Natural Language Parsing Techonlogy

Foundations of Language Science and Technology (WS 2014/2015)

Bernd Kiefer

Language Technology Lab, DFKI GmbH

Department of Computational Linguistics
Saarland University

November 2014

Outline

Overview

Basic Parsing Algorithms
- Parsing Strategies
- CYK Algorithm
- Earley's Algorithm

Parsing with Probabilistic Context-Free Grammar
- PCFG
- Inside-Outside Algorithm

Recent Advances in Parsing Technology

Outline

Overview

Basic Parsing Algorithms

- Parsing Strategies
- CYK Algorithm
- Earley's Algorithm

Parsing with Probabilistic Context-Free Grammar

- PCFG
- Inside-Outside Algorithm

Recent Advances in Parsing Technology

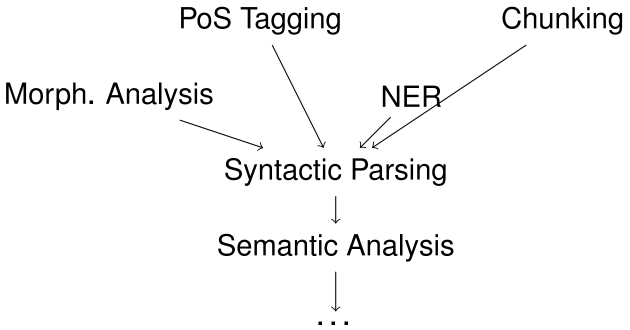
Language & Grammar

- Language
  - Structural
  - Productive
  - Ambiguous, yet efficient in human-human communication
- Grammar
  - Generalization of regularities in language structures
  - Morphology & syntax, often complemented by phonetics, phonology, semantics, and pragmatics

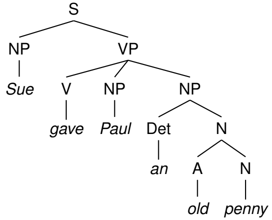
Ambiguity

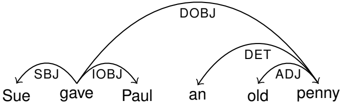
- Human languages are ambiguous on almost every layer
- Grammar frameworks are designed to represent necessary ambiguities, and eliminate unnecessary ones
- Parsing models are responsible for retrieving valid analyses according to the grammar

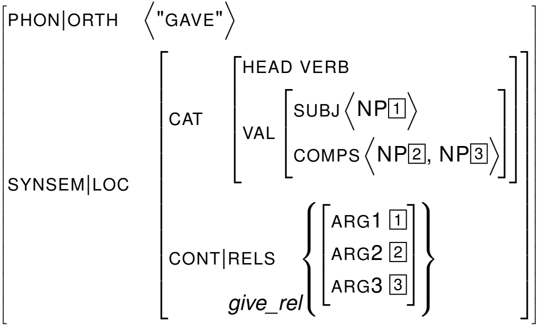
Syntactic Parser as NLP Component



Trees (or not)


S
NP
VP
Sue
V
NP
NP
gave
Paul
Det
N
an
A
N
old
penny


DOBJ
DET
SBJ
IOBJ
ADJ
Sue
gave
Paul
an
old
penny


PHON|ORTH
"GAVE"
SYNSEM|LOC
CAT
HEAD VERB
VAL
SUBJ
NP1
COMPS
NP2, NP3
CONT|RELS
give_rel
ARG1 1
ARG2 2
ARG3 3

Chomsky Hierarchy

- Type 0 (unrestricted rewriting system)

$$\alpha \rightarrow \beta \quad \alpha, \beta \in (V_N \cup V_T)^*$$

- Type 1 (context sensitive grammars)

$$\phi A \omega \rightarrow \phi \beta \omega \quad A \in V_N, \beta, \phi, \omega \in (V_N \cup V_T)^*$$

- Type 2 (context free grammars)

$$A \rightarrow \beta \quad A \in V_N, \beta \in (V_N \cup V_T)^*$$

- Type 3 (regular grammars)

$$A \rightarrow xB \vee A \rightarrow x \quad A, B \in V_N, x \in V_T$$

Context-Free Grammar

A CFG is a quadruple: $\langle V_T, V_N, \mathscr{P}, S\rangle$

- V_T: terminal symbols
- V_N: non-terminal symbols
- $\mathscr{P}$: context-free productions

$$A \rightarrow \beta \quad A \in V_N, \beta \in (V_N \cup V_T)^*$$

- S: start symbol

Context-Free Phrase Structure Grammar

- $S \rightarrow NP\ VP$
- $NP \rightarrow Det\ N$
- $N \rightarrow Adj\ N$
- $VP \rightarrow V$
- $VP \rightarrow V\ NP$
- $VP \rightarrow Adv\ VP$

- $N \rightarrow dog|cat$
- $Det \rightarrow the|a$
- $V \rightarrow chases|sleeps$
- $Adj \rightarrow gray|lazy$
- $Adv \rightarrow fiercely$

CFG Derivation

- If $\phi = \beta A\gamma$, $\omega = \beta\alpha\gamma$ and $A \rightarrow \alpha \in \mathscr{P}$
  then ω follows ϕ, $\phi \Rightarrow \omega$
- If a sequence of strings $\phi_1, \phi_2, \ldots, \phi_m$ where for all i
  $(1 \leq i \leq m - 1)$, $\phi_i \Rightarrow \phi_{i+1}$
  then $\phi_1, \phi_2, \ldots, \phi_m$ is a derivation from ϕ_1 to ϕ_m
- **"Derivable"** relation: transitive, reflexive

$$\phi_1 \overset{*}{\Rightarrow} \phi_m$$

Outline

Overview

Basic Parsing Algorithms
- Parsing Strategies
- CYK Algorithm
- Earley's Algorithm

Parsing with Probabilistic Context-Free Grammar
- PCFG
- Inside-Outside Algorithm

Recent Advances in Parsing Technology

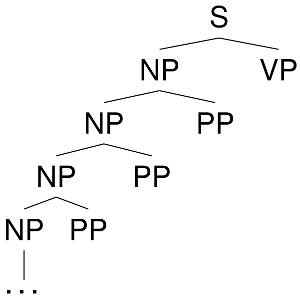
Parsing Strategies

- Top-down: start from the start symbol, and expand the tree with grammar rules (e.g. replace LHS symbol with RHS sequences of CFG productions)
- Bottom-up: start from the input sequence, and apply grammar rules to build trees upwards (e.g. reducing RHS sequence into LHS symbols)

Top-Down Parsing

- Goal-directed search
- Waste time on trees that do not match input sentence
- Pure top-down (left-first) approach cannot parse (left-)recursion grammars

1. $S \rightarrow NP\ VP$
2. $NP \rightarrow NP\ PP$
3. $\ldots$



Bottom-Up Parsing

- Use the input to guide the search (data-driven)
- Waste time on trees that don't result in S
- Recursive unary rules still create an infinite parse forest for a finite length sentence

1. $A \rightarrow B|a$
2. $B \rightarrow A$
3. ...

Problems

- Left-recursion $NP \rightarrow NP\ PP$
- Ambiguity
- Repeated parsing of subtrees

Dynamic Programming (DP)

- Divisibility: the optimal solution of a sub problem is part of the optimal solution of the whole problem
- Memoization: solve small problems only once and remember the answers

Example

Calculating Fibonacci numbers:

$$F_n = F_{n-1} + F_{n-2} \quad (F_0 = 0, F_1 = 1)$$

Pascal Triangle (Binomial Coefficients):

$$\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$$

CYK Algorithm

- Cocke-Younger-Kasami, also known as CKY algorithm
- Essentially a bottom-up chart parsing algorithm using dynamic programming
- CFG is in Chomsky Normal Form (CNF)
  - $A \rightarrow BC$
  - $A \rightarrow a$
  - $S \rightarrow \epsilon$
  - $A, B, C \in V_N, \quad a \in V_T, \quad B, C \neq S$
- Fill in a two-dimension array: $\mathbb{C}[i][j]$ contains all the possible syntactic interpretations of the substring $w_{i+1} \ldots w_j$
- Complexity $O(n^3)$

CYK Algorithm

for all $i, j \quad 0 \leq i < j \leq n$ **do**
    $\mathbb{C}[i][j] \Leftarrow \emptyset$
end for
for all $A \rightarrow w_i \in \mathscr{P}$ **do**
    $\mathbb{C}[i-1][i] \Leftarrow \{A\} \cup \mathbb{C}[i-1][i]$
end for
for $s = \langle 2 \ldots n \rangle$ **do**
    for all $A \rightarrow B\ C \in \mathscr{P}, i, k : 0 \leq i < k < i + s$ **do**
        if $B \in \mathbb{C}[i][k] \wedge C \in \mathbb{C}[k][i+s]$ **then**
            $\mathbb{C}[i][i+s] \Leftarrow \{A\} \cup \mathbb{C}[i][i+s]$
        end if
    end for
end for

CYK Chart Example

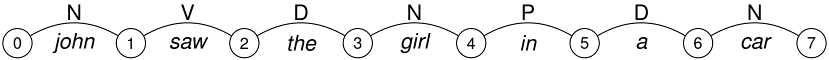
S →NP VP|N VP|N V|NP V
VP→V NP|V N|VP PP
NP→D N|NP PP|N PP
PP→P NP|P N
N →*john, girl, car*
V →*saw, walks*
P →*in*
D →*the, a*

(0) *john* (1) *saw* (2) *the* (3) *girl* (4) *in* (5) *a* (6) *car* (7)

CYK Chart Example

N						
	V					
		D				
			N			
				P		
					D	
						N

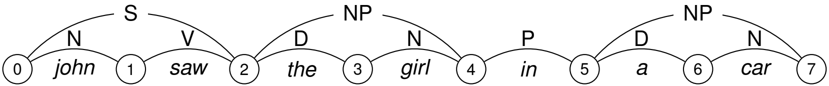
S $\rightarrow$NP VP|N VP|N V|NP V
VP$\rightarrow$V NP|V N|VP PP
NP$\rightarrow$D N|NP PP|N PP
PP$\rightarrow$P NP|P N
N $\rightarrow$*john, girl, car*
V $\rightarrow$*saw, walks*
P $\rightarrow$*in*
D $\rightarrow$*the, a*



CYK Chart Example

N	S					
	V					
		D	NP			
			N			
				P		
					D	NP
						N

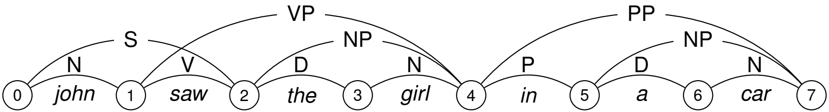
S $\rightarrow$ NP VP | N VP | N V | NP V
VP $\rightarrow$ V NP | V N | VP PP
NP $\rightarrow$ D N | NP PP | N PP
PP $\rightarrow$ P NP | P N
N $\rightarrow$ *john, girl, car*
V $\rightarrow$ *saw, walks*
P $\rightarrow$ *in*
D $\rightarrow$ *the, a*



CYK Chart Example

N	S					
	V		VP			
		D	NP			
			N			
				P		PP
					D	NP
						N

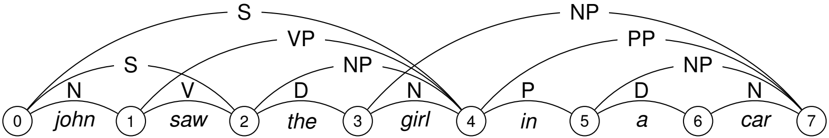
S $\rightarrow$NP VP|N VP|N V|NP V
VP$\rightarrow$V NP|V N|VP PP
NP$\rightarrow$D N|NP PP|N PP
PP$\rightarrow$P NP|P N
N $\rightarrow$*john, girl, car*
V $\rightarrow$*saw, walks*
P $\rightarrow$*in*
D $\rightarrow$*the, a*



CYK Chart Example

N	S		S			
	V		VP			
		D	NP			
			N			NP
				P		PP
					D	NP
						N

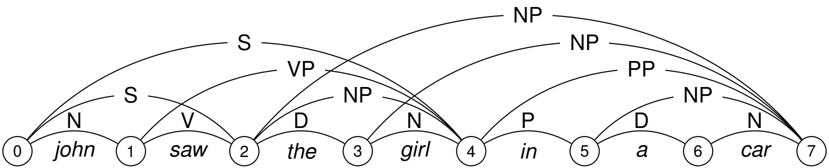
S →NP VP|N VP|N V|NP V
VP→V NP|V N|VP PP
NP→D N|NP PP|N PP
PP→P NP|P N
N →*john, girl, car*
V →*saw, walks*
P →*in*
D →*the, a*



CYK Chart Example

N	S		S			
	V		VP			
		D	NP			NP
			N			NP
				P		PP
					D	NP
						N

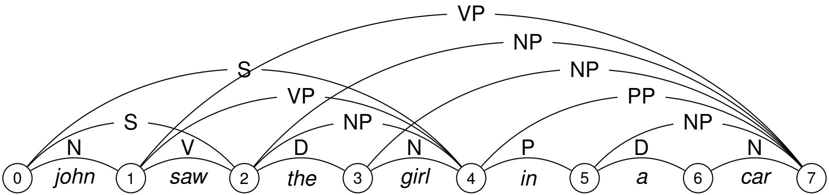
S $\rightarrow$ NP VP|N VP|N V|NP V
VP $\rightarrow$ V NP|V N|VP PP
NP $\rightarrow$ D N|NP PP|N PP
PP $\rightarrow$ P NP|P N
N $\rightarrow$ *john, girl, car*
V $\rightarrow$ *saw, walks*
P $\rightarrow$ *in*
D $\rightarrow$ *the, a*



CYK Chart Example

N	S		S			
	V		VP			VP
		D	NP			NP
			N			NP
				P		PP
					D	NP
						N

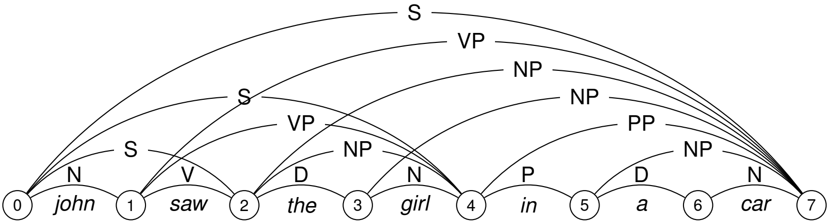
S $\rightarrow$ NP VP | N VP | N V | NP V
VP $\rightarrow$ V NP | V N | VP PP
NP $\rightarrow$ D N | NP PP | N PP
PP $\rightarrow$ P NP | P N
N $\rightarrow$ *john, girl, car*
V $\rightarrow$ *saw, walks*
P $\rightarrow$ *in*
D $\rightarrow$ *the, a*



CYK Chart Example

N	S		S			S
	V		VP			VP
		D	NP			NP
			N			NP
				P		PP
					D	NP
						N

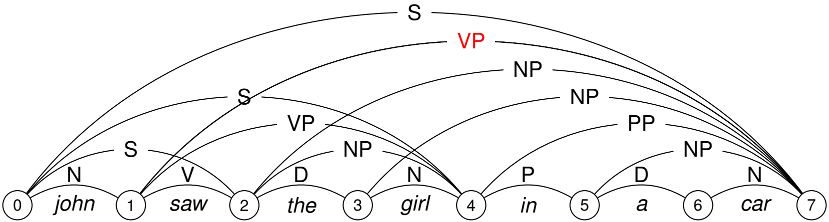
S →NP VP|N VP|N V|NP V
VP→V NP|V N|VP PP
NP→D N|NP PP|N PP
PP→P NP|P N
N →*john, girl, car*
V →*saw, walks*
P →*in*
D →*the, a*



CYK Chart Example

N	S		S			S
	V		VP			VP
		D	NP			NP
			N			NP
				P		PP
					D	NP
						N

S $\rightarrow$ NP VP|N VP|N V|NP V
VP $\rightarrow$ V NP|V N|VP PP
NP $\rightarrow$ D N|NP PP|N PP
PP $\rightarrow$ P NP|P N
N $\rightarrow$ *john, girl, car*
V $\rightarrow$ *saw, walks*
P $\rightarrow$ *in*
D $\rightarrow$ *the, a*



Earley's Algorithm

- Use dynamic programming to do top-down search
- Chart: a set of items $\langle h, i, A \to \alpha \cdot \beta \rangle$
  - h, i: positions in the input $0 \leq h \leq i \leq n$
  - $A \to \alpha \cdot \beta$: dotted rule ($A \to \alpha\beta \in \mathscr{P}$)
    - α: RHS prefix that has already been applied to input from h to i
    - β: RHS suffix yet to be found

Earley's Algorithm

- **Initialize**
  foreach $S \rightarrow \alpha \in \mathscr{P}$
  $\mathbb{C} \Leftarrow \langle 0, 0, S \rightarrow \cdot\alpha \rangle$
- **Scan(i)**
  if $w_i = a \wedge \langle h, i-1, A \rightarrow \alpha \cdot a\beta \rangle \in \mathbb{C}$
  $\mathbb{C} \Leftarrow \langle h, i, A \rightarrow \alpha a \cdot \beta \rangle$
- **Complete(i)**
  foreach $\langle h, i, A \rightarrow \alpha\cdot \rangle \in \mathbb{C}$
  foreach $\langle k, h, B \rightarrow \beta \cdot A\gamma \rangle \in \mathbb{C}$
  $\mathbb{C} \Leftarrow \langle k, i, B \rightarrow \beta A \cdot \gamma \rangle$
- **Predict(i)**
  foreach $\langle h, i, A \rightarrow \alpha \cdot B\beta \rangle \in \mathbb{C}$
  foreach $B \rightarrow \gamma \in \mathscr{P}$
  $\mathbb{C} \Leftarrow \langle i, i, B \rightarrow \cdot\gamma \rangle$
- **Parse**
  Initialize
  for $i = \langle 1 \ldots n \rangle$
  Predict($i-1$)
  Scan(i)
  Complete(i)
  if $\exists \langle 0, n, S \rightarrow \alpha\cdot \rangle \in \mathbb{C}$
  return *success*
  else
  return *failed*

Earley Chart: Example

$_0$ the/det $_1$ dog/n $_2$ chases/v $_3$ a/det $_4$ cat/n $_5$

	0	1	2	3	4	5
0	$S \rightarrow \cdot NP\ VP$ $NP \rightarrow \cdot det\ n$					
1	$NP \rightarrow det \cdot n$					
2	$NP \rightarrow det\ n \cdot$ $S \rightarrow NP \cdot VP$		$VP \rightarrow \cdot v$ $VP \rightarrow \cdot v\ NP$			
3	$S \rightarrow NP\ VP \cdot$		$VP \rightarrow v \cdot$ $VP \rightarrow v \cdot NP$	$NP \rightarrow \cdot det\ n$		
4				$NP \rightarrow det \cdot n$		
5	$S \rightarrow NP\ VP \cdot$		$VP \rightarrow v\ NP \cdot$	$NP \rightarrow det\ n \cdot$		

1. $S \rightarrow NP\ VP$
2. $VP \rightarrow v\ NP$
3. $VP \rightarrow v$
4. $NP \rightarrow det\ n$

Outline

Overview

Basic Parsing Algorithms

- Parsing Strategies
- CYK Algorithm
- Earley's Algorithm

Parsing with Probabilistic Context-Free Grammar

- PCFG
- Inside-Outside Algorithm

Recent Advances in Parsing Technology

Probabilistic Context-Free Grammar

An PCFG is a quintuple: $\langle V_T, V_N, \mathscr{P}, S, \mathbf{Pr} \rangle$

- $\mathbf{Pr} : \mathscr{P} \rightarrow [0, 1]$ s.t.

$$\forall A \in V_N, \sum_{A \rightarrow \alpha \in \mathscr{P}} \mathbf{Pr}(A \rightarrow \alpha) = 1$$

- $\mathbf{Pr}(A \rightarrow \alpha)$ can be understood as the conditional probability of observing $A \rightarrow \alpha$ in the derivation given A: $P(A \rightarrow \alpha | A)$

Various Probabilities

Joint Probability: $P(x, y)$

- Input sequence: x
- A Parse: y with corresponding derivation sequence:

$$S \underset{r_1}{\Rightarrow} \phi_1 \underset{r_2}{\Rightarrow} \phi_2 \underset{r_3}{\Rightarrow} \ldots \underset{r_k}{\Rightarrow} x$$

  where r_i is the production rule used in th i^{th} derivation step
- $P(x, y) = \prod_{i=1}^{k} \mathbf{Pr}(r_i)$
- $\sum_{y \in \mathscr{T}(G), x = yield(y)} P(x, y) = 1$
- More generally, $P(x, y|A) = \prod_{i=1}^{k} \mathbf{Pr}(r_i)$ is the probability of a sub-parse y rooted by A and generate input x by derivation sequence

$$A \underset{r_1}{\Rightarrow} \phi_1 \underset{r_2}{\Rightarrow} \phi_2 \underset{r_3}{\Rightarrow} \ldots \underset{r_k}{\Rightarrow} x$$

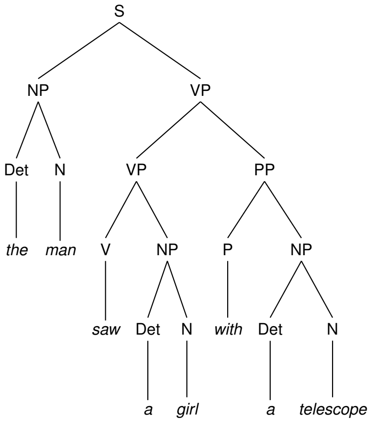
Various Probabilities

Structural Language Model: $P(x)$

- $P(x) = \sum_{y \in \mathscr{T}(x)} P(x, y)$
- $\mathscr{T}(x)$ is the set of parse trees for input sequence x

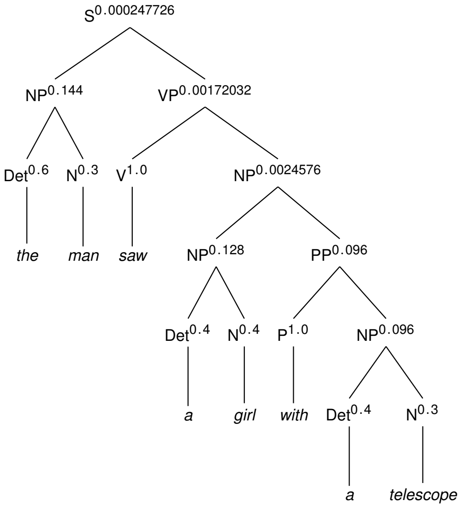
PCFG Example

1. $S \rightarrow NP\ VP$ 1.0
2. $NP \rightarrow Det\ N$ 0.8
3. $NP \rightarrow NP\ PP$ 0.2
4. $VP \rightarrow V\ NP$ 0.7
5. $VP \rightarrow VP\ PP$ 0.3
6. $PP \rightarrow P\ NP$ 1.0
7. $V \rightarrow saw$ 1.0
8. $N \rightarrow man$ 0.3
9. $N \rightarrow girl$ 0.4
10. $N \rightarrow telescope$ 0.3
11. $Det \rightarrow a$ 0.4
12. $Det \rightarrow the$ 0.6
13. $P \rightarrow with$ 1.0



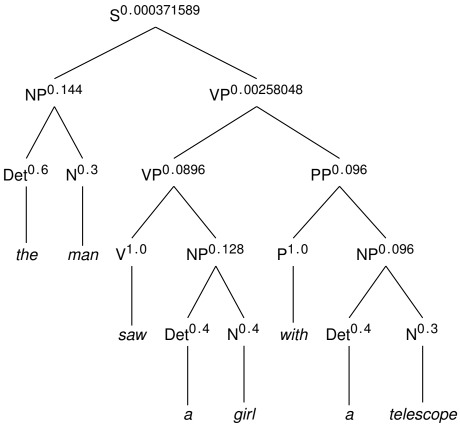
PCFG Example

1. $S \rightarrow NP\ VP$ 1.0
2. $NP \rightarrow Det\ N$ 0.8
3. $NP \rightarrow NP\ PP$ 0.2
4. $VP \rightarrow V\ NP$ 0.7
5. $VP \rightarrow VP\ PP$ 0.3
6. $PP \rightarrow P\ NP$ 1.0
7. $V \rightarrow saw$ 1.0
8. $N \rightarrow man$ 0.3
9. $N \rightarrow girl$ 0.4
10. $N \rightarrow telescope$ 0.3
11. $Det \rightarrow a$ 0.4
12. $Det \rightarrow the$ 0.6
13. $P \rightarrow with$ 1.0



PCFG Example

1. $S \rightarrow NP\ VP$ 1.0
2. $NP \rightarrow Det\ N$ 0.8
3. $NP \rightarrow NP\ PP$ 0.2
4. $VP \rightarrow V\ NP$ 0.7
5. $VP \rightarrow VP\ PP$ 0.3
6. $PP \rightarrow P\ NP$ 1.0
7. $V \rightarrow saw$ 1.0
8. $N \rightarrow man$ 0.3
9. $N \rightarrow girl$ 0.4
10. $N \rightarrow telescope$ 0.3
11. $Det \rightarrow a$ 0.4
12. $Det \rightarrow the$ 0.6
13. $P \rightarrow with$ 1.0



Parsing with PCFG

- Earley and CYK algorithms can be adapted to carry probabilities
- Best parse tree y^* for a sentence x

$$y^* = \underset{y \in \mathscr{T}(x)}{\operatorname{argmax}} P(x, y)$$

- $N-$best parse can be recovered with Viterbi-like algorithm

Learning PCFG Probabilities

- Given a treebank, with Maximum-Likelihood Estimation (MLE):

$$\mathbf{Pr}(A \to \beta) = \frac{\#(A \to \beta)}{\#(A)}$$

- When the grammar is large (e.g. by lexicalization), smoothing is necessary to overcome data sparseness

Inside-Outside Algorithm

- When there is no labeled data (treebank), probabilities of a PCFG can be updated to maximize the likelihood over a set of unlabeled sentences

$$\mathbf{Pr}^* = \operatorname*{argmax}_{\mathbf{Pr}} \prod_x P(x) = \operatorname*{argmax}_{\mathbf{Pr}} \prod_x \sum_{y \in \mathscr{T}(x)} P(x, y)$$

- An Expectation-Maximization procedure can be used to iteratively find $\mathbf{Pr}^*$

Inside Probability

Definition

Inside probability $\beta_j(p,q)$ is the probability of sequence $w_{p+1} \dots w_q$ being generated with a tree rooted by node N^j

$$\beta_j(p,q) = P(w_{p+1} \dots w_q | N^j_{pq})$$

- $\beta_1(0,n) = P(w_1 w_2 \dots w_n) \qquad N^1 = S$
- Calculation can be carried out bottom-up

$$\beta_j(k-1,k) = Pr(N^j \rightarrow w_k) \qquad N^j \in V_N \tag{1}$$

$$\beta_j(p,q) = \sum_{r,s} \sum_{d=p+1}^{q-1} Pr(N^j \rightarrow N^r\ N^s) \cdot \beta_r(p,d) \cdot \beta_s(d,q) \tag{2}$$

Outside Probability

Definition

Outside probability $\alpha_j(p, q)$ is the total probability of beginning with the start symbol and generating N^j_{pq} and all the words outside

$$\alpha_j(p, q) = P(w_1 \dots w_p, N^j_{pq}, w_{q+1} \dots w_n)$$

- N^j_{pq} means

$$N^j \stackrel{*}{\Rightarrow} w_{p+1} \dots w_q$$

- $P(w_1 w_2 \dots w_n, N^j_{pq}) = \alpha_j(p, q) \cdot \beta_j(p, q)$
- $P(w_1 w_2 \dots w_n) = \sum_j \alpha_j(k-1, k) Pr(N^j \rightarrow w_k)$ for any k

Outside Probability (cont.)

- Calculation is top-down

$$\alpha_j(0,n) = \begin{cases} 1 & N^j = S \\ 0 & otherwise \end{cases} \quad (3)$$

$$\begin{aligned}\alpha_j(p,q) = &\sum_{f,g} \sum_{q<e<n} \alpha_f(p,e) \cdot Pr(N^f \to N^j\ N^g) \cdot \beta_g(q,e) \\ &+ \sum_{f,g} \sum_{0<e<p} \alpha_f(e,q) \cdot Pr(N^f \to N^g\ N^j) \cdot \beta_g(e,p) \quad (4)\end{aligned}$$

Calculating Expected Counts

The expected times N^j is used in the derivation for sentence $w_1 \dots w_n$

$$E[N^j | w_1 \dots w_n] = \sum_{p=0}^{n-1} \sum_{q=p+1}^{n} P(N^j_{pq} | w_1 \dots w_n) \tag{5}$$

$$= \sum_{p=0}^{n-1} \sum_{q=p+1}^{n} \frac{P(N^j_{pq}, w_1 \dots w_n)}{P(w_1 \dots w_n)} = \sum_{p=0}^{n-1} \sum_{q=p+1}^{n} \frac{\alpha_j(p,q) \cdot \beta_j(p,q)}{P(w_1 \dots w_n)}$$

Calculating Expected Counts (cont.)

The expected times $N^j \rightarrow N^r N^s$ and N^j is used in the derivation for sentence $w_1 \ldots w_n$

$$E[N^j \rightarrow N^r N^s | w_1 \ldots w_n] = \sum_{p=0}^{n-1} \sum_{q=p+1}^{n} P(N^j_{pq}, N^j \rightarrow N^r N^s | w_1 \ldots w_n) \tag{6}$$

$$= \frac{\sum_{p=0}^{n-1} \sum_{q=p+1}^{n} \sum_{d=p+1}^{q-1} \alpha_j(p,q) \cdot Pr(N^j \rightarrow N^r N^s) \cdot \beta_r(p,d) \cdot \beta_s(d,q)}{P(w_1 \ldots w_n)}$$

Update Formula

For a single sentence, rule probabilities can be reestimated

$$\hat{Pr}(N^j \rightarrow N^r N^s) = \frac{E[N^j \rightarrow N^r N^s, N^j | w_1 \ldots w_n]}{E[N^j | w_1 \ldots w_n]} \tag{7}$$

$$= \frac{\sum_{p=0}^{n-1} \sum_{q=p+1}^{n} \sum_{d=p+1}^{q-1} \alpha_j(p,q) \cdot Pr(N^j \rightarrow N^r N^s) \cdot \beta_r(p,d) \cdot \beta_s(d,q)}{\sum_{p=0}^{n-1} \sum_{q=p+1}^{n} \alpha_j(p,q) \cdot \beta_j(p,q)}$$

Similarly, for unary rules,

$$\hat{Pr}(N^j \rightarrow w^k) = \frac{\sum_{h=1}^{n} \alpha_j(h-1,h) \cdot P(w_h = w^k) \cdot \beta_j(h-1,h)}{\sum_{p=0}^{n-1} \sum_{q=p+1}^{n} \alpha_j(p,q) \cdot \beta_j(p,q)} \tag{8}$$

Multiple Training Sentences

For each sentence $\vec{w}^i$ in the training corpus

$$f_i(p,q,j,r,s) = \frac{\sum_{d=p+1}^{q-1} \alpha_j(p,q) \cdot Pr(N^j \to N^r N^s) \cdot \beta_r(p,d) \cdot \beta_s(d,q)}{P(w_1 \ldots w_n)} \tag{9}$$

$$g_i(h,j,k) = \frac{\alpha_j(h-1,h) \cdot P(w_h = w^k) \cdot \beta_j(h-1,h)}{P(w_1 \ldots w_n)} \tag{10}$$

$$h_i(p,q,j) = \frac{\alpha_j(p,q) \cdot \beta_j(p,q)}{P(w_1 \ldots w_n)} \tag{11}$$

then

$$\hat{P}r(N^j \to N^r N^s) = \frac{\sum_{i=1}^{m} \sum_{p=0}^{n_i-1} \sum_{q=p+1}^{n_i} f_i(p,q,j,r,s)}{\sum_{i=1}^{m} \sum_{p=0}^{n_i-1} \sum_{q=p+1}^{n_i} h_i(p,q,j)} \tag{12}$$

$$\hat{P}r(N^j \to w^k) = \frac{\sum_{i=1}^{m} \sum_{h=1}^{n_i} g_i(h,j,k)}{\sum_{i=1}^{m} \sum_{p=0}^{n_i-1} \sum_{q=p+1}^{n_i} h_i(p,q,j)} \tag{13}$$

Inside-Outside Algorithm

Initialize an arbitrary set of rule probabilities $\mathbf{Pr^0}$

repeat

$F = G = H \Leftarrow 0$

for $\vec{w}^k = w_1^k \ldots w_n^k$ in the corpus **do**

Calculate inside probabilities $\beta_j(p, q)$

Calculate outside probabilities $\alpha_j(p, q)$

Accumulate counts F G and H

end for

Update rule probabilities $Pr^{i+1}(N^j \rightarrow N^r N^s)$ and $Pr^{i+1}(N^j \rightarrow w^h)$

until $|P_{Pr^{i+1}}(W) - P_{Pr^i}(W)| \leq \epsilon$

Outline

Overview

Basic Parsing Algorithms

- Parsing Strategies
- CYK Algorithm
- Earley's Algorithm

Parsing with Probabilistic Context-Free Grammar

- PCFG
- Inside-Outside Algorithm

Recent Advances in Parsing Technology

Statistical Constituent Parsing

- Collins parser [Collins, 1997]
- Reranking model [Charniak and Johnson, 2005]
- Self-training [McClosky et al., 2006]
- Latent-Variable PCFG [Petrov et al., 2006]

Statistical Dependency Parsing

- Graph-based approach [Eisner, 1996, McDonald et al., 2005]
  - Edge-factorized scoring model
  - Efficient algorithms to find maximal spanning tree
  - Allows non-projective dependency structures
- Transition-based approach [Nivre et al., 2007, Sagae and Tsujii, 2008]
  - (Near) deterministic parsing
  - Projective/pseudo-projective

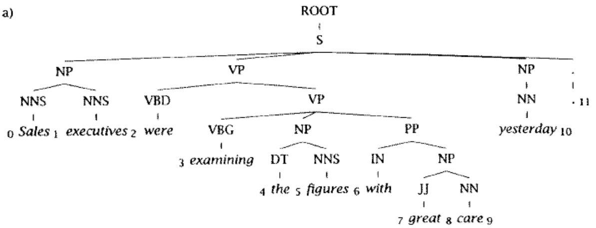
Parsing with Richer Formalisms

- TAG
- CCG
- LFG
- HPSG

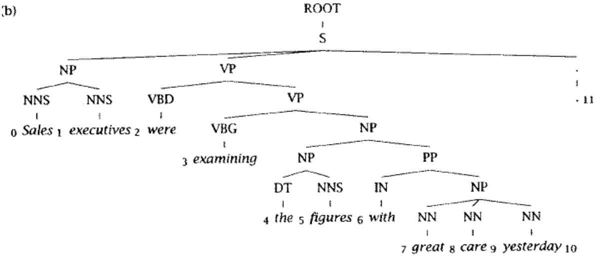
Parser Evaluation

- Evaluation against “gold-standard”
  - E.g. PARSEVAL
- Application-based evaluation

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%		

Domain Adaptability and Multilinguality

- Statistical parsing models usually performs well in in-domain tests and suffer significant accuracy drop when tested on out-of-domain data
- Differences between languages require different parsing models (morphology, word order, etc.)

Open Questions

- How relevant is linguistic study to the development of parsers?
- How do we evaluate a parser?
- How to make trade-offs between adequacy, accuracy and efficiency?

References I

Charniak, E. and Johnson, M. (2005).
Coarse-to-fine n-best parsing and maxent discriminative reranking.
In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan.

Collins, M. (1997).
Three Generative, Lexicalised Models for Statistical Parsing.
In *Proceedings of the 35th annual meeting of the association for computational linguistics*, pages 16–23, Madrid, Spain.

Eisner, J. (1996).
Three new probabilistic models for dependency parsing: An exploration.
In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, Denmark.

References II

McClosky, D., Charniak, E., and Johnson, M. (2006).
Effective self-training for parsing.
In *Proceedings of HLT-NAACL-2006*, pages 152–159, New York, USA.

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005).
Non-Projective Dependency Parsing using Spanning Tree Algorithms.
In *Proceedings of HLT-EMNLP 2005*, pages 523–530, Vancouver, Canada.

Nivre, J., Nilsson, J., Hall, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007).
Maltparser: A language-independent system for data-driven dependency parsing.
Natural Language Engineering, 13(1):1–41.

References III

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006).
Learning accurate, compact, and interpretable tree annotation.
In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia.

Sagae, K. and Tsujii, J. (2008).
Shift-reduce dependency dag parsing.
In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 753–760, Manchester, UK.