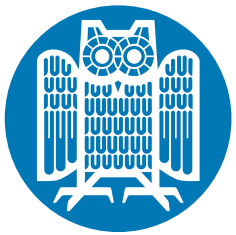
Mathematische Grundlagen III: Statistische Methoden
July 1, 2008

Probabilistic Context-free Grammars: Overview

Matthew Crocker

Computerlinguistik
Universität des Saarlandes

crocker@coli.uni-sb.de



UNIVERSITÄT
DES
SAARLANDES

Overview

- probabilistic parsing
- definition of a PCFG
- properties and assumptions
- computing the probability of a parse
- computing the probability of a sentence

We will follow Manning and Schuetze, 1999.

Parsing versus Classification

So far, we've applied machine learning to solving very specialized, and finite, classification tasks.

Parsing: recovering the syntactic structure of a sentence, usually the basis for determining the meaning.

Learning: can we use data about how to assign structures to sentences to automatically build parsers, as we did with classifiers?

Key differences:

- Infinite: most grammars generate an infinite number of syntactic structures, so there is no finite target class to be learned
- Data Sparseness: many sentence structures, and sentences, are never seen in training
- Ambiguity: sentences may be generated by more than one possible structure

Probabilistic Parsing

Basic problem in computational linguistics: ambiguity

- Lexical: words have different meanings and parts-of-speech
- Structural: different POS lead to different structure
- Structural: some structural ambiguity is independent of POS

Solution: use statistics to help

- Compute the best POS for each word (using n-grams, HMMs)
- Select the best parse tree for the sentence
- Also can help efficiency, and for language modelling

We want to compute to $\hat{t}$, the most probable parse for a string w_{1m}:

(1) $$\hat{t} = \arg\max_t P(t|w_{1m}, G)$$

Probabilistic Context-free Grammars

A PCFG G consists of:

$\{w^1, \ldots, w^V\}$	Terminal vocabulary
$\{N^1, \ldots, N^n\}$	Nonterminal vocabulary
N^1	Start symbol
$\{N^i \rightarrow \zeta^j\}$	Grammar rules, where ζ^j is a sequence of terminals and nonterminals
$\{P(N^i \rightarrow \zeta^j)\}$	Rule probabilities

Other notational conventions:

$\mathcal{L}$	Language (generated or accepted by the grammar)
t	Parse tree
N^j_{pq}	Nonterminal N^j spans positions p through q in string
w_{ab}	Sequence of words $w_a \cdots w_b$
$\alpha_j(p,q)$	Outside probabilities
$\beta_j(p,q)$	Inside probabilities

Properties of PCFGs

Intuitively, the main properties are:

- A PCFG is a standard CFG where the grammar rules are annotated with probabilities;
- the probabilities of all rules with the same lefthand side have to sum to one;
- the probability of a parse is the product of the probabilities of all rules applied in the parse;
- the probability of a sentence is the sum of the probabilities of all parse trees.

More formally, a PCFG has the property:

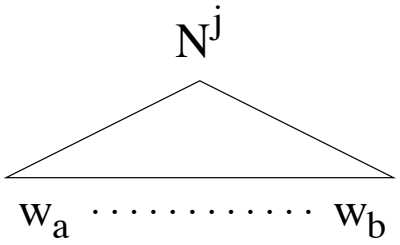
(2) $$\forall i \quad \sum_j P(N^i \rightarrow \zeta^j) = 1$$

Given a grammar G, a sentence w_{1m} has the probability:

(3) $$P(w_{1m}) = \sum_t P(w_{1m}, t) \quad \text{where } t \text{ is a parse tree of the sentence}$$
$$= \sum_{\{t:\text{yield}(t)=w_{1n}\}} P(t)$$

Properties of PCFGs

N^j dominates the words $w_a \cdots w_b$:



This is written as $N^j \Rightarrow^* w_a \cdots w_b$, or, alternatively as $\text{yield}(N^j) = w_a \cdots w_b = w_{ab}$.

Example for a PCFG

S $\rightarrow$ NP VP	1.0	NP $\rightarrow$ NP PP	0.4
PP $\rightarrow$ P NP	1.0	NP $\rightarrow$ astronomers	0.1
VP $\rightarrow$ V NP	0.7	NP $\rightarrow$ ears	0.18
VP $\rightarrow$ VP PP	0.3	NP $\rightarrow$ saw	0.04
P $\rightarrow$ with	1.0	NP $\rightarrow$ stars	0.18
V $\rightarrow$ saw	1.0	NP $\rightarrow$ telescopes	0.1

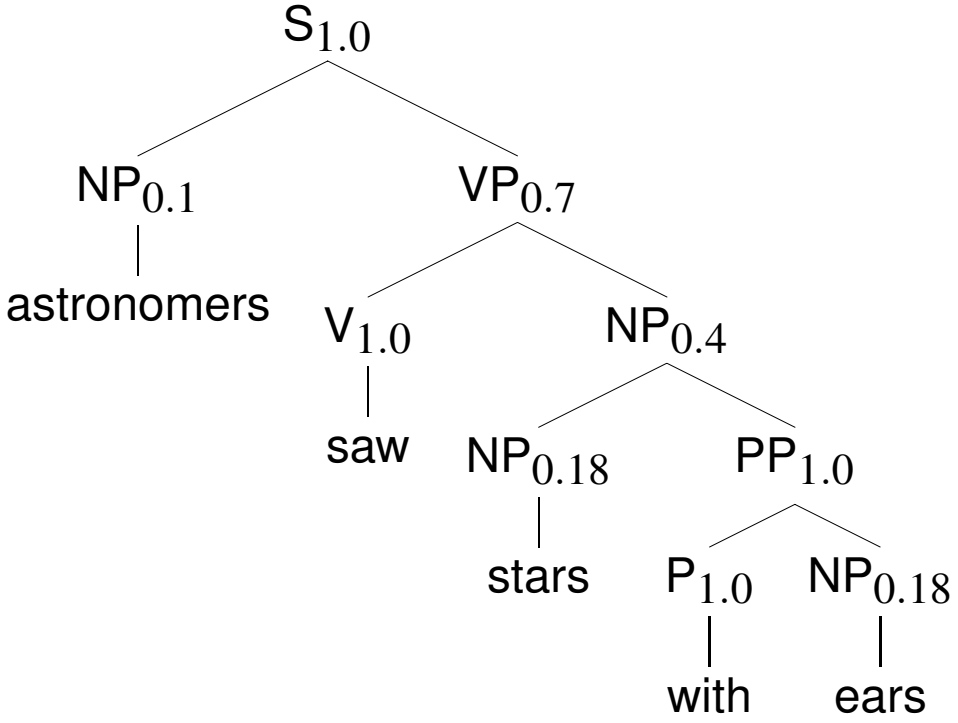
Computing the parse probability is:

$$P(t,s) = \prod_{n \in t} P(r(n)) \tag{4}$$

Product of all rules r used to expand each node n in the parse tree t of sentence s.

Example Parse Tree

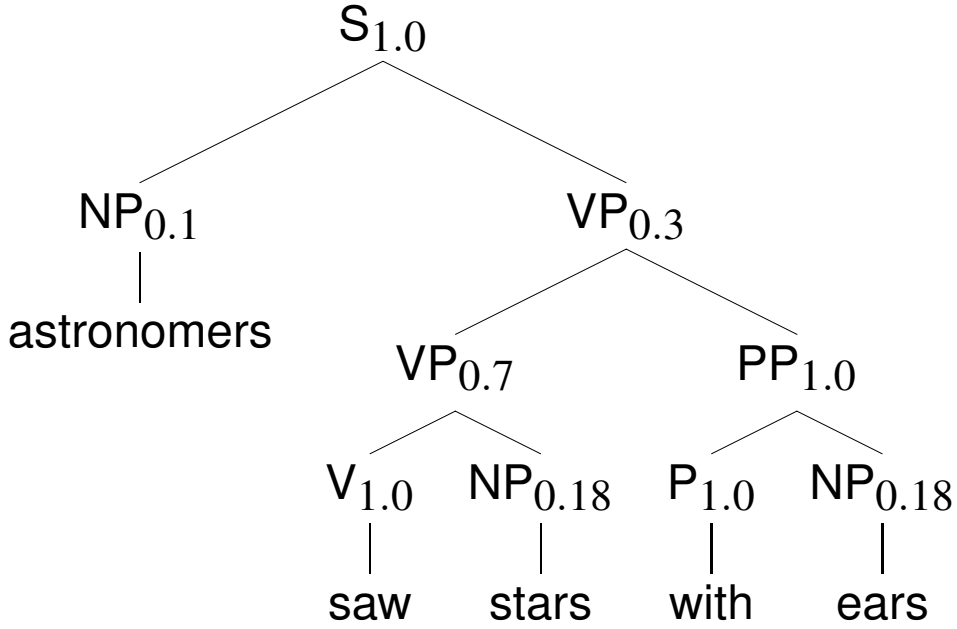
t_1:



$$P(t_1) = 1.0 \cdot 0.1 \cdot 0.7 \cdot 1.0 \cdot 0.4 \cdot 0.18 \cdot 1.0 \cdot 1.0 \cdot 0.18 = 0.0009072$$

Example Parse Tree

t_2:



$$P(t_2) = 1.0 \cdot 0.1 \cdot 0.3 \cdot 0.7 \cdot 1.0 \cdot 0.18 \cdot 1.0 \cdot 1.0 \cdot 0.18 = 0.0006804$$

Overall probability of the sentence:
$P(w_{15}) = P(t_1) + P(t_2) = 0.0015876$

Parse & Sentence Probabilities

The *parsing model*:

(5) $P(t|s,G)$ where, $\sum_t P(t|s,G) = 1$

The *language model*:

(6) $$P(s|G) = \sum_t P(s,t) = \sum_{t:yield(t)=s} P(t)$$

Now, to turn the language model into a parsing model, divide P(t,s) by P(s). Then the most probably parse $\hat{t}$, for a string w_{1m}:

(7) $$\hat{t} = \arg\max_t P(t|w_{1m},G) = \arg\max_t \frac{P(t,w_{1m})}{P(w_{1m})} = \arg\max_t P(t,w_{1m})$$

Assumptions Underlying PCFGs

Place invariance: the probability of a subtree does not depend on where in the string the words it dominates are (cf. time invariance in HMMs):

(8) $\forall k \quad P(N^j_{k(k+c)} \rightarrow \zeta)$ is the same

Context-free: the probability of a subtree does not depend on words not dominated by the subtree:

(9) $P(N^j_{kl} \rightarrow \zeta |$ anything outside k through $l) = P(N^j_{kl} \rightarrow \zeta)$

Ancestor-free: the probability of a subtree does not depend on the nodes in the derivation outside the subtree:

(10) $P(N^j_{kl} \rightarrow \zeta |$ any ancestor nodes outside $N^j_{kl}) = P(N^j_{kl} \rightarrow \zeta)$

Three Fundamental Questions

There are three fundamental question that we have to solve about PCFGs to be able to use them:

1. What is the probability of a sentence w_{1m} according to a grammar G: $P(w_{1m}|G)$?
2. What is the most probable parse for a sentence: $\arg\max_t P(t|w_{1m}, G)$?
3. How can we choose rule probabilities for the grammar G that maximize the probability of a sentence, $\arg\max_G P(w_{1m}|G)$?

There are efficient algorithms for this:

1. inside or outside probabilities
2. Viterbi Algorithm
3. Inside-Outside Algorithm

These algorithms are analogous to the ones used for HMMs; forward and backward probabilities in HMMs correspond to inside and outside probabilities in PCFGs.

Chomsky Normal Form

We will only consider Grammars in Chomsky Normal Form, i.e., grammars that have only unary and binary rules of the form:

$N^i \rightarrow N^j\ N^k$
$N^i \rightarrow w^j$

The parameters of a grammar in Chomsky Normal Form are:

$P(N^j \rightarrow N^r\ N^s|G)$ if n nonterminals, an n^3 matrix of parameters
$P(N^j \rightarrow w^k|G)$ if V terminals, $n \cdot V$ parameters

For $j = 1, \ldots, n$:

$$\sum_{r,s} P(N^j \rightarrow N^r\ N^s|G) + \sum_k P(N^j \rightarrow w^k|G) = 1 \tag{11}$$

Any CFG can be represented as a weakly equivalent CFG in Chomsky Normal Form.

Learning grammars and probabilities

There are two ways to determine the parameters, or probabilities, of PCFGs:

1. Try to learning them automatically: find the parameters which are most likely to generate some training corpus.
2. Find the probabilities somewhere else: e.g. determine the relative frequency of use for each rule (branch) type in a treebank

Automatic learning rarely works well. Need either a good grammar in advance, otherwise too many grammars are considered. Many local maxima.

Reading grammars and probabilities from treebanks limits the corpora and languages we can train on. Sparse data.

Problems with PCFGs

1. Not affected by syntactic context: e.g. Pronouns are more likely in subject position than object position
2. No influence of lexical items: subcategorisation, selectional restrictions, etc.
3. No sensitivity to global structural preferences: e.g. high versus low attachment of modifiers (PPs, relative clauses, adverbs, etc.)

Some solutions:

1. Lexicalization: each NT is associated with it's head
2. Parentization: Each NT production is condition on parent and grandparent
3. Other trick: subcategorization, traces, punctuation, clustering ...

Lexicalizing a PCFG

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 → telescopes	0.1

Naive lexicalization of only the VP rules:

VP(saw) → V(saw) NP(astronomers)	0.1	V(saw) → saw	1.0
VP(saw) → V(saw) NP(ears)	0.15	NP → astronomers	0.1
VP(saw) → V(saw) NP(saw)	0.05	NP → ears	0.18
VP(saw) → V(saw) NP(stars)	0.3	NP → saw	0.04
VP(saw) → V(saw) NP(telescopes)	0.1	NP → stars	0.18
VP(saw) → VP(saw) PP(with)	0.3	NP → telescopes	0.1

Now, suppose we had 10 verbs ... we would have 60 VP rules.

Lexicalizing a PCFG

A fully lexicalized grammar requires that for all rules:

$N^j \rightarrow N^r\ N^s$

we must estimate:

$P(N^j \rightarrow N^r\ N^s | j, h(N^j), h(N^r), h(N^s))$

However, this requires us to estimate an impossible number of parameters, for which there will be insufficient training data. Thus, typically, we condition on just the lexical head:

$P(N^j \rightarrow N^r\ N^s | j, h(N^j))$

In other words, our grammar now has only two VP rules again:

VP(saw) $\rightarrow$ V(saw) NP 0.7
VP(saw) $\rightarrow$ VP(saw) PP 0.3

Now, suppose we had 10 verbs ... we would have 20 VP rules.

Evaluation of PCFGs

PARSEVAL

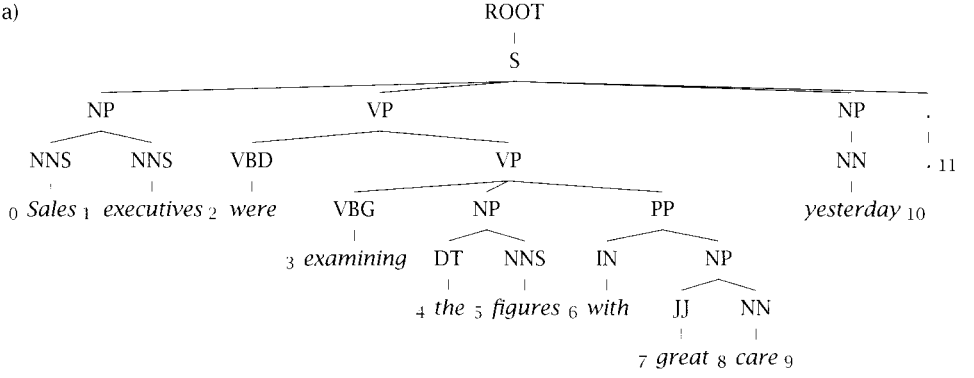
Compare the output of the parser to some “Gold Standard”, usually a Treebank. A constituent is labelled correctly if there is a constituent in the treebank with the same start-point, end-point, and same non-terminal symbol.

1. labeled recall: $\frac{\text{\# correct constituents in candidate parse of s}}{\text{\# correct constituents in treebank parse of s}}$
2. labeled precision: $\frac{\text{\# correct constituents in candidate parse of s}}{\text{\# total constituents in candidate parse of s}}$
3. crossed-brackets: how many constituents have crossing brackets, e.g. ((A B) C) instead of (A (B C))

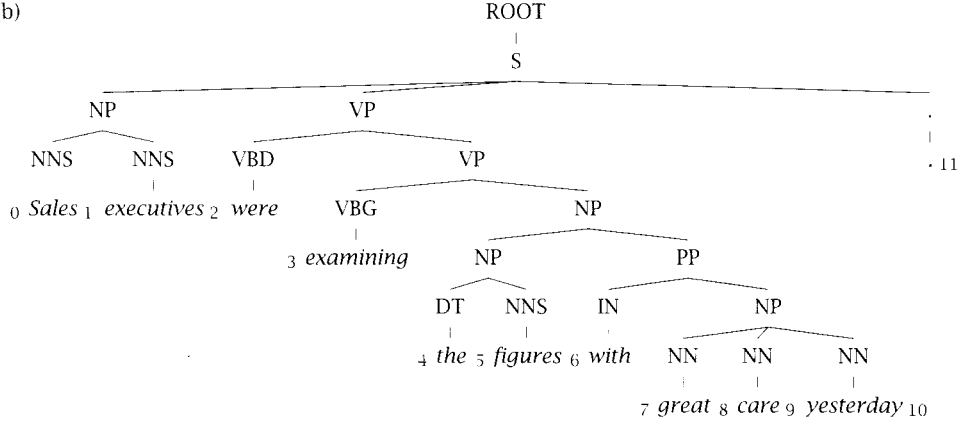
Performance:

1. Standard PCFG: LR=71.7, LP=75.8
2. Lexicalized PCFG: LR=83.4, LP=84.1
3. Charniak (2000): LR=91.1, LP=90.1

(a)



(b)



(c) Brackets in gold standard tree (a.):
S-(0:11), **NP-(0:2)**, VP-(2:9), VP-(3:9), **NP-(4:6)**, PP-(6-9), NP-(7,9), *NP-(9:10)

(d) Brackets in candidate parse (b.):
S-(0:11), **NP-(0:2)**, VP-(2:10), VP-(3:10), NP-(4:10), **NP-(4:6)**, PP-(6-10), NP-(7,10)

(e) Precision:	3/8 = 37.5%	Crossing Brackets:	0
Recall:	3/8 = 37.5%	Crossing Accuracy:	100%
Labeled Precision:	3/8 = 37.5%	Tagging Accuracy:	10/11 = 90.9%
Labeled Recall:	3/8 = 37.5%		

Discussion of PCFGs

1. Help deal with increasing ambiguity as we grow the coverage of our grammars
2. This enables us to use more robust grammars, that allows some ungrammaticalities
3. PCFGs can be learned ”in the limit”, from positive examples, while CFGs cannot
4. PCFGs provide a language model, but typically worse than n-gram models
5. PCFGs are biased: probabilities of smaller trees are greater than larger trees (independent of corpus frequency), also favours non-terminals with few expansions
6. No direct notion of plausibility, but lexicalized PCFGs may implicitly have this information