

# Natural Language Processing (CSE 490U): Phrase Structure

Noah Smith

© 2017

University of Washington  
nasmith@cs.washington.edu

February 8–17, 2017

# Finite-State Automata

A **finite-state automaton** (plural “automata”) consists of:

- ▶ A finite set of states  $\mathcal{S}$ 
  - ▶ Initial state  $s_0 \in \mathcal{S}$
  - ▶ Final states  $\mathcal{F} \subseteq \mathcal{S}$
- ▶ A finite alphabet  $\Sigma$
- ▶ Transitions  $\delta : \mathcal{S} \times \Sigma \rightarrow 2^{\mathcal{S}}$ 
  - ▶ Special case: **deterministic** FSA defines  $\delta : \mathcal{S} \times \Sigma \rightarrow \mathcal{S}$

A string  $x \in \Sigma^n$  is recognizable by the FSA iff there is a sequence  $\langle s_0, \dots, s_n \rangle$  such that  $s_n \in \mathcal{F}$  and

$$\bigwedge_{i=1}^n [[s_i \in \delta(s_{i-1}, x_i)]]$$

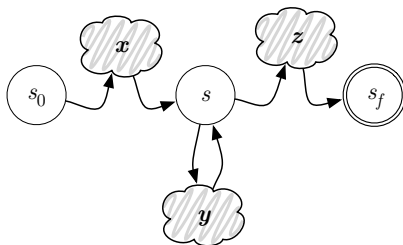
This is sometimes called a **path**.

# Terminology from Theory of Computation

- ▶ A **regular expression** can be:
  - ▶ an empty string (usually denoted  $\epsilon$ ) or a symbol from  $\Sigma$
  - ▶ a **concatentation** of regular expressions (e.g.,  $abc$ )
  - ▶ an **alternation** of regular expressions (e.g.,  $ab|cd$ )
  - ▶ a **Kleene star** of a regular expression (e.g.,  $(abc)^*$ )
- ▶ A **language** is a set of strings.
- ▶ A **regular language** is a language expressible by a regular expression.
- ▶ Important theorem: every regular language can be recognized by a FSA, and every FSA's language is regular.

## Proving a Language Isn't Regular

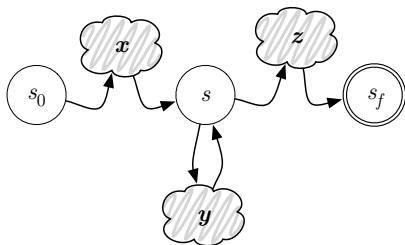
Pumping lemma (for regular languages): if  $L$  is an infinite regular language, then there exist strings  $x$ ,  $y$ , and  $z$ , with  $y \neq \epsilon$ , such that  $xy^n z \in L$ , for all  $n \geq 0$ .



If  $L$  is infinite and  $x$ ,  $y$ ,  $z$  do not exist, then  $L$  is not regular.

## Proving a Language Isn't Regular

Pumping lemma (for regular languages): if  $L$  is an infinite regular language, then there exist strings  $x$ ,  $y$ , and  $z$ , with  $y \neq \epsilon$ , such that  $xy^n z \in L$ , for all  $n \geq 0$ .

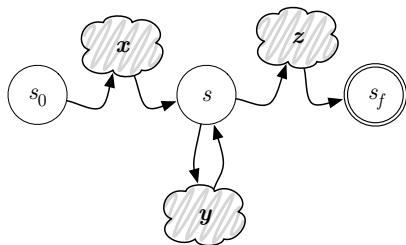


If  $L$  is infinite and  $x$ ,  $y$ ,  $z$  do not exist, then  $L$  is not regular.

If  $L_1$  and  $L_2$  are regular, then  $L_1 \cap L_2$  is regular.

## Proving a Language Isn't Regular

Pumping lemma (for regular languages): if  $L$  is an infinite regular language, then there exist strings  $x$ ,  $y$ , and  $z$ , with  $y \neq \epsilon$ , such that  $xy^n z \in L$ , for all  $n \geq 0$ .



If  $L$  is infinite and  $x$ ,  $y$ ,  $z$  do not exist, then  $L$  is not regular.

If  $L_1$  and  $L_2$  are regular, then  $L_1 \cap L_2$  is regular.

If  $L_1 \cap L_2$  is not regular, and  $L_1$  is regular, then  $L_2$  is not regular.

Claim: English is not regular.

$L_1 = (\text{the cat|mouse|dog})^*(\text{ate|bit|chased})^* \text{ likes tuna fish}$

$L_2 = \text{English}$

$L_1 \cap L_2 = (\text{the cat|mouse|dog})^n(\text{ate|bit|chased})^{n-1} \text{ likes tuna fish}$

$L_1 \cap L_2$  is not regular, but  $L_1$  is  $\Rightarrow L_2$  is not regular.

the cat likes tuna fish

the cat the dog chased likes tuna fish

the cat the dog the mouse scared chased likes tuna fish

the cat the dog the mouse the elephant squashed scared chased  
likes tuna fish

the cat the dog the mouse the elephant the flea bit squashed  
scared chased likes tuna fish

the cat the dog the mouse the elephant the flea the virus infected  
bit squashed scared chased likes tuna fish



# Linguistic Debate

# Linguistic Debate

Chomsky put forward an argument like the one we just saw.

# Linguistic Debate

Chomsky put forward an argument like the one we just saw.

(Chomsky gets credit for formalizing a hierarchy of types of languages: regular, context-free, context-sensitive, recursively enumerable. This was an important contribution to CS!)

# Linguistic Debate

Chomsky put forward an argument like the one we just saw.

(Chomsky gets credit for formalizing a hierarchy of types of languages: regular, context-free, context-sensitive, recursively enumerable. This was an important contribution to CS!)

Some are unconvinced, because after a few center embeddings, the examples become unintelligible.

# Linguistic Debate

Chomsky put forward an argument like the one we just saw.

(Chomsky gets credit for formalizing a hierarchy of types of languages: regular, context-free, context-sensitive, recursively enumerable. This was an important contribution to CS!)

Some are unconvinced, because after a few center embeddings, the examples become unintelligible.

Nonetheless, most agree that natural language syntax isn't well captured by FSAs.

# Noun Phrases

What, exactly makes a noun phrase? Examples (Jurafsky and Martin, 2008):

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

# Constituents

More general than noun phrases: **constituents** are groups of words.

Linguists characterize constituents in a number of ways, including:

# Constituents

More general than noun phrases: **constituents** are groups of words.

Linguists characterize constituents in a number of ways, including:

- ▶ where they occur (e.g., “NPs can occur before verbs”)
- ▶ where they can *move* in variations of a sentence
  - ▶ On September 17th, I'd like to fly from Atlanta to Denver
  - ▶ I'd like to fly on September 17th from Atlanta to Denver
  - ▶ I'd like to fly from Atlanta to Denver on September 17th



# Constituents

More general than noun phrases: **constituents** are groups of words.

Linguists characterize constituents in a number of ways, including:

- ▶ where they occur (e.g., “NPs can occur before verbs”)
- ▶ where they can *move* in variations of a sentence
  - ▶ On September 17th, I'd like to fly from Atlanta to Denver
  - ▶ I'd like to fly on September 17th from Atlanta to Denver
  - ▶ I'd like to fly from Atlanta to Denver on September 17th
- ▶ what parts can move and what parts can't
  - ▶ \*On September I'd like to fly 17th from Atlanta to Denver

# Constituents

More general than noun phrases: **constituents** are groups of words.

Linguists characterize constituents in a number of ways, including:

- ▶ where they occur (e.g., “NPs can occur before verbs”)
- ▶ where they can *move* in variations of a sentence
  - ▶ On September 17th, I'd like to fly from Atlanta to Denver
  - ▶ I'd like to fly on September 17th from Atlanta to Denver
  - ▶ I'd like to fly from Atlanta to Denver on September 17th
- ▶ what parts can move and what parts can't
  - ▶ \*On September I'd like to fly 17th from Atlanta to Denver
- ▶ what they can be conjoined with
  - ▶ I'd like to fly from Atlanta to Denver on September 17th and in the morning

# Recursion and Constituents

this is the house

this is the house that Jack built

this is the cat that lives in the house that Jack built

this is the dog that chased the cat that lives in the house that Jack built

this is the flea that bit the dog that chased the cat that lives in the house the Jack built

this is the virus that infected the flea that bit the dog that chased the cat that lives in the house that Jack built

# Not Constituents

(Pullum, 1991)

- ▶ *If on a Winter's Night a Traveler* (by Italo Calvino)
- ▶ *Nuclear and Radiochemistry* (by Gerhart Friedlander et al.)
- ▶ *The Fire Next Time* (by James Baldwin)
- ▶ *A Tad Overweight, but Violet Eyes to Die For* (by G.B. Trudeau)
- ▶ *Sometimes a Great Notion* (by Ken Kesey)
- ▶ [how can we know the] *Dancer from the Dance* (by Andrew Holleran)

# Context-Free Grammar

A **context-free grammar** consists of:

- ▶ A finite set of nonterminal symbols  $\mathcal{N}$ 
  - ▶ A start symbol  $S \in \mathcal{N}$
- ▶ A finite alphabet  $\Sigma$ , called “terminal” symbols, distinct from  $\mathcal{N}$
- ▶ Production rule set  $\mathcal{R}$ , each of the form “ $N \rightarrow \alpha$ ” where
  - ▶ The lefthand side  $N$  is a nonterminal from  $\mathcal{N}$
  - ▶ The righthand side  $\alpha$  is a sequence of zero or more terminals and/or nonterminals:  $\alpha \in (\mathcal{N} \cup \Sigma)^*$ 
    - ▶ Special case: **Chomsky normal form** constrains  $\alpha$  to be either a single terminal symbol or two nonterminals

# An Example CFG for a Tiny Bit of English

From Jurafsky and Martin (2008)

S → NP VP

S → Aux NP VP

S → VP

NP → Pronoun

NP → Proper-Noun

NP → Det Nominal

Nominal → Noun

Nominal → Nominal Noun

Nominal → Nominal PP

VP → Verb

VP → Verb NP

VP → Verb NP PP

VP → Verb PP

VP → VP PP

PP → Preposition NP

Det → that | this | a

Noun → book | flight | meal | money

Verb → book | include | prefer

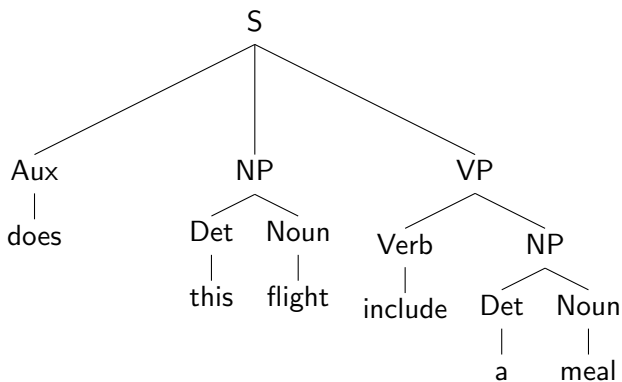
Pronoun → I | she | me

Proper-Noun → Houston | NWA

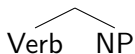
Aux → does

Preposition → from | to | on | near  
| through

## Example Phrase Structure Tree

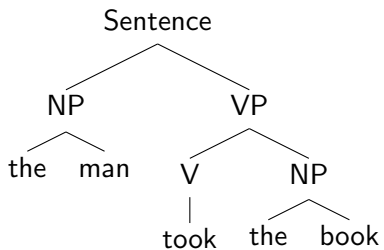


The phrase-structure tree represents both the syntactic structure of the sentence and the **derivation** of the sentence under the grammar. E.g., VP corresponds to the rule  $VP \rightarrow \text{Verb NP}$ .



# The First Phrase-Structure Tree

(Chomsky, 1956)





# Where do natural language CFGs come from?

As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.

# Where do natural language CFGs come from?

As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.

- ▶ Need lots of categories to make sure all and only grammatical sentences are included.

# Where do natural language CFGs come from?

As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.

- ▶ Need lots of categories to make sure all and only grammatical sentences are included.
- ▶ Categories tend to start exploding combinatorially.

# Where do natural language CFGs come from?

As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.

- ▶ Need lots of categories to make sure all and only grammatical sentences are included.
- ▶ Categories tend to start exploding combinatorially.
- ▶ Alternative grammar formalisms are typically used for manual grammar construction; these are often based on constraints and a powerful algorithmic tool called *unification*.

## Where do natural language CFGs come from?

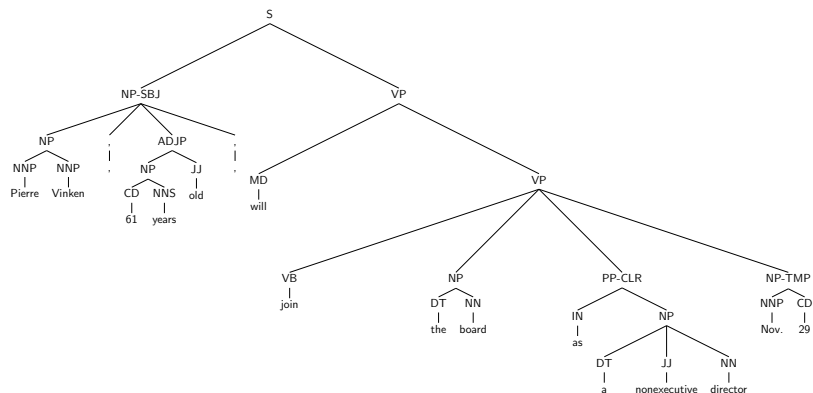
As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.

- ▶ Need lots of categories to make sure all and only grammatical sentences are included.
- ▶ Categories tend to start exploding combinatorially.
- ▶ Alternative grammar formalisms are typically used for manual grammar construction; these are often based on constraints and a powerful algorithmic tool called *unification*.

Standard approach today:

1. Build a corpus of annotated sentences, called a **treebank**. (Memorable example: the Penn Treebank, Marcus et al., 1993.)
2. Extract rules from the treebank.
3. Optionally, use statistical models to generalize the rules.

# Example from the Penn Treebank



# LISP Encoding in the Penn Treebank

```
( (S
  (NP-SBJ-1
    (NP (NNP Rudolph) (NNP Agnew) )
    ( , , )
    (UCP
      (ADJP
        (NP (CD 55) (NNS years) )
        (JJ old) )
      (CC and)
      (NP
        (NP (JJ former) (NN chairman) )
        (PP (IN of)
          (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC) )))
      ( , , ) )
    (VP (VBD was)
      (VP (VBN named)
        (S
          (NP-SBJ (-NONE- *-1) )
          (NP-PRD
            (NP (DT a) (JJ nonexecutive) (NN director) )
            (PP (IN of)
              (NP (DT this) (JJ British) (JJ industrial) (NN conglomerate) )))
          ( . . ) )
        ( . . ) )
      ( . . ) )
    ( . . ) )
  ( . . ) )
```

## Some Penn Treebank Rules with Counts

40717 PP → IN NP	100 VP → VBD PP-PRD
33803 S → NP-SBJ VP	100 PRN → : NP :
22513 NP-SBJ → -NONE-	100 NP → DT JJS
21877 NP → NP PP	100 NP-CLR → NN
20740 NP → DT NN	99 NP-SBJ-1 → DT NNP
14153 S → NP-SBJ VP .	98 VP → VBN NP PP-DIR
12922 VP → TO VP	98 VP → VBD PP-TMP
11881 PP-LOC → IN NP	98 PP-TMP → VBG NP
11467 NP-SBJ → PRP	97 VP → VBD ADVP-TMP VP
11378 NP → -NONE-	...
11291 NP → NN	10 WHNP-1 → WRB JJ
...	10 VP → VP CC VP PP-TMP
989 VP → VBG S	10 VP → VP CC VP
985 NP-SBJ → NN	ADVP-MNR
983 PP-MNR → IN NP	10 VP → VBZ S , SBAR-ADV
983 NP-SBJ → DT	10 VP → VBZ S ADVP-TMP
969 VP → VBN VP	
...	

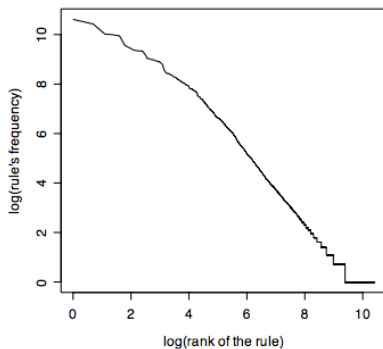


## Penn Treebank Rules: Statistics

32,728 rules in the training section (not including 52,257 lexicon rules)

4,021 rules in the development section

overlap: 3,128



# (Phrase-Structure) Recognition and Parsing

Given a CFG  $(\mathcal{N}, S, \Sigma, \mathcal{R})$  and a sentence  $x$ , the **recognition** problem is:

Is  $x$  in the language of the CFG?

Related problem: **parsing**:

Show one or more derivations for  $x$ , using  $\mathcal{R}$ .

# (Phrase-Structure) Recognition and Parsing

Given a CFG  $(\mathcal{N}, S, \Sigma, \mathcal{R})$  and a sentence  $x$ , the **recognition** problem is:

Is  $x$  in the language of the CFG?

The proof is a derivation.

Related problem: **parsing**:

Show one or more derivations for  $x$ , using  $\mathcal{R}$ .

# (Phrase-Structure) Recognition and Parsing

Given a CFG  $(\mathcal{N}, S, \Sigma, \mathcal{R})$  and a sentence  $x$ , the **recognition** problem is:

Is  $x$  in the language of the CFG?

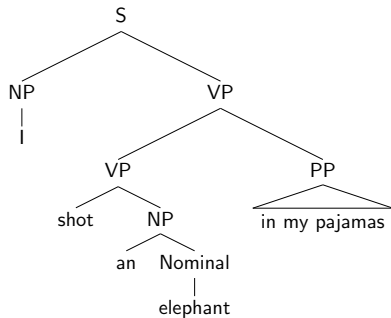
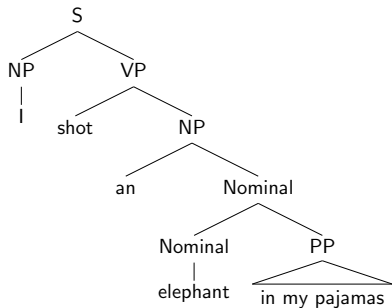
The proof is a derivation.

Related problem: **parsing**:

Show one or more derivations for  $x$ , using  $\mathcal{R}$ .

With reasonable grammars, the number of parses is exponential in  $|x|$ .

# Ambiguity



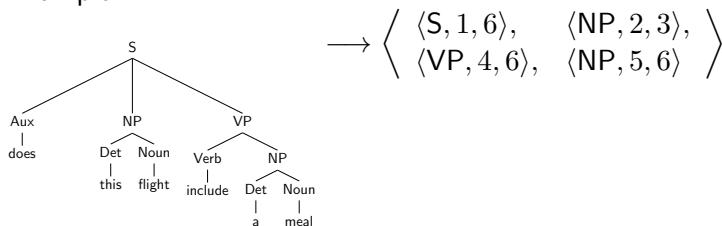
# Parser Evaluation

Represent a parse tree as a collection of tuples

$\langle \langle \ell_1, i_1, j_1 \rangle, \langle \ell_2, i_2, j_2 \rangle, \dots, \langle \ell_n, i_n, j_n \rangle \rangle$ , where

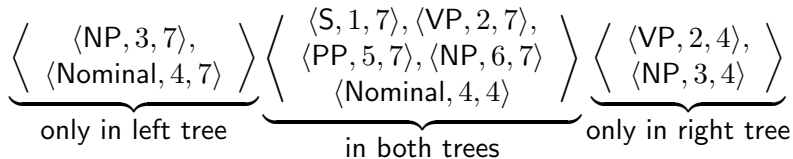
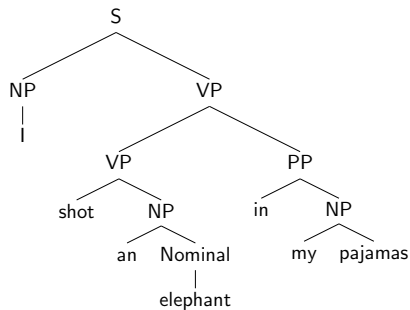
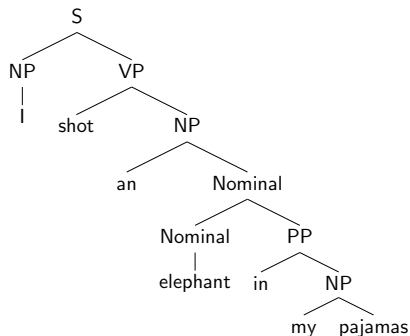
- ▶  $\ell_k$  is the nonterminal labeling the  $k$ th phrase
- ▶  $i_k$  is the index of the first word in the  $k$ th phrase
- ▶  $j_k$  is the index of the last word in the  $k$ th phrase

Example:



Convert gold-standard tree and system hypothesized tree into this representation, then estimate precision, recall, and  $F_1$ .

# Tree Comparison Example



# Two Views of Parsing



# Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.

# Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.
  - ▶ Often **greedy**, with a statistical classifier deciding what action to take in every state.

# Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.
  - ▶ Often **greedy**, with a statistical classifier deciding what action to take in every state.
2. Discrete optimization: define a scoring function and seek the tree with the highest score.

# Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.
  - ▶ Often **greedy**, with a statistical classifier deciding what action to take in every state.
2. Discrete optimization: define a scoring function and seek the tree with the highest score.
  - ▶ Today: scores are defined using the rules.

$$\text{predict}(\mathbf{x}) = \operatorname{argmax}_{\mathbf{t}} \prod_{r \in \mathcal{R}} s(r)^{c_{\mathbf{t}}(r)} = \operatorname{argmax}_{\mathbf{t}} \sum_{r \in \mathcal{R}} c_{\mathbf{t}}(r) \log s(r)$$

where  $\mathbf{t}$  is constrained to include grammatical trees with  $\mathbf{x}$  as their yield. Denote this set  $\mathcal{T}_{\mathbf{x}}$ .

# Probabilistic Context-Free Grammar

A **probabilistic context-free grammar** consists of:

- ▶ A finite set of nonterminal symbols  $\mathcal{N}$ 
  - ▶ A start symbol  $S \in \mathcal{N}$
- ▶ A finite alphabet  $\Sigma$ , called “terminal” symbols, distinct from  $\mathcal{N}$
- ▶ Production rule set  $\mathcal{R}$ , each of the form “ $N \rightarrow \alpha$ ” where
  - ▶ The lefthand side  $N$  is a nonterminal from  $\mathcal{N}$
  - ▶ The righthand side  $\alpha$  is a sequence of zero or more terminals and/or nonterminals:  $\alpha \in (\mathcal{N} \cup \Sigma)^*$ 
    - ▶ Special case: **Chomsky normal form** constrains  $\alpha$  to be either a single terminal symbol or two nonterminals
- ▶ For each  $N \in \mathcal{N}$ , a probability distribution over the rules where  $N$  is the lefthand side,  $p(* \mid N)$ .

# PCFG Example

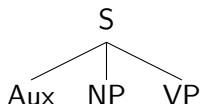
S

Write down the start symbol. Here: S

Score:

1

## PCFG Example

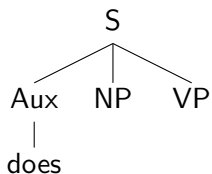


Choose a rule from the “S” distribution. Here:  $S \rightarrow \text{Aux NP VP}$

Score:

$$p(\text{Aux NP VP} \mid S)$$

## PCFG Example



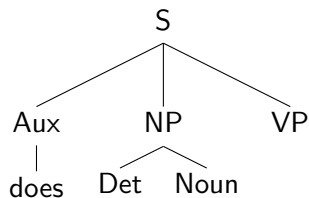
Choose a rule from the “Aux” distribution. Here:  $\text{Aux} \rightarrow \text{does}$

Score:

$$p(\text{Aux NP VP} \mid S) \cdot p(\text{does} \mid \text{Aux})$$



## PCFG Example

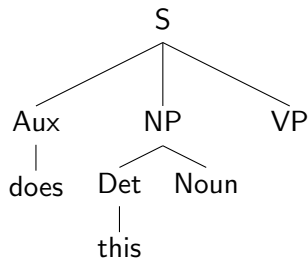


Choose a rule from the “NP” distribution. Here: NP  $\rightarrow$  Det Noun

Score:

$$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP})$$

## PCFG Example

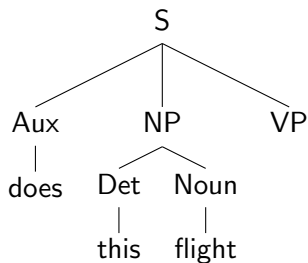


Choose a rule from the “Det” distribution. Here: Det  $\rightarrow$  this

Score:

$$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det})$$

## PCFG Example

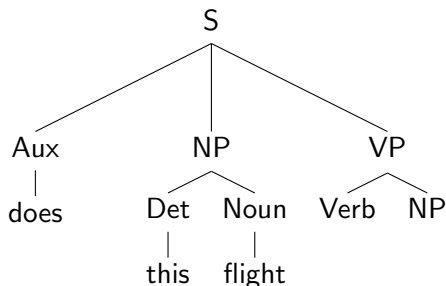


Choose a rule from the “Noun” distribution. Here: Noun  $\rightarrow$  flight

Score:

$$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det}) \\ \cdot p(\text{flight} \mid \text{Noun})$$

## PCFG Example

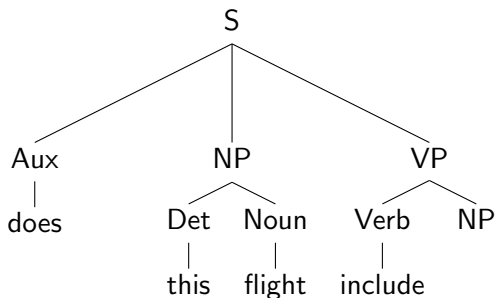


Choose a rule from the “VP” distribution. Here:  $VP \rightarrow \text{Verb NP}$

Score:

$$p(\text{Aux NP VP} \mid S) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det}) \\ \cdot p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP})$$

## PCFG Example

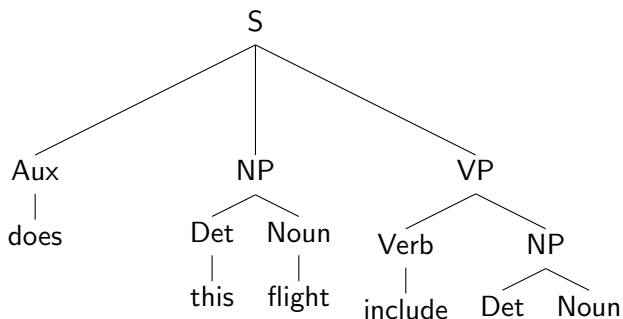


Choose a rule from the “Verb” distribution. Here: Verb  $\rightarrow$  include

Score:

$$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det}) \\ \cdot p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP}) \cdot p(\text{include} \mid \text{Verb})$$

## PCFG Example

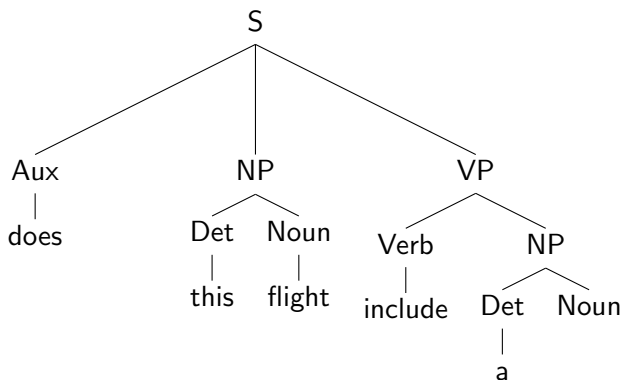


Choose a rule from the “NP” distribution. Here: NP  $\rightarrow$  Det Noun

Score:

$$\begin{aligned} & p(\text{Aux NP VP} \mid S) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det}) \\ & \cdot p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP}) \cdot p(\text{include} \mid \text{Verb}) \\ & \cdot p(\text{Det Noun} \mid \text{NP}) \end{aligned}$$

## PCFG Example

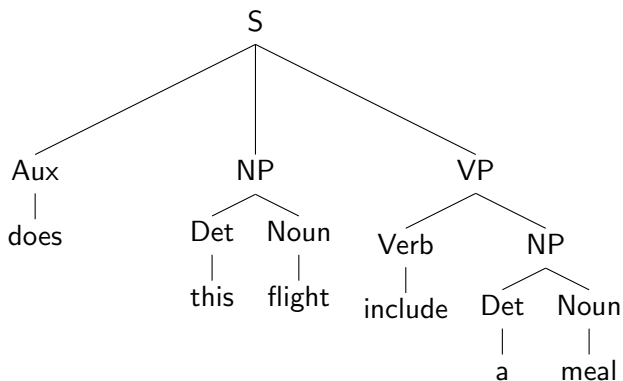


Choose a rule from the “Det” distribution. Here: Det  $\rightarrow$  a

Score:

$$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det}) \\ \cdot p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP}) \cdot p(\text{include} \mid \text{Verb}) \\ \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{a} \mid \text{Det})$$

## PCFG Example

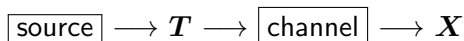


Choose a rule from the “Noun” distribution. Here: Noun  $\rightarrow$  meal  
Score:

$$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det}) \\ \cdot p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP}) \cdot p(\text{include} \mid \text{Verb}) \\ \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{a} \mid \text{Det}) \cdot p(\text{meal} \mid \text{Noun})$$



# PCFG as a Noisy Channel



The PCFG defines the source model.

The channel is deterministic: it erases everything except the tree's leaves (the yield).

Decoding:

$$\begin{aligned} & \operatorname{argmax}_{\mathbf{t}} p(\mathbf{t}) \cdot \begin{cases} 1 & \text{if } \mathbf{t} \in \mathcal{T}_x \\ 0 & \text{otherwise} \end{cases} \\ & = \operatorname{argmax}_{\mathbf{t} \in \mathcal{T}_x} p(\mathbf{t}) \end{aligned}$$

# Probabilistic Parsing with CFGs

- ▶ How to set the probabilities  $p(\text{righthand side} \mid \text{lefthand side})$ ?
- ▶ How to decode/parse?

# Probabilistic CKY

(Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967)

Input:

- ▶ a PCFG  $(\mathcal{N}, S, \Sigma, \mathcal{R}, p(* | *))$ , in **Chomsky normal form**
- ▶ a sentence  $x$  (let  $n$  be its length)

Output:  $\operatorname{argmax}_{t \in \mathcal{T}_x} p(t | x)$  (if  $x$  is in the language of the grammar)

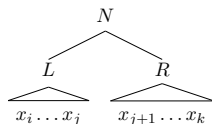
# Probabilistic CKY

Base case: for  $i \in \{1, \dots, n\}$  and for each  $N \in \mathcal{N}$ :

$$s_{i:i}(N) = p(x_i | N)$$

For each  $i, k$  such that  $1 \leq i < k \leq n$  and each  $N \in \mathcal{N}$ :

$$s_{i:k}(N) = \max_{L, R \in \mathcal{N}, j \in \{i, \dots, k-1\}} p(L R | N) \cdot s_{i:j}(L) \cdot s_{(j+1):k}(R)$$



Solution:

$$s_{1:n}(S) = \max_{t \in \mathcal{T}_x} p(t)$$

# Parse Chart

$x_1$				
	$x_2$			
		$x_3$		
			$x_4$	
				$x_5$

# Parse Chart

	$s_{1:1}(*)$				
$x_1$		$s_{2:2}(*)$			
	$x_2$		$s_{3:3}(*)$		
		$x_3$		$s_{4:4}(*)$	
			$x_4$		$s_{5:5}(*)$
				$x_5$	

# Parse Chart

	$s_{1:1}(*)$	$s_{1:2}(*)$			
$x_1$		$s_{2:2}(*)$	$s_{2:3}(*)$		
	$x_2$		$s_{3:3}(*)$	$s_{3:4}(*)$	
		$x_3$		$s_{4:4}(*)$	$s_{4:5}(*)$
			$x_4$		$s_{5:5}(*)$
				$x_5$	

# Parse Chart

	$s_{1:1}(*)$	$s_{1:2}(*)$	$s_{1:3}(*)$		
$x_1$		$s_{2:2}(*)$	$s_{2:3}(*)$	$s_{2:4}(*)$	
			$s_{3:3}(*)$	$s_{3:4}(*)$	$s_{3:5}(*)$
$x_2$				$s_{4:4}(*)$	$s_{4:5}(*)$
					$s_{5:5}(*)$
			$x_3$		
				$x_4$	
					$x_5$



# Parse Chart

	$s_{1:1}(*)$	$s_{1:2}(*)$	$s_{1:3}(*)$	$s_{1:4}(*)$	
$x_1$		$s_{2:2}(*)$	$s_{2:3}(*)$	$s_{2:4}(*)$	$s_{2:5}(*)$
$x_2$			$s_{3:3}(*)$	$s_{3:4}(*)$	$s_{3:5}(*)$
				$s_{4:4}(*)$	$s_{4:5}(*)$
					$s_{5:5}(*)$

$x_5$

# Parse Chart

	$s_{1:1}(*)$	$s_{1:2}(*)$	$s_{1:3}(*)$	$s_{1:4}(*)$	$s_{1:5}(*)$
$x_1$		$s_{2:2}(*)$	$s_{2:3}(*)$	$s_{2:4}(*)$	$s_{2:5}(*)$
$x_2$			$s_{3:3}(*)$	$s_{3:4}(*)$	$s_{3:5}(*)$
$x_3$				$s_{4:4}(*)$	$s_{4:5}(*)$
$x_4$					$s_{5:5}(*)$
$x_5$					

# Remarks

- ▶ Space and runtime requirements?

## Remarks

- ▶ Space and runtime requirements?  $O(|\mathcal{N}|n^2)$  space,  $O(|\mathcal{R}|n^3)$  runtime.

## Remarks

- ▶ Space and runtime requirements?  $O(|\mathcal{N}|n^2)$  space,  $O(|\mathcal{R}|n^3)$  runtime.
- ▶ Recovering the best tree?

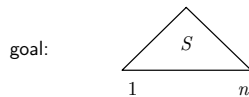
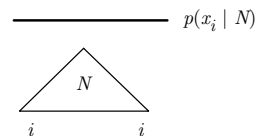
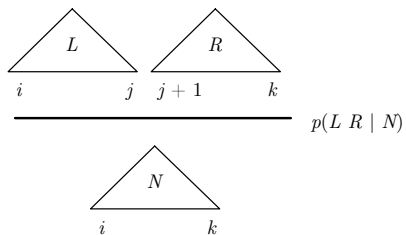
## Remarks

- ▶ Space and runtime requirements?  $O(|\mathcal{N}|n^2)$  space,  $O(|\mathcal{R}|n^3)$  runtime.
- ▶ Recovering the best tree? Backpointers.

## Remarks

- ▶ Space and runtime requirements?  $O(|\mathcal{N}|n^2)$  space,  $O(|\mathcal{R}|n^3)$  runtime.
- ▶ Recovering the best tree? Backpointers.
- ▶ Probabilistic **Earley's** algorithm does not require the grammar to be in Chomsky normal form.

# The Declarative View of CKY





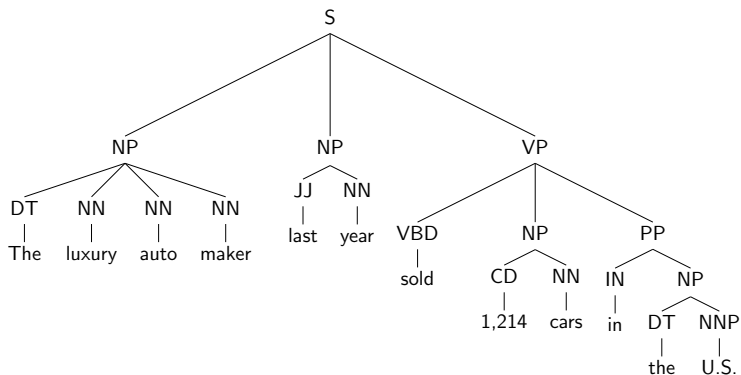
# Probabilistic CKY with an Agenda

1. Initialize every item's value in the **chart** to the “default” (zero).
2. Place all initializing updates onto the **agenda**.
3. While the agenda is not empty or the goal is not reached:
  - ▶ Pop the highest-priority update from the agenda (item  $I$  with value  $v$ )
  - ▶ If  $I = \text{goal}$ , then return  $v$ .
  - ▶ If  $v > \text{chart}(I)$ :
    - ▶  $\text{chart}(I) \leftarrow v$
    - ▶ Find all combinations of  $I$  with other items in the chart, generating new possible updates; place these on the agenda.

Any priority function will work! But smart ordering will save time.

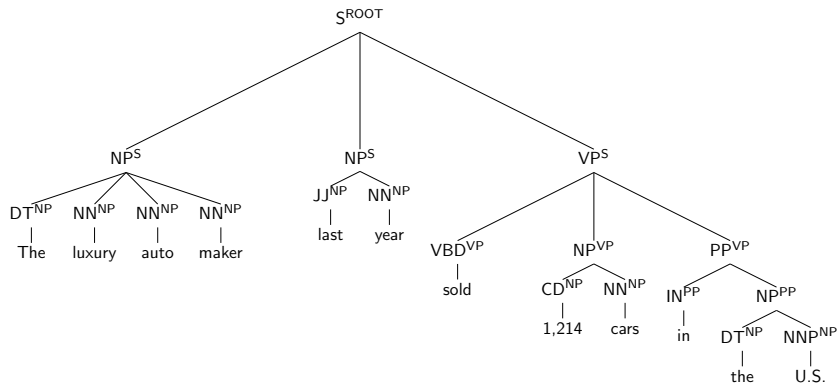
This idea can also be applied to other algorithms (e.g., Viterbi).

# Starting Point: Phrase Structure



# Parent Annotation

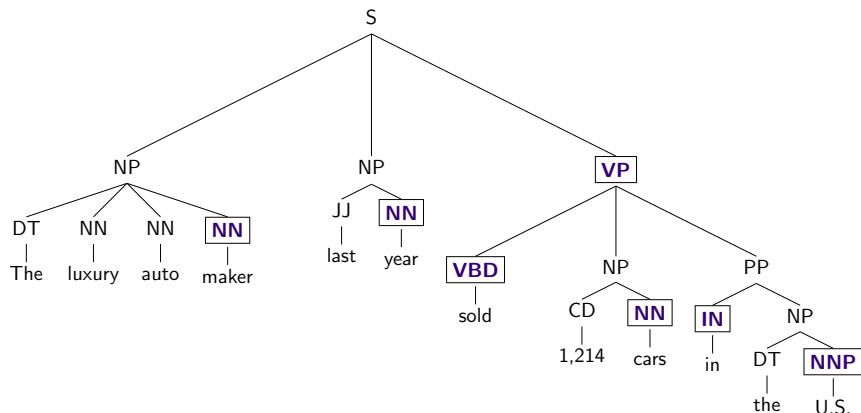
(Johnson, 1998)



Increases the “vertical” Markov order:

$$p(\text{children} \mid \text{parent, grandparent})$$

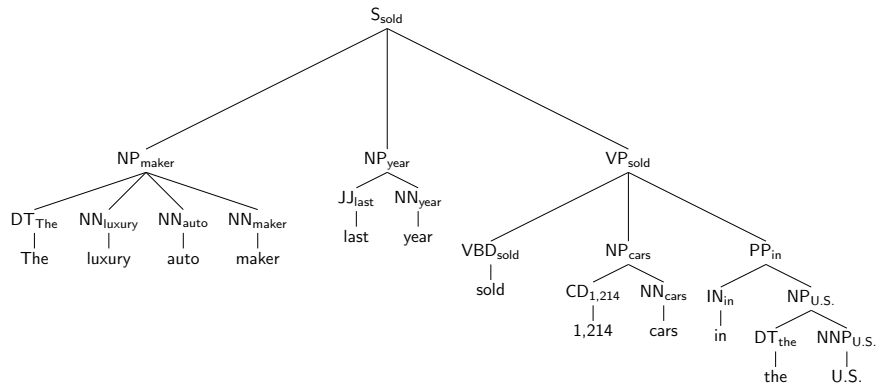
# Headedness



Suggests "horizontal" markovization:

$$p(\text{children} \mid \text{parent}) = p(\text{head} \mid \text{parent}) \cdot \prod_i p(i\text{th sibling} \mid \text{head}, \text{parent})$$

# Lexicalization



Each node shares a **lexical head** with its head child.

# Transformations on Trees

Starting around 1998, many different ideas—both linguistic and statistical—about how to transform treebank trees.

All of these make the grammar larger—and therefore all frequencies became sparser—so a lot of research on *smoothing* the probability rules.

Parent annotation, headedness, markovization, and lexicalization; also category *refinement* by linguistic rules (Klein and Manning, 2003).

- ▶ These are reflected in some versions of the popular Stanford and Berkeley parsers.

# Tree Decorations

(Klein and Manning, 2003)

- ▶ Mark nodes with only 1 child as UNARY
- ▶ Mark DTs (determiners), RBs (adverbs) when they are only children
- ▶ Annotate POS tags with their parents
- ▶ Split IN (prepositions; 6 ways), AUX, CC, %
- ▶ NPs: temporal, possessive, base
- ▶ VPs annotated with head tag (finite vs. others)
- ▶ DOMINATES-V
- ▶ RIGHT-RECURSIVE NP

# Machine Learning and Parsing



# Machine Learning and Parsing

- ▶ Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses, then train a log-linear model to rerank (Charniak and Johnson, 2005).
  - ▶ K-best parsing: Huang and Chiang (2005)

# Machine Learning and Parsing

- ▶ Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses, then train a log-linear model to rerank (Charniak and Johnson, 2005).
  - ▶ K-best parsing: Huang and Chiang (2005)
- ▶ Define rule-local features on trees (and any part of the input sentence); minimize hinge or log loss.
  - ▶ These exploit dynamic programming algorithms for training (CKY for arbitrary scores, and the sum-product version).

# Machine Learning and Parsing

- ▶ Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses, then train a log-linear model to rerank (Charniak and Johnson, 2005).
  - ▶ K-best parsing: Huang and Chiang (2005)
- ▶ Define rule-local features on trees (and any part of the input sentence); minimize hinge or log loss.
  - ▶ These exploit dynamic programming algorithms for training (CKY for arbitrary scores, and the sum-product version).
- ▶ Learn refinements on the constituents, as latent variables (Petrov et al., 2006).

# Machine Learning and Parsing

- ▶ Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses, then train a log-linear model to rerank (Charniak and Johnson, 2005).
  - ▶ K-best parsing: Huang and Chiang (2005)
- ▶ Define rule-local features on trees (and any part of the input sentence); minimize hinge or log loss.
  - ▶ These exploit dynamic programming algorithms for training (CKY for arbitrary scores, and the sum-product version).
- ▶ Learn refinements on the constituents, as latent variables (Petrov et al., 2006).
- ▶ Neural, too:
  - ▶ Socher et al. (2013) define **compositional vector grammars** that associate each phrase with a vector, calculated as a function of its subphrases' vectors. Used essentially to rerank.
  - ▶ Dyer et al. (2016): **recurrent neural network grammars**, generative models like PCFGs that encode arbitrary previous derivation steps in a vector. Parsing requires some tricks.

# To-Do List

- ▶ Collins (2011)
- ▶ Assignment 3 is due February 20.

## Extras

# Structured Perceptron

Collins (2002)

Perceptron algorithm for **parsing**:

- ▶ For  $t \in \{1, \dots, T\}$ :
  - ▶ Pick  $i_t$  uniformly at random from  $\{1, \dots, n\}$ .
  - ▶  $\hat{t}_{i_t} \leftarrow \operatorname{argmax}_{t \in \mathcal{T}_{x_{i_t}}} \mathbf{w} \cdot \Phi(\mathbf{x}_{i_t}, t)$
  - ▶  $\mathbf{w} \leftarrow \mathbf{w} - \alpha (\Phi(\mathbf{x}_{i_t}, \hat{t}_{i_t}) - \Phi(\mathbf{x}_{i_t}, t_{i_t}))$

This can be viewed as stochastic subgradient descent on the *structured* hinge loss:

$$\sum_{i=1}^n \underbrace{\max_{t \in \mathcal{T}_{x_{i_t}}} \mathbf{w} \cdot \Phi(\mathbf{x}_{i_t}, t)}_{\text{fear}} - \underbrace{\mathbf{w} \cdot \Phi(\mathbf{x}_{i_t}, t_{i_t})}_{\text{hope}}$$

# Beyond Structured Perceptron (I)

**Structured support vector machine** (also known as **max margin parsing**; Taskar et al., 2004):

$$\sum_{i=1}^n \max_{\mathbf{t} \in \mathcal{T}_{\mathbf{x}_{i_t}}} \underbrace{\mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{t}) + \text{cost}(\mathbf{t}_{i_t}, \mathbf{t})}_{\text{fear}} - \underbrace{\mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{t}_i)}_{\text{hope}}$$

where  $\text{cost}(\mathbf{t}_i, \mathbf{t})$  is the number of local errors (either constituent errors or “rule” errors).



## Beyond Structured Perceptron (II)

Log-loss, which gives parsing models analogous to **conditional random fields** (Miyao and Tsujii, 2002; Finkel et al., 2008):

$$\sum_{i=1}^n \log \underbrace{\sum_{t \in \mathcal{T}_{\mathbf{x}_i}} \exp \mathbf{w} \cdot \Phi(\mathbf{x}_i, t)}_{\text{fear}} - \underbrace{\mathbf{w} \cdot \Phi(\mathbf{x}_i, t_i)}_{\text{hope}}$$

# References I

- Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*, 2005.
- Noam Chomsky. Three models for the description of language. *Information Theory, IEEE Transactions on*, 2(3):113–124, 1956.
- John Cocke and Jacob T. Schwartz. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University, 1970.
- Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*, 2002.
- Michael Collins. Probabilistic context-free grammars, 2011. URL <http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/pcfgs.pdf>.
- Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars, 2016. To appear.
- Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. Efficient, feature-based, conditional random field parsing. In *Proc. of ACL*, 2008.
- Liang Huang and David Chiang. Better  $k$ -best parsing. In *Proc. of IWPT*, 2005.
- Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–32, 1998.
- Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, second edition, 2008.

## References II

- Tadao Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, 1965.
- Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proc. of ACL*, 2003.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2): 313–330, 1993.
- Yusuke Miyao and Jun'ichi Tsujii. Maximum entropy estimation for feature forests. In *Proc. of HLT*, 2002.
- Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL*, 2006.
- Geoffrey K. Pullum. *The Great Eskimo Vocabulary Hoax and Other Irreverent Essays on the Study of Language*. University of Chicago Press, 1991.
- Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *Proc. of ACL*, 2013.
- Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*. 2004.
- Daniel H. Younger. Recognition and parsing of context-free languages in time  $n^3$ . *Information and Control*, 10(2), 1967.