Basic Parsing with Context Free Grammars
Chapter 13

Lecture 6

September/October 2012

Analyzing Linguistic Units

- Morphological parsing:
  - analyze words into morphemes and affixes
  - rule-based, FSAs, FSTs
- Phonological parsing:
  - analyze sounds into words and phrases
- POS Tagging
- Syntactic parsing:
  - identify component parts and how related
  - to see if a sentence is grammatical
  - to assign an abstract representation of meaning

Syntactic Parsing

- Declarative formalisms like CFGs define the legal strings of a language but don't specify how to recognize or assign structure to them
- Parsing algorithms specify how to recognize the strings of a language and assign each string one or more syntactic structures
- Parse trees useful for grammar checking, semantic analysis, MT, QA, information extraction, speech recognition…and almost every task in NLP

Parsing is a Form of Search

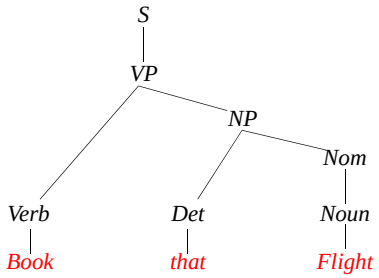
- Searching FSAs
  - Finding the right path through the automaton
  - Search space defined by structure of FSA
- Searching CFGs
  - Finding the right parse tree among all possible parse trees
  - Search space defined by the grammar
- Constraints provided by the input sentence and the automaton or grammar

CFG for Fragment of English

S → NP VP	VP → V
S → Aux NP VP	Det → that | this | a
S → VP	N → book | flight | meal | money
NP → Det Nom	V → book | include | prefer
NP →PropN	Aux → does
Nom → N Nom	Prep →from | to | on
Nom → N	PropN → Houston | TWA
Nom → Nom PP	
VP → V NP	

TopD BotUp E.g. LC's

Parse Tree for "Book that flight" for Prior CFG

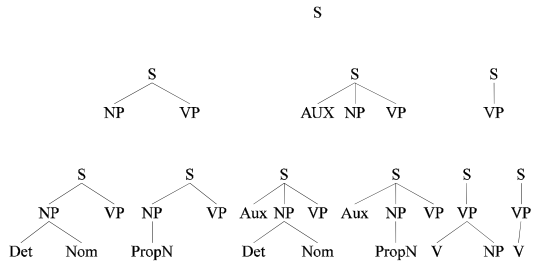


Top-Down Parser

- Builds from the root S node to the leaves
- Find a rule to apply by matching the left hand side of a rule
- Build a tree by replacing LHS with the right hand side
- Assuming we build all trees in parallel:
  - Find all trees with root S (or all rules w/lhs S)
  - Next expand all constituents in these trees/rules
  - Continue until leaves are pos
  - Candidate trees failing to match pos of input string are rejected (e.g. Book that flight can only match subtree 5)

7

Top Down Space



8

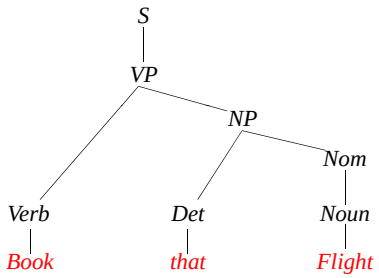
CFG for Fragment of English

S → NP VP	VP → V
S → Aux NP VP	Det → that (5) | this | a
S → VP (1)	N → book | flight (7) | meal | money
NP → Det Nom (4)	V → book (3) | include | prefer
NP →PropN	Aux → does
Nom → N Nom	Prep →from | to | on
Nom → N (6)	PropN → Houston | TWA
Nom → Nom PP	
VP → V NP (2)	

TopD BotUp E.g. LC's

9

Parse Tree for "Book that flight" for Prior CFG



10

Bottom-Up Parsing

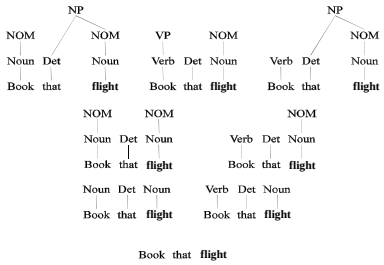
- Parser begins with words of input and builds up trees, applying grammar rules whose right hand side match
  - Book that flight

N	Det	N	V	Det	N
Book	that	flight	Book	that	flight

  - 'Book' ambiguous
  - Parse continues until an S root node reached or no further node expansion possible

11

Bottom-Up Space



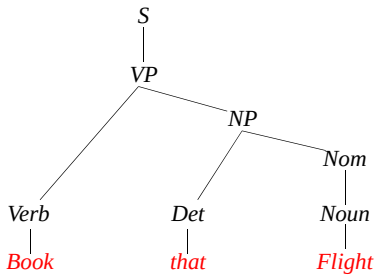
12

CFG for Fragment of English

S → NP VP	VP → V
S → Aux NP VP	Det → that (2) | this | a
S → VP (7)	N → book | flight (3) | meal | money
NP → Det Nom (5)	V → book (1) | include | prefer
NP →PropN	Aux → does
Nom → N Nom	Prep →from | to | on
Nom → N (4)	PropN → Houston | TWA
Nom → Nom PP	
VP → V NP (6)	

TopD BotUp E.g. LC's

Parse Tree for "Book that flight" for Prior CFG



Control

- Of course, we left out how to keep track of the spaces and how to make choices
  - Which node to try to expand next
  - Which grammar rule to use to expand a node

A Top-Down Parsing Strategy

- **Depth-first search**:
  - Agenda of search states: expand search space incrementally, exploring most recently generated state (tree) each time
  - When you reach a state (tree) inconsistent with input, backtrack to most recent unexplored state (tree)
- Which node to expand?
  - **Leftmost** or rightmost
- Which grammar rule to use?
  - Order in the grammar??

Top-Down, Depth-First, Left-Right Strategy

- Initialize agenda with 'S' tree and ptr to first word and make this current search state (cur)
- Loop until successful parse or empty agenda
  - Apply all applicable grammar rules to leftmost unexpanded node of cur
    - If this node is a POS category and matches that of the current input, push this onto agenda
    - O.w. push new trees onto agenda
  - Pop new cur from agenda
- Does this flight include a meal?

Top-Down, Depth-First, Left-to-Right Search

Curr: S

[Does]

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP →PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

Top-Down, Depth-First, Left-to-Right Search

Curr: S

[Does]

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP →PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

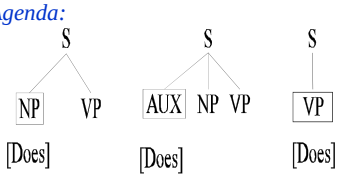
19

Top-Down, Depth-First, Left-to-Right Search

Curr: S

[Does]

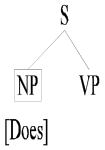
Agenda:



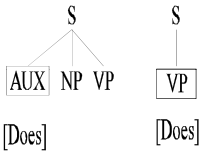
20

Top-Down, Depth-First, Left-to-Right Search

Curr:



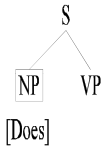
Agenda:



21

Top-Down, Depth-First, Left-to-Right Search

Curr:



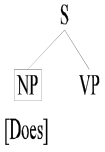
Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP →PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

22

Top-Down, Depth-First, Left-to-Right Search

Curr:



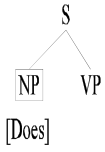
Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP →PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

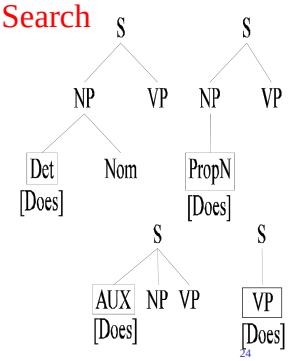
23

Top-Down, Depth-First, Left-to-Right Search

Curr:



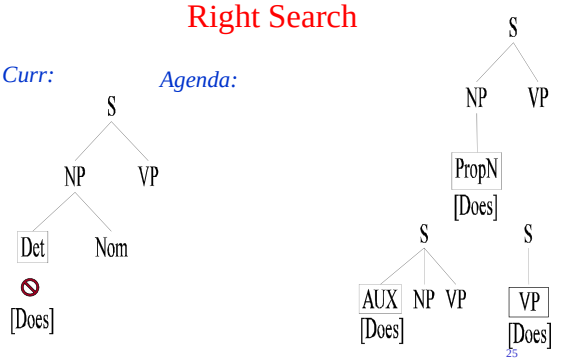
Agenda:



24

Top-Down, Depth-First, Left-to-Right Search

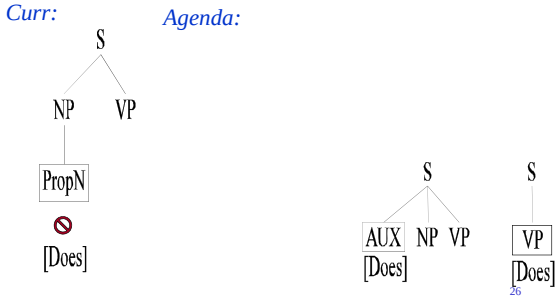
Curr: *Agenda:*



25

Top-Down, Depth-First, Left-to-Right Search

Curr: *Agenda:*



26

Top-Down, Depth-First, Left-to-Right Search

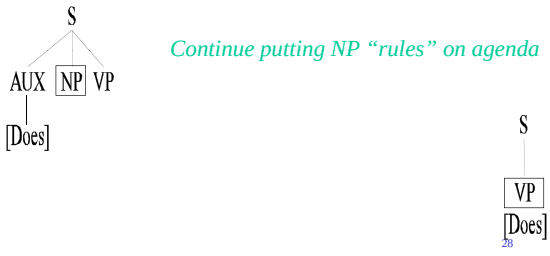
Curr: *Agenda:*

27

Top-Down, Depth-First, Left-to-Right Search

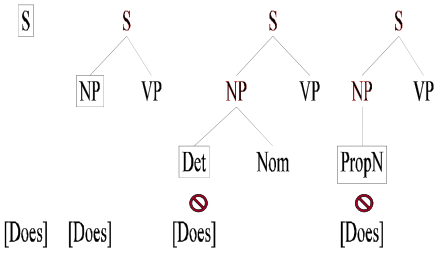
Curr: *Agenda:*

Continue putting NP "rules" on agenda



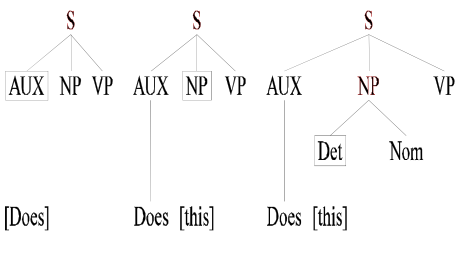
28

"Does this flight include a meal?" Parsing Overview



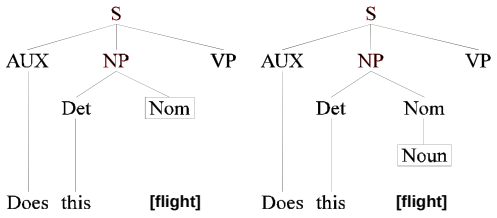
29

"Does this flight include a meal?" Parsing Overview (cont.)



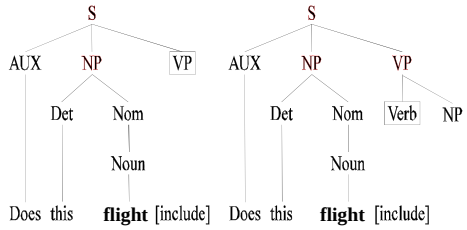
30

"Does this flight include a meal?" Parsing Overview (cont.)



31

"Does this flight include a meal?" Parsing Overview (cont.)



32

A Bottom-Up Parsing Strategy

- **Depth-first search**:
  - State of parse is going to be initialized to the input words
  - At each step, look for Right Hand Side of a rule in the state, replace the matched right hand side with the Left Hand Side of the rule and continue
  - Agenda of search states: expand search space incrementally, exploring most recently generated state each time
  - When you reach a state that contains only the start symbol, you have successfully parsed

33

Bottom Up: "Book that flight"

- Curr: N det N
  Agenda: V det N
- Curr: Nom det N
  Agenda: N det Nom, V det N
- Curr: Nom det Nom
  Agenda: N det Nom, V det N
- Curr: Nom NP
  Agenda: N det Nom, V det N
- Curr: N det Nom
  Agenda: V det N

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP →PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

34

Bottom Up: "Book that flight"

- Curr: V det N
  Agenda:
- Curr: VP det N
  Agenda: V det Nom
- Curr: VP NP
  Agenda: V det Nom
- Curr: S NP
  Agenda: V det Nom

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP →PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

35

Bottom Up: "Book that flight"

- Curr: V det Nom
  Agenda:
- Curr: V NP
  Agenda:
- Curr: VP
  Agenda:
- Curr: S
  Agenda:
- SUCCESS!!!!

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP →PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

36

What's wrong with….

- Top-Down parsers never explore illegal parses (e.g. can't form an S) -- but waste time on trees that can never match the input
- Bottom-Up parsers never explore trees inconsistent with input -- but waste time exploring illegal parses (no S root)
- For both: control strategy -- how explore search space?
  - Pursuing all parses in parallel or backtrack or …?
  - Which rule to apply next?
  - Which node to expand next?

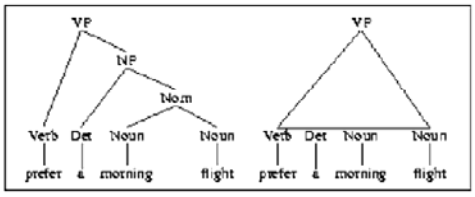
37

Left Corners: Top-Down Parsing with Bottom-Up Filtering

- We saw: Top-Down, depth-first, L2R parsing
  - Expands non-terminals along the tree's left edge down to leftmost leaf of tree
  - Moves on to expand down to next leftmost leaf…
  - Note: In successful parse, current input word will be first word in derivation of node the parser currently processing
  - So….look ahead to left-corner of the tree
    - B is a left-corner of A if A =*=> Bα
    - Build table with left-corners of all non-terminals in grammar and consult before applying rule

38

Left Corners



39

Calculating Left Corners

For each constituent on the LHS of a rule, follow through LHS until you find a preterminal (lexical category). That's the left corner.

Consider S – one rule at a time
Det
PropN
Aux
V

Same procedure for other constituents

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP →PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

40

Left-Corner Table for CFG

Category	Left Corners
S	Det, PropN, Aux, V
NP	Det, PropN
Nom	N
VP	V

41

Summing Up

- Parsing is a search problem which may be implemented with many control strategies
  - Top-Down or Bottom-Up approaches each have problems
  - Combining the two solves some but not all issues…
  - Look at some more solutions next week (Thursday)
- Next time: Read Ch 10:3-6

42