Lexicalized and Probabilistic Parsing – Part 1

ICS 482 Natural Language Processing

Lecture 14: Lexicalized and Probabilistic Parsing – Part 1

Husni Al-Muhtaseb

بسم الله الرحمن الرحيم

ICS 482 Natural Language Processing

Lecture 14: Lexicalized and Probabilistic Parsing – Part 1

Husni Al-Muhtaseb

NLP Credits and Acknowledgment

These slides were adapted from presentations of the Authors of the book

---

SPEECH and LANGUAGE PROCESSING:

An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition

and some modifications from presentations found in the WEB by several scholars including the following

NLP Credits and Acknowledgment

If your name is missing please contact me
muhtaseb
At
Kfupm.
Edu.
sa

NLP Credits and Acknowledgment

Husni Al-Muhtaseb
James Martin
Jim Martin
Dan Jurafsky
Sandiway Fong
Song young in
Paula Matuszek
Mary-Angela Papalaskari
Dick Crouch
Tracy Kin
L. Venkata Subramaniam
Martin Volk
Bruce R. Maxim
Jan Hajič
Srinath Srinivasa
Simeon Ntafos
Paolo Pirjanian
Ricardo Vilalta
Tom Lenaerts

Heshaam Feili
Björn Gambäck
Christian Korthals
Thomas G. Dietterich
Devika Subramanian
Duminda Wijesekera
Lee McCluskey
David J. Kriegman
Kathleen McKeown
Michael J. Ciaraldi
David Finkel
Min-Yen Kan
Andreas Geyer-Schulz
Franz J. Kurfess
Tim Finin
Nadjet Bouayad
Kathy McCoy
Hans Uszkoreit
Azadeh Maghsoodi

Khurshid Ahmad
Staffan Larsson
Robert Wilensky
Feiyu Xu
Jakub Piskorski
Rohini Srihari
Mark Sanderson
Andrew Elks
Marc Davis
Ray Larson
Jimmy Lin
Marti Hearst
Andrew McCallum
Nick Kushmerick
Mark Craven
Chia-Hui Chang
Diana Maynard
James Allan

Martha Palmer
julia hirschberg
Elaine Rich
Christof Monz
Bonnie J. Dorr
Nizar Habash
Massimo Poesio
David Goss-Grubbs
Thomas K Harris
John Hutchins
Alexandros Potamianos
Mike Rosner
Latifa Al-Sulaiti
Giorgio Satta
Jerry R. Hobbs
Christopher Manning
Hinrich Schütze
Alexander Gelbukh
Gina-Anne Levow
Guitao Gao
Qing Ma
Zeynep Altan

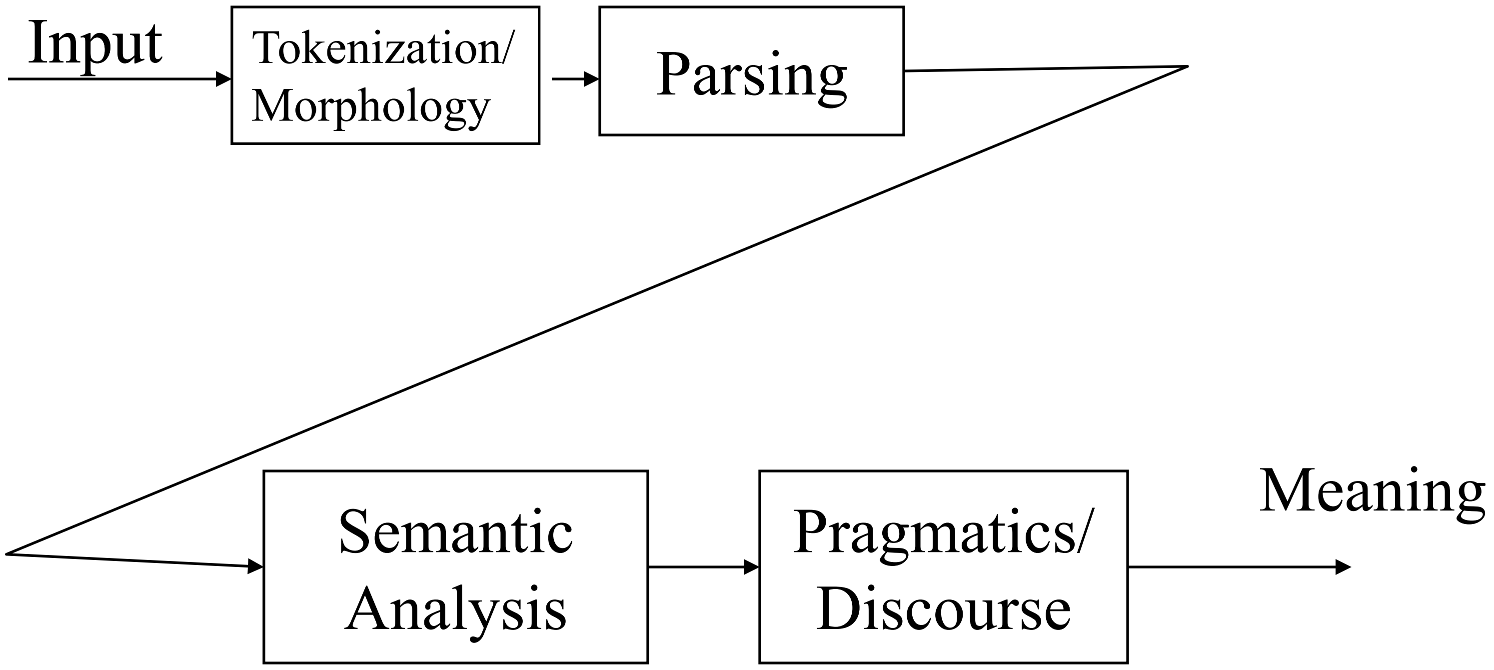
Previous Lectures

- Introduction and Phases of an NLP system
- NLP Applications - Chatting with Alice
- Finite State Automata, Regular Expressions &languages
- Morphology: Inflectional & Derivational
- Parsing and Finite State Transducers
- Stemming & Porter Stemmer
- Statistical NLP – Language Modeling
- N Grams, Smoothing : Add-one & Witten-Bell
- Parts of Speech - Arabic Parts of Speech
- Syntax: Context Free Grammar (CFG) & Parsing
- Parsing: Top-Down, Bottom-Up, Top-down parsing with bottom-up filtering
- Earley's Algorithm – Pop quiz on Earley's Algorithm

Today's Lecture

- Quiz 2 – 25 minutes
- Lexicalized and Probabilistic Parsing

Natural Language Understanding



Lexicalized and Probabilistic Parsing

- Resolving structural ambiguity: choose the most probable parse
- Use lexical dependency (relationship between words)

Probability Model (1)

- A derivation (tree) consists of the set of grammar rules that are in the tree

- The probability of a derivation (tree) is just the product of the probabilities of the rules in the derivation

Probability Model (1.1)

- The probability of a word sequence (sentence) is the probability of its tree in the unambiguous case
- It's the sum of the probabilities of the trees in the ambiguous case

Formal

$$P(T,S) = \prod_{n \in T} p(r(n))$$

$$P(T,S) = P(T)P(S \mid T)$$

Since $P(S \mid T) = 1, \;\; P(T,S) = P(T)$

T	Parse tree
r	rule
n	node in the pars tree
$p(r(n))$	probability of the rule expanded from node n

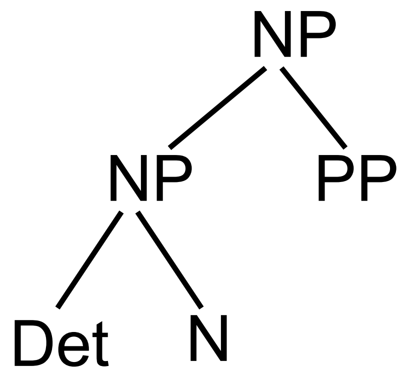
Probability Model

- Attach probabilities to grammar rules
- The expansions for a given non-terminal sum to 1

VP → Verb	.55
VP → Verb NP	.40
VP → Verb NP NP	.05

Probabilistic Context-Free Grammars

NP → Det N	:	0.4
NP → NPposs N	:	0.1
NP → Pronoun	:	0.2
NP → NP PP	:	0.1
NP → N	:	0.2



P(subtree above) = 0.1 x 0.4 = 0.04

Probabilistic Context-Free Grammars

- PCFG
- Also called Stochastic CFG (SCFG)
- $G = (N, \Sigma, P, S, D)$
  - A set of non-terminal symbols (or variables) N
  - A set of terminal symbols Σ $(N \cap \sum = \emptyset)$
  - A set of productions P, each of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$
    - $*$ denotes finite length of the infinite set of strings $(\Sigma \cup N)$
  - A designated start symbol $S \in N$
  - A function D that assigns a probability to each rule in P
- $P(A \rightarrow \alpha)$ or $P(A \rightarrow \alpha | A)$

Probabilistic Context-Free Grammars

$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.05] | the [.80] | a [.15]	
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$	[.10]
$S \rightarrow VP$	[.05]	$Noun \rightarrow flights$	[.50]
$NP \rightarrow Det\ Nom$	[.20]	$Noun \rightarrow meal$	[.40]
$NP \rightarrow Proper\text{-}Noun$	[.35]	$Verb \rightarrow book$	[.30]
$NP \rightarrow Nom$	[.05]	$Verb \rightarrow include$	[.30]
$NP \rightarrow Pronoun$	[.40]	$Verb \rightarrow want$	[.40]
$Nom \rightarrow Noun$	[.75]	$Aux \rightarrow can$	[.40]
$Nom \rightarrow Noun\ Nom$	[.20]	$Aux \rightarrow does$	[.30]
$Nom \rightarrow Proper\text{-}Noun\ Nom$	[.05]	$Aux \rightarrow do$	[.30]
$VP \rightarrow Verb$	[.55]	$Proper\text{-}Noun \rightarrow TWA$	[.40]
$VP \rightarrow Verb\ NP$	[.40]	$Proper\text{-}Noun \rightarrow Denver$	[~~.40~~].60
$VP \rightarrow Verb\ NP\ NP$	[.05]	$Pronoun \rightarrow you$ [.40] | I [.60]	

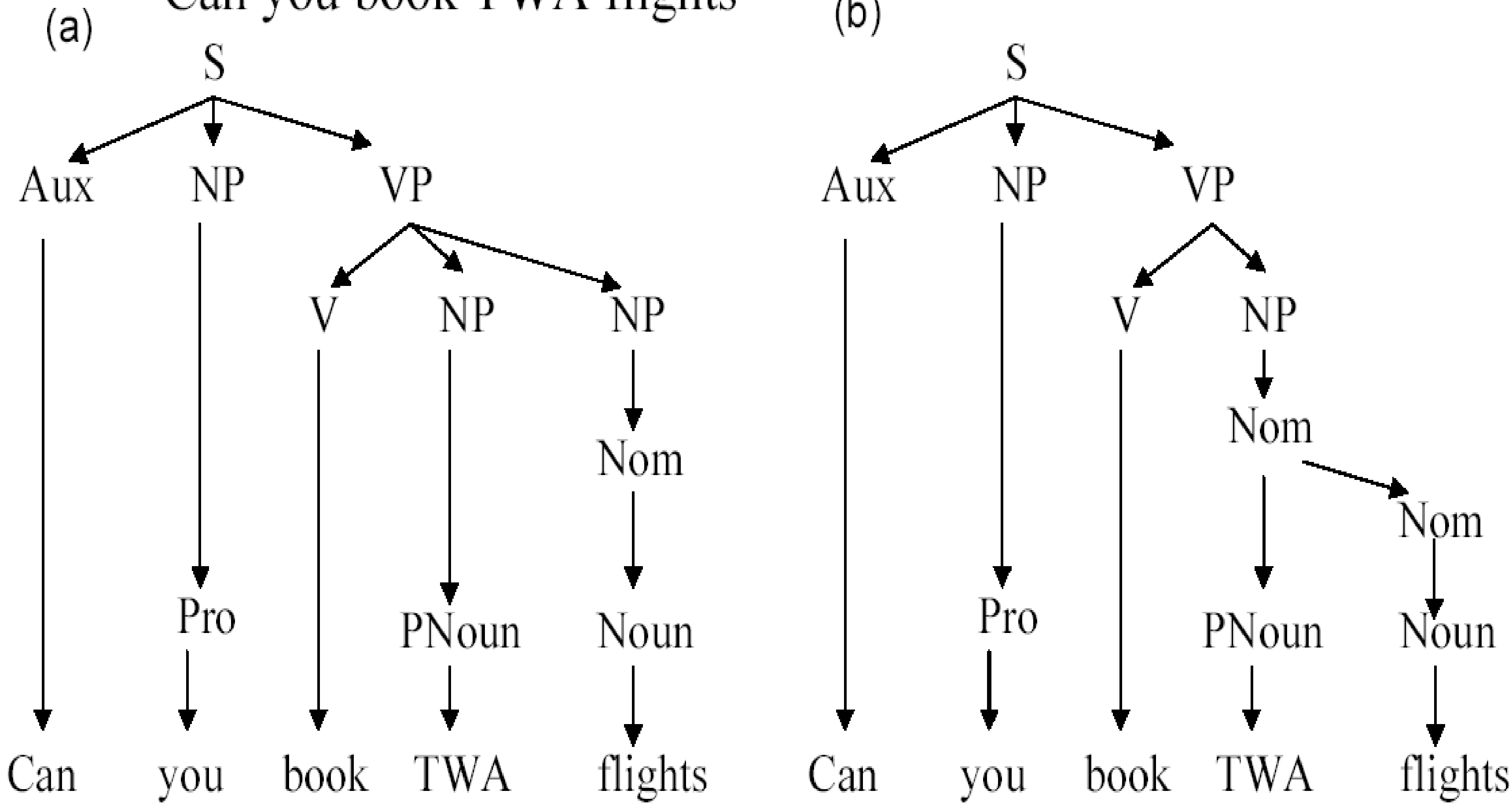
English practice

- What do you understand from the sentence:

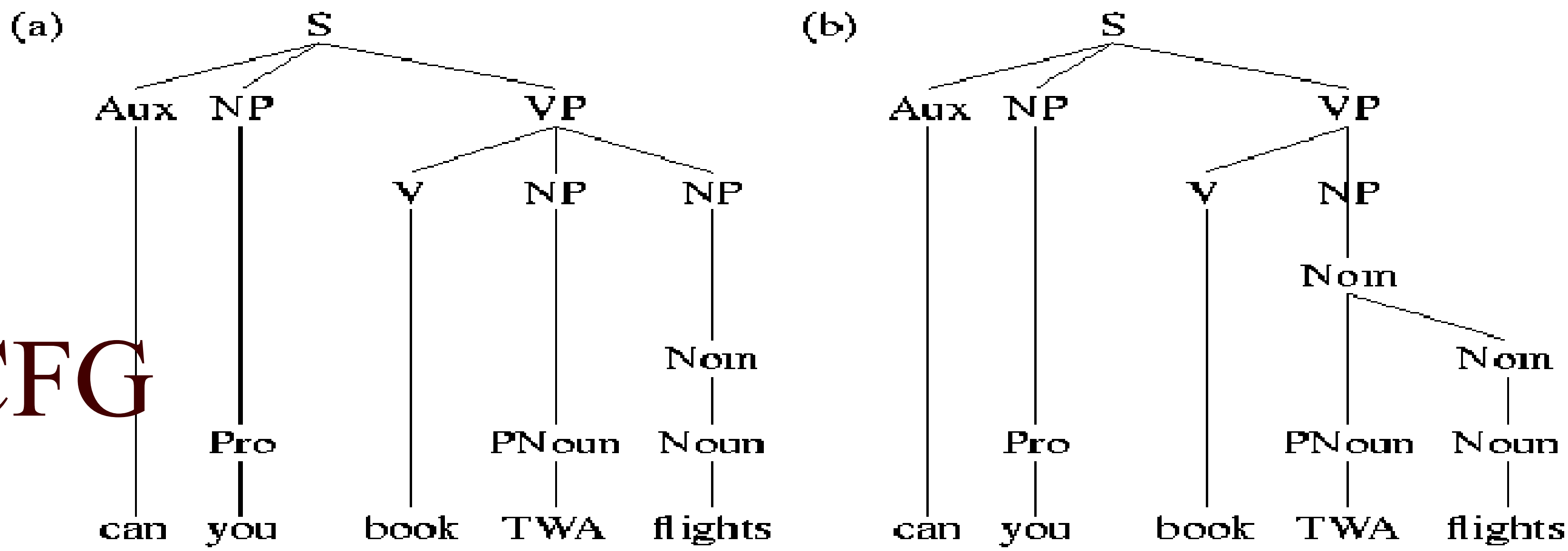
  “Can you book TWA flights?”

  - Can you book flights on behalf of TWA?
    - → [TWA] [flights]
  - Can you book flights run by TWA?
    - → [TWA flights]

Can you book TWA flights



PCFG



Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP NP	.05
NP	→	Nom	.05
NP	→	PNoun	.35
Nom	→	Noun	.75
Aux	→	Can	.40
~~NP~~	~~→~~	~~Pro~~	~~.40~~
Pro	→	you	.40
Verb	→	book	.30
PNoun	→	TWA	.40
Noun	→	flights	.50

Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP	.40
NP	→	Nom	.05
Nom	→	PNoun Nom	.05
Nom	→	Noun	.75
Aux	→	Can	.40
~~NP~~	~~→~~	~~Pro~~	~~.40~~
Pro	→	you	.40
Verb	→	book	.30
Pnoun	→	TWA	.40
Noun	→	flights	.50

PCFG

$$P(T,S) = \prod_{n \in T} p(r(n))$$

$$P(T,S) = P(T)P(S \mid T)$$

Since $P(S \mid T) = 1, \;\; P(T,S) = P(T)$

T	Parse tree
r	rule
n	node in the pars tree
$p(r(n))$	propability of the role expanded from node n

$$P(T_l) = .15 \times .40 \times .05 \times .05 \times .35 \times .75 \times .40 \times .40 \times .30 \times .40 \times .50 = 3.78 \times 10^{-7}$$

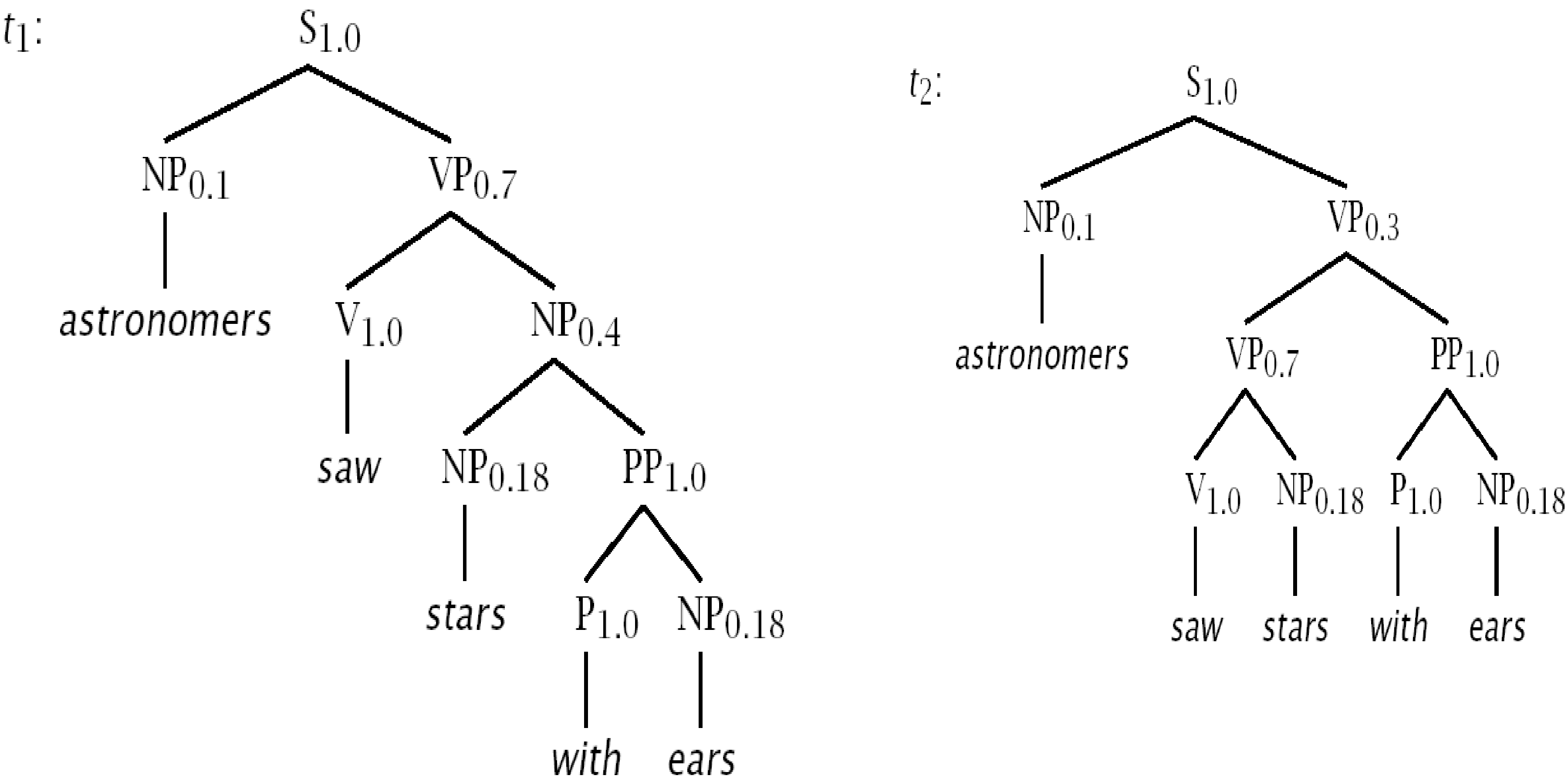
$$P(T_r) = .15 \times .40 \times .40 \times .05 \times .05 \times .75 \times .40 \times .40 \times .30 \times .40 \times .50 = 4.32 \times 10^{-7}$$

$$\hat{T}(S) = \arg\max_T P(T \mid S) = \arg\max_T \frac{P(T,S)}{P(S)} = \arg\max_T P(T,S) = \arg\max_T P(T)$$

A simple PCFG (in CNF)

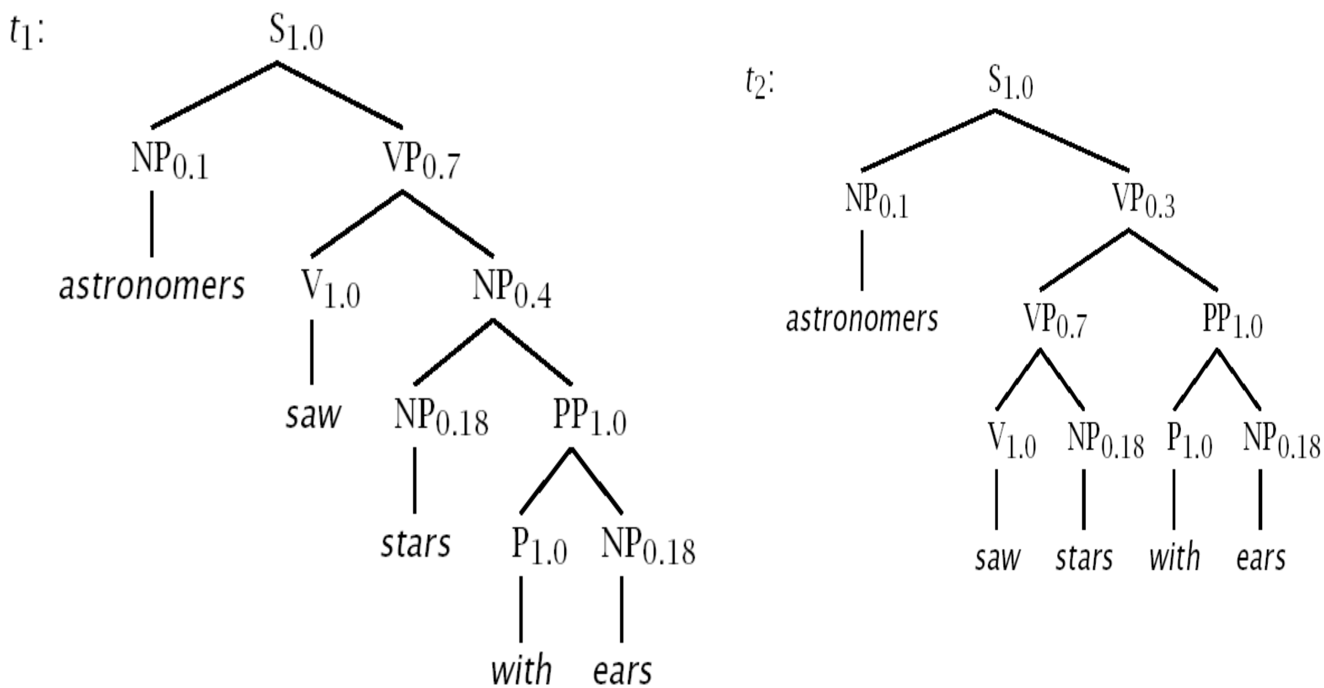
- $S \rightarrow NP\ VP\ 1.0$
- $PP \rightarrow P\ NP\ 1.0$
- $VP \rightarrow V\ NP\ 0.7$
- $VP \rightarrow VP\ PP\ 0.3$
- $P \rightarrow$ with 1.0
- $V \rightarrow$ saw 1.0
- $NP \rightarrow NP\ PP\ 0.4$
- $NP \rightarrow$ astronomers 0.1
- $NP \rightarrow$ ears 0.18
- $NP \rightarrow$ saw 0.04
- $NP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

Ex: Astronomers saw stars with ears



The two parse trees' probabilities & the sentence probability

- $P(t_1) = 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 1.0 \times 0.18 = 0.0009072$
- $P(t_2) = 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18 = 0.0006804$
- $P(w_{15}) = P(t_1) + P(t_2) = 0.0015876$



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

- $P(t_1) = 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 1.0 \times 0.18 = 0.0009072$
- $P(t_2) = 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18 = 0.0006804$
- $P(w_{15}) = P(t_1) + P(t_2) = 0.0015876$

Probabilistic CFGs

- The probabilistic model
  - Assigning probabilities to parse trees
- Getting the probabilities for the model
- Parsing with probabilities
  - Slight modification to dynamic programming approach
  - Task is to find the max probability tree for an input

Getting the Probabilities

- From an annotated database (a treebank)
- Learned from a corpus

Treebank

- Get a large collection of parsed sentences
- Collect counts for each non-terminal rule expansion in the collection
- Normalize
- Done

Learning

- What if you don’t have a treebank (and can’t get one)
- Take a large collection of text and parse it.
- In the case of syntactically ambiguous sentences collect all the possible parses
- Prorate the rule statistics gathered for rules in the ambiguous case by their probability
- Proceed as you did with a treebank.
- **Inside-Outside** algorithm

Assumptions

- We're assuming that there is a grammar to be used to parse with.
- We're assuming the existence of a large robust dictionary with parts of speech
- We're assuming the ability to parse (i.e. a parser)
- Given all that… we can parse probabilistically

Typical Approach

- Bottom-up dynamic programming approach
- Assign probabilities to constituents as they are completed and placed in the table
- Use the max probability for each constituent going up

Max probability

- Say we're talking about a final part of a parse
  - $S_0 \rightarrow NP_i VP_j$

The probability of the S is…

P(S → NP VP)*P(NP)*P(VP)

The green stuff is already known. We're doing bottom-up parsing

Max

- The P(NP) is known.
- What if there are multiple NPs for the span of text in question (*0* to *i*)?
- Take the max (Why?)
- Does not mean that other kinds of constituents for the same span are ignored (i.e. they might be in the solution)

Probabilistic Parsing

- Probabilistic CYK (Cocke-Younger-Kasami) algorithm for parsing PCFG
- Bottom-up dynamic programming algorithm
- Assume PCFG is in Chomsky Normal Form (production is either $A \rightarrow B\ C$ or $A \rightarrow a$)

Chomsky Normal Form (CNF)

All rules have form:

$$A \rightarrow BC \quad \text{and} \quad A \rightarrow a$$

Non-Terminal   Non-Terminal   terminal

Examples:

$$S \to AS$$
$$S \to a$$
$$A \to SA$$
$$A \to b$$

Chomsky Normal Form

$$S \to AS$$
$$S \to AAS$$
$$A \to SA$$
$$A \to aa$$

Not Chomsky Normal Form

Observations

- Chomsky normal forms are good for parsing and proving theorems
- It is possible to find the Chomsky normal form of any context-free grammar

Probabilistic CYK Parsing of PCFGs

- CYK Algorithm: bottom-up parser
- Input:
  - A Chomsky normal form PCFG, G= (N, Σ, P, S, D) Assume that the N non-terminals have indices 1, 2, …, |N|, and the start symbol S has index 1
  - n words $w_1, \ldots, w_n$
- Data Structure:
  - A dynamic programming array $\pi[i,j,a]$ holds the maximum probability for a constituent with non-terminal index a spanning words $i..j$.
- Output:
  - The maximum probability parse $\pi[1,n,1]$

Base Case

- CYK fills out $\pi[i,j,a]$ by induction
- Base case
  - Input strings with length = 1 (individual words w_i)
  - In CNF, the probability of a given non-terminal A expanding to a single word w_i must come only from the rule $A \rightarrow w_i$ i.e., $P(A \rightarrow w_i)$

Probabilistic CYK Algorithm [Corrected]

Function CYK(*words, grammar*)
  return the most probable parse and its probability
For i ←1 **to** *num_words*
  for $a \leftarrow 1$ **to** *num_nonterminals*
    If $(A \rightarrow w_i)$ is in grammar **then** $\pi[i, i, a] \leftarrow P(A \rightarrow w_i)$
For *span* ←2 **to** *num_words*
  For *begin* ←1 **to** *num_words* – *span* + 1
    end ←*begin* + *span* – 1
    For *m* ←*begin* **to** *end* – 1
     For *a* ←1 **to** *num_nonterminals*
      For *b* ←1 **to** *num_nonterminals*
       For *c* ←1 **to** *num_nonterminals*
        $prob \leftarrow \pi[begin, m, b] \times \pi[m+1, end, c] \times P(A \rightarrow BC)$
        If $(prob > \pi[begin, end, a])$ **then**
          $\pi[begin, end, a] = prob$
          $back[begin, end, a] = \{m, b, c\}$
Return *build_tree(back[1, num_words, 1]),* π*[1, num_words, 1]*

The CYK Membership Algorithm

Input:

- Grammar G in Chomsky Normal Form
- String w

Output:

find if $w \in L(G)$

The Algorithm

Input example:

- Grammar G:

$$S \to AB$$

$$A \to BB$$

$$A \to a$$

$$B \to AB$$

$$B \to b$$

- String : w $aabbb$

aabbb

All substrings of length 1	a	a	b	b	b
All substrings of length 2	aa	ab	bb	bb	
All substrings of length 3	aab	abb	bbb		
All substrings of length 4	aabb	abbb			
All substrings of length 5	aabbb				

$$S \rightarrow AB$$

$$A \rightarrow BB$$

$$A \rightarrow a$$

$$B \rightarrow AB$$

$$B \rightarrow b$$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
aab	abb	bbb		
aabb	abbb			
aabbb				

$$S \rightarrow AB$$

$$A \rightarrow BB$$

$$A \rightarrow a$$

$$B \rightarrow AB$$

$$B \rightarrow b$$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
	S,B	A	A	
aab	abb	bbb		
aabb	abbb			
aabbb				

$S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

a A	a A	b B	b B	b B
aa	ab S,B	bb A	bb A	
aab S,B	abb A	bbb S,B		
aabb A	abbb S,B			
aabbb S,B				

Therefore: $aabbb \in L(G)$

CYK Algorithm for Deciding Context Free Languages

IDEA: For each substring of a given input *x*, find all variables which can derive the substring. Once these have been found, telling which variables generate *x* becomes a simple matter of looking at the grammar, since it's in Chomsky normal form

Thank you

- السلام عليكم ورحمة الله