Probabilistic Context Free Grammars

Simon Clematide
Institute of Computational Linguistics
University of Zurich

http://www.cl.unizh.ch/siclemat/talks/pcfg/

Synopsis

Program

- CFG grammars, rewriting, derivations, parse trees
  - Shift-reduce parsing CFGs
- PCFG grammars
  - Probability of rules, derivations, parse trees
  - Tasks for PCFGs
- One Task: Most probable parse tree
  - Naive solution
  - Viterbi chart parsing
  - Pseudocode for viterbi
- Assumptions, features and use of PCFGs

Context Free Grammars

Symbols

Start Symbol → *S*

Nonterminals 'syntactic categories': *S*, *VP*, *P*, *N*, *V*, *NP*

Terminals 'words': *astronomers*, *saw*, *with*, *stars*, *ears*, *telescopes*

Rules

'Syntactic' Rules:
- $S \rightarrow NP\ VP$
- $PP \rightarrow P\ NP$
- $VP \rightarrow V\ NP$
- $VP \rightarrow VP\ PP$
- $NP \rightarrow NP\ PP$

'Lexical' Rules:
- $NP \rightarrow$ *astronomers*
- $NP \rightarrow$ *telescopes*
- $NP \rightarrow$ *saw*
- $NP \rightarrow$ *ears*
- $NP \rightarrow$ *stars*
- $P \rightarrow$ *with*
- $V \rightarrow$ *saw*

Formal Definition

A **context-free grammar** G is a quadrupel (N, T, S, R) where

- N is a finite set of **non-terminal symbols**
- T is a finite set of **terminal symbols** ($T \cap N = \emptyset$)
  - V is the set of symbols; short cut for $N \cup T$
- S is a distinguished **start symbol** ($S \in N$)
- R is a finite set of production **rules** ($R \subseteq N \times V^*$)

Arrow notation for rule: $B \rightarrow \beta$ where

- $B \in N$ and $\beta \in V^*$

From Rewriting to Derivation

One rewriting step

- The string $\alpha\mathbf{B}\gamma \in V^*$ can be rewritten as $\alpha\beta\gamma$ iff $\mathbf{B} \rightarrow \beta$ is in R.

$$\alpha B\gamma \Rightarrow \alpha\beta\gamma$$

Finite number of steps

- If a string $\phi \in V^*$ can be rewritten as ψ in a finite number of steps, this is denoted $\phi \Rightarrow^* \psi$.

Language of a grammar *G*

$$L(G) = \{W \in T^* : S \Rightarrow^* W\}$$

S
⇒ ***NP*** *VP*
⇒ *astronomers* **VP**
⇒ astronomers ***V*** *NP*
⇒ astronomers *saw* **NP**
⇒ astronomers saw *stars*

Example Derivation:
S ⇒ astronomers saw stars*

Derivations and Parse Trees

A leftmost derivation

of a terminal sequence $W \in T^*$ is the sequence

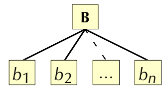
$$S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow W$$

where the leftmost non-terminal symbol is rewritten in each step.

A parse tree

of *W* is the tree representation of a (leftmost) derivation.

- Root is labelled *S*.
- Leafs are labelled in the order they appear in *W*.
- Inner nodes are labelled according to the rewritten symbols.



Labelling closeup of one derivation step involving rule $\mathbf{B} \rightarrow \beta$ $(\beta = b_1, b_2, \ldots, b_n)$

Bottom Up CFG Recognizing in Prolog

```
:- op(1150, xfx, '--->').

% sr_parse(Words, Startsymbol)
sr_parse(W, S):-
   sr_parse([], W, S).

% Ok, if no more words
% and startsymbol on stack
sr_parse([S], [], S).

% Reduce Stack by one rule
sr_parse([V2,V1|Stack], W, S):-
    (LHS ---> (V1,V2)),
    sr_parse([LHS|Stack], W, S).

% Consume word and shift its cat on stack
sr_parse(Stack,[W1|W],S):-
    (LHS ---> [W1]),
    sr_parse([LHS|Stack], W, S).
```

Shift-Reduce Parser for Grammars in Chomsky Normal Form

```
s  ---> np, vp.
pp ---> p, np.
vp ---> v, np.
vp ---> vp, pp.
np ---> np, pp.
```

Syntactic Rules

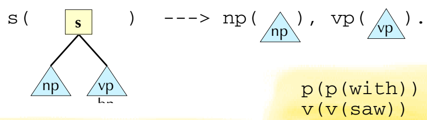
```
p ---> [with].
v ---> [saw].
np ---> [saw].
np ---> [astronomers].
np ---> [ears].
np ---> [stars].
np ---> [telescopes].
```

Lexical Rules

Bottom Up CFG Parsing in Prolog

Allow complex grammar symbols

for storing the partial trees of each rule.




```
s(s(NP,VP))    ---> np(NP), vp(VP).
pp(pp(P,NP))   ---> p(P), np(NP).
vp(vp(V,NP))   ---> v(V), np(NP).
vp(vp(VP,PP))  ---> vp(VP), pp(PP).
np(np(NP,PP))  ---> np(NP), pp(PP).
```

Syntactic Rules

```
p(p(with))     ---> [with].
v(v(saw))      ---> [saw].
np(np(saw))    ---> [saw].
np(np(astronomers)) --->
                    [astronomers].
np(np(ears))   ---> [ears].
np(np(stars))  ---> [stars].
np(np(telescopes)) --->
                    [telescopes].
```

Lexical Rules

```
?- sr_parse([astronomers,saw,stars], s(Tree)).
```

PCFG: Formal Definition

A **Probabilistic CFG** is a quintupel G=(*N, T, S, R, P*) where

- *(N,T,S,R)* build up a normal CFG.
    - Let *V* again denote $N \cup T$.
- *P* is a **conditional probabilistic function** $(N \times V^*) \to [0,1]$.

Remarks

- $P(B,\beta)$ is normally notated as $P(B\to\beta)$ which means $P(B\to\beta\,|\,B)$. The probability of rewriting with β given B.
- for all $B \in N$,

$$\sum_{\beta \in V^*} P(B \to \beta) = 1$$

$P(VP \to V\ NP) = 0.7$
$P(VP \to VP\ PP) = 0.3$

Probability of Trees and Strings

The probability of a parse tree of a string *W*

is the probability of its leftmost derivation $P(S \Rightarrow^* W)$.

- Or the probability of some other derivation strategy ...

The probability of a derivation

is the product of the rules' probabilities that are used in the derivation.

The probability of a string

is the sum of the probabilities of its parse trees.

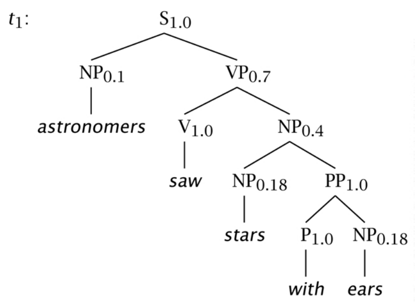
The probability of a language

is the sum of the probabilities of its strings.

- May be improper...

Toy Grammar with Silly Sentence

P(t1) = …

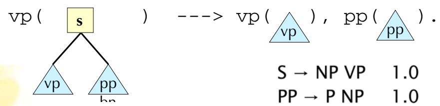


S → NP VP	1.0
PP → P NP	1.0
VP → V NP	0.7
VP → VP PP	0.3
P → *with*	1.0
V → *saw*	1.0

NP → NP PP	0.4
NP → *astronomers*	0.1
NP → *ears*	0.18
NP → *saw*	0.04
NP → *stars*	0.18
NP → *telescopes*	0.1

Toy Grammar in Prolog

Also allow probabilities in symbols…




```
vp(vp(V,NP)@0.7*(PV*PNP)) --->
   v(V@PV),
   np(NP@PNP).
vp(vp(VP,PP)@0.3*(PVP*PPP)) --->
   vp(VP@PVP),
   pp(PP@PPP).
...
```

```
p(p(with)@1) ---> [with].
v(v(saw)@1) ---> [saw].
np(np(astronomers)@0.1) ---> [astronomers].
```

```
?- sr_parse([astronomers,saw,stars],
            s(Tree-Prob)).
```

S → NP VP	1.0
PP → P NP	1.0
VP → V NP	0.7
VP → VP PP	0.3
P → *with*	1.0
V → *saw*	1.0

NP → NP PP	0.4
NP → *astronomers*	0.1
NP → *ears*	0.18
NP → *saw*	0.04
NP → *stars*	0.18
NP → *telescopes*	0.1

Three Tasks for PCFGs

Language modelling

- Give the probability for each string generated by a grammar!

$$P(w_1 \ldots w_m \mid G)$$

Best parse tree

- Select the most probable parse tree for a given string!

$$\arg\max_{tree} P(tree \mid w_1 \ldots w_m, G)$$

Grammar learning

- Optimize the rule probabilities of a given grammar for some sentences!

$$\arg\max_{G} P(w_1 \ldots w_m \mid G)$$

Most Probable Parse Tree

Which is the most probable parse tree for a given sentence?

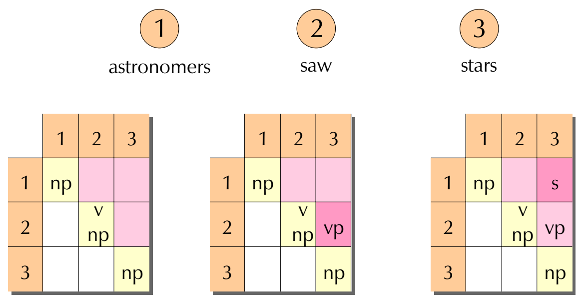
- Straight-forward solution
  - Enumerate all possible parse trees and take the maximum!

```
?- findall(P-T, sr_parse([stars,saw,ears],s(T@P), Pairs),
   max_key(Pairs,MaxP-MaxTree).
```

- This is naive for longer sentences with decently ambiguous grammars!
  - Exponentional complexity: $O(n^{|\text{sentence length}|})$
- Solution: Use a caching algorithm!
  - Saying probabilistic chart parsing, Viterbi algorithm for PCFG or inside algorithm!

First: Building CFG Charts

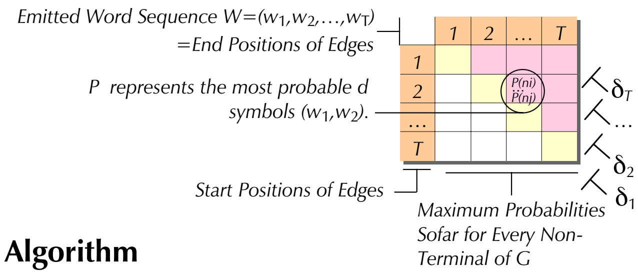
Draw edges!



(Passive) Chart as Matrix: Lexical Information | *First Syntactic Edge* | *Second syntactic edge* | *Fundamental Rule of Chart Parsing*

Viterbi for CFG in Chomsky Normal F.

Delta, the caching data structure



The Viterbi Algorithm

- initializes the delta matrix diagonally in δ_1 with lexical rules.
- fills delta diagonally to the right upper corner with syntactic rules.
- computes the most probable tree back from the upper right cell δ_T.

Induction of Delta

V1 Initialization: For each word in $W = w_1, \ldots, w_T \in T^*$

$$\delta_{t,t}(X_i) = P(X_i \rightarrow w_t) \quad \text{for } 1 \le t \le T \text{ and } X_i \in N$$

V2 Induction

$$\delta_{r,t}(X_i) = \max_{\substack{X_j, X_k \\ r,s,t}} P(X_i \rightarrow X_j X_k)\delta_{r,s}(X_j)\delta_{s+1,t}(X_k) \quad \text{for } 1 \le r \le s < t \le T \text{ and } X_i \in N$$

$$\psi_{r,t}(X_i) = \arg\max_{X_j, X_k, s} P(X_i \rightarrow X_j X_k)\delta_{r,s}(X_j)\delta_{s+1,t}(X_k)$$

V3 Termination

$$P_{\max}(W \mid G) = \delta_{1,T}(S)$$

- Construct parse tree by working backwards through Ψ.

Pseudo Code for CYK Algorithm

```
# Given
# Sentence: Array w
# Nonterminals: NT
# Terminals: T
# Lexical rule probability:
#   Matrix  lex[NT][T] -> Prob
# Syntactic rule probability:
#   Matrix syn [NT][NT][NT]-> Prob
# Delta: Matrix delta[int][int][NT] -> Prob
# Chart (Psi):
#    Matrix chart[int][int][NT] -> <int,NT,NT>
# Returns
# Total probability: p

# V1: Lexical level
for i := 1 to length(w) do
  foreach a in NT do
    delta[i][i][a] := lex[a][w[i]]
  done
done

# V2 Syntactic levels
for span := 2 to length(w) do
  for from := 1 to length(w) - span - 1
    end := from + span - 1
    for middle := begin to end - 1
      foreach a in keys(syn) do
        foreach b in keys(syn[a]) do
          foreach c in keys(syn[a][b]) do
            p:= syn[a][b][c]*delta[begin][middle][b]
                *delta[middle+1][end][c]
            if (p > delta[begin][end][a] then
              delta[begin][end][a] := p
              chart[begin][end][a] := <middle,b,c>
          done
        done
      done
    done
  done
done
```


Assumptions of PCFG

Place invariance

- Identical subtrees have the same probability wherever they appear in a syntax tree.

Context-free

- The probability of a subtree does not take into account the words before or after.

Ancestor-free

- The dominating nodes of a subtree have no influence on its probability.

Usefullness of PCFGs

PCFG as language models

- Probabilistic language models of raw PCFGs are (too) simple. Raw trigram models are better.
  - Independence assumptions are too strong.
  - Lexicalization and contextualization is needed.
  - This can be added in various fashions. (see M&S Ch. 12)

PCFG for parsing

- Good robustness if you allows everything with low probability.
- May help in some cases to make the right desambiguation decision.
  - But certain biases are typical, e.g. preference for smaller trees.

PCFG is apted for grammar induction