Syntax Analysis

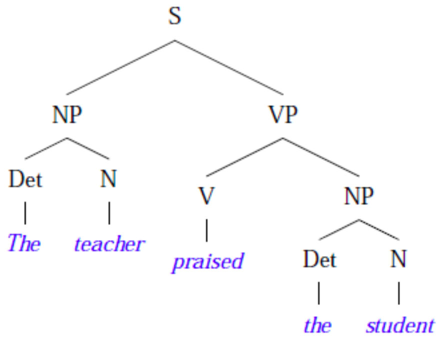
What is Syntax?

- Refers to the way words are arranged together, and the relationship between then.
- **Language Models:** Importance of modeling word order
- **POS categories:** An equivalence class for words
- More complex notions: constituency, grammatical relations, subcategorization etc.

What is Syntax?

It is the set of rules and processes that govern the structure of sentences (sentence structure) in a given language.

Syntax Tree: Example



Constituency

Constituent

A group of words acts as a single unit - phrases, clauses etc.

Part of Speech - “Substitution Test"

The {sad, intelligent, green, fat, …} one is in the corner.

Constituency: Noun Phrase

- *Kermit the frog*
- *they*
- *December twenty-sixth*
- *the reason he is running for president*

Constituency

- Working based on Constituency (Phrase structure)
  - Organizing words into nested constituents
  - Showing that groups of words within utterances can act as single units
  - Forming coherent classes from these units that can behave in similar ways
    - With respect to their internal structure
    - With respect to other units in the language
  - Considering a head word for each constituent

Constituency

the writer talked to the audiences about his new book.

the writer talked about his new book to the audiences. C

about his new book the writer talked to the audiences. C

the writer talked book to the audiences about his new. C

Constituent Phrases

Usually named based on the word that heads the constituent:

- the man from India: **Noun Phrase (head = man)**
- extremely clever: **Adjective Phrase (head = clever)**
- on December 26th: **Preposition Phrase (head = on)**

Single word can also act as a phrase. Any example?

Words can also act as phrases

Joe grew potatoes

Joe and *potatoes* are both nouns and noun phrases

Compare with: *The man from Amherst grew beautiful russet potatoes.*

Joe appears in a place that a larger noun phrase could have been.

Evidence that constituency exists

They appear in similar environments

Kermit the frog comes on stage
They come to Massachusetts every summer
December twenty-sixth comes after Christmas
The reason he is running for president comes out only now.
But not each individual word in the consituent
**The comes our...* **is comes out...* **for comes out...*

Can be placed in a number of different locations

Consituent = Prepositional phrase: *On December twenty-sixth*
On December twenty-sixth I'd like to fly to Florida.
I'd like to fly on December twenty-sixth to Florida.
I'd like to fly to Florida on December twenty-sixth.
But not split apart
**On December I'd like to fly twenty-sixth to Florida.*
**On I'd like to fly December twenty-sixth to Florida.*

Evidences of Constituency

Appear in similar syntactic environments (ex: before a verb)

- three parties from Brooklyn arrive ...
- a high-class spot such as Mindy's attracts ...

But, '**from arrive**'..., '**as attracts**'..., is incorrect.

Can be placed in a number of different location in a sentence

- On September seventeenth, I'd like to fly from Atlanta to Denver
- I'd like to fly on September seventeenth from Atlanta to Denver
- I'd like to fly from Atlanta to Denver on September seventeenth

But, '**On September, I'd like to fly seventeenth from Atlanta to Denver**' is incorrect.

Main Grammar Fragments

- Sentence
- Noun Phrase
  - Agreement
- Verb Phrase
  - Sub-categorization

Grammar Fragments: Sentence

- Declaratives
  A plane left.
  $S \rightarrow NP\ VP$

- Imperatives
  Leave!
  $S \rightarrow VP$

- Yes-No Questions
  Did the plane leave?
  $S \rightarrow Aux\ NP\ VP$

- WH Questions
  When did the plane leave?
  $S \rightarrow NP_{WH}\ Aux\ NP\ VP$

Grammar Fragments: NP

- Each NP has a central critical noun called head
- The head of an NP can be expressed using
  - Pre-nominals: the words that can come before the head
  - Post-nominals: the words that can come after the head

Grammar Fragments: NP

- Pre-nominals
  - Simple lexical items: *the, this, a, an, ...*
    a car
  - Simple possessives
    John's car
  - Complex recursive possessives
    John's sister's friend's car
  - Quantifiers, cardinals, ordinals...
    three cars
  - Adjectives
    large cars

Grammar Fragments: NP

- Post-nominals
  - Prepositional phrases
    flight from Seattle
  - Non-finite clauses
    flight arriving before noon
  - Relative clauses
    flight that serves breakfast

Agreement

- Having constraints that hold among various constituents
- Considering these constraints in a rule or set of rules

  Example: determiners and the head nouns in NPs have to agree in number

  This flight C
  Those flights C
  This flights C
  Those flight C

- Grammars that do not consider constraints will over-generate
  - Accepting and assigning correct structures to grammatical examples (*this flight*)
  - But also accepting incorrect examples (*these flight*)

Agreement at sentence level

- Considering similar constraints at sentence level

Example: subject and verb in sentences have to agree in number and person

John flies C
We fly C
John fly C
We flies C

Agreement

- Possible CFG solution

$S_{sg} \rightarrow NP_{sg}\, VP_{sg}$

$S_{pl} \rightarrow NP_{pl}\, VP_{pl}$

$NP_{sg} \rightarrow Det_{sg}\, N_{sg}$

$NP_{pl} \rightarrow Det_{pl}\;\; N_{pl}$

$VP_{sg} \rightarrow V_{sg}\, NP_{sg}$

$VP_{pl} \rightarrow V_{pl}\, NP_{pl}$

...

- Shortcoming:
    - Introducing many rules in the system

Grammar Fragments: VP

- VPs consist of a head verb along with zero or more constituents called arguments

$VP \rightarrow V$	*disappear*
$VP \rightarrow V \;\; NP$	*prefer a morning flight*
$VP \rightarrow V \;\; PP$	*fly on Thursday*
$VP \rightarrow V \;\; NP \;\; PP$	*leave Boston in the morning*
$VP \rightarrow V \;\; NP \;\; NP$	*give me the flight number*

- Arguments
  - Obligatory: complement
  - Optional: adjunct

Sub-categorization

- Even though there are many valid VP rules, not all verbs are allowed to participate in all VP rules

  disappear a morning flight C

- Solution:
  - Subcategorizing the verbs according to the sets of VP rules that they can participate in
  - This is a modern take on the traditional notion of transitive/intransitive
  - Modern grammars may have 100s or such classes

Sub-categorization

- Example:

Sneeze	*John sneezed*
Find	*Please find [a flight to NY]*$_{NP}$
Give	*Give [me]*$_{NP}$*[a cheaper fair]*$_{NP}$
Help	*Can you help [me]*$_{NP}$*[with a flight]*$_{PP}$
Prefer	*I prefer [to leave earlier]*$_{TO\text{-}VP}$
Told	*I was told [United has a flight]*$_{S}$

John sneezed the book C
I prefer United has a flight C
Give with a flight C

Sub-categorization

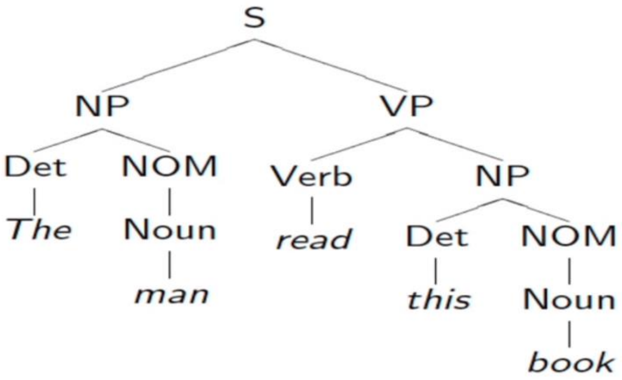
- The over-generation problem also exists in VP rules
  - Permitting the presence of strings containing verbs and arguments that do not go together

John sneezed the book
$VP \rightarrow V\ NP$

- Solution:
  - Similar to agreement phenomena, we need a way to formally express the constraints

For details refer following
Speech and Language Processing by Dan Jurafsky and James H. Martin

Modeling Constituency: what tool do we need?



Modeling Constituency

Context-free grammar

The most common way of modeling constituency

Consists of production Rules

These rules express the ways in which the symbols of the language can be grouped and ordered together

Example

Noun phrase can be composed of either a ProperNoun or a determiner (Det) followed by a Nominal; a Nominal can be more than one nouns

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

CFG for Languages

CFG: $G = (T, N, S, R)$

- T : set of terminals
- N: set of non-terminals
  - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R: Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)*$

Terminals and pre-terminals

Terminals mainly correspond to words in the language while pre-terminals mainly correspond to POS categories

CFG

- Terminals
  - The set of words in the text
- Non-Terminals
  - The constituents in a language
    (noun phrase, verb phrase, ...)
- Start symbol
  - The main constituent of the language
    (sentence)
- Rules
  - Equations that consist of a single non-terminal on the left and any number of terminals and non-terminals on the right

CFG for Languages

Example

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

CFG for Languages

Example

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

Now, these can be combined with other rules, that express facts about a lexicon.

CFG for Languages

Example

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

Now, these can be combined with other rules, that express facts about a lexicon.

$Det \rightarrow$ a

$Det \rightarrow$ the

$Noun \rightarrow$ flight

CFG for Languages

Example

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

Now, these can be combined with other rules, that express facts about a lexicon.

$Det \rightarrow$ a

$Det \rightarrow$ the

$Noun \rightarrow$ flight

Can you identify the terminal, non-terminals and preterminals?

CFG as a generator

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

$Det \rightarrow$ a

$Det \rightarrow$ the

$Noun \rightarrow$ flight

Generating ‘a flight’:

$NP \rightarrow$ Det Nominal

$\rightarrow$ Det Noun $\rightarrow$ a Noun $\rightarrow$ a flight

- Thus a CFG can be used to randomly generate a series of strings
- This sequence of rule expansions is called a derivation of the string of words, usually represented as a tree

CFGs and Grammaticality

A CFG defines a formal language = set of all sentences (string of words) that can be derived by the grammar

- Sentences in this set are said to be **grammatical**
- Sentences outside this set are said to be **ungrammatical**

CFGs and Recursion

Recursive Definition

- PP $\rightarrow$ Prep NP
- NP $\rightarrow$ Noun PP

Example Sentence

$[_S$The mailman ate his $[_{NP}$ lunch $[_{PP}$ with his friend $[_{PP}$ from the cleaning staff $[_{PP}$ of the building $[_{PP}$ at the intersection $[_{PP}$ on the north end $[_{PP}$ of town]]]]]]].

What does Context stand for in CFG?

- The notion of *context* has nothing to do with the ordinary meaning of word context in language
- All it really means is that the non-terminal on the left-hand side of a rule is out there all by itself (free of context)

$A \rightarrow BC$

- I can rewrite A as *B followed by C* regardless of the context in which A is found
- Or when I see a *B followed by a C*, I can infer an A regardless of the surrounding context

Noun Phrase → Proper Noun | Det Nominal

Nominal → noun | noun Nominal

$A \rightarrow B\,C$

$x\,A\,y \rightarrow x\,B\,C\,y$

Grammar Rewrite Rules

S → NP VP
S → Aux NP VP
S → VP
NP → Det NOM
NOM → Noun
NOM → Noun NOM
VP → Verb
VP → Verb NP

Det → *that* | *this* | *a* | *the*
Noun → *book* | *flight* | *meal* | *man*
Verb → *book* | *include* | *read*
Aux → *does*

S → NP VP
→ Det NOM VP
→ *The* NOM VP
→ *The* Noun VP
→ *The man* VP
→ *The man* Verb NP
→ *The man read* NP
→ *The man read* Det NOM
→ *The man read this* NOM
→ *The man read this* Noun
→ *The man read this book*

The man read this book

S → NP VP

→ Det Nom VP

→ The Nom VP

→ The Noun VP

→ The man VP

→ The man Verb NP

→ The man read NP

→ The man read Det Nom

this Noun → book

Parse Tree

$S \rightarrow \text{NP VP}$

$\rightarrow \text{Det NOM VP}$

$\rightarrow \textit{The} \text{ NOM VP}$

$\rightarrow \textit{The} \text{ Noun VP}$

$\rightarrow \textit{The man} \text{ VP}$

$\rightarrow \textit{The man} \text{ Verb NP}$

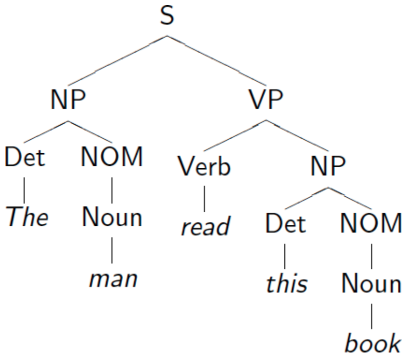
$\rightarrow \textit{The man read} \text{ NP}$

$\rightarrow \textit{The man read} \text{ Det NOM}$

$\rightarrow \textit{The man read this} \text{ NOM}$

$\rightarrow \textit{The man read this} \text{ Noun}$

$\rightarrow \textit{The man read this book}$



What is Parsing?

- The process of taking a string and a grammar and returning all possible parse trees for that string
- That is, find all trees, whose root is the start symbol S, which cover exactly the words in the input

What are the constraints? “book that flight”

- There must be three leaves, *book*, *that* and *flight*
- The tree must have one root, the start symbol S
- Give rise to two search strategies: *top-down* (goal-oriented) and *bottom-up* (data-directed)

Parsing

Grammar

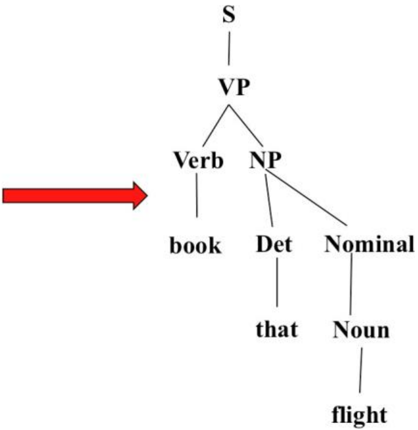
$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\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

Lexicon

Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | he | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

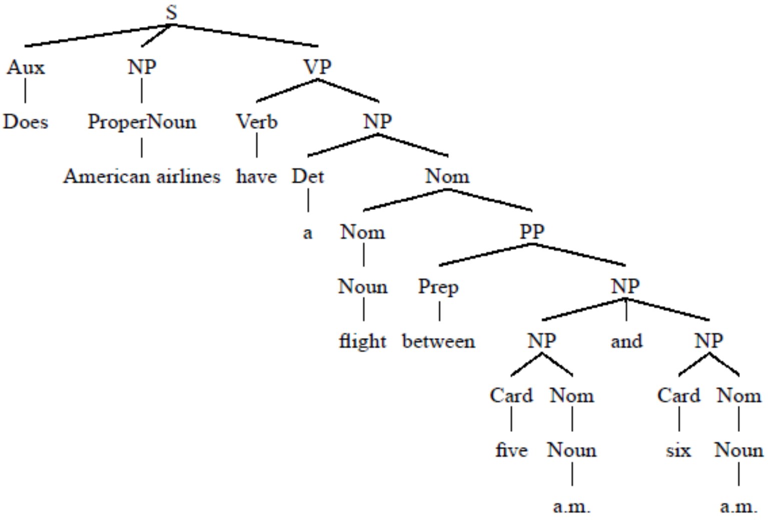
Parsing

book that flight



Draw the trees

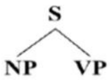
Does American airlines have a flight between five a.m. and six a.m.?



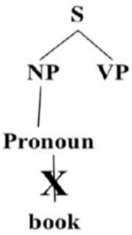
Top-Down Parsing

- Searches for a parse tree by trying to build upon the root node S down to the leaves
- Start by assuming that the input can be derived by the designated start symbol S
- Find all trees that can start with S, by looking at the grammar rules with S on the left-hand side
- Trees are grown downward until they eventually reach the POS categories at the bottom
- Trees whose leaves fail to match the words in the input can be rejected

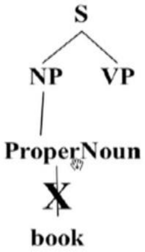
Top-Down Parsing



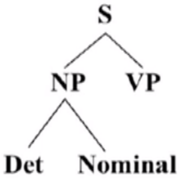
Top-Down Parsing



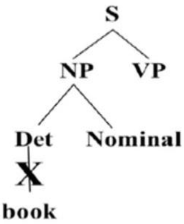
Top-Down Parsing



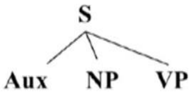
Top-Down Parsing



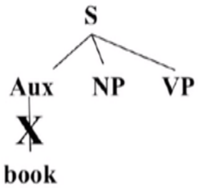
Top-Down Parsing



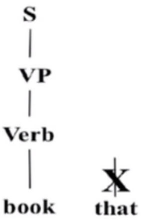
Top-Down Parsing



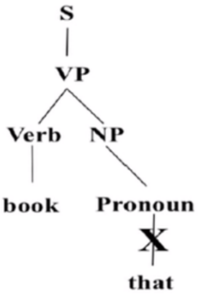
Top-Down Parsing



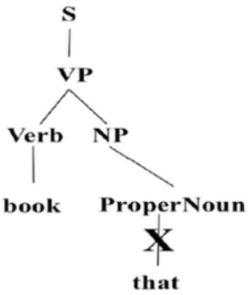
Top-Down Parsing



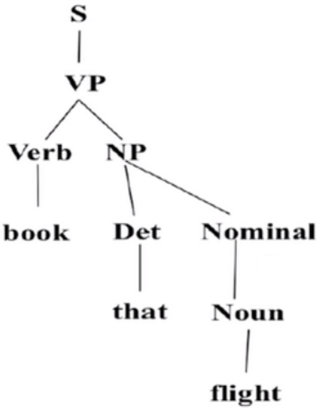
Top-Down Parsing



Top-Down Parsing



Top-Down Parsing



Bottom-Up Parsing

- The parser starts with the words of the input, and tries to build trees from the words up, by applying rules from the grammar one at a time
- Parser looks for the places in the parse-in-progress where the right-hand-side of some rule might fit.

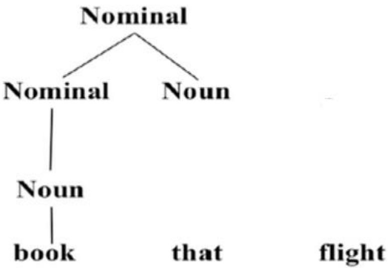
Bottom-up Parsing

- The parser starts from the words of the input and builds the tree up.
- It looks for the places in the current parse-tree where some right side of the grammar can fit.
- Trees that do not match any derivation are rejected.

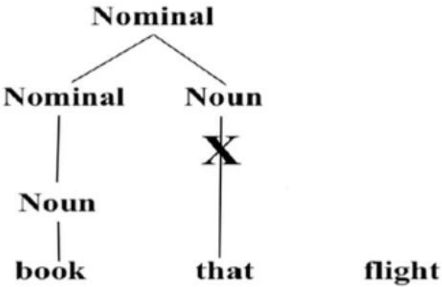
Bottom-Up Parsing

book that flight

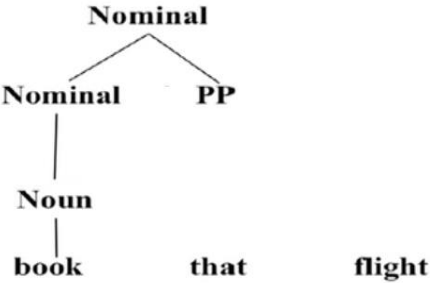
Bottom-Up Parsing



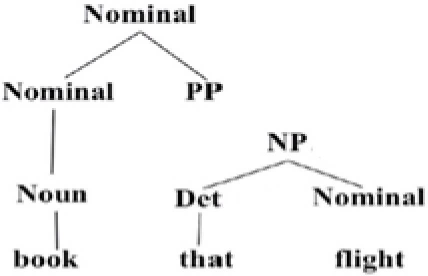
Bottom-Up Parsing



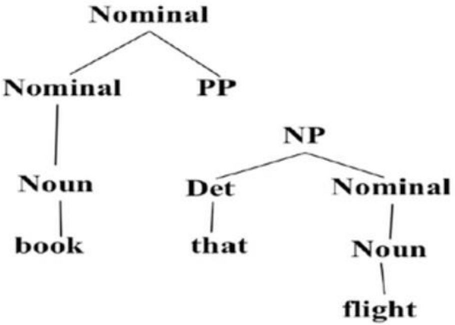
Bottom-Up Parsing



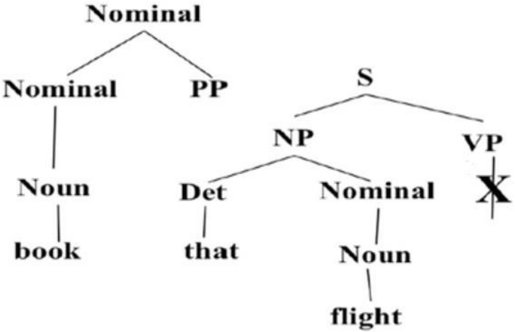
Bottom-Up Parsing



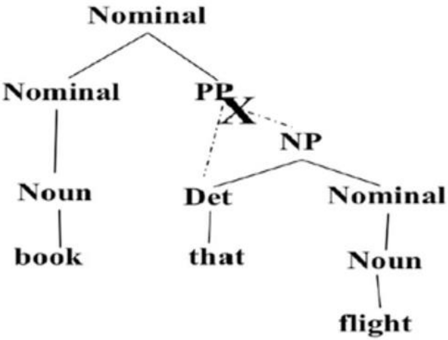
Bottom-Up Parsing



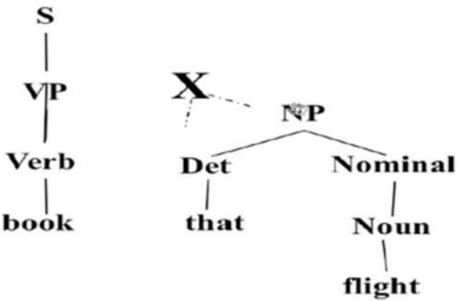
Bottom-Up Parsing



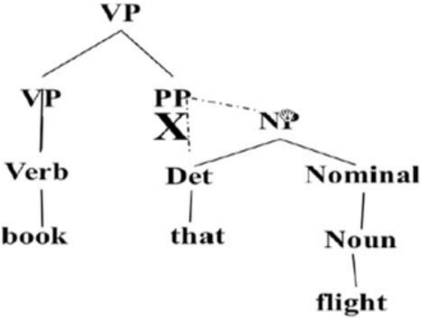
Bottom-Up Parsing



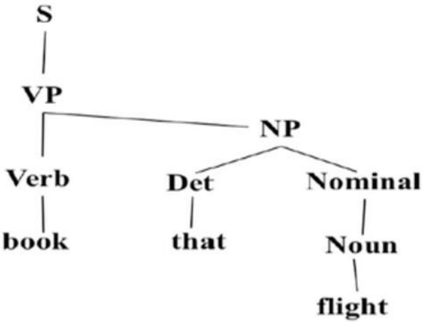
Bottom-Up Parsing



Bottom-Up Parsing



Bottom-Up Parsing



Difference between top-down and bottom-up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.
- Bottom up never explores options that do not connect to the actual sentence but can explore options that can never lead to a full parse.
- Relative amounts of wasted search depend on how much the grammar branches in each direction.

Top-Down vs. Bottom-Up

- In both cases, we left out how to keep track of the search space and how to make choices

- Solutions
  - Backtracking
    - Making a choice, if it works out then fine
    - If not, then back up and make a different choice
      $\Rightarrow$ duplicated work
  - Dynamic programming
    - Avoiding repeated work
    - Solving exponential problems in polynomial time
    - Storing ambiguous structures efficiently

Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.
- Caching (memoizing) critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs.
- Dynamic programming algorithms based on both top-down and bottom-up search can achieve $O(n^3)$ recognition time where n is the length of the input string.

Dynamic Programming Parsing Methods

Cache the intermediate results.

- CKY (Cocke-Kasami-Younger) algorithm: bottom-up, requires normalizing the grammar
- Earley Parser - top-down, does not require normalizing grammar, more complex
- More generally, *chart parsers* retain completed phrases in a chart and can combine top-down and bottom-up searches.

CKY Algorithm

- Grammar must be converted to Chomsky normal form (CNF) in which all productions must have
  - Either, exactly two non-terminals on the RHS
  - Or, 1 terminal symbol on the RHS
- Parse bottom-up storing phrases formed from all substrings in a triangular table (chart)

Chomsky Normal Form

Each grammar can be represented by a set of binary rules

$$A \rightarrow B\ C$$

$$A \rightarrow w$$

A, B, C are non-terminals w is a terminal

$A \to \gamma$ CFG prod^n

CNF $A \to BC$ non-terminals

$A \to a \to$ terminal

Chomsky Normal Form

Converting to Chomsky normal form

$$A \rightarrow B\ C\ D$$

$$X \rightarrow B\ C$$

$$A \rightarrow X\ D$$

X does not occur anywhere else in the the grammar

$$S \rightarrow \underbrace{\underline{Aux}\ \underline{NP}}_{X1}\ \underline{VP}$$

$S \rightarrow X1\ VP$ ✓ CNF

$X1 \rightarrow Aux\ NP$ ✓

Chomsky Normal Form

Converting to Chomsky normal form

$$A \rightarrow B$$

$$B \rightarrow C\ D$$

$$A \rightarrow C\ D$$

CKY Parsing

$$A \rightarrow B\ C$$

If there is an A somewhere in the input, then there must be a B followed by a C in the input

If the A spans from i to j in the input, then there must be a k such that $i < k < j$

- B spans from i to k
- C spans from k to j

Converting to CNF

Original Grammar

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 → VP PP
PP → Prep NP
Pronoun → I | he | she | me
Noun → book | flight | meal | money
Verb → book | include | prefer
Proper-Noun → Houston | NWA

Converting to CNF

Original Grammar

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 → VP PP
PP → Prep NP
Pronoun → I | he | she | me
Noun → book | flight | meal | money
Verb → book | include | prefer
Proper-Noun → Houston | NWA

Chomsky Normal Form

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → VP PP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → VP PP
PP → Prep NP
Pronoun → I | he | she | me
Noun → book | flight | meal | money
Verb → book | include | prefer
Proper-Noun → Houston | NWA

CKY Algorithm

- Let n be the number of words in the input. Think about $n+1$ lines separating them, numbered 0 to n.
- x_{ij} will denote the words between line i and j
- We build a table so that x_{ij} contains all the possible non-terminal spanning for words between line i and j.
- We build the Table bottom-up.

$| w_1 | w_2 | \cdots | w_n |$

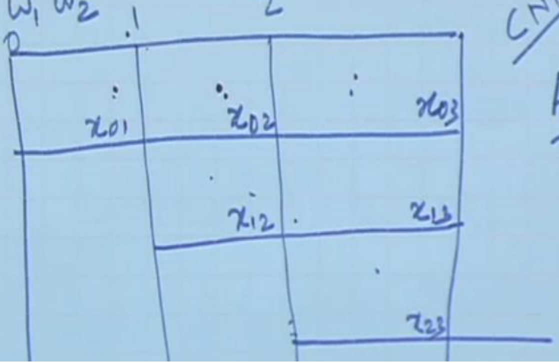
$0 \quad 1 \quad 2 \quad \cdots \quad n-1 \quad n$

CKY

~~CNF~~

~~$A \to a.$~~

$x_{02} = w_1 w_2$



CKY for CFG

a 1	pilot 2	likes 3	flying 4	planes 5

$S \longrightarrow NP\ VP$
$VP \longrightarrow VBG\ NNS$
$VP \longrightarrow VBZ\ VP$
$VP \longrightarrow VBZ\ NP$
$NP \longrightarrow DT\ NN$
$NP \longrightarrow JJ\ NNS$
$DT \longrightarrow a$
$NN \longrightarrow pilot$
$VBZ \longrightarrow likes$
$VBG \longrightarrow flying$
$JJ \longrightarrow flying$
$NNS \longrightarrow planes$

0 a 1 pilot 2 likes 3 flying 4 planes 5

x_{01}	x_{02}	x_{03}	x_{04}	x_{05}
	x_{12}	x_{13}	x_{14}	x_{15}
		x_{23}	x_{24}	x_{25}
			x_{34}	x_{35}
				x_{45}

0 a 1 pilot 2 likes 3 flying 4 planes 5

DT x_{01}	x_{02}	x_{03}	x_{04}	x_{05}
	NN x_{12}	x_{13}	x_{14}	x_{15}
		VBZ x_{23}	x_{24}	x_{25}
			VBG JJ x_{34}	x_{35}
				NNS x_{45}

0 a 1 pilot 2 likes 3 flying 4 planes 5

DT x_{01}	NP x_{02}	x_{03}	x_{04}	x_{05}
	NN x_{12}	— x_{13}	x_{14}	x_{15}
		VBZ x_{23}	— — x_{24}	x_{25}
			VBG JJ x_{34}	VP NP x_{35}
				NNS x_{45}

0 a 1 pilot 2 likes 3 flying 4 planes 5

DT x_{01}	NP x_{02}	— x_{03}	x_{04}	x_{05}
	NN x_{12}	— x_{13}	x_{14}	x_{15}
		VBZ x_{23}	— — x_{24}	x_{25}
			VBG JJ x_{34}	VP NP x_{35}
				NNS x_{45}

$x_{03} = x_{01}/x_{13} + x_{02}/x_{23}$

DT

$14 \to 12/24$, $13/34$

0 a 1 pilot 2 likes 3 flying 4 planes 5

DT x_{01}	NP x_{02}	— x_{03}	x_{04}	x_{05}
	NN x_{12}	— x_{13}	— x_{14}	x_{15}
		VBZ x_{23}	— — x_{24}	VP → VBZ VP x_{25}
			VBG JJ x_{34}	VP NP x_{35}
				NNS x_{45}

$x_{03} = x_{01}/x_{13} + x_{02}/x_{23}$

$14 \to 12/24$, $13/34$

$04 \to 01/14$, $02/24$, $03/34$

CKY for CFG

a 1	pilot 2	likes 3	flying 4	planes 5
DT	NP	-	-	S S
	NN	-	-	-
		VBZ	-	VP VP
			JJ VBG	NP VP
				NNS

$S \longrightarrow NP\ VP$
$VP \longrightarrow VBG\ NNS$
$VP \longrightarrow VBZ\ VP$
$VP \longrightarrow VBZ\ NP$
$NP \longrightarrow DT\ NN$
$NP \longrightarrow JJ\ NNS$
$DT \longrightarrow a$
$NN \longrightarrow pilot$
$VBZ \longrightarrow likes$
$VBG \longrightarrow flying$
$JJ \longrightarrow flying$
$NNS \longrightarrow planes$

Use CKY algorithm to find the parse tree for "Book the flight through Houston" using the CNF form shown in the previous slide.

CFG

$S \rightarrow NP \;\; VP$
$S \rightarrow VP$
$NP \rightarrow N$
$NP \rightarrow Det\ N$
$NP \rightarrow NP \;\; NP$
$NP \rightarrow NP \;\; PP$
$VP \rightarrow V$
$VP \rightarrow VP \;\; PP$
$VP \rightarrow VP \;\; NP$
$PP \rightarrow Prep \;\; NP$

$N \rightarrow$ book
$V \rightarrow$ book
$Det \rightarrow$ the
$N \rightarrow$ flight
$Prep \rightarrow$ through
$N \rightarrow$ Houston

CKY Parsing

[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	[1,2]	[1,3]	[1,4]	[1,5]
		[2,3]	[2,4]	[2,5]
			[3,4]	[3,5]
				[4,5]
Book	the	flight	through	Houston

0 1 2 3 4 5

CKY Parsing

N → book$_{[0,1]}$ **V → book**$_{[0,1]}$ [0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	[1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ [2,3]	[2,4]	[2,5]
			Prep → through$_{[3,4]}$ [3,4]	[3,5]
				N → houston$_{[4,5]}$ [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

CKY Parsing

N → book$_{[0,1]}$ **V → book**$_{[0,1]}$ **NP → N**$_{[0,1]}$ **VP → V**$_{[0,1]}$ **S → VP**$_{[0,1]}$ [0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	[1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ **NP → N**$_{[2,3]}$ [2,3]	[2,4]	[2,5]
			Prep → through$_{[3,4]}$ [3,4]	[3,5]
				N → houston$_{[4,5]}$ **NP → N**$_{[4,5]}$ [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

CKY Parsing

N → book$_{[0,1]}$ **V → book**$_{[0,1]}$ **NP → N**$_{[0,1]}$ **VP → V**$_{[0,1]}$ **S → VP**$_{[0,1]}$ [0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	**NP → Det**$_{[1,2]}$**, N**$_{[2,3]}$ [1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ **NP → N**$_{[2,3]}$ [2,3]	[2,4]	[2,5]
			Prep → through$_{[3,4]}$ [3,4]	[3,5]
				N → houston$_{[4,5]}$ **NP → N**$_{[4,5]}$ [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

CKY Parsing

N → book$_{[0,1]}$ **V → book**$_{[0,1]}$ **NP → N**$_{[0,1]}$ **VP → V**$_{[0,1]}$ **S → VP**$_{[0,1]}$ [0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	**NP → Det**$_{[1,2]}$**, N**$_{[2,3]}$ [1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ **NP → N**$_{[2,3]}$ [2,3]	[2,4]	[2,5]
			Prep → through$_{[3,4]}$ [3,4]	**PP→Prep**$_{[3,4]}$**,NP**$_{[4,5]}$ [3,5]
				N → houston$_{[4,5]}$ **NP → N**$_{[4,5]}$ [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

CKY Parsing

N → book$_{[0,1]}$ V → book$_{[0,1]}$ NP → N$_{[0,1]}$ VP → V$_{[0,1]}$ S → VP$_{[0,1]}$ [0,1]	[0,2]	NP → NP$_{[0,1]}$, NP$_{[1,3]}$ VP → VP$_{[0,1]}$, NP$_{[1,3]}$ S → VP$_{[0,3]}$ [0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	NP → Det$_{[1,2]}$, N$_{[2,3]}$ [1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ NP → N$_{[2,3]}$ [2,3]	[2,4]	[2,5]
			Prep → through$_{[3,4]}$ [3,4]	PP→Prep$_{[3,4]}$,NP$_{[4,5]}$ [3,5]
				N → houston$_{[4,5]}$ NP → N$_{[4,5]}$ [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

CKY Parsing

N → book$_{[0,1]}$ V → book$_{[0,1]}$ NP → N$_{[0,1]}$ VP → V$_{[0,1]}$ S → VP$_{[0,1]}$ [0,1]	[0,2]	NP → NP$_{[0,1]}$, NP$_{[1,3]}$ VP → VP$_{[0,1]}$, NP$_{[1,3]}$ S → VP$_{[0,3]}$ [0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	NP → Det$_{[1,2]}$, N$_{[2,3]}$ [1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ NP → N$_{[2,3]}$ [2,3]	[2,4]	NP → NP$_{[2,3]}$, PP$_{[3,5]}$ [2,5]
			Prep → through$_{[3,4]}$ [3,4]	PP→Prep$_{[3,4]}$,NP$_{[4,5]}$ [3,5]
				N → houston$_{[4,5]}$ NP → N$_{[4,5]}$ [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

CKY Parsing

$N \rightarrow book_{[0,1]}$ $V \rightarrow book_{[0,1]}$ $NP \rightarrow N_{[0,1]}$ $VP \rightarrow V_{[0,1]}$ $S \rightarrow VP_{[0,1]}$ [0,1]	[0,2]	$NP \rightarrow NP_{[0,1]}, NP_{[1,3]}$ $VP \rightarrow VP_{[0,1]}, NP_{[1,3]}$ $S \rightarrow VP_{[0,3]}$ [0,3]	[0,4]	[0,5]
	$Det \rightarrow the_{[1,2]}$ [1,2]	$NP \rightarrow Det_{[1,2]}, N_{[2,3]}$ [1,3]	[1,4]	$NP \rightarrow NP_{[1,3]}, PP_{[3,5]}$ [1,5]
		$N \rightarrow flight_{[2,3]}$ $NP \rightarrow N_{[2,3]}$ [2,3]	[2,4]	$NP \rightarrow NP_{[2,3]}, PP_{[3,5]}$ [2,5]
			$Prep \rightarrow through_{[3,4]}$ [3,4]	$PP \rightarrow Prep_{[3,4]}, NP_{[4,5]}$ [3,5]
				$N \rightarrow houston_{[4,5]}$ $NP \rightarrow N_{[4,5]}$ [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

CKY Parsing

$N \rightarrow book_{[0,1]}$ $V \rightarrow book_{[0,1]}$ $NP \rightarrow N_{[0,1]}$ $VP \rightarrow V_{[0,1]}$ $S \rightarrow VP_{[0,1]}$ [0,1]	[0,2]	$NP \rightarrow NP_{[0,1]}, NP_{[1,3]}$ $VP \rightarrow VP_{[0,1]}, NP_{[1,3]}$ $S \rightarrow VP_{[0,3]}$ [0,3]	[0,4]	$VP \rightarrow VP_{[0,1]}, NP_{[1,5]}$ $VP' \rightarrow VP_{[0,3]}, PP_{[3,5]}$ $S \rightarrow VP_{[0,5]}$ $S \rightarrow VP'_{[0,5]}$ [0,5]
	$Det \rightarrow the_{[1,2]}$ [1,2]	$NP \rightarrow Det_{[1,2]}, N_{[2,3]}$ [1,3]	[1,4]	$NP \rightarrow NP_{[1,3]}, PP_{[3,5]}$ [1,5]
		$N \rightarrow flight_{[2,3]}$ $NP \rightarrow N_{[2,3]}$ [2,3]	[2,4]	$NP \rightarrow NP_{[2,3]}, PP_{[3,5]}$ [2,5]
			$Prep \rightarrow through_{[3,4]}$ [3,4]	$PP \rightarrow Prep_{[3,4]}, NP_{[4,5]}$ [3,5]
				$N \rightarrow houston_{[4,5]}$ $NP \rightarrow N_{[4,5]}$ [4,5]

ambiguity

Book the flight through Houston

0 1 2 3 4 5

Probabilistic Context-free grammars (PCFGs)

Definition

Also known as a weighted grammar, a probabilistic context-free grammar (PCFG) is one that assigns a probability to each production rule.

It can be used to assign a probability to every string in the language (language model) and to every structure in the language.

Probabilistic Context-free grammars (PCFGs)

PCFG: $G = (T, N, S, R, P)$

- T : set of terminals
- N: set of non-terminals
  - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R: Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)*$
- $P(R)$ gives the probability of each rule.

$$\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$$

Computing rule probabilities

Computing rule probabilities

$$P(A \to \beta) = \frac{Count(A \to \beta)}{\sum_\lambda Count(A \to \lambda)} = \frac{Count(A \to \beta)}{Count(A)} \quad (1)$$

Computing rule probabilities

$$P(A \rightarrow \beta) = \frac{Count(A \rightarrow \beta)}{\sum_{\lambda} Count(A \rightarrow \lambda)} = \frac{Count(A \rightarrow \beta)}{Count(A)} \tag{1}$$

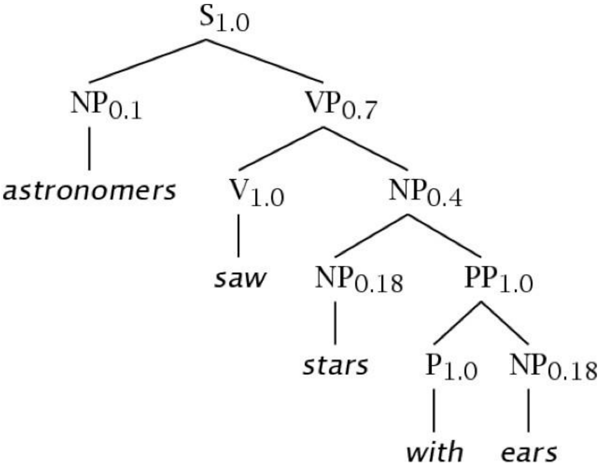
$$\forall X \in N, \sum_{\beta} P(X \rightarrow \beta) = 1$$

A Simple PCFG (in CNF)

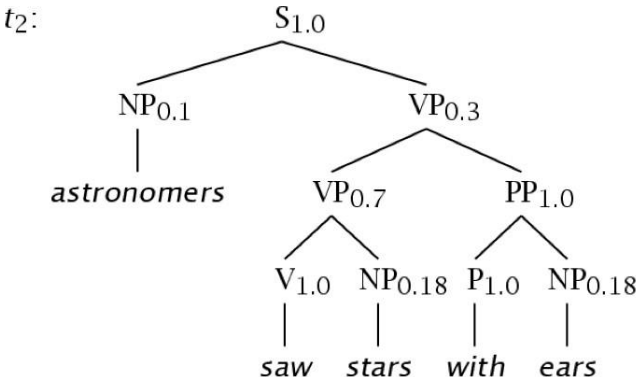
S	→	NP VP	1.0	NP	→	NP PP	0.4
VP	→	V NP	0.7	NP	→	*astronomers*	0.1
VP	→	VP PP	0.3	NP	→	*ears*	0.18
PP	→	P NP	1.0	NP	→	*saw*	0.04
P	→	*with*	1.0	NP	→	*stars*	0.18
V	→	*saw*	1.0	NP	→	*telescope*	0.1

Example Trees

t_1:



Example Trees



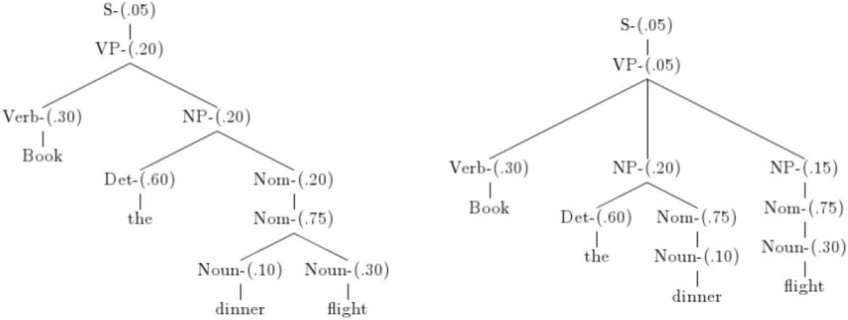
Probability of trees and strings

- $P(t)$: The probability of tree is the product of the probabilities of the rules used to generate it
- $P(w_{1n})$: The probability of the string is the sum of the probabilities of the trees which have that string as their yield

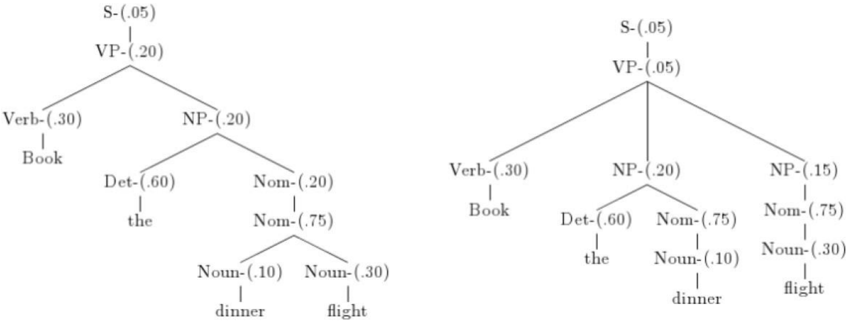
Tree and String probabilities

$$
\begin{aligned}
w_{15} &= \textit{astronomers saw stars with ears} \\
P(t_1) &= 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 \\
&\quad * 1.0 * 1.0 * 0.18 \\
&= 0.0009072 \\
P(t_2) &= 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 \\
&\quad * 1.0 * 1.0 * 0.18 \\
&= 0.0006804 \\
P(w_{15}) &= P(t_1) + P(t_2) \\
&= 0.0009072 + 0.0006804 \\
&= 0.0015876
\end{aligned}
$$

“Book the dinner flight”



“Book the dinner flight”



Probabilities

- Parse tree 1: $.05 \times .20 \times .30 \times .20 \times .60 \times .20 \times .75 \times .10 \times .30 = 1.62 \times 10^{-6}$
- Parse tree 2: $.05 \times .05 \times .30 \times .20 \times .60 \times .75 \times .10 \times .15 \times .75 \times .30 = 2.28 \times 10^{-7}$

Features of PCFGs

- As the number of possible trees for a given input grows, a PCFG gives some idea of the plausibility of a particular parse
- *But* the probability estimates are based purely on structural factors, and do not factor in lexical co-occurrence. Thus, PCFG does not give a very good idea of the plausibility.
- Real text tends to have grammatical mistakes. PCFG avoids this problem by ruling out nothing, but by giving implausible sentences a low probability
- A PCFG is a **worse language model** for English than an n-gram model
- The probability of a smaller tree is greater than a larger tree.

How to find the most likely parse?: CKY for PCFG

How to find the most likely parse?: CKY for PCFG

a 1	pilot 2	likes 3	flying 4	planes 5

$S \longrightarrow NP\ VP$	[1.0]
$VP \longrightarrow VBG\ NNS$	[0.1]
$VP \longrightarrow VBZ\ VP$	[0.1]
$VP \longrightarrow VBZ\ NP$	[0.3]
$NP \longrightarrow DT\ NN$	[0.3]
$NP \longrightarrow JJ\ NNS$	[0.4]
$DT \longrightarrow a$	[0.3]
$NN \longrightarrow pilot$	[0.1]
$VBZ \longrightarrow likes$	[0.4]
$VBG \longrightarrow flying$	[0.5]
$JJ \longrightarrow flying$	[0.1]
$NNS \longrightarrow planes$	[.34]

CKY for PCFG

a 1	pilot 2	likes 3	flying 4	planes 5
DT [0.3]	NP [.009]	-	-	S [$1.4688x10^{-5}$] S [$6.12x10^{-6}$]
	NN [0.1]	-	-	-
		VBZ [0.4]	-	VP [.001632] VP [.00068]
			JJ [0.1] VBG [0.5]	NP [.0136] VP [.017]
				NNS [.34]

$S \longrightarrow NP\ VP$	[1.0]
$VP \longrightarrow VBG\ NNS$	[0.1]
$VP \longrightarrow VBZ\ VP$	[0.1]
$VP \longrightarrow VBZ\ NP$	[0.3]
$NP \longrightarrow DT\ NN$	[0.3]
$NP \longrightarrow JJ\ NNS$	[0.4]
$DT \longrightarrow a$	[0.3]
$NN \longrightarrow pilot$	[0.1]
$VBZ \longrightarrow likes$	[0.4]
$VBG \longrightarrow flying$	[0.5]
$JJ \longrightarrow flying$	[0.1]
$NNS \longrightarrow planes$	[.34]

**= P(NP->DT NN)*
P(DT-> a)*
P(NN->pilot)
= 0.3*0.3*0.1 = 0.009**

$0.009 \times 0.00068 \times 1.0 = 6.12x10^{-6}$

Important Questions?

Let W_{1m} be a sentence, G a grammar, t a parse tree

- What is the most likely parse of sentence?

$$argmax_t P(t|w_{1m}, G)$$

- What is the probability of a sentence?

$$P(w_{1m}|G)$$

- How to learn the rule probabilities in the grammar G?

Probability of a String

$$P(w_{1m}|G)$$

- In general, simply summing the probabilities of all possible parse trees is not an efficient way to calculate the string probability
- We use *inside algorithm*, a dynamic programming algorithm based on inside probabilities.

Q1: You are given the grammar below. How many parse trees can you derive for the sentence:

Radha drove to Agra and Delhi in November.

Draw each parse tree. The rules of the CFG grammar where S is the start symbol are:

S → NP V P,

V P → V NP | V P P | V P P P,

NP → NP P P | NP CNJ NP,

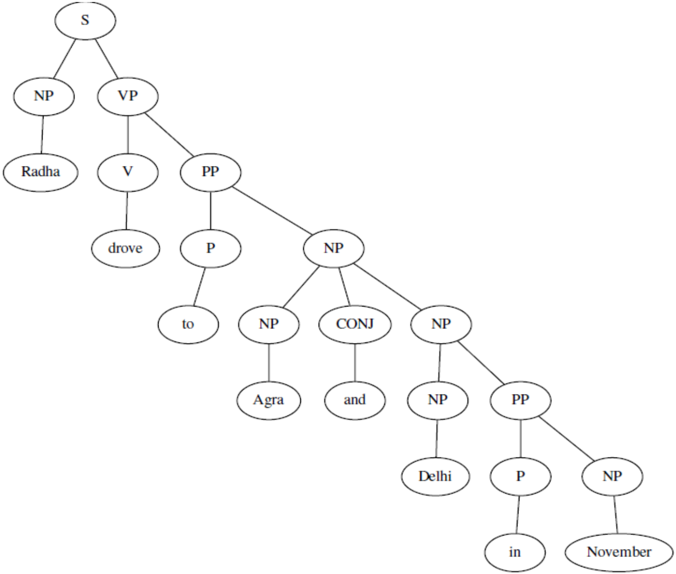
P P → P NP,

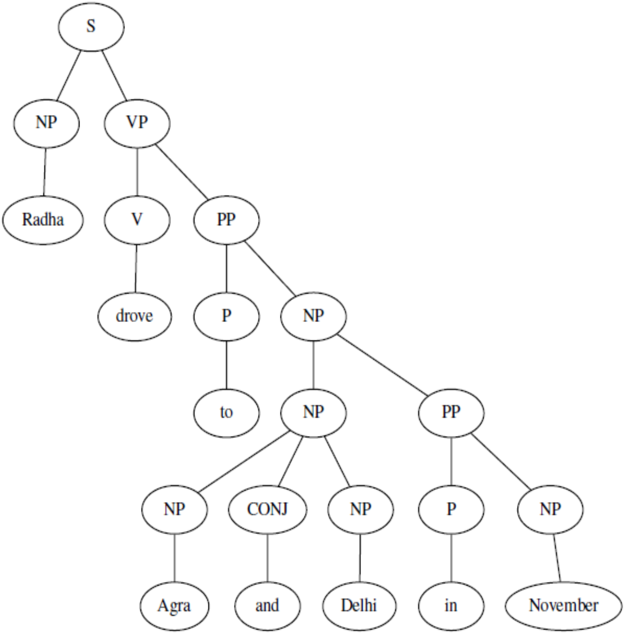
NP → Radha | Agra | Delhi | November,

V → drove, P → to | in,

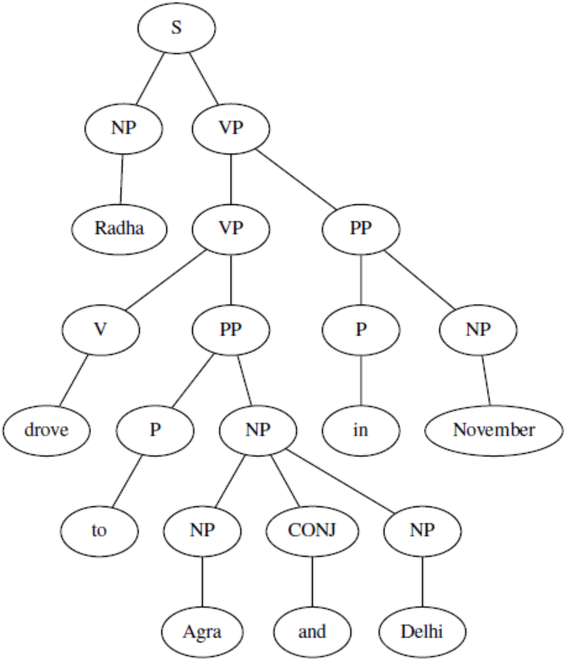
CNJ → and

Solution: **There are three parses:**





S
NP
VP
Radha
V
PP
drove
P
NP
to
NP
PP
NP
CONJ
NP
P
NP
Agra
and
Delhi
in
November


S
NP
VP
Radha
VP
PP
V
PP
P
NP
drove
P
NP
in
November
to
NP
CONJ
NP
Agra
and
Delhi

Q2: Given the following PCFG G (where S is the start symbol)

Rule	**Probability**
$S \rightarrow VP$	1.0
$VP \rightarrow V\ NP$	0.7
$VP \rightarrow V\ NP\ PP$	0.3
$NP \rightarrow NP\ PP$	0.3
$NP \rightarrow DET\ N$	0.7
$PP \rightarrow P\ N$	1.0
$DET \rightarrow \text{the}$	0.1
$V \rightarrow \text{cut} \mid \text{ask} \mid \text{find}$	0.1
$P \rightarrow \text{with} \mid \text{in}$	0.1
$N \rightarrow \text{envelope} \mid \text{grandma} \mid \text{scissors} \mid \text{suits}$	0.1

Answer the following questions.
a) Is the PCGF G a proper PCFG? Why?

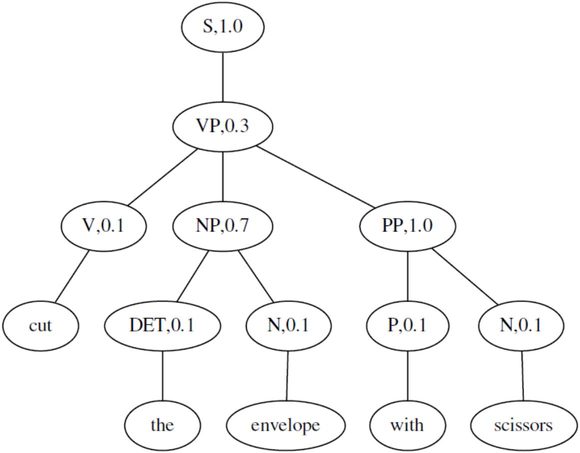
Solution:
No - not a proper PCFG. The total probability for several non-terminals does not add up to 1.0 - for example N; DET; V.

(b) For the sentence:

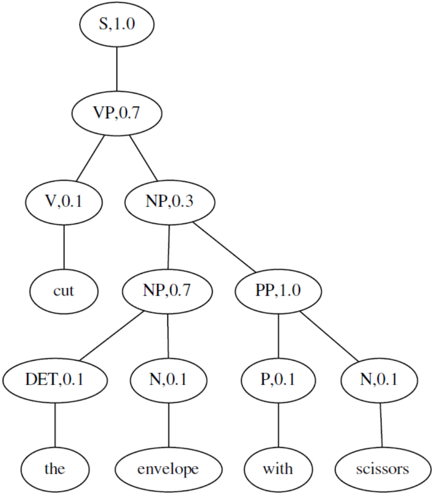
Cut the envelope with scissors.

Find the parse trees and their probabilities. Does the result look right? Justify.

Solution:



$$\text{Probability} = 0.3 \times 0.7 \times 0.1^5 = 21 \times 10^{-7}$$



$$\text{Probability} = 0.3 \times 0.7 \times 0.7 \times 0.1^5 = 14.7 \times 10^{-7}$$

The first tree has higher probability and it is the correct parse since `with scissors' as the instrument for cutting should attach to `cut' rather than `envelope'.