i = 1 \ldots n$, for $X \in N$

  $$\pi[i, i, X] = q(X \rightarrow w_i)$$

  (note: define $q(X \rightarrow w_i) = 0$ if $X \rightarrow w_i$ is not in the grammar)

- Recursive definition: for all $i = 1 \ldots n$, $j = (i + 1) \ldots n$, $X \in N$,

  $$\pi(i, j, X) = \max_{\substack{X \rightarrow Y Z \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow Y Z) \times \pi(i, s, Y) \times \pi(s + 1, j, Z)\right)$$


X
Y
Z
W_i
W_s
W_{s+1}
W_n

An Example

$$\pi(i,j,X) = \max_{\substack{X \to YZ \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \to YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

the dog saw the man with the telescope

The Full Dynamic Programming Algorithm

Input: a sentence $s = x_1 \ldots x_n$, a PCFG $G = (N, \Sigma, S, R, q)$.

Initialization:

For all $i \in \{1 \ldots n\}$, for all $X \in N$,

$$\pi(i, i, X) = \begin{cases} q(X \to x_i) & \text{if } X \to x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

Algorithm:

- For $l = 1 \ldots (n - 1)$ **l:长度** **w1,w2, ..., wl, ...,wn**
  - For $i = 1 \ldots (n - l)$
    - Set $j = i + l$
    - For all $X \in N$, calculate

      $$\pi(i, j, X) = \max_{\substack{X \to Y Z \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \to Y Z) \times \pi(i, s, Y) \times \pi(s + 1, j, Z) \right)$$

      and

      $$bp(i, j, X) = \arg \max_{\substack{X \to Y Z \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \to Y Z) \times \pi(i, s, Y) \times \pi(s + 1, j, Z) \right)$$

$S = w_1 \ w_2 \ w_3 \cdots\cdots w_k \cdots w_n$

$l=1$

$i=1 \quad j=2 \quad \pi(1,2,X) = \max(q(X \to YZ) \times \pi(1,1,Y) \times \pi(2,2,Z))$

$i=2 \quad j=3$

$i=3 \quad j=4$

$\vdots$

$i=n-1 \quad j=n$

$l=2$

$i=1, \ j=3 \quad \pi(1,3,X) = \max(q(X \to YZ) \times \pi(1,1,Y) \times \pi(2,3,Z))$

$i=2, \ j=4 \quad \pi(2,4,X) =$

$\vdots$

$l=n-1$

$i=1 \quad j=n \quad \pi(1,n,X) = \max(\quad\quad)\quad)$

A Dynamic Programming Algorithm for the Sum

- Given a PCFG and a sentence s, how do we find

$$\sum_{t \in \mathcal{T}(s)} p(t)$$

- Notation:

$$n = \text{number of words in the sentence}$$

$$w_i = i\text{'th word in the sentence}$$

$$N = \text{the set of non-terminals in the grammar}$$

$$S = \text{the start symbol in the grammar}$$

- Define a dynamic programming table

$$\pi[i, j, X] \quad = \quad \text{sum of probabilities for constituent with non-terminal } X \text{ spanning words } i \ldots j \text{ inclusive}$$

- Our goal is to calculate $\sum_{t \in \mathcal{T}(s)} p(t) = \pi[1, n, S]$

Summary

- PCFGs augments CFGs by including a probability for each rule in the grammar.
- The probability for a parse tree is the product of probabilities for the rules in the tree
- To build a PCFG-parsed parser:
  1. Learn a PCFG from a treebank
  2. Given a test data sentence, use the CKY algorithm to compute the highest probability tree for the sentence under the PCFG