Natural Language Processing (CSE 517): Phrase Structure Syntax and Parsing

Noah Smith

University of Washington
nasmith@cs.washington.edu

May 9, 2018

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

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

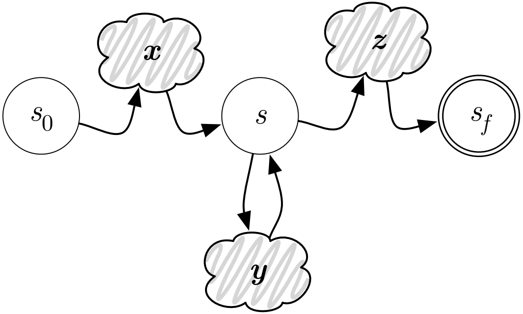
This is sometimes called a **path**.

Terminology from Theory of Computation

- A **regular expression** can be:
  - an empty string (usually denoted ϵ) or a symbol from Σ
  - 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

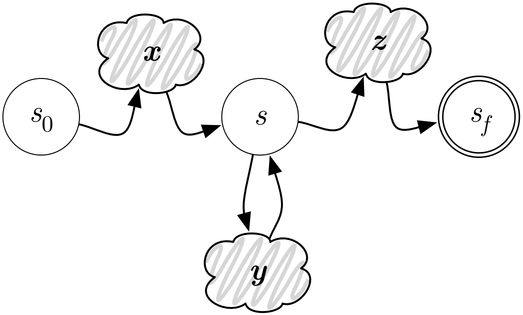
Pumping lemma (for regular languages): if L is an infinite regular language, then there exist strings $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, with $\boldsymbol{y} \neq \epsilon$, such that $\boldsymbol{x}\boldsymbol{y}^n\boldsymbol{z} \in L$, for all $n \geq 0$.



If L is infinite and $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ 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 $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, with $\boldsymbol{y} \neq \epsilon$, such that $\boldsymbol{x}\boldsymbol{y}^n\boldsymbol{z} \in L$, for all $n \geq 0$.

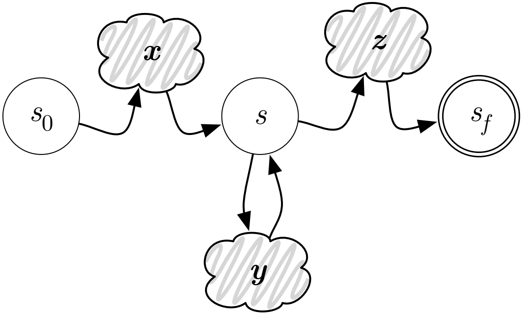


If L is infinite and $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ do not exist, then L is not regular.

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 $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, with $\boldsymbol{y} \neq \epsilon$, such that $\boldsymbol{x}\boldsymbol{y}^n\boldsymbol{z} \in L$, for all $n \geq 0$.



If L is infinite and $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ 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.

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

$L_1 \cap L_2$ 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 Σ, called "terminal" symbols, distinct from $\mathcal{N}$
- Production rule set $\mathcal{R}$, each of the form "$N \rightarrow \boldsymbol{\alpha}$" where
  - The lefthand side N is a nonterminal from $\mathcal{N}$
  - The righthand side $\boldsymbol{\alpha}$ is a sequence of zero or more terminals and/or nonterminals: $\boldsymbol{\alpha} \in (\mathcal{N} \cup \Sigma)^*$
    - Special case: **Chomsky normal form** constrains $\boldsymbol{\alpha}$ to be either a single terminal symbol or two nonterminals

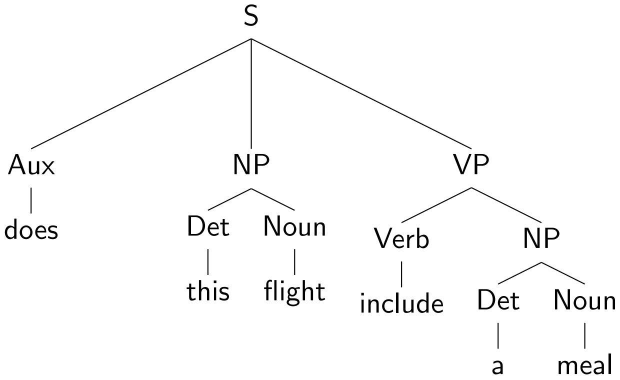
An Example CFG for a Tiny Bit of English

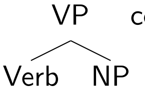
From Jurafsky and Martin (2008)

$S \rightarrow$ NP VP
$S \rightarrow$ Aux NP VP
$S \rightarrow$ VP
NP $\rightarrow$ Pronoun
NP $\rightarrow$ Proper-Noun
NP $\rightarrow$ Det 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 $\rightarrow$ VP PP
PP $\rightarrow$ Preposition NP

Det $\rightarrow$ that | this | a
Noun $\rightarrow$ book | flight | meal | money
Verb $\rightarrow$ book | include | prefer
Pronoun $\rightarrow$ I | she | me
Proper-Noun $\rightarrow$ Houston | NWA
Aux $\rightarrow$ does
Preposition $\rightarrow$ from | to | on | near
| through

Example Phrase Structure Tree

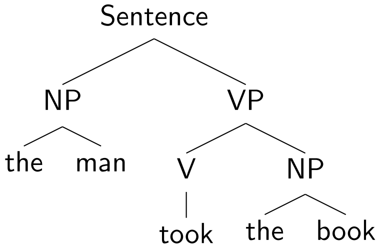


The phrase-structure tree represents both the syntactic structure of the sentence and the **derivation** of the sentence under the grammar. E.g., 

corresponds to the rule VP $\rightarrow$ 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?

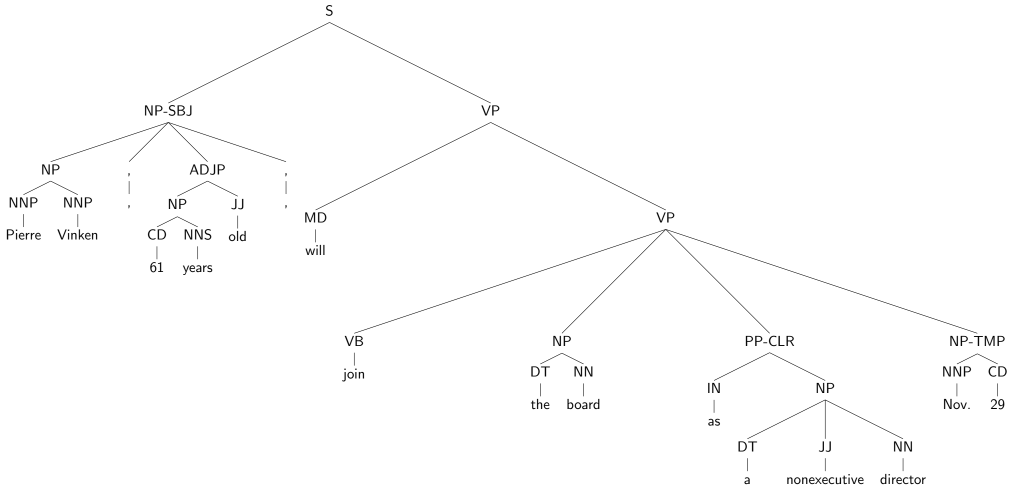
As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.

- 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) ))))))
    (. .) ))
```

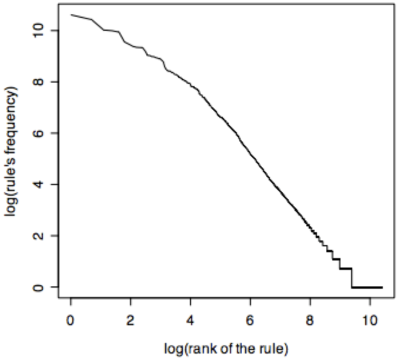
Some Penn Treebank Rules with Counts

40717 PP → IN NP
33803 S → NP-SBJ VP
22513 NP-SBJ → -NONE-
21877 NP → NP PP
20740 NP → DT NN
14153 S → NP-SBJ VP .
12922 VP → TO VP
11881 PP-LOC → IN NP
11467 NP-SBJ → PRP
11378 NP → -NONE-
11291 NP → NN
. . .
989 VP → VBG S
985 NP-SBJ → NN
983 PP-MNR → IN NP
983 NP-SBJ → DT
969 VP → VBN VP

100 VP → VBD PP-PRD
100 PRN → : NP :
100 NP → DT JJS
100 NP-CLR → NN
99 NP-SBJ-1 → DT NNP
98 VP → VBN NP PP-DIR
98 VP → VBD PP-TMP
98 PP-TMP → VBG NP
97 VP → VBD ADVP-TMP VP
. . .
10 WHNP-1 → WRB JJ
10 VP → VP CC VP PP-TMP
10 VP → VP CC VP ADVP-MNR
10 VP → VBZ S , SBAR-ADV
10 VP → VBZ S ADVP-TMP

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 $\boldsymbol{x}$, the **recognition** problem is:

Is $\boldsymbol{x}$ in the language of the CFG?

Related problem: **parsing**:

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

(Phrase-Structure) Recognition and Parsing

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

Is $\boldsymbol{x}$ in the language of the CFG?

The proof is a derivation.

Related problem: **parsing**:

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

(Phrase-Structure) Recognition and Parsing

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

Is $\boldsymbol{x}$ in the language of the CFG?

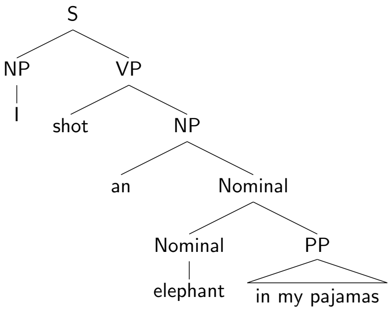
The proof is a derivation.

Related problem: **parsing**:

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

With reasonable grammars, the number of parses is exponential in $|\boldsymbol{x}|$.

Ambiguity

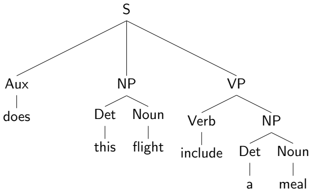

S
NP
VP
I
shot
NP
an
Nominal
Nominal
PP
elephant
in my pajamas
S
NP
VP
I
VP
PP
shot
NP
in my pajamas
an
Nominal
elephant

Parser Evaluation

Represent a parse tree as a collection of tuples $\langle\langle \ell_1, i_1, j_1\rangle, \langle \ell_2, i_2, j_2\rangle, \ldots, \langle \ell_n, i_n, j_n\rangle$, where

- ℓ_k is the nonterminal labeling the kth phrase
- i_k is the index of the first word in the kth phrase
- j_k is the index of the last word in the kth phrase

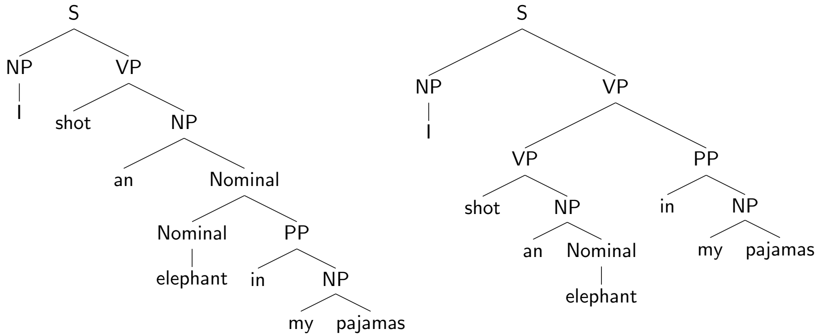
Example:



$$\longrightarrow \left\langle \begin{array}{ll} \langle \mathsf{S}, 1, 6\rangle, & \langle \mathsf{NP}, 2, 3\rangle, \\ \langle \mathsf{VP}, 4, 6\rangle, & \langle \mathsf{NP}, 5, 6\rangle \end{array} \right\rangle$$

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

Tree Comparison Example



$$\underbrace{\left\langle \begin{matrix} \langle \text{NP}, 3, 7\rangle, \\ \langle \text{Nominal}, 4, 7\rangle \end{matrix} \right\rangle}_{\text{only in left tree}} \underbrace{\left\langle \begin{matrix} \langle \text{NP}, 1, 1\rangle \\ \langle \text{S}, 1, 7\rangle, \langle \text{VP}, 2, 7\rangle, \\ \langle \text{PP}, 5, 7\rangle, \langle \text{NP}, 6, 7\rangle \\ \langle \text{Nominal}, 4, 4\rangle \end{matrix} \right\rangle}_{\text{in both trees}} \underbrace{\left\langle \begin{matrix} \langle \text{VP}, 2, 4\rangle, \\ \langle \text{NP}, 3, 4\rangle \end{matrix} \right\rangle}_{\text{only in right tree}}$$

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.

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

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

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 Σ, called "terminal" symbols, distinct from $\mathcal{N}$
- Production rule set $\mathcal{R}$, each of the form "$N \rightarrow \boldsymbol{\alpha}$" where
  - The lefthand side N is a nonterminal from $\mathcal{N}$
  - The righthand side $\boldsymbol{\alpha}$ is a sequence of zero or more terminals and/or nonterminals: $\boldsymbol{\alpha} \in (\mathcal{N} \cup \Sigma)^*$
    - Special case: **Chomsky normal form** constrains $\boldsymbol{\alpha}$ 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

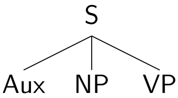
S

Write down the start symbol. Here: S

Score:

$$1$$

PCFG Example

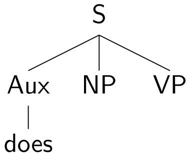


Choose a rule from the "S" distribution. Here: S $\rightarrow$ Aux NP VP

Score:

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

PCFG Example

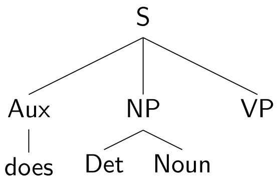


Choose a rule from the "Aux" distribution. Here: Aux $\rightarrow$ does

Score:

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

PCFG Example

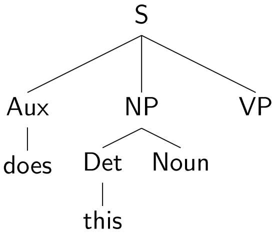


Choose a rule from the "NP" distribution. Here: NP $\rightarrow$ Det Noun

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

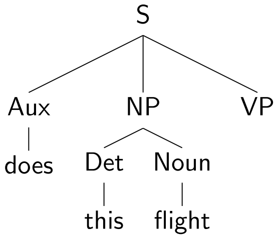


Choose a rule from the “Det” distribution. Here: Det $\rightarrow$ this

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

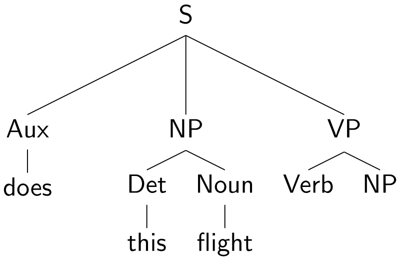


Choose a rule from the "Noun" distribution. Here: Noun $\rightarrow$ flight

Score:

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

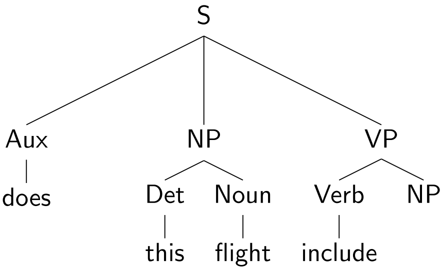
PCFG Example



Choose a rule from the "VP" distribution. Here: VP $\rightarrow$ Verb NP
Score:

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

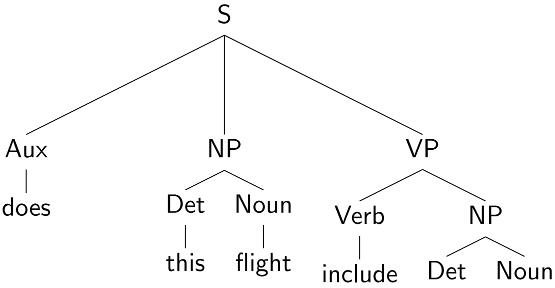
PCFG Example



Choose a rule from the "Verb" distribution. Here: Verb $\rightarrow$ include
Score:

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

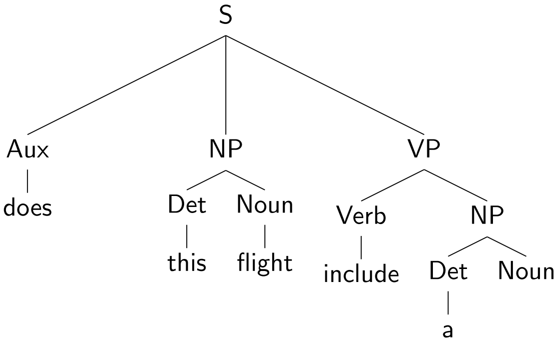
PCFG Example



Choose a rule from the "NP" distribution. Here: NP $\rightarrow$ Det Noun
Score:

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

PCFG Example

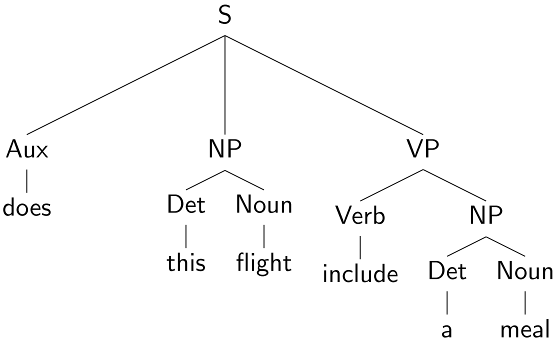


Choose a rule from the "Det" distribution. Here: Det $\rightarrow$ a

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

$$\boxed{\text{source}} \longrightarrow \boldsymbol{T} \longrightarrow \boxed{\text{channel}} \longrightarrow \boldsymbol{X}$$

The PCFG defines the source model.

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

Decoding:

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

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

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

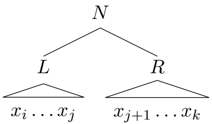
Probabilistic CKY

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

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

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

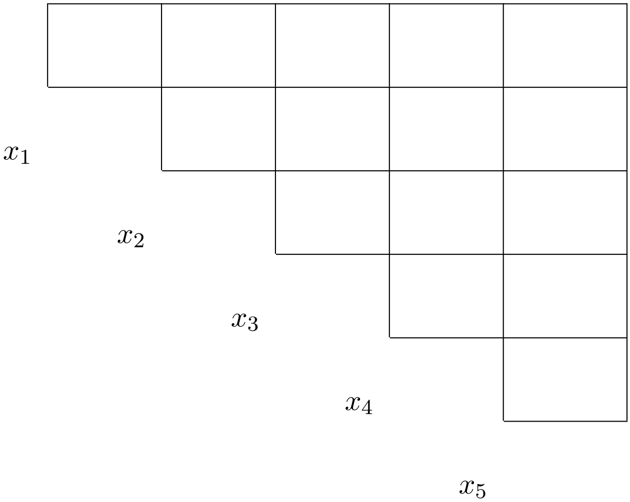
$$s_{i:k}(N) = \max_{L,R \in \mathcal{N}, j \in \{i,\ldots,k-1\}} p(L\ R \mid N) \cdot s_{i:j}(L) \cdot s_{(j+1):k}(R)$$



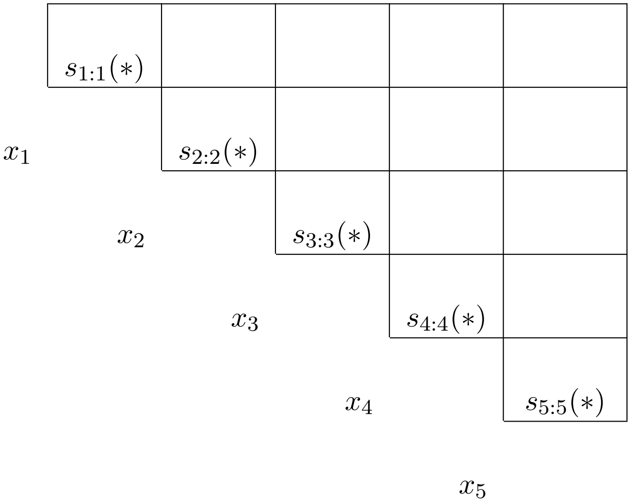
Solution:

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

Parse Chart



Parse Chart



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		$s_{2:2}(*)$	$s_{2:3}(*)$	$s_{2:4}(*)$	
	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	

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		$s_{3:3}(*)$	$s_{3:4}(*)$	$s_{3:5}(*)$
		x_3		$s_{4:4}(*)$	$s_{4:5}(*)$
			x_4		$s_{5:5}(*)$
				x_5	

Parse Chart

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

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 $I = \mathrm{goal}$, then return v.
   - If $v > \mathrm{chart}(I)$:
     - $\mathrm{chart}(I) \leftarrow v$
     - 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).

References I

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.

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.

Tadao Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, 1965.

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.

Geoffrey K. Pullum. *The Great Eskimo Vocabulary Hoax and Other Irreverent Essays on the Study of Language*. University of Chicago Press, 1991.

Daniel H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10 (2), 1967.