CS395-T
Topics in Natural Language Processing

LECTURE 3

Sanda Harabagiu
Department of Computer Sciences
University of Texas at Austin
Tu, Th 11AM-12:20 PM
sanda@cs.utexas.edu
http://www.cs.utexas.edu/users/sanda/cs395T.html
http://www.engr.smu.edu/~sanda/cs395T.html

What is a grammar?

- A vogue in the 19th Century to describe the structures of an area of knowledge

(e.g. Busby's "A Grammar of Music"
Field's "A Grammar of Colouring")

- The meaning of grammar has changed:
  - It used to be "a listing of principles or structures"
  - Now: "principles or structures as a field of inquiry"

Syntax

- Greek *sýntaxis* : "setting out together or arrangement"
- Covered: part-of-speech categories (equivalence classes for words)
- New ideas:
  - constituency
  - grammatical relations
  - subcategorization and dependencies

3

What is a constituent?

- The idea: groups of words may behave as a single unit or phrase → a constituent
- Example: noun phrases act like a unit
  - "she"
  - "Michael"
  - "the house"
  - "Russian Hill"
  - "well-weathered three-story structure"
- How can we model the constituency?

  → with context-free grammars

4

Context-Free Grammars

- Chomsky (1956) Backus (1959)
- set of rules (productions) – expressing the way symbols of the language can be grouped and ordered together

+

- lexicon
- Example:

NP → Det Nominal
NP → Proper Noun
Nominal → Noun | Noun Nominal

- Embed rules about the lexicon:

Det → a Det → the Noun → flight

5

Lexicon Example

Noun → *flights* | *breeze* | *trip* | *morning* | …
Verb → *is* | *prefer* | *like* | *need* | *want* | *fly* | …
Adjective → *cheapest* | *non-stop* | *first* | *latest* | *other* | *direct* | …
Pronoun → *me* | *I* | *you* | *it* | …
Proper-Noun → *Alaska* | *Baltimore* | *Los Angeles* | *Chicago* | *United* | *American* | …
Determiner → *the* | *a* | *an* | *this* | *these* | *that* |…
Preposition → *from* | *to* | *on* | *near*
Conjunction → *and* | *or* | *but* | …

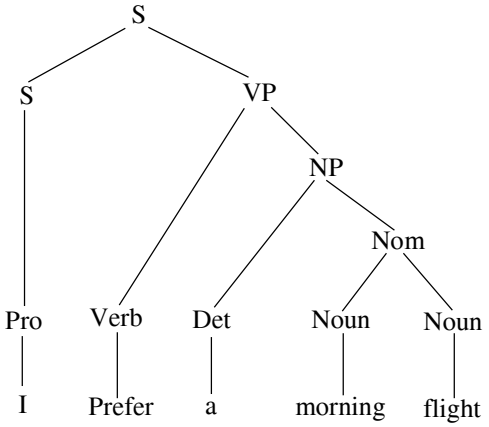
6

A Grammar

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

7

A Parse



8

Formal Definition of CFG

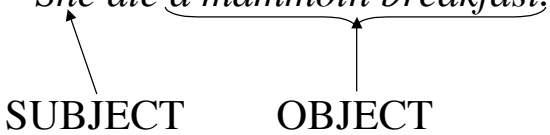
1. Set of non-terminal symbols N
2. Set of terminals Σ
3. Set Productions $A \rightarrow \alpha$

   $A \in N$, α-string $(\Sigma \cup \mathbf{N})^*$
4. A designated start symbol

9

Grammatical Relations

- Formalizations of ideas from traditional grammar about SUBJECTS and OBJECTS.
- Sentence:



SUBJECT OBJECT

10

Recursion in Grammars

- When an expansion of a non-terminal includes the non-terminal itself

  Nominal → Nominal PP

- Problem for Finite-State Grammars !
  - Why ? When building the finite-state model for some of the rules:
  - Example: generate a regular expression for an NP:

    (Det)(Card)(Ord)(Quant)(AP)Nominal

    Is this all ? No ! Post-modifiers are possible !

11

Finite-state Grammars

- The regular expression becomes

  *(Det)(Card)(Ord)(Quant)(AP)Nominal(PP)**
- We need to expand inline the definition of PP

  → *(Det)(Card)(Ord)(Quant)(AP)Nominal (P NP)**

  We use NP in the definition of an NP !
- This means we can expand the rule into:

  *(Det)(Card)(Ord)(Quant)(AP)Nominal(P(Det)(Card)(Ord)(Quant)(AP)Nominal(P NP)**

Problem: NP has a recursive rule !

12

Solution ???

- Sneaky solution
- Recursive Transition Network

13

Verb Phrases and SubCategorization

The Verb Phrase consists of a verb and a numbr of other constituents (NP_S, PP_S)

For example:

VP → Verb (disappear)

VP → Verb NP (prefer a morning flight)

VP → Verb NP PP (leave Boston in the morning)

VP → Verb PP (leaving on Thursday)

14

Sentential Complements

- Examples:
  - You[$_{VP}$ [$_{V}$ said [$_{S}$ there were two flights that were the cheapest]]]
  - You [$_{VP}$ [$_{V}$ said [$_{S}$ you had a two hundred sixty six dollar fare]]]
  - [$_{VP}$ [$_{V}$ Tell] [$_{NP}$ me] [$_{S}$ how to get from the airport in Philadelphia to downtown]]
  - I [$_{VP}$ [$_{V}$ think [$_{S}$ I would like to take the nine thirty flight]]
- Rules
  - VP → Verb S

15

Other Constituents

- A VP can be the constituent of another VP.
  - I want [$_{VP}$ to fly from Milwakee to Orlando]
  - Hi, I want [$_{VP}$ to arrange three flights]
  - Hello, I'm trying [$_{VP}$ to find a flight that goes from Pittsburgh to Denver after two p.m.]

16

Other Constituents

- Important observation: *While a verb phrase can have many possible kinds of constituents, not every verb is compatible with every verb phrase.*
- Example: verb *want* can be used
  - with a VP complement: "I want a flight"
  - with an infinitive VP complement: "I want to fly to Dallas"
- But verb *find* cannot take such a VP complement
  - "I find to fly to Dallas"

17

Parsing as Search

- Take the sentence: Book that flight
- Use the grammar:

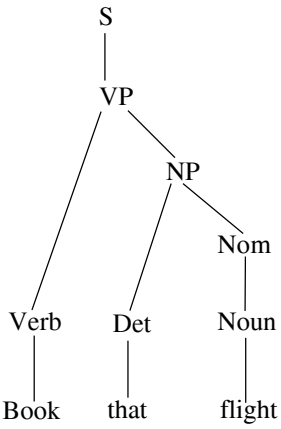
S → NP VP	Det → that | this | a
S → Aux NP VP	Noun → book | flight | meal | money
S → VP	Verb → book | include | prefer
NP → Det Nominal	Aux → does
Nominal → Noun	
Nominal → Noun Nominal	Prep → from | to | on
NP → Proper-Noun	Proper-Noun → Houston | TWA
VP → Verb	
VP → Verb NP	Nominal → Nominal PP

18

Generate the Parse

- How? Parsing search
  - find all trees (*a*) root is S
    (*b*) cover all words
- 2 kinds of constraints
  1) 3 leaves
  2) grammar rules

2 search strategies



top-down
(goal-directed search)

bottom-up
(data-directed search)

19

Top-Down Parsing

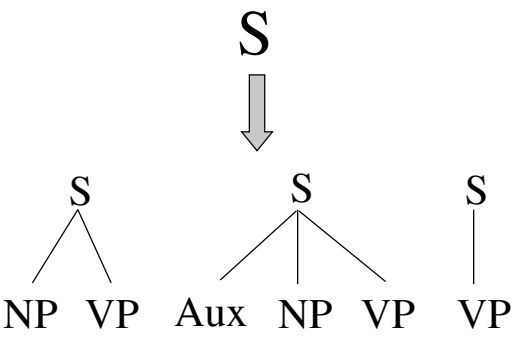
- Searches for a parse tree by trying to build the root **S** down to the leaves.

S → NP VP

S → Aux NP VP

S → VP

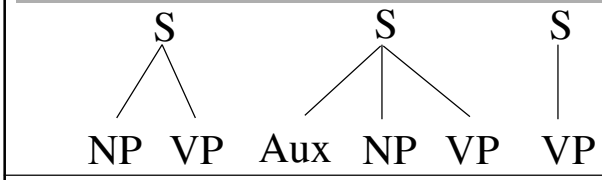
3 rules

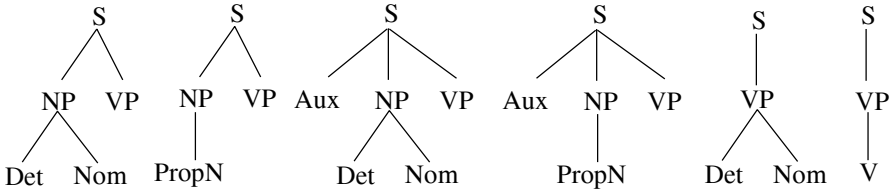


2nd ply has 3 trees!

20

Continuing the Search





NP → Det Nominal

NP → Proper-Noun

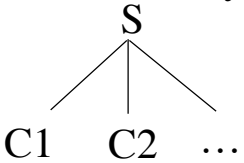
VP → Verb

VP → Verb NP

21

What happens?

- At each ply of the search space, we use the right-hand-sides of the rules to provide new sets of expectations for the parser – used to recursively generate the rest of the trees



$\begin{cases} S \rightarrow \ldots \\ S \rightarrow \ldots \\ S \rightarrow \ldots \end{cases}$ n_1 rules

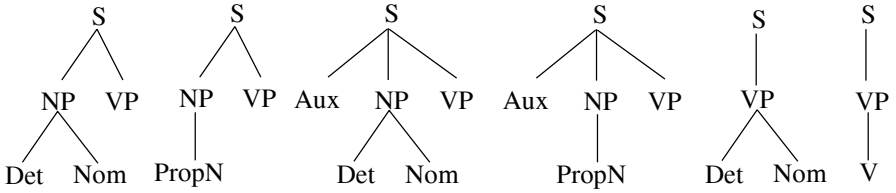
$\begin{cases} C1 \rightarrow abc \\ C2 \rightarrow AcD \\ \ldots \end{cases}$ n_2 rules

n_1 X n_2 trees

22

How much do we grow the trees?

- until they eventually reach the PoS categories (the tree leaves)
- What then?
  - Trees whose leaves fail to match all words in the input are rejected!
- For the sentence: *Book that flight*



23

Bottom-Up Parsing

- it is used in shift-reduce parsers common in computer languages.
- The parser starts with the words!
  - The tree is built from the words up!

⟹ Scan the grammar rules right-to-left

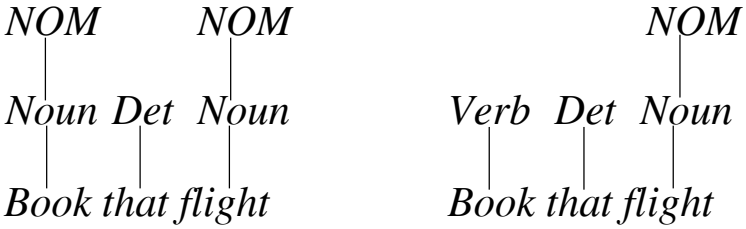
Book that flight

Noun Det Noun *Verb Det Noun*

Book that flight *Book that flight*

24

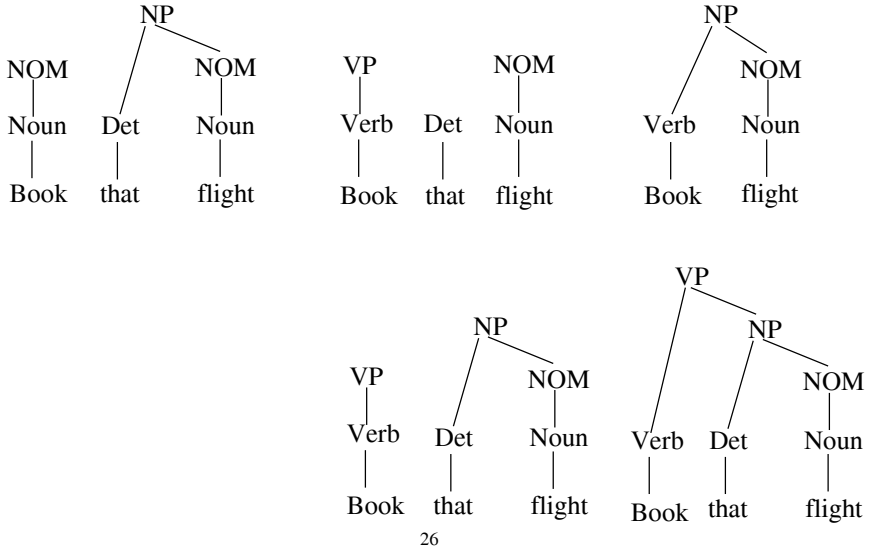
The Search Continues



- The rule *Nominal → Noun* is applied to both trees!

25

Finish The Search



26

How Does It Work ?

- The parser extends one ply to the next by looking for places in the parse-in-progress whose the right-hand-side of some rule might fit !

27

Comparing Top-Down and Bottom-Up

- Top-down: advantage → never wastes time on trees that cannot result in an S
- Bottom-up → trees that have no hope of leading to an S (or fitting with any of their neighbors) are generated with wild abandon
- Top-down: disadvantage → spends considerable time on S trees that are not consistent with the input
- Bottom-up → never suggests trees that are not at least locally grounded in the actual input

28

Can We Get Started ?

- Now we have an idea on how to parse !
- We can use a POS-tagger !
- Let's go to work
  - ➔ Where is the grammar ?

Problem: Collecting grammar rules from a corpus.

What corpus: on the CS net, at /projects/nlp

What kind of grammars are there ?

29

Chomsky's Classification

4 classes of grammars:

- Class 0: unrestricted phrase-structure grammars (No restriction of types of rules)
- Class 1: Context-sensitive grammars

$$x \rightarrow y$$

x and y are sequences of T and/or N symbols.

The length of y ≥ the length of x

30

More Classes

- Class 2: Context-free grammars

  $A \rightarrow x$

  A is a non-terminal

  x is a sequence of T and/or N symbols

- Class 3: Regular grammars

  $A \rightarrow Bt$ or $A \rightarrow t$

  $A, B \in N$

  $t \in T$

Note: The higher the class, the more restrictive it is.

31