Natural Language Processing

AIMA Ch 23

[Some slides were adapted from slides by James H. Martin at U Colorado Boulder, by Albert Gatt at University of Malta, and by Eric Xing of CMU]

Additional Reference

- [SLP] Speech and Language Processing, Daniel Jurafsky and James H. Martin
  - 2nd edition, 2008
  - 3rd edition, Oct. 2019

- Sequence labeling
  - [SLP 2nd ed.] Ch 5, 6
  - [SLP 3rd ed.] Ch 8, 9
- Parsing
  - [AIMA] Ch.23
  - [SLP 2nd ed.] Ch 12, 13, 14
  - [SLP 3rd ed.] Ch 12, 13, 14, 15

Natural Language Processing

- Get computers to perform useful and interesting tasks involving human languages.
  - Understanding
  - Generation
- Big applications
  - Question answering, conversational agents (ChatBot)
  - Financial document processing
  - Machine translation
  - News generation

Levels of NLP Research

Phonetics and phonology	knowledge about linguistic sounds
Morphology	knowledge of the meaningful components of words
Syntax	knowledge of the structural relationships between words
Lexical semantics	knowledge of word meaning
Compositional semantics	knowledge of the meaning of sentences
Pragmatics	knowledge of the relationship of meaning to the goals and intentions of the speaker
Discourse	knowledge about linguistic units larger than a single sentence

Sequence Labeling

Sequence Labeling

- Problem Definition
  - Known
    - A set of labels $C = \{c_1, c_2, \dots, c_J\}$
  - Input
    - Sentence $s = \{x^1, x^2, \dots, x^m\}$
  - Output
    - For each word x^i, predict a label $c^i \in C$

Examples

- Part-of-speech tagging
  - Input

Pierre	Vinken	,	61	years	old	,	will	join	…

  - Output

NNP	NNP	,	CD	NNS	JJ	,	MD	VB

NNP = Proper noun, singular

CD = Cardinal number

NNS = Noun, plural

JJ = Adjective

…

Examples

- Chinese word segmentation
  - Input

    瓦 里 西 斯 的 船 只 中 …

  - Output

B	I	I	E	S	B	E	S	
(瓦	里	西	斯)	(的)	(船	只)	(中)	…

B = beginning of a word

I = inside of a word

E = end of a word

S = single character word

Examples

- Named entity recognition
  - Input

Michael	Jeffrey	Jordan	was	born	in	Brooklyn	…

  - Output

B-PER	I-PER	E-PER	O	O	O	S-LOC

Michael Jeffrey Jordan — Person

Brooklyn — Location

B = beginning of an entity
I = inside of an entity
E = end of an entity
S = single word entity
O = outside of any entity

-PER = person
-LOC = location
-ORG = organization
…

Examples

- Semantic role labeling
  - Input

The cat loves hats …

  - Output

B-ARG0 E-ARG0 S-PRED S-ARG1

The cat $\xleftarrow{\text{arg0}}$ loves $\xrightarrow{\text{arg1}}$ hats

B = beginning of an entity	-PRED = predicate
I = inside of an entity	-ARG0 = agent
E = end of an entity	-ARG1 = patient
S = single word entity	…
O = outside of any entity	

The simplest method

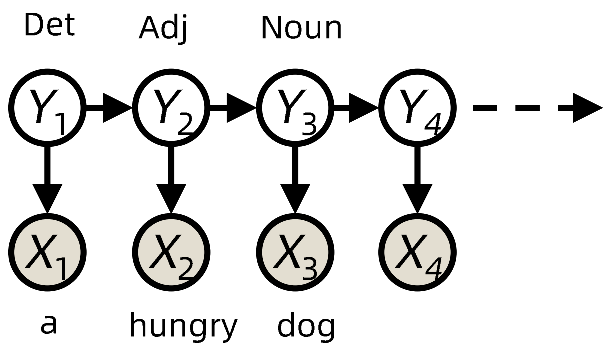
- For each word, predict its most frequent label
  - 90% accuracy on POS tagging!
  - Disadvantages:
    1. It does not consider the contextual info
       - “book a flight” vs. “read a book”
       - 我骑车差点摔倒，好在我一把把把把住了
    2. It does not consider relations between adjacent labels
       - In BIOES: "B-I" and "B-E" are OK, but "B-O" and "B-S" are not

Methods

- Hidden Markov Models (HMM)
- Max-Entropy Markov Models (MEMM)
- Conditional Random Fields (CRF)

Hidden Markov Model (HMM)

- Variables
  - X: word
  - Y: label (hidden state)
- Parameters
  - Transition model $P(y_t|y_{t-1})$
  - Emission model $P(x_t|y_t)$
  - Initial distribution $P(y_1)$
    - Can be seen as transition from Y_0=START to Y_1
  - Final distribution $P(y_n)$
    - Can be seen as transition from Y_n to Y_{n+1}=STOP



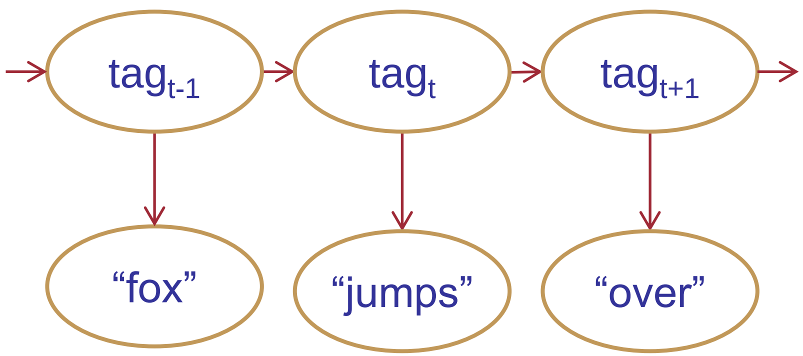
HMM Example

Transition

Y_{t-1}	$P(Y_t \mid Y_{t-1})$			
	N	V	P	…
START	0.5	0.1	0.1	…
N	0.4	0.3	0.1	…
V	0.5	0	0.3	…
P	0.3	0.1	0	…
…	…	…	…	…

Emission

Y_t	$P(X_t \mid Y_t)$			
	"fox"	"dog"	"run"	…
N	0.02	0.03	0.01	…
V	0	0	0.05	…
P	0	0	0	…
…	…	…	…	…



HMM Inference

- Find the most likely label sequence of the input sentence
  - $\arg\max_{y_{0:t}} P(y_{0:t} \mid x_{1:t})$
- Algorithm?
  - Viterbi algorithm

$$\boldsymbol{m}_{1:t+1} = \text{VITERBI}(\boldsymbol{m}_{1:t}, e_{t+1})$$

$$= P(e_{t+1}|X_{t+1}) \max_{x_t} P(X_{t+1} \mid x_t)\, \boldsymbol{m}_{1:t}[x_t]$$

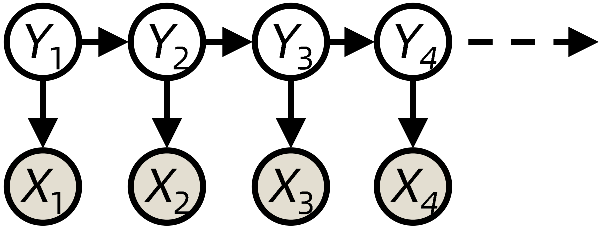
Beyond HMM

- The simplest method: for each word, predict its most frequent label
  - Problems:

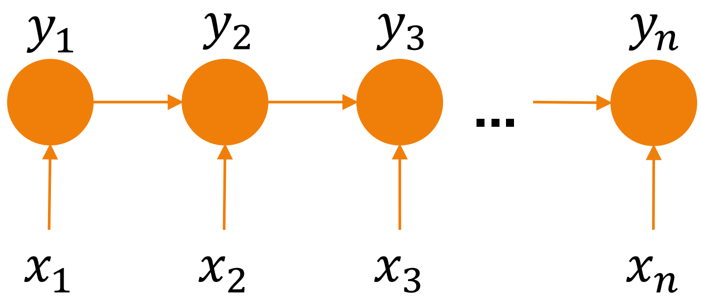
☹ 1. It does not consider the contextual info

☺ 2. It does not consider relations between adjacent labels

- HMM handles problem 2, but not 1



Max-Entropy Markov Models (MEMM)



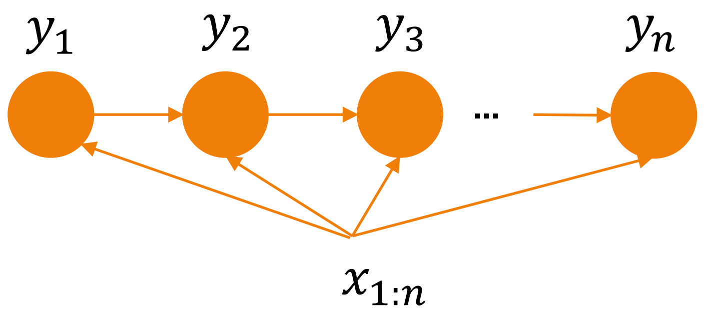
$$P(y_{1:n}|x_{1:n}) = P(y_1|x_1)\prod_{t=2}^{n} P(y_t|y_{t-1}, x_t)$$

$$P(y_t|y_{t-1}, x_t) = \frac{\exp(W^T \boxed{f(y_{t-1}, y_t, x_t)})}{Z(y_{t-1}, x_t)}$$

Possible features:

- y_{t-1} is B and y_t is E?
- y_{t-1} is B and y_t is O?
- x_t is a noun?
- x_t is capitalized?
- …

Max-Entropy Markov Models (MEMM)

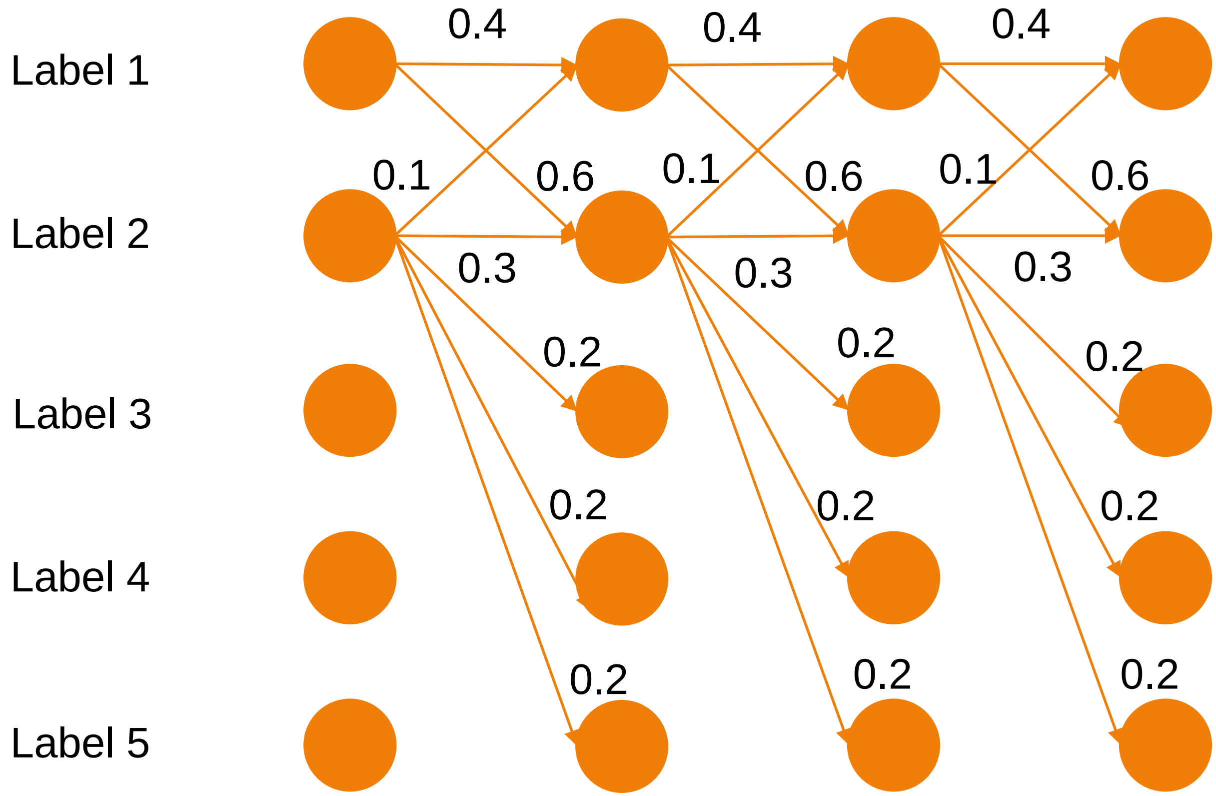


$$P(y_{1:n}|x_{1:n}) = P(y_1|x_{1:n}) \prod_{t=2}^{n} P(y_t|y_{t-1}, x_{1:n})$$

$$P(y_t|y_{t-1}, x_{1:n}) = \frac{\exp(W^T f(y_{t-1}, y_t, x_{1:n}))}{Z(y_{t-1}, x_{1:n})}$$

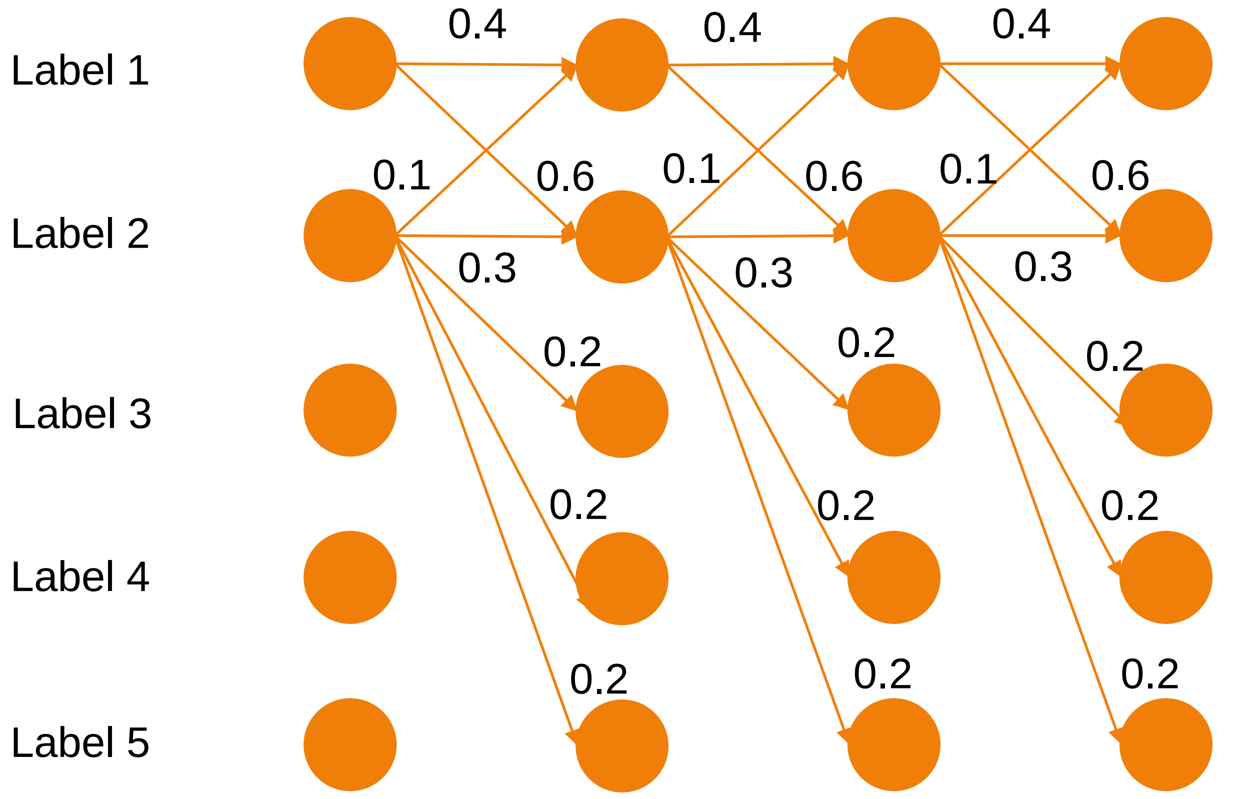
- MEMM considers both contextual info and relations between adjacent labels!
- But… MEMM suffers from label bias problem

Label Bias Problem



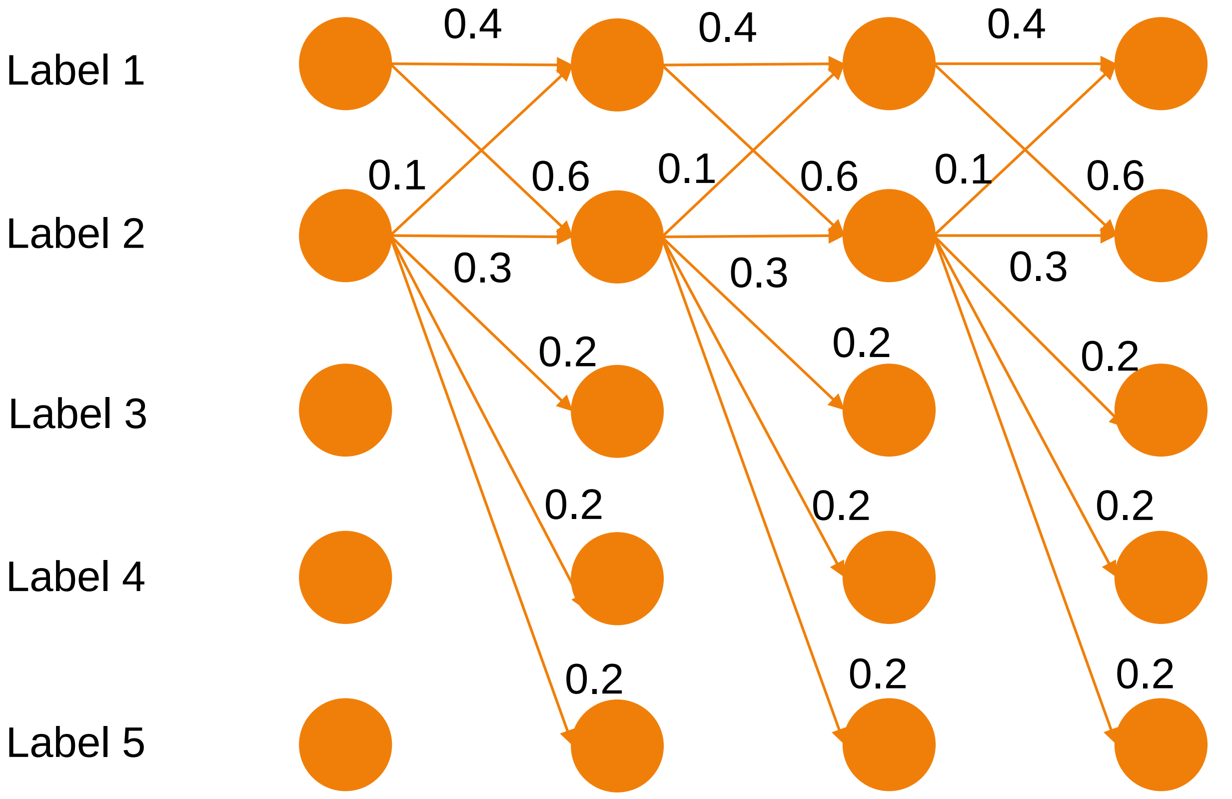
- What the local transition probabilities say:
  - Label 1 prefers to go to label 2
  - Label 2 prefers to stay at label 2

Label Bias Problem



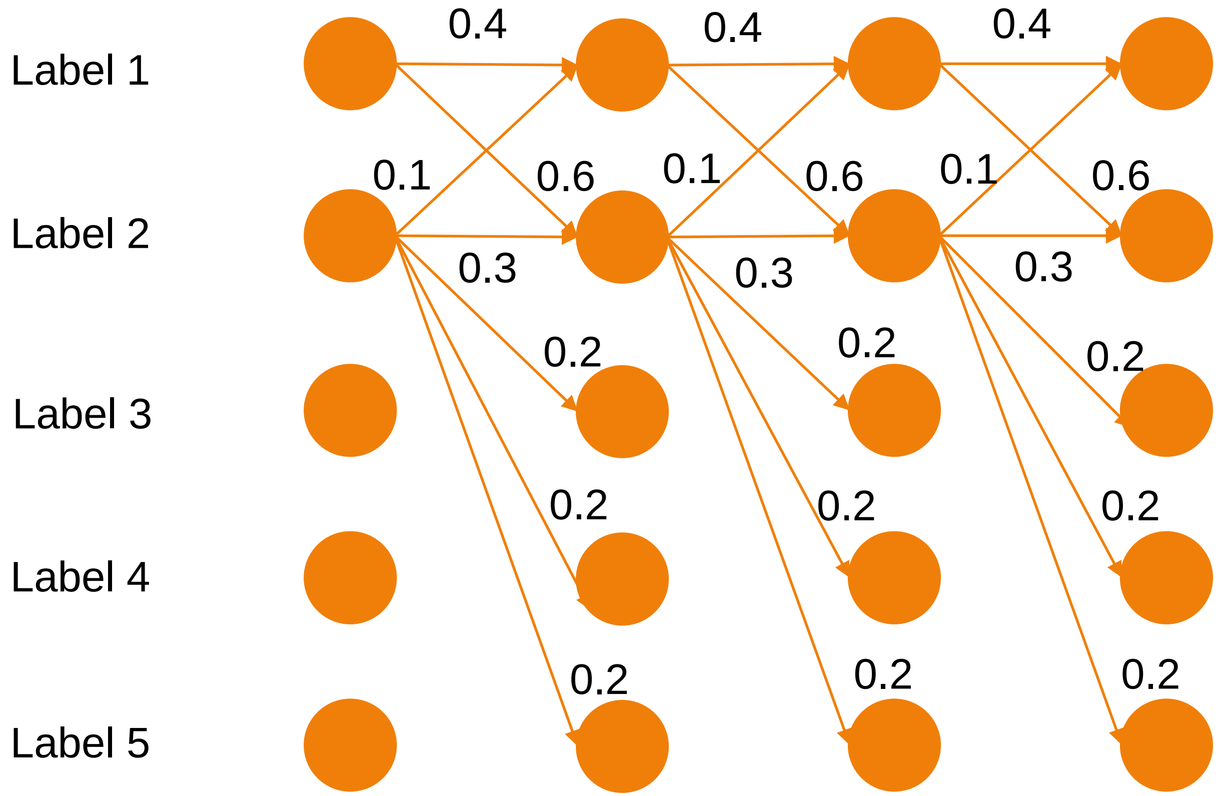
- $\boxed{P(1 \rightarrow 1 \rightarrow 1 \rightarrow 1)=0.4^3=0.064}$
- $P(1 \rightarrow 2 \rightarrow 1 \rightarrow 2)=0.6*0.1*0.6 =0.036$
- $P(2 \rightarrow 2 \rightarrow 2 \rightarrow 2)=0.3^3=0.027$
- $P(2 \rightarrow 1 \rightarrow 2 \rightarrow 1)=0.1*0.6*0.1 =0.006$

Label Bias Problem



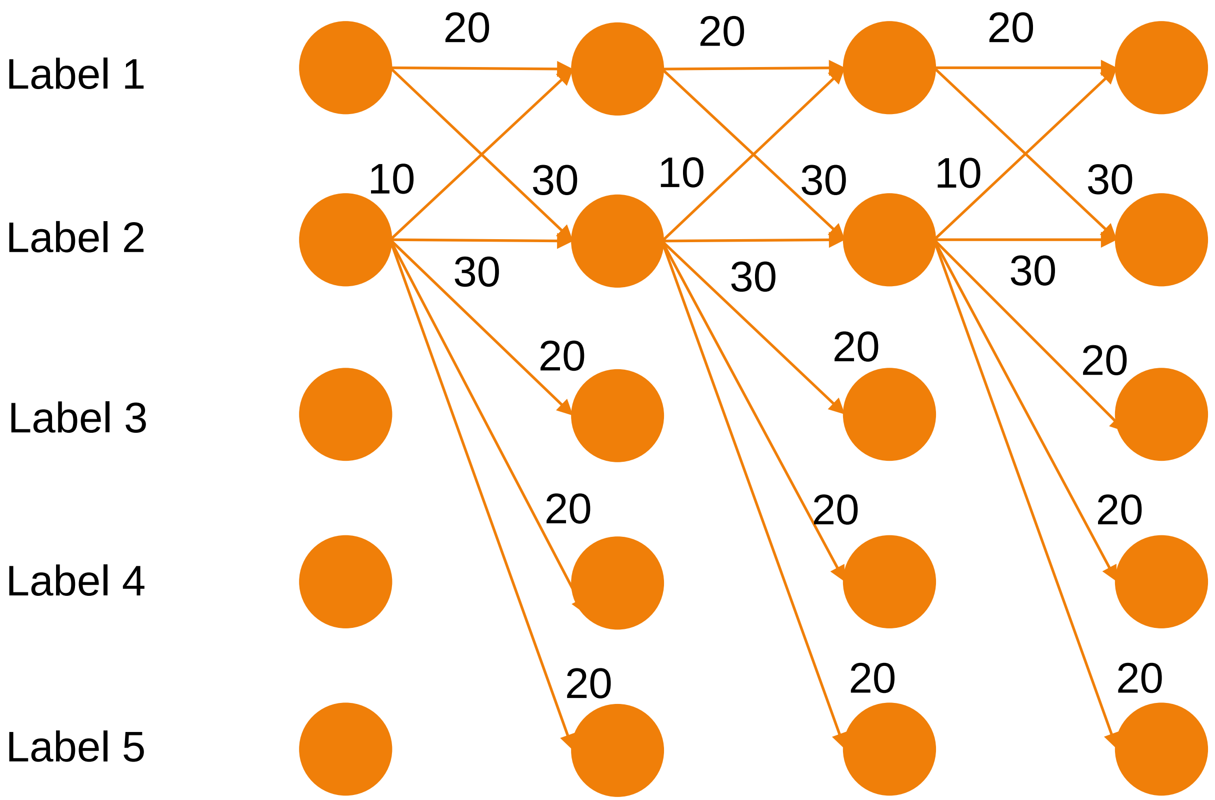
- Label 1 has only two transitions but label 2 has five
- Transition probabilities from label 2 are lower

Label Bias Problem



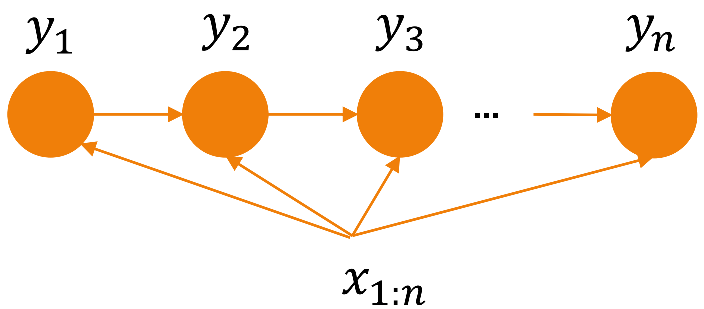
- Label bias in MEMM
  - Preference of states with lower number of transitions

Label Bias Problem



- Solution
  - From local probabilities to local potentials

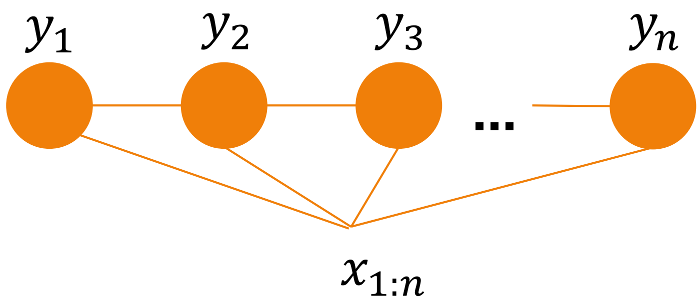
From MEMM to CRF



$$P(y_{1:n}|x_{1:n}) = P(y_1|x_{1:n}) \prod_{t=2}^{n} P(y_t|y_{t-1}, x_{1:n})$$

$$P(y_t|y_{t-1}, x_{1:n}) = \frac{\exp(W^T f(y_{t-1}, y_t, x_{1:n}))}{Z(y_{t-1}, x_{1:n})}$$

From MEMM to CRF



$$P(y_{1:n}|x_{1:n}) = \frac{1}{Z(x_{1:n})} \prod_{t=1}^{n} \exp(W^T f(y_{t-1}, y_t, x_{1:n}))$$

- Conditional Random Field (CRF) is an undirected graphical model
  - Global normalization instead of local normalization
  - Inference: Viterbi

Summary

- Sequence labeling
  - Predict a label for each word of a sentence
  - Many NLP tasks can be seen as sequence labeling
- Methods
  - HMM
  - MEMM
  - CRF

Parsing

Formal Grammars

Constituency

- Constituents
  - Groups of words within sentences can be shown to act as single units.
  - Ex: (The fox)(jumps (over (the dog)))
- These units form coherent classes
  - Units in the same class behave in similar ways
    - …with respect to their *internal* structure
    - …and with respect to other (*external*) units in the language
  - E.g., noun phrases

Constituency

- For example, it makes sense to say that the following are all ***noun phrases*** in English...

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

- Why?
  - Similar internal structures
    - e.g., determiner + modifier + noun + modifier
  - They can all precede verbs (external evidence)

Grammars and Constituency

- Grammar
  - the set of constituents and the rules that govern how they combine
- Lots of different theories of grammar
- Context-free grammars (CFGs)
  - Also known as: Phrase structure grammars
  - One of the simplest and most basic grammar formalisms

Context-Free Grammars

- A context-free grammar has four components
  - A set Σ of terminals (words)
  - A set N of nonterminals (phrases)
  - A start symbol $S \in N$
  - A set R of production rules
    - Specifies how a nonterminal can produce a string of terminals and/or nonterminals

Example Grammar

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

Example Grammar

$$
\begin{aligned}
Noun &\rightarrow flights \mid breeze \mid trip \mid morning \\
Verb &\rightarrow is \mid prefer \mid like \mid need \mid want \mid fly \\
Adjective &\rightarrow cheapest \mid non\text{-}stop \mid first \mid latest \\
&\quad \mid other \mid direct \\
Pronoun &\rightarrow me \mid I \mid you \mid it \\
Proper\text{-}Noun &\rightarrow Alaska \mid Baltimore \mid Los\ Angeles \\
&\quad \mid Chicago \mid United \mid American \\
Determiner &\rightarrow the \mid a \mid an \mid this \mid these \mid that \\
Preposition &\rightarrow from \mid to \mid on \mid near \\
Conjunction &\rightarrow and \mid or \mid but
\end{aligned}
$$

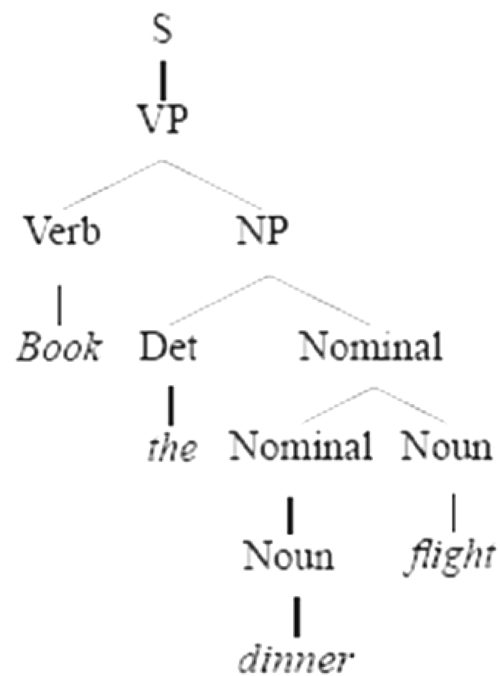
Sentence Generation

- A grammar can be used to generate a string
  - starting from a string containing only the start symbol S
  - recursively applying the rules to rewrite the string
  - until the string contains only terminals
- The generative process specifies the grammatical structure (parse tree) of the string

Example

$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$NP \rightarrow Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow Verb\ NP\ NP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

……



Book the dinner flight

Sentence Parsing

- Parsing is the process of taking a string and a grammar and returning one or more parse tree(s) for that string
  - If no parse tree can be found, then the string does not belong to the language
  - Parsing algorithms: CYK, Earley, etc.
    - To be introduced later

Probabilistic Grammars

- Also called stochastic grammars
- Each rule is associated with a probability

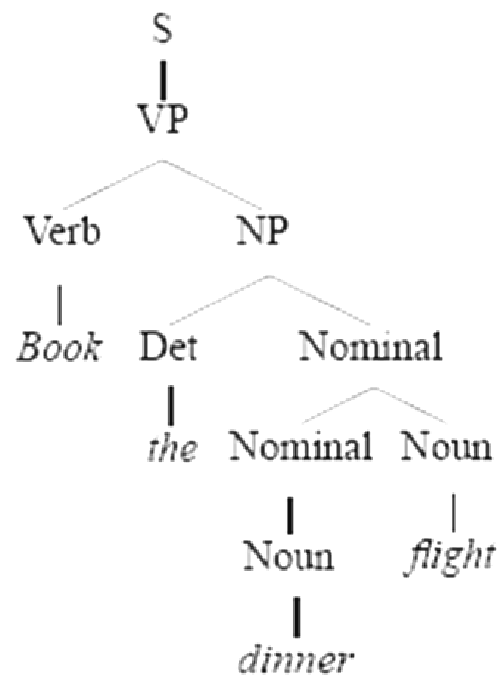
$$\alpha \to \beta : P(\alpha \to \beta | \alpha)$$

- The probability of a parse tree is the product of the probabilities of all the rules used in generating the parse tree

Example

S	→	*NP VP*	[.80]
S	→	*Aux NP VP*	[.15]
S	→	*VP*	[.05]
NP	→	*Pronoun*	[.35]
NP	→	*Proper-Noun*	[.30]
NP	→	*Det Nominal*	[.20]
NP	→	*Nominal*	[.15]
Nominal	→	*Noun*	[.75]
Nominal	→	*Nominal Noun*	[.20]
Nominal	→	*Nominal PP*	[.05]
VP	→	*Verb*	[.35]
VP	→	*Verb NP*	[.20]
VP	→	*Verb NP PP*	[.10]
VP	→	*Verb PP*	[.15]
VP	→	*Verb NP NP*	[.05]
VP	→	*VP PP*	[.15]
PP	→	*Preposition NP*	[1.0]

……



Book the dinner flight

$P(T) = .05 \times .20 \times .20 \times .20 \times .75 \times .30 \times .60 \times .10 \times .40 = 2.2 \times 10^{-6}$

Ambiguity

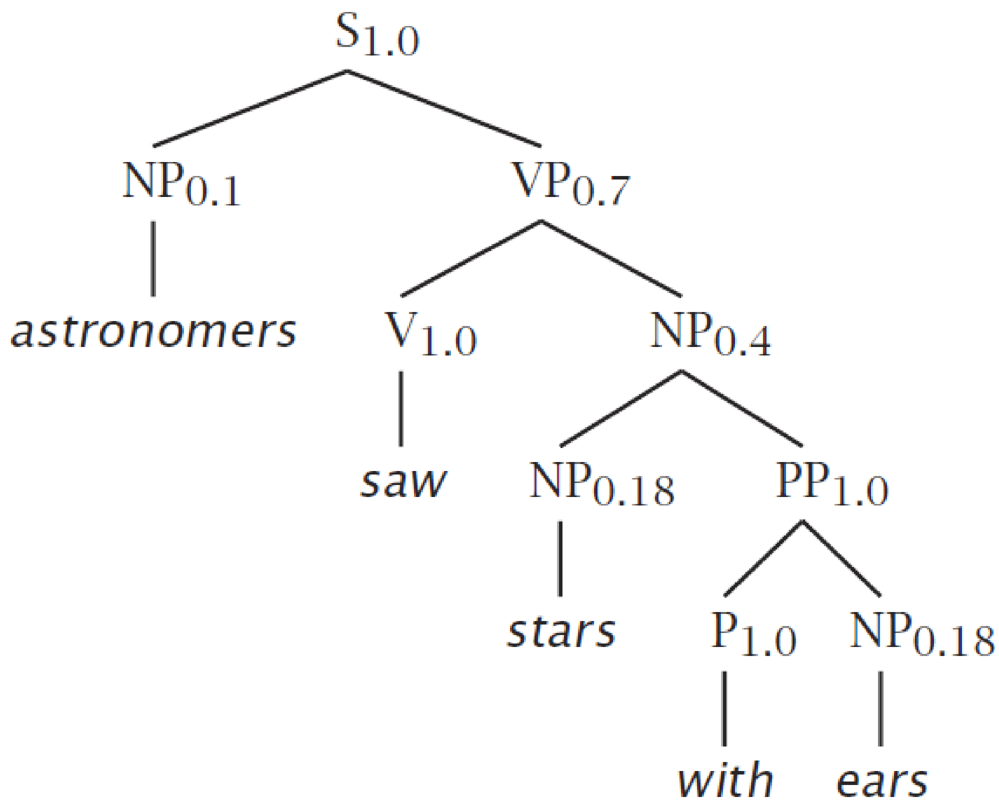
- A sentence is ambiguous if it has more than one possible parse tree
  - …and hence more than one interpretation
- Examples
  - Time flies like an arrow.
  - Astronomers saw stars with ears.

Example

$S \rightarrow NP\ VP$	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

Example

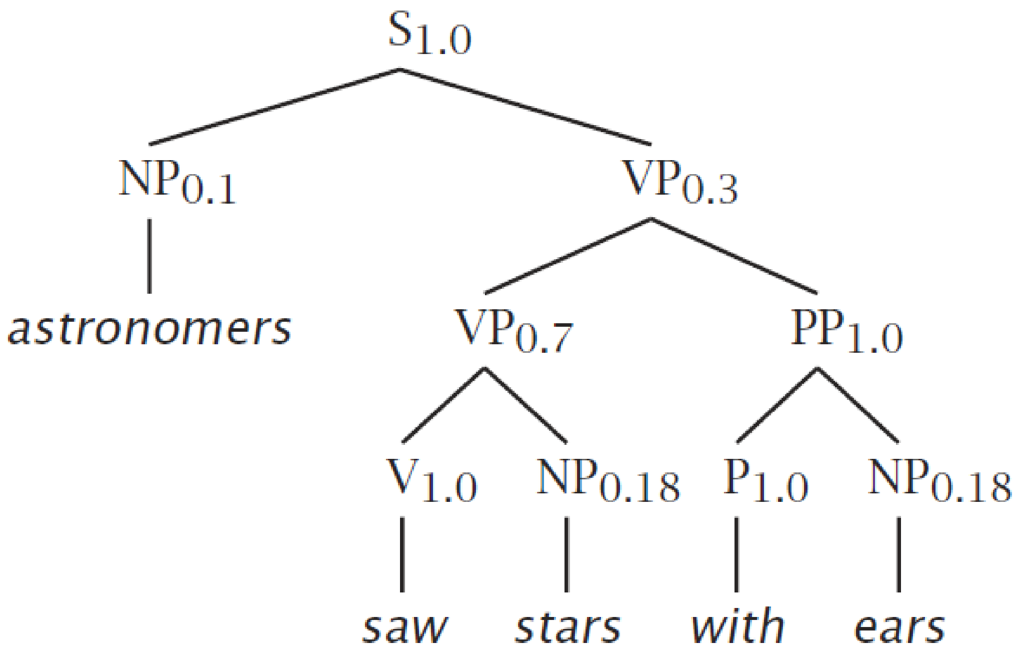
t_1:



$$
\begin{aligned}
P(t_1) &= 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \\
&\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0009072
\end{aligned}
$$

Example

t_2:



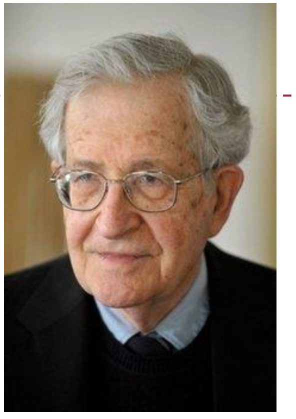
$$
\begin{aligned}
P(t_2) &= 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \\
&\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0006804
\end{aligned}
$$

Chomsky Normal Form (CNF)

- Only two types of production rules in CNF

  $$A \rightarrow B\ C$$

  $$A \rightarrow w$$



Noam Chomsky

- Any arbitrary CFG can be rewritten into CNF automatically
  - The resulting grammar accepts (and rejects) the same set of strings as the original grammar
  - But the resulting parse trees are different (i.e., binarized)

Conversion to CNF

- Eliminate chains of unary productions.
  - So… $A \rightarrow B$, $B \rightarrow C$ turns into $A \rightarrow C$
- Introduce new intermediate non-terminals into the grammar that distribute rules with length > 2 over several rules.
  - So… $S \rightarrow A\ B\ C$ turns into
  - $S \rightarrow X\ C$ and
  - $X \rightarrow A\ B$
  - Where X is a symbol that doesn't occur anywhere else in the grammar.

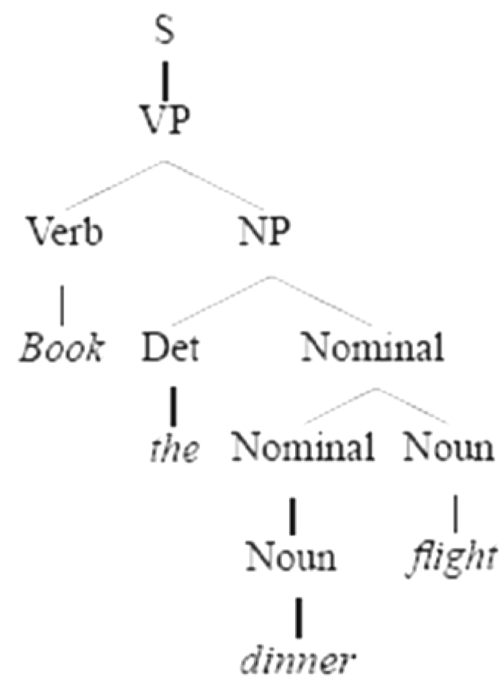
$\mathscr{L}_1$ **Grammar**	$\mathscr{L}_1$ **in CNF**
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

Parsing

Parsing

- Parsing with CFGs is the task of assigning proper parse trees to input strings

Book the dinner flight

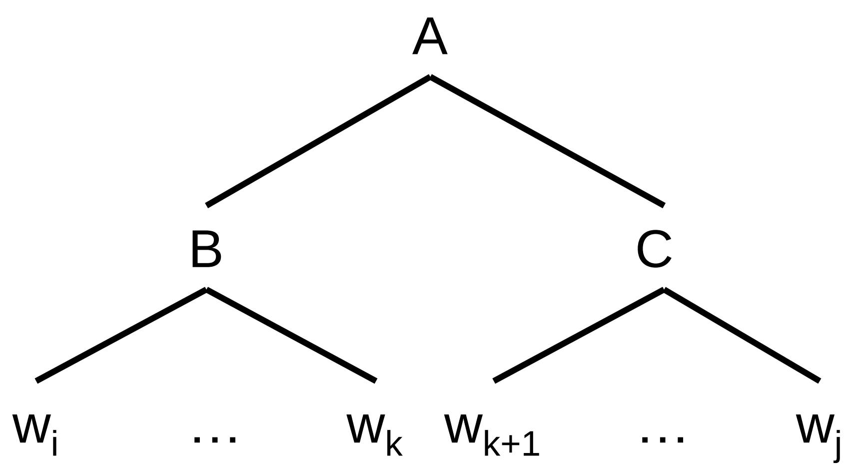


Parsing

- A brute-force approach
  - Enumerate all parse trees consistent with the input string
- Problem
  - Number of binary trees with *n* leaves is the Catalan number C_{n-1}
  - (Exponential growth)

Parsing

- Dynamic programming
  - Divide the problem into many sub-problems
    - Sub-problem: parsing the substring between positions *i* and *j*
  - Solutions to smaller sub-problems are reused in solving larger sub-problems



Cocke–Younger–Kasami Algorithm (CYK)

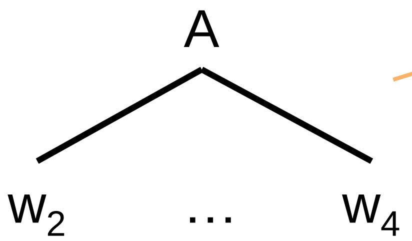
- A bottom-up dynamic programming algorithm
- Applies to CFG in Chomsky Normal Form (CNF)
  - Only two types of production rules

$A \rightarrow B\ C$

$A \rightarrow w$

CYK

- Build a table so that a non-terminal A spanning from i to j in the input is placed in cell [i-1, j] in the table.



	1	2	3	4	5
0					
1				A	
2					
3					
4					

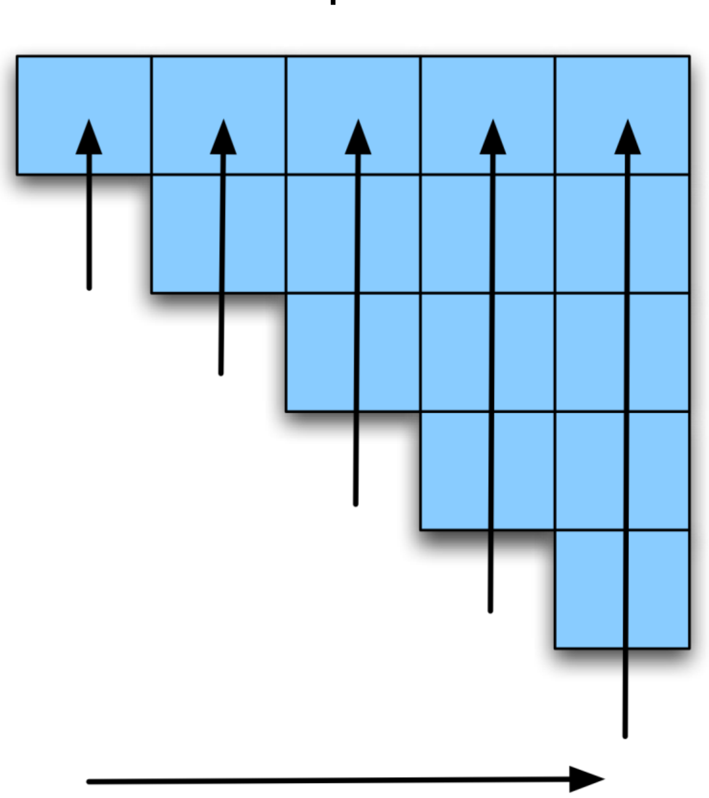
- So a non-terminal spanning an entire string will sit in cell [0, n]
  - Hopefully an *S*

Example

A completed table for input "Book the flight through Houston"

Book	the	flight	through	Houston
S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

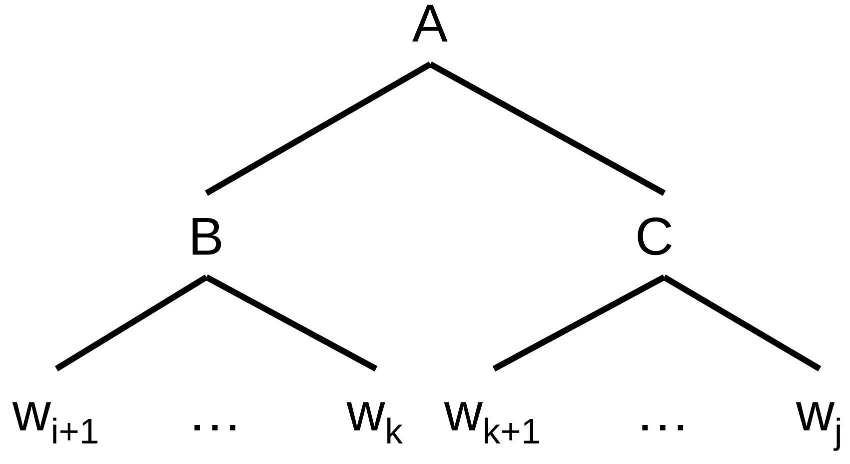
We fill the table from bottom up



CYK

- Base case:
  - A is in cell [i-1,i] iff. there exists a rule $A \rightarrow w_i$

- Recursion:
  - A is in cell [i,j] iff. for some rule $A \rightarrow B\ C$ there is a B in cell [i,k] and a C in cell [k,j] for some k.



CYK Algorithm

function CKY-PARSE(*words, grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**

$table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$

for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**

for $k \leftarrow i+1$ **to** $j-1$ **do**

$$table[i,j] \leftarrow table[i,j] \cup \{A \mid A \rightarrow BC \in grammar, B \in table[i,k], C \in table[k,j]\}$$

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0					
1					
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det				
1					
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det				
1		N			
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3				Det	
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3				Det	
4					N

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3				Det	NP
4					N

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		VP
3				Det	NP
4					N

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			S
1		N			
2			V		VP
3				Det	NP
4					N

CYK Parsing

- Is that really a parser?
  - We want a parse tree, not a yes/no answer
- Simple changes
  - Add back-pointers so that each state knows where it came from.
  - After filling the table, recursively retrieve the constituents from the top (i.e., the start symbol) down

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			S
1		N			
2			V		VP
3				Det	NP
4					N

Ambiguity

- NP → Det N
- NP → N N

	1	2	3	4	5
0	Det N	??			
1		N			
2					
3					
4					

Ambiguity

- NP → Det N
- NP → N N

	1	2	3	4	5
0	Det N	NP			
1		N			
2					
3					
4					

Ambiguity

- NP → Det N
- NP → N N

	1	2	3	4	5
0	Det N	NP			
1		N			
2					
3					
4					

Ambiguity

- NP → Det NP
- NP → NP PP

	1	2	3	4	5
0	Det	NP	??		
1		N	NP		
2			PP		
3					
4					

Ambiguity

- NP → Det NP
- NP → NP PP

	1	2	3	4	5
0	Det	NP	NP		
1		N	NP		
2			PP		
3					
4					

Ambiguity

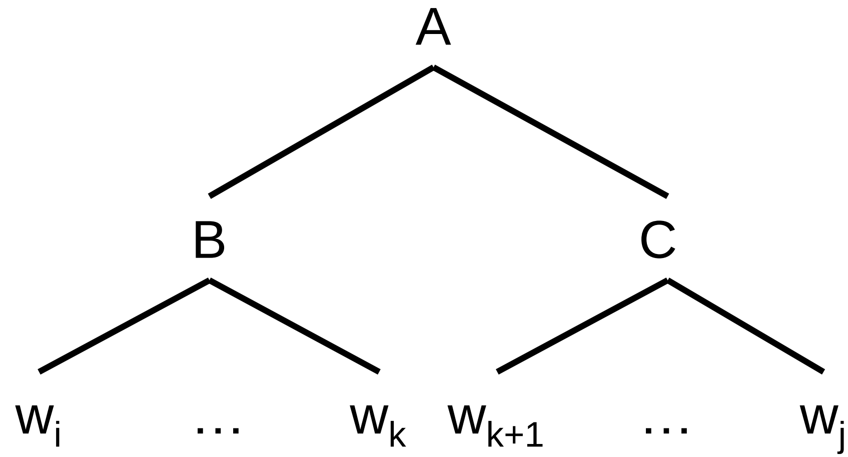
- NP → Det NP
- NP → NP PP

	1	2	3	4	5
0	Det	NP	NP		
1		N	NP		
2			PP		
3					
4					

Probabilistic Parsing

- We have a probabilistic grammar, e.g., PCFG
- We want to find the parse tree of an input string with the highest probability
- In cell [i-1,j] of the table, associate each nonterminal A with the probability of the best parse tree rooted at A covering substring from i to j
- Recursive computation

$$P_{A,i,j} = \max_{B,C,k} P(A \rightarrow BC) \times P_{B,i,k} \times P_{C,k,j}$$



CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0					
1					
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4				
1					
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4				
1		N 0.02			
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3				Det 0.4	
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3				Det 0.4	
4					N 0.01

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3				Det 0.4	NP 0.001
4					N 0.01

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		VP .00001
3				Det 0.4	NP 0.001
4					N 0.01

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	**2**	**3**	**4**	**5**
0	Det 0.4	NP .0024			S .0000000192
1		N 0.02			
2			V .05		VP .00001
3				Det 0.4	NP 0.001
4					N 0.01

Ambiguity

- NP → Det N [0.7]
- NP → N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8				
1		N 0.02			
2					
3					
4					

Ambiguity

- NP → Det N [0.7]
- NP → N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8	NP .0056			
1		N 0.02			
2					
3					
4					

Ambiguity

- NP → Det N [0.7]
- NP → N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8	NP .0048			
1		N 0.02			
2					
3					
4					

Ambiguity

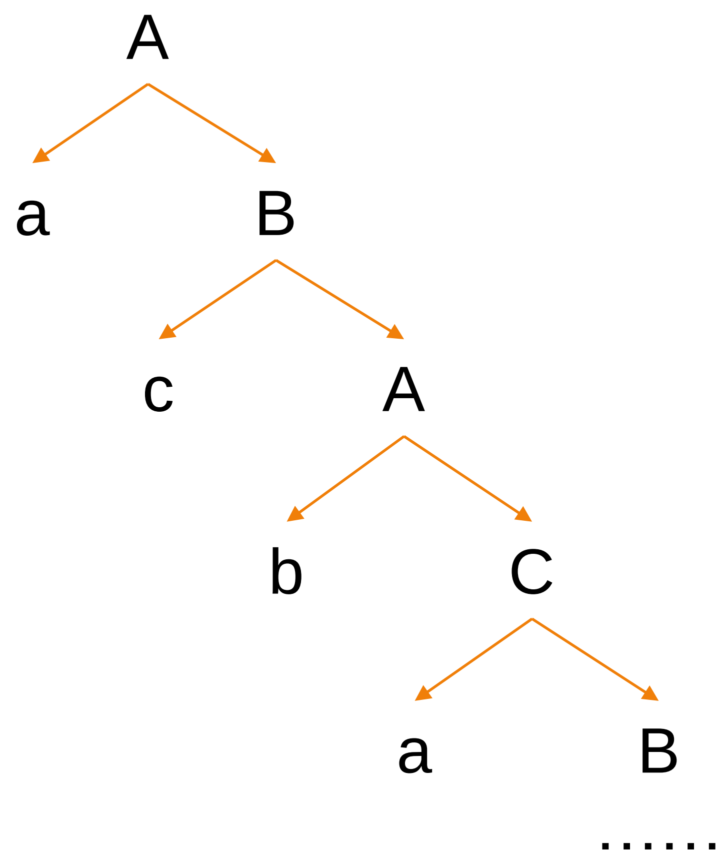
- NP → Det N [0.7]
- NP → N N [0.3]

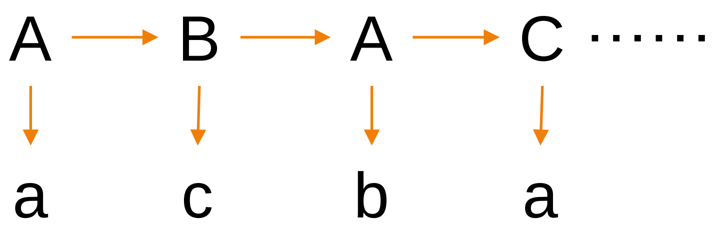
	1	2	3	4	5
0	Det 0.4 N 0.8	NP .0056			
1		N 0.02			
2					
3					
4					

Regular Grammar

Regular Grammars

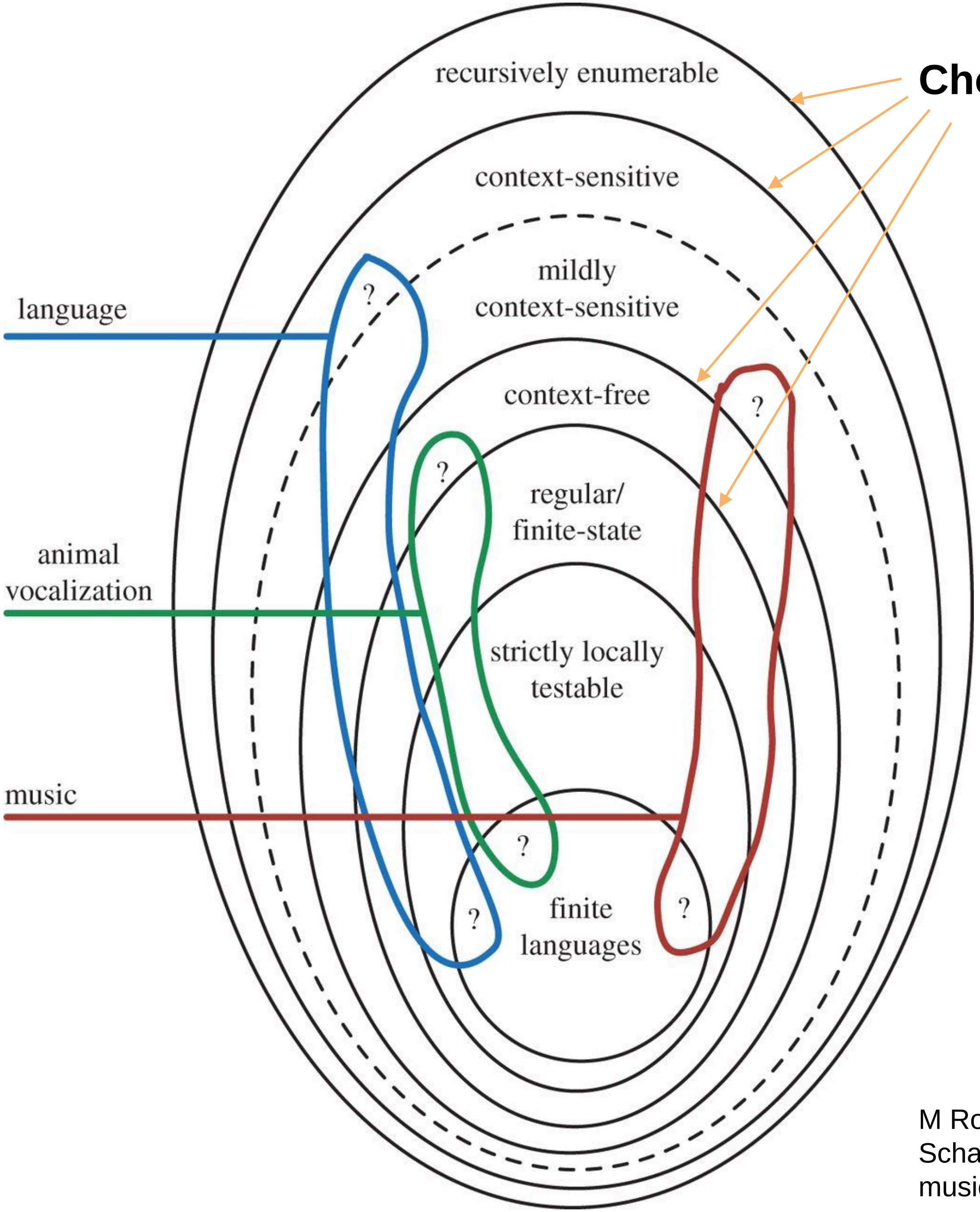
- Production rules are of the form *A → aB or A → a*

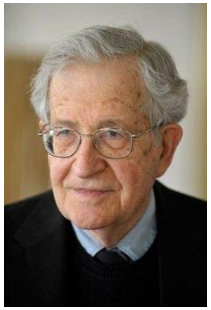




Probabilistic RG = HMM

Chomsky Hierarchy





M Rohrmeier, W Zuidema, G Wiggins, C Scharff. Principles of structure building in music, language and animal song.

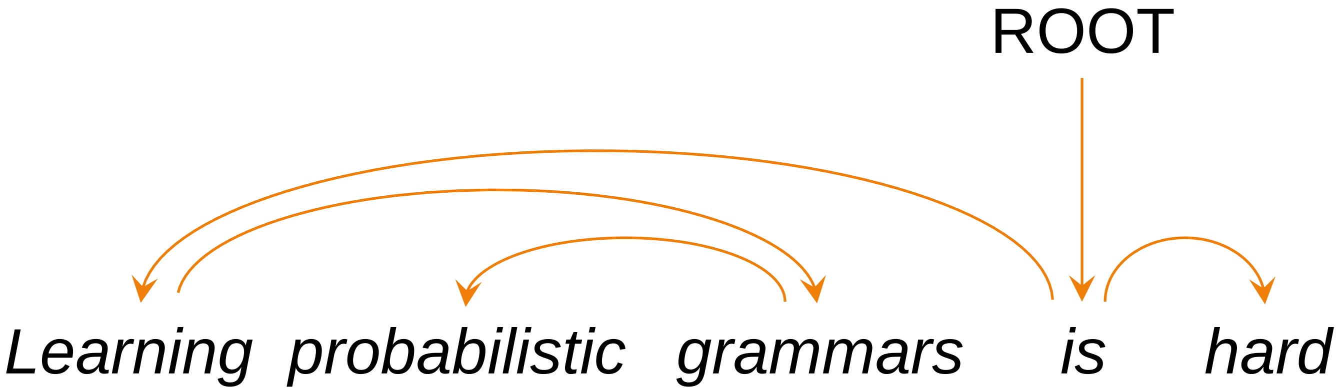
Dependency Grammar

Dependency Grammars

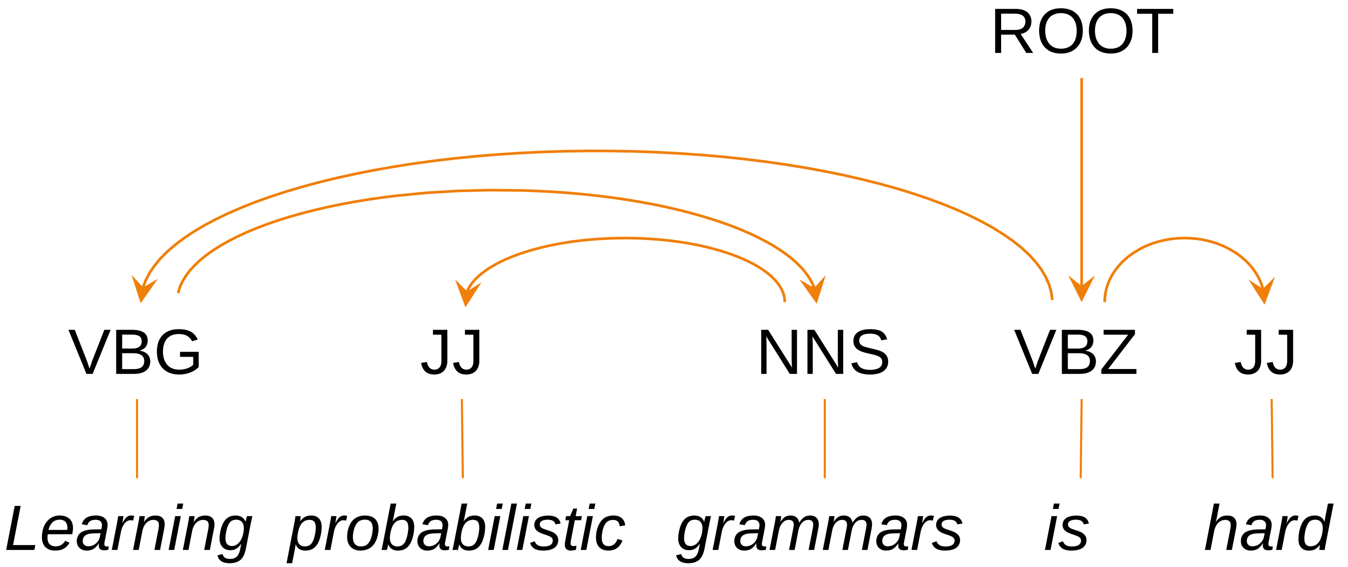
- CFG focuses on *constituents*.

- A dependency grammar focuses on just binary relations among the words in a sentence
- A dependency parse is a tree where
  - the nodes are the words in a sentence
  - The links between the words represent their dependency relations.
    - Relations may be typed (labeled)

Dependency Parse



Dependency Parse



Dependency Types

Argument Dependencies	Description
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition
Modifier Dependencies	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

Dependency Parse

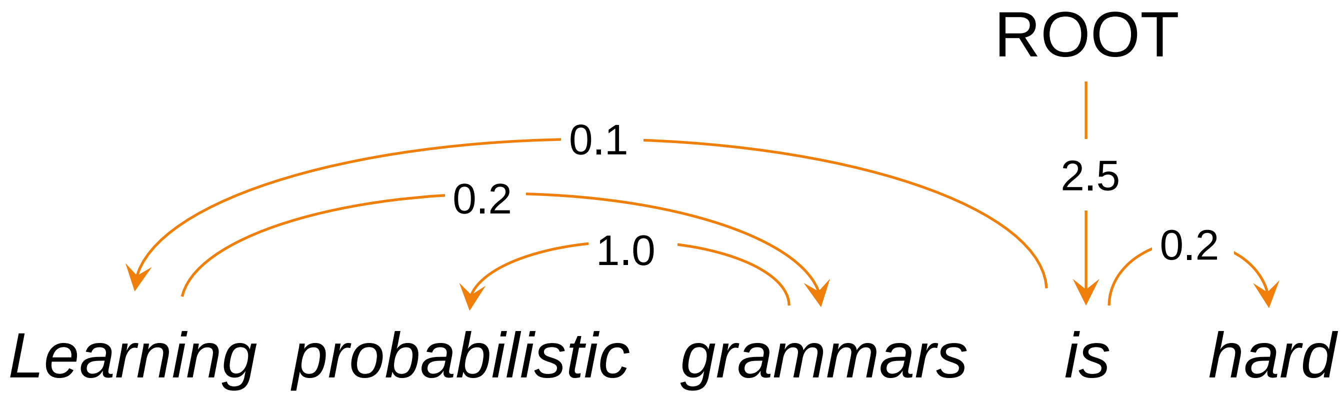
- Advantages
  - Deals well with free word order languages where the constituent structure is quite fluid
    - Ex: Czech, Turkish
  - Parsing is much faster than CFG-bases parsers
  - Dependency structure often captures the syntactic relations needed by later applications
    - CFG-based approaches often extract this same information from trees anyway.

Dependency Parsing

- Parsing: taking a string and a grammar and returning one or more parse tree(s) for that string
- There are two modern approaches to dependency parsing
  - Graph-based approach: finding the (maximum) spanning trees of the complete graph over words
  - Transition-based (shift-reduce) approach: reading words from left to right and taking a sequence of actions to construct a tree

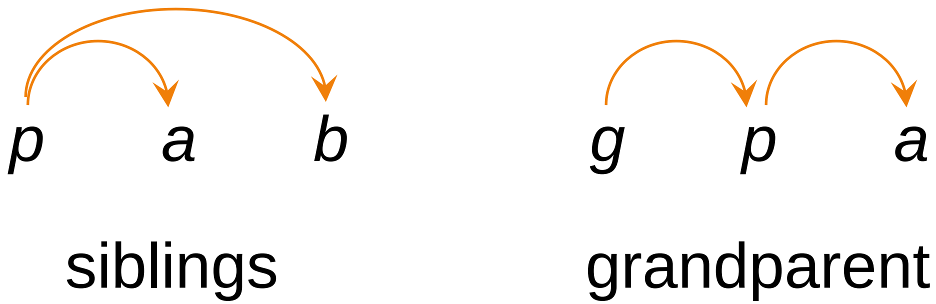
Parse tree scoring

- Purpose of scoring: resolve ambiguity
- First-order graph-based parsing
  - Each arc has a score. The tree score is the sum of arc scores.
  - An arc score is often computed from features of the two words



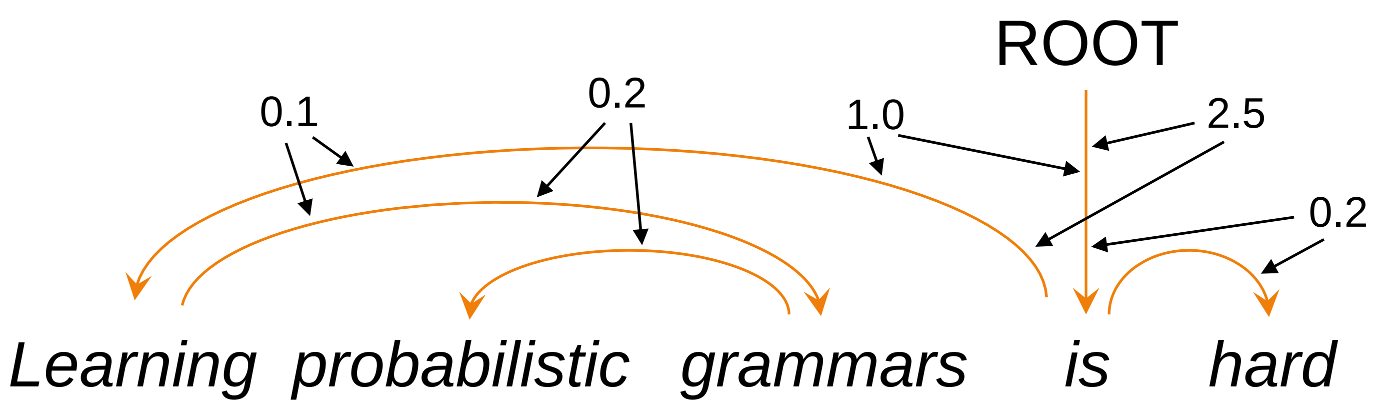
Parse tree scoring

- Purpose of scoring: resolve ambiguity
- Second-order graph-based parsing
  - Each connected pair of arcs has a score. The tree score is the sum of arc-pair scores.



Parse tree scoring

- Purpose of scoring: resolve ambiguity
- Second-order graph-based parsing
  - Each connected pair of arcs has a score. The tree score is the sum of arc-pair scores.



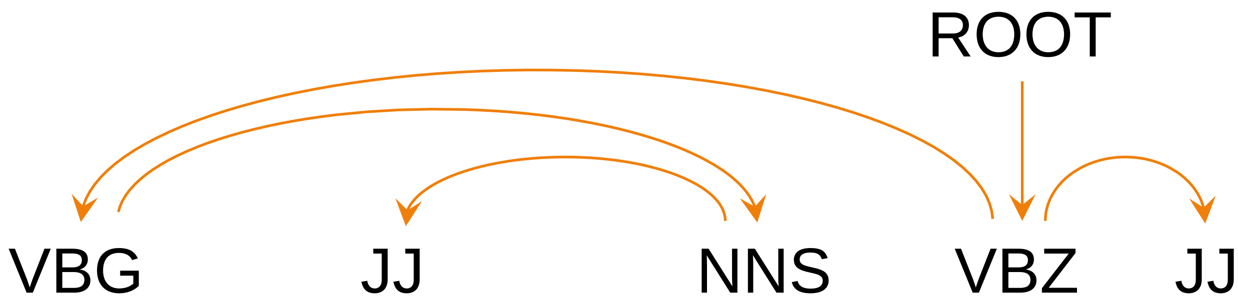
Dependency Grammar vs. CFG

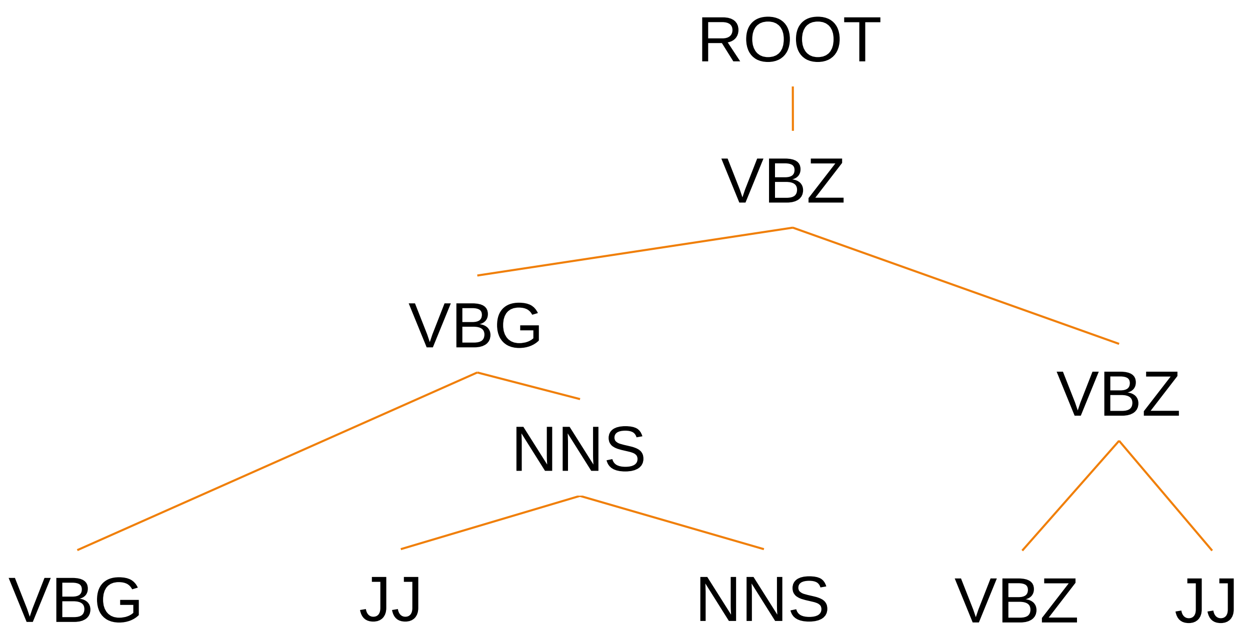
DG vs. CFG

- Dependency grammars are a subclass of CFGs

$a \rightarrow b$		$A \rightarrow AB$
$a \rightarrow c$	$\Rightarrow$	$A \rightarrow AC$
$d \leftarrow a$		$A \rightarrow DA$
$e \leftarrow a$		$A \rightarrow EA$
		$A \rightarrow a$
		$B \rightarrow b$
		$C \rightarrow c$
		$D \rightarrow d$
		$E \rightarrow e$

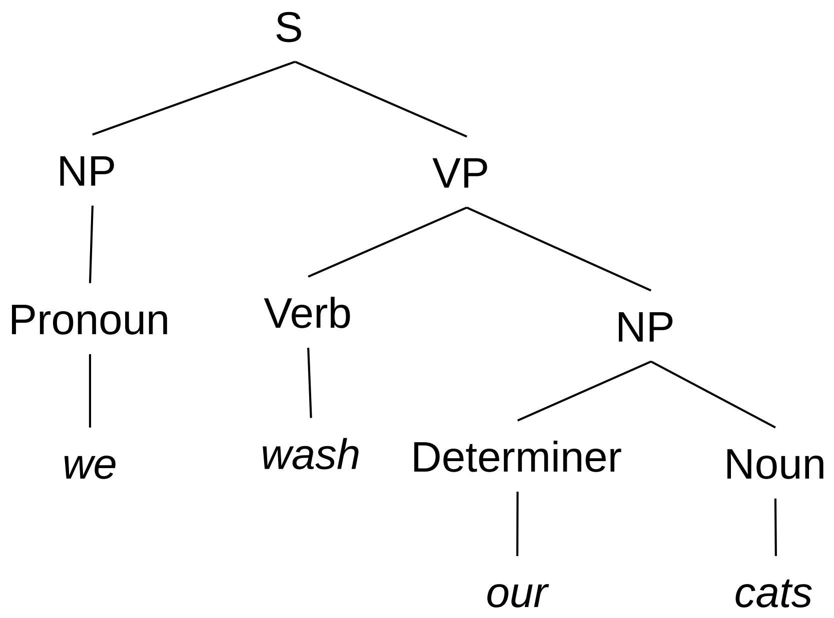
DG vs. CFG





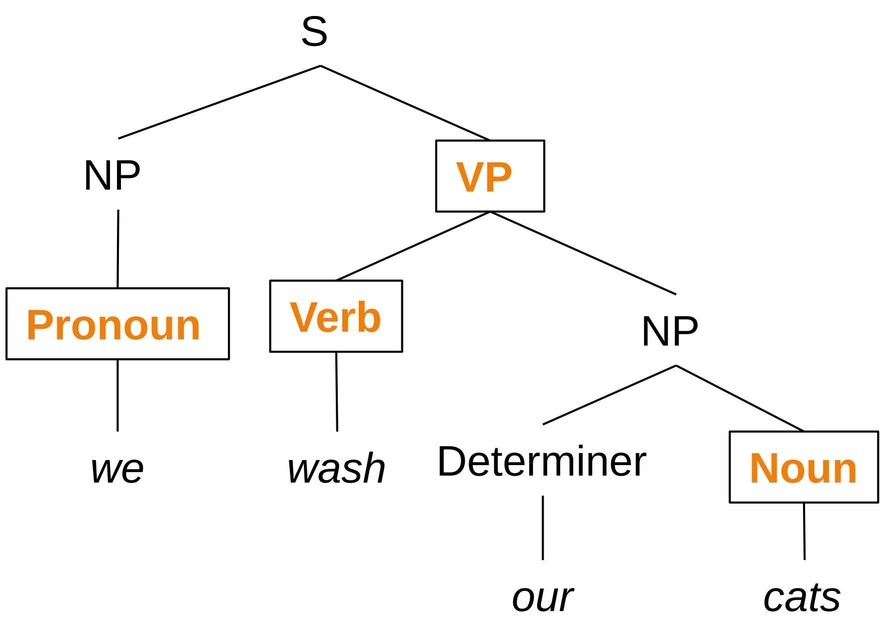
Conversion

- From a constituent tree to a dependency tree
  - Constituent tree



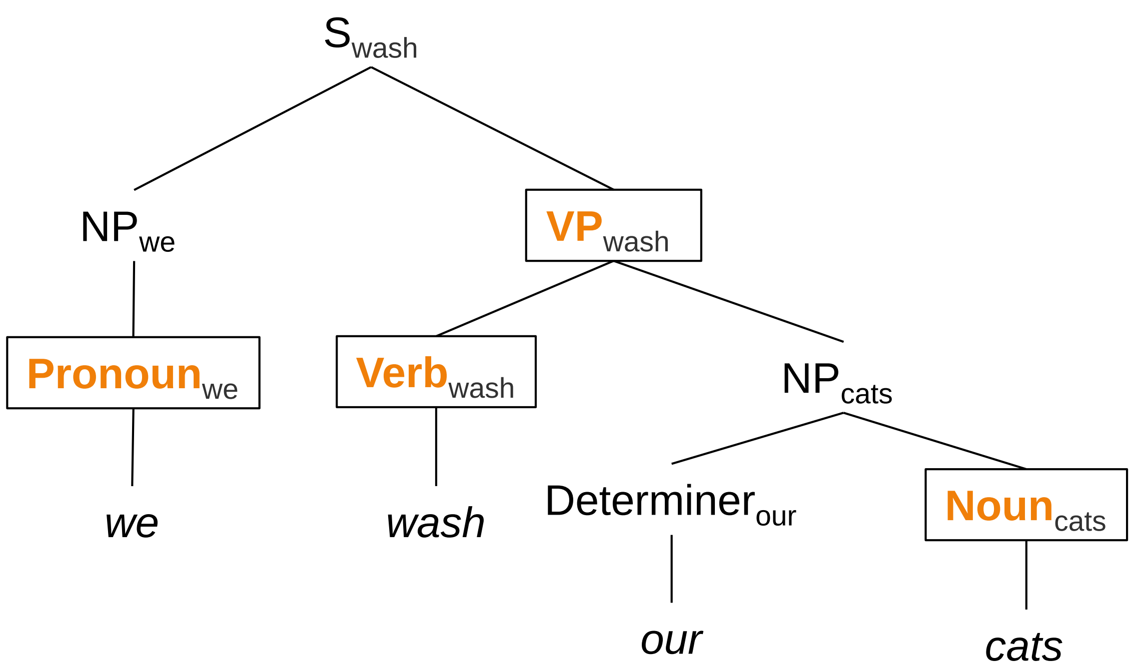
Conversion

- From a constituent tree to a dependency tree
  - Constituent tree with heads



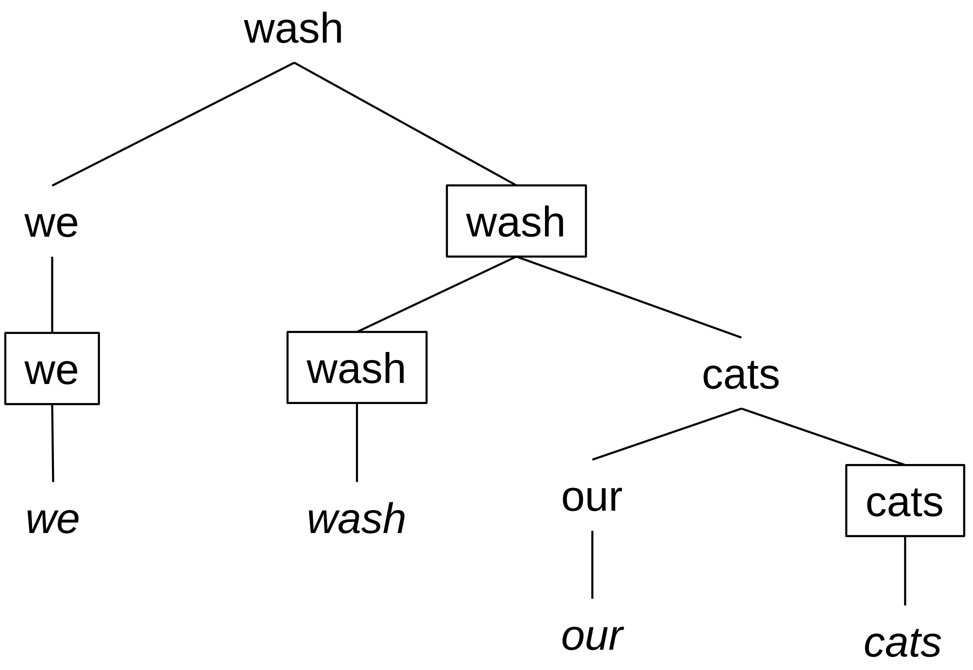
Conversion

- From a constituent tree to a dependency tree
  - Constituent tree with heads, lexicalized



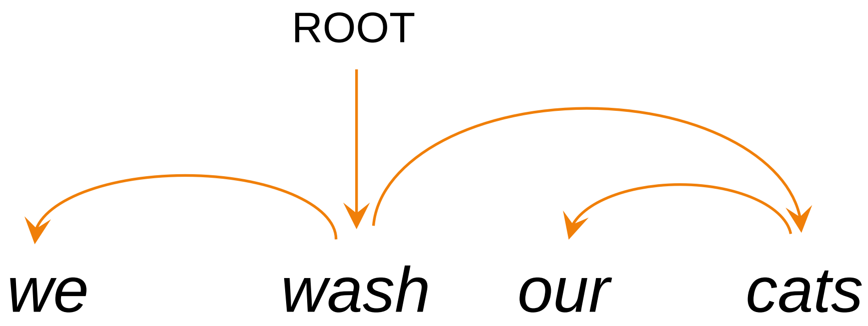
Conversion

- From a constituent tree to a dependency tree
  - Constituent tree with heads, lexicalized



Conversion

- From a constituent tree to a dependency tree
  - Dependency tree



Summary

- Grammars
  - (Probabilistic) context-free grammars
  - Regular grammars
  - Dependency grammars
- Parsing
  - CYK