Probability estimation used in Tree Tagger: Suffix Lexicon

- Probability of unknown words are estimated according to its suffix, p(t | w)

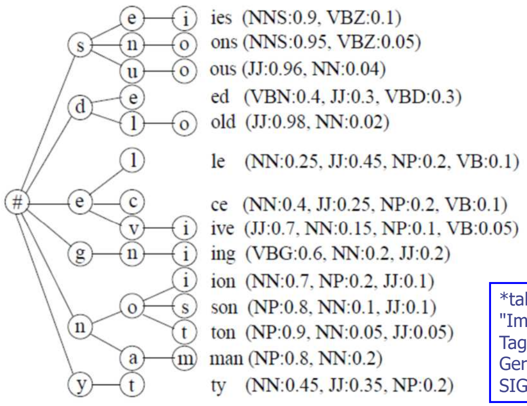


Figure 2: A sample suffix tree of maximal length 3.

*taken from: Helmut Schmid, "Improvements in Part-ofSpeech Tagging with an Application to German," Proceedings of the ACL SIGDAT-Workshop. March 1995.

Estimation of the probabilities 確率値の推定法

- Unsupervised learning (no tagged data)
  - Baum-Welch algorithm (Hidden Markov Model)：隠れマルコフモデル
- Supervised learning (with tagged data)
  - Maximum likelihood（最尤推定）
  - Decision Trees（決定木）
  - Maximum Entropy model（最大エントロピー法）
  - Conditional Random Fields（CRF:条件付き確率場）
  - Recurrent Neural Networks（再帰型NN）+ CRF

Contents of this lecture

- Morphological Analysis
  - Tokenization
  - Part-of-speech tagging
- Syntactic Parsing Algorithms
  - Phrase structure parsing
  - Word dependency parsing

Formal Grammars（形式文法）

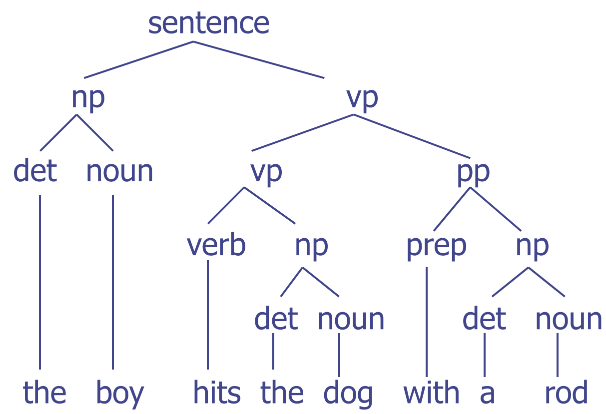
- G=(N, T, P, S)
  - N: a finite set of nonterminal symbols (phrases)
  - T: a finite set of terminal symbols (words)
  - P: a finite set of grammar rules
  - S: start symbol ($S \in N$)
- Grammar rule: $\alpha \rightarrow \beta$ （文法規則の形）
  - Context free grammar（文脈自由文法）：
    $A \rightarrow \beta$ ($\beta \in (N \cup T)^*$)
  - Regular grammar（正規文法）：
    $A \rightarrow B$ or $A \rightarrow aB$
    ($A, B \in N$, $a \in T$)」

A sample context-free grammar

(a) sentence → np, vp
(b) np → det, noun
(c) vp → verb, np
(d) vp → vp, pp
(e) pp → prep, np
(f) det → [the] | [a]
(g) noun → [boy] | [dog] | [rod]
(h) verb → [hits]
(i) prep → [with]

Note:
- [the] means “the” is a terminal symbol
- “|” means “or”

A Phrase Structure Tree
句構造木



Parsing Algorithms
統語解析アルゴリズム

Context free grammar parsing

Concurrent

CKY
Chart Parsing
- Bottom-up
- Top-down
  = Earley parsing

Generalized LR parsing
(Tomita algorithm)

Backtracking

Top-down
Bottom-up
(Shift-reduce)
Left-corner

Deterministic
(Look-ahead)
LR parsing

Dependency parsing

Deterministic

Transition-based
(Shift-reduce)

Concurrent

CKY
(Eisner algorithm)

Global optimization

MST (Maximum Spanning Tree algorithm)

ILP (Integer Linear Programming)

Concurrent Parsing Algorithm

- CKY(Cocke-Kasami-Younger) algorithm
  - Assumes grammar rules to be in Chomsky Normal Form (right-hand side of rules have no more than two children)
    - Grammar rules: A → B, C or A → [a]
  - CKY algorithm is an application of bottom-up dynamic programming to grammar rules
    - Box b[i,j] (1≦i≦j≦n) is defined as the set of phrases that span between i-th and j-th words of the input sentence (length n)
    - Construction of b[i,j] by bottom-up DP method:
      b[i,j] = {A | rule A→B,C exists, and B∈b[i,k] ∧ C∈b[k+1,j] (i≦k≦j-1) }

CKY Table

- Table construction goes toward right-upwards

b[1,1]	b[1,2]	b[1,3]	…	b[1,j]	…	b[1,n]
	b[2,2]	b[2,3]	…	b[2,j]	…	b[2,n]
		b[3,3]	…	b[3,j]	…	b[3,n]
			…	…	…	…
				b[j,j]	…	b[i,n]
					…	…
						b[n,n]

CKY Table

	b[i,i]	b[i,i+1]	…	b[i,j-1]	b[i,j]	
			…		b[i+1,j]	
				…	b[i+1,j]	
					b[j,j]	

$b[i,j] = \{A \mid \text{rule } A \rightarrow B,C \text{ exists, and}$

$B \in b[i,k] \wedge C \in b[k+1,j]$

$(i \leqq k \leqq j-1)\ \}$

CKY Table

	b[i,i]	b[i,i+1]	…	b[i,j-1]	b[i,j]	
			…		b[i+1,j]	
				…	b[i+2,j]	
					b[j,j]	

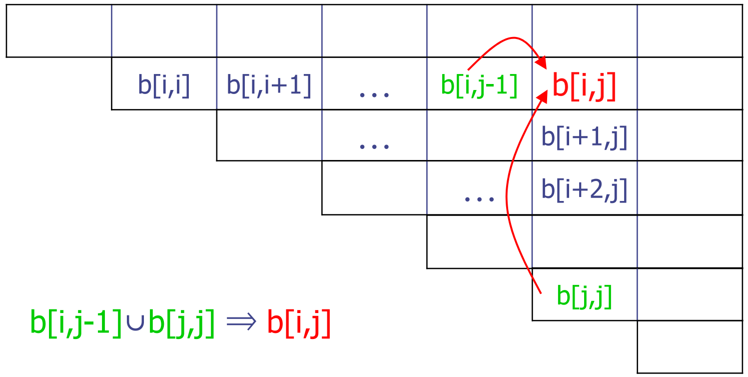
$b[i,i] \cup b[i+1,j] \Rightarrow b[i,j]$

CKY Table

	b[i,i]	b[i,i+1]	…	b[i,j-1]	b[i,j]	
			…		b[i+1,j]	
				…	b[i+2,j]	
					b[j,j]	

$b[i,i+1] \cup b[i+2,j] \Rightarrow b[i,j]$

CKY Table



$b[i,j-1] \cup b[j,j] \Rightarrow b[i,j]$

Sample Process of CKY parsing algorithm

input sentence: The boy hits the dog with a rod.

Grammar rules:

sentence → np, vp
np → det, noun
vp → verb, np
vp → vp, pp
pp → prep, np
det → [the] | [a]
noun → [boy] | [dog] | | [rod] | [hits]
verb → [hits]
prep → [with]

→ CKY.ppt

CKY parsing algorithm requires $O(n^3)$ processing time, while the backtracking algorithms needs $O(c^n)$ time in worst case. Time complexity of Chart parsing is $O(n^3)$.

Chart Parsing

- An extension of CKY parsing to a general context-free grammars
  - CKY parsing handles grammar rules of Chomsky normal form (right-hand side of grammar rule has at most 2 non-terminal symbols)
  - General rules can have 3 or more non-terminal symbols on the right-hand side

  ⇒ introduce special notation indicating which part has been analyzed

  Example: For a rule like vp → verb np pp

  the following notations are used ("・" shows until which part has been analyzed):

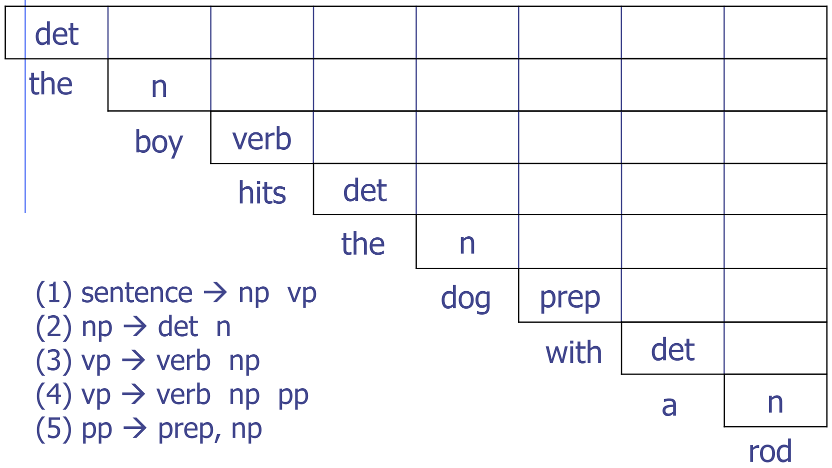
  vp → verb ・ np pp, vp → verb np ・ pp

Bottom-up Chart Parsing

- Mainly consisting of two operations:
  - Invocation (of a new rule): If B is in b[i,j], and there are rules like A → B C … D having B as the first symbol on the right-hand, then for each rule add a dotted rule like A → B ・ C … D , in b[i,j]
  - Update (of a dotted rule): If there is a dotted rule, A → B … ・ C D … E , in b[i,j], and there is a completed symbol C in b[j+1,k], then add A → B … C ・ D … E in b[i,k].

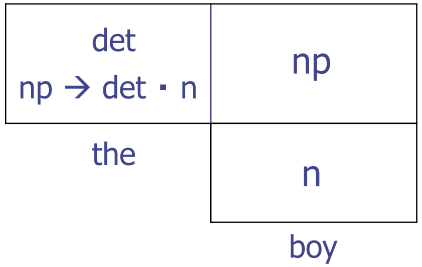
    If the dotted rule included in b[i,j] is A → B … ・ C (having C as the final symbol), add a completed symbol A in b[i,k].

Bottom-up Chart Parsing for
"the boy hits the dog with a rod" (Initial state)



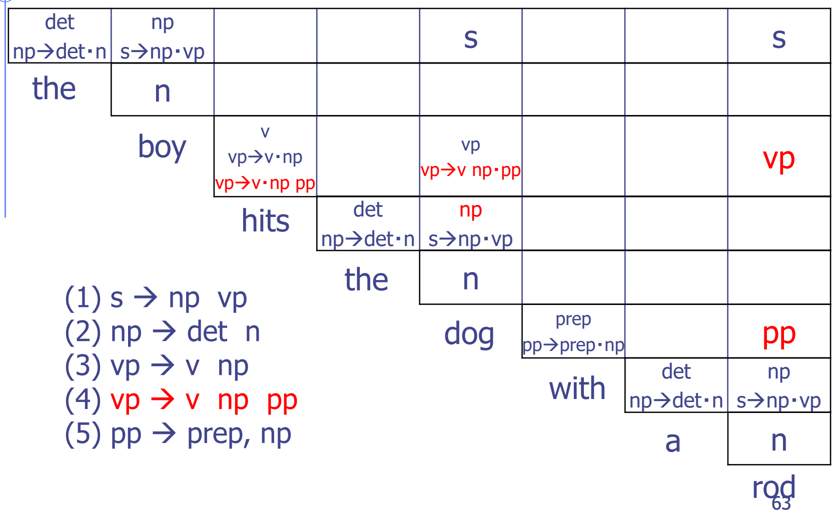
(1) sentence → np vp
(2) np → det n
(3) vp → verb np
(4) vp → verb np pp
(5) pp → prep, np

Steps of Bottom-up Chart Parsing



(2) np → det n

Bottom-up Chart Parsing for
"the boy hits the dog with a rod"



(1) s → np vp
(2) np → det n
(3) vp → v np
(4) vp → v np pp
(5) pp → prep, np