CSE440: Natural Language Processing II

Farig Sadeque
Assistant Professor
Department of Computer Science and Engineering
BRAC University

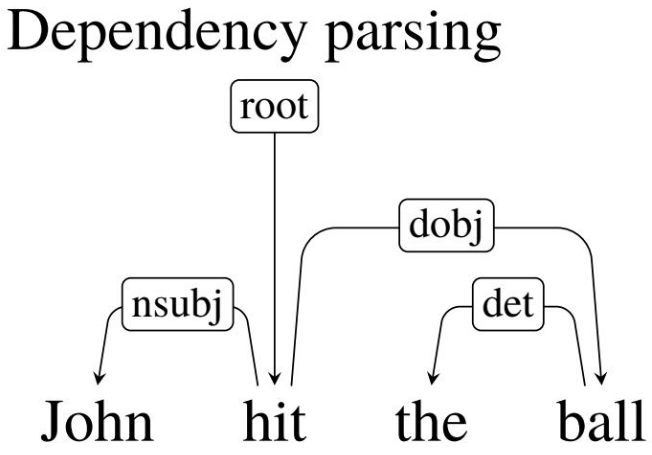
Lecture 8: Parsing

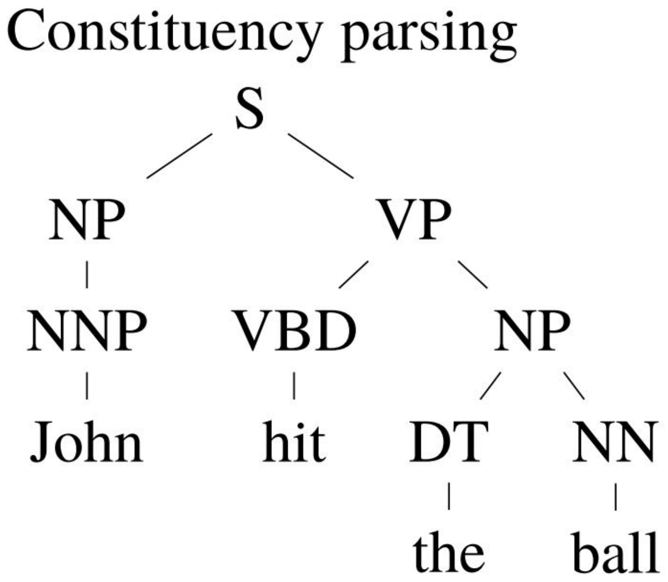
Outline

- Constituency parsing (SLP 17)
- Cocke-Kasami-Younger algorithm (SLP 17)
- Probabilistic parsing (Lecture)
- Dependency parsing (SLP 18)

Structured prediction

A structured prediction problem cannot easily be decomposed into simple text or word classification.





Constituency parsing

What is a constituent?

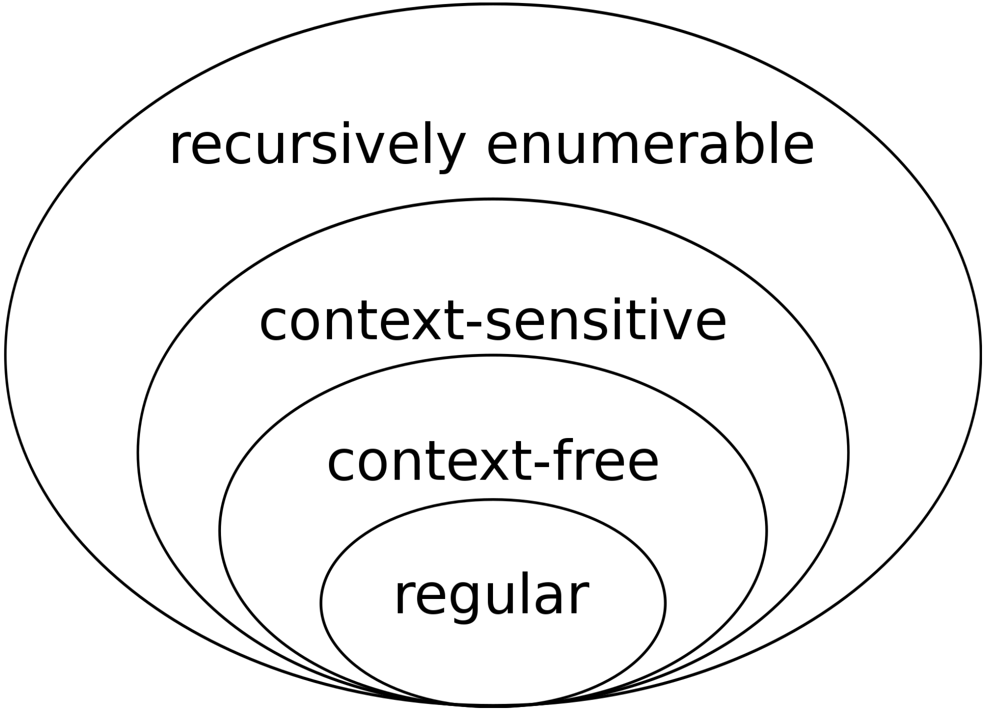
A constituent is a group of words behaving as a single unit.

- *I'm flying from Tucson to Minneapolis [next week].*
- *I'm flying [next week] from Tucson to Minneapolis.*
- *[Next week] I'm flying from Tucson to Minneapolis.*
- **[Next] I'm flying [week] from Tucson to Minneapolis.*
- **I'm flying [next] from Tucson to Minneapolis [week].*

Constituency parsing

- Gives us a parse tree with indicates which constituents should connect to which one to create a meaningful sentence
- To understand how constituency parsing works, we first need to understand Context free grammar

Chomsky hierarchy of languages



Context free grammar

- Also known as phrase-structure grammar
- They can describe sentence structure of natural language easily
- Was formalized by Noam Chomsky in 1956, but idea of this grammar dates back to early 1900s

Context free grammar

A context-free grammar consists of:

- N, a set of non-terminal symbols
- Σ, a set of terminal symbols (disjoint from N)
- R, a set of productions of the form $A \rightarrow \beta$, where
  - A is a non-terminal,
  - β is a string of symbols from $(\Sigma \cup N)*$
- S, a designated start symbol from N

Examples for production rules

$$
\begin{array}{lll}
\text{S} \longrightarrow \text{VP} & \text{DET} \longrightarrow \textit{that} & \text{DET} \longrightarrow \textit{my} \\
\text{VP} \longrightarrow \text{VERB NP} & \text{NOUN} \longrightarrow \textit{cat} & \text{NOUN} \longrightarrow \textit{thief} \\
\text{NP} \longrightarrow \text{DET NOUN} & \text{VERB} \longrightarrow \textit{stop} & \text{VERB} \longrightarrow \textit{pet}
\end{array}
$$

In natural language processing, our units are words, and all words are terminals. Terms that are written in lowercase are terminals, whereas terms written in uppercase are non-terminals.

Derivation

If $A \longrightarrow \beta$ is a production of R and α and γ are in $(\Sigma \cup N)*$, then $\alpha A \gamma$ directly derives $\alpha\beta\gamma$ (written as $\alpha A\gamma \Rightarrow \alpha\beta\gamma$).

Example:

stop NP $\Rightarrow$ *stop* DET NOUN from the grammar:

S $\longrightarrow$ VP	DET $\longrightarrow$ *that*	DET $\longrightarrow$ *my*
VP $\longrightarrow$ VERB NP	NOUN $\longrightarrow$ *cat*	NOUN $\longrightarrow$ *thief*
NP $\longrightarrow$ DET NOUN	VERB $\longrightarrow$ *stop*	VERB $\longrightarrow$ *pet*

If $\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m$

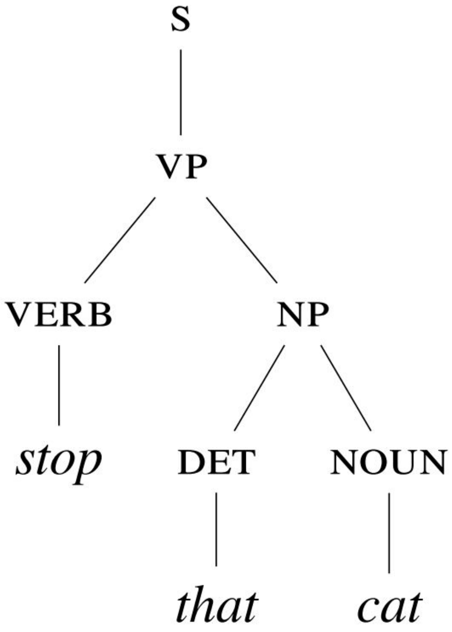
then α_1 derives α_m (written as $\alpha_1 \overset{*}{\Rightarrow} \alpha_m$)

Example:

S $\overset{*}{\Rightarrow}$ *stop that cat* from the grammar above.

Derivation tree

$$
\begin{aligned}
\text{S} &\longrightarrow \text{VP} \\
\text{VP} &\longrightarrow \text{VERB NP} \\
\text{NP} &\longrightarrow \text{DET NOUN} \\
\text{DET} &\longrightarrow \textit{that} \\
\text{DET} &\longrightarrow \textit{my} \\
\text{NOUN} &\longrightarrow \textit{cat} \\
\text{NOUN} &\longrightarrow \textit{thief} \\
\text{VERB} &\longrightarrow \textit{stop} \\
\text{VERB} &\longrightarrow \textit{pet}
\end{aligned}
$$



This derivation tree is popularly known as a constituency tree

Constituency parsing

- Ambiguity in phrase-structure grammars
- Chomsky normal form
- Cocke-Kasami-Younger (CKY) algorithm

Parsing is difficult

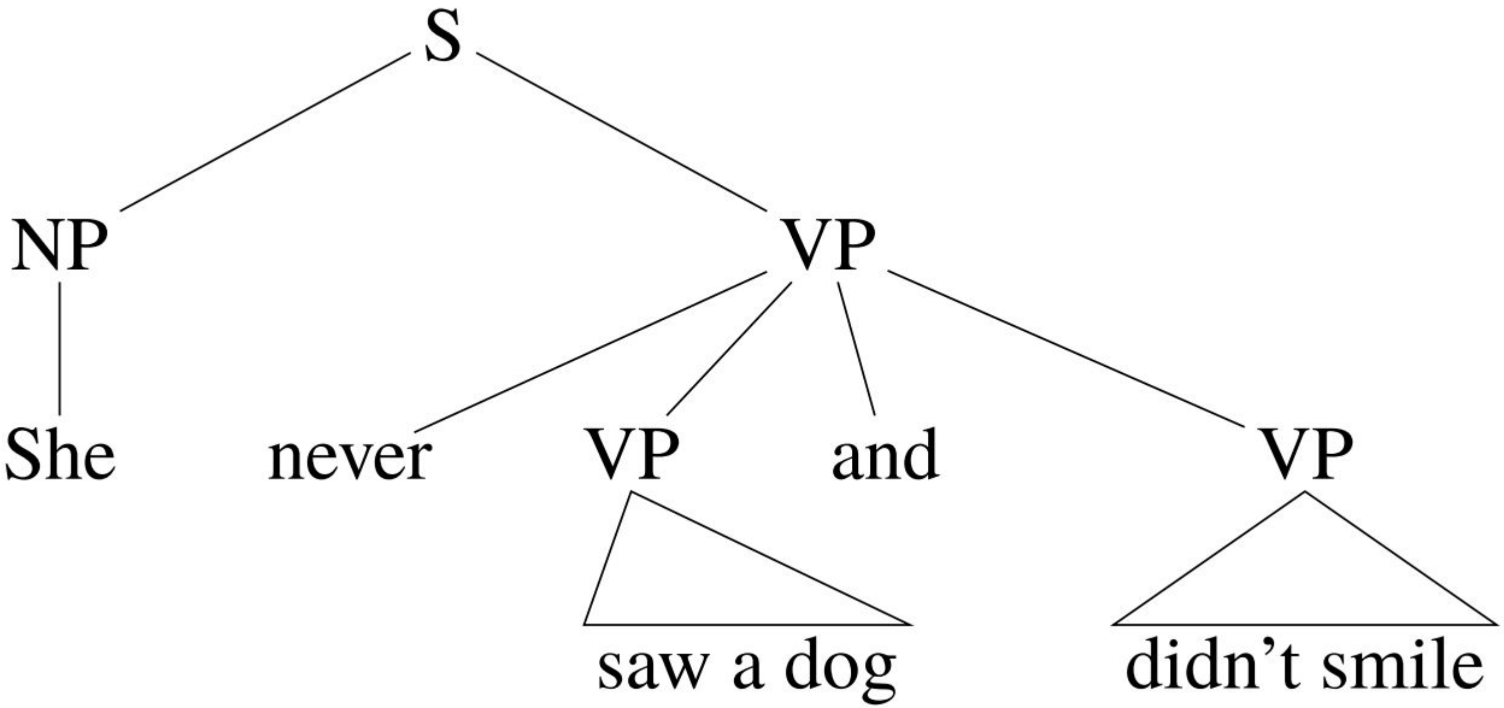
Why?

- Ambiguity
- *I shot an elephant in my pajamas*

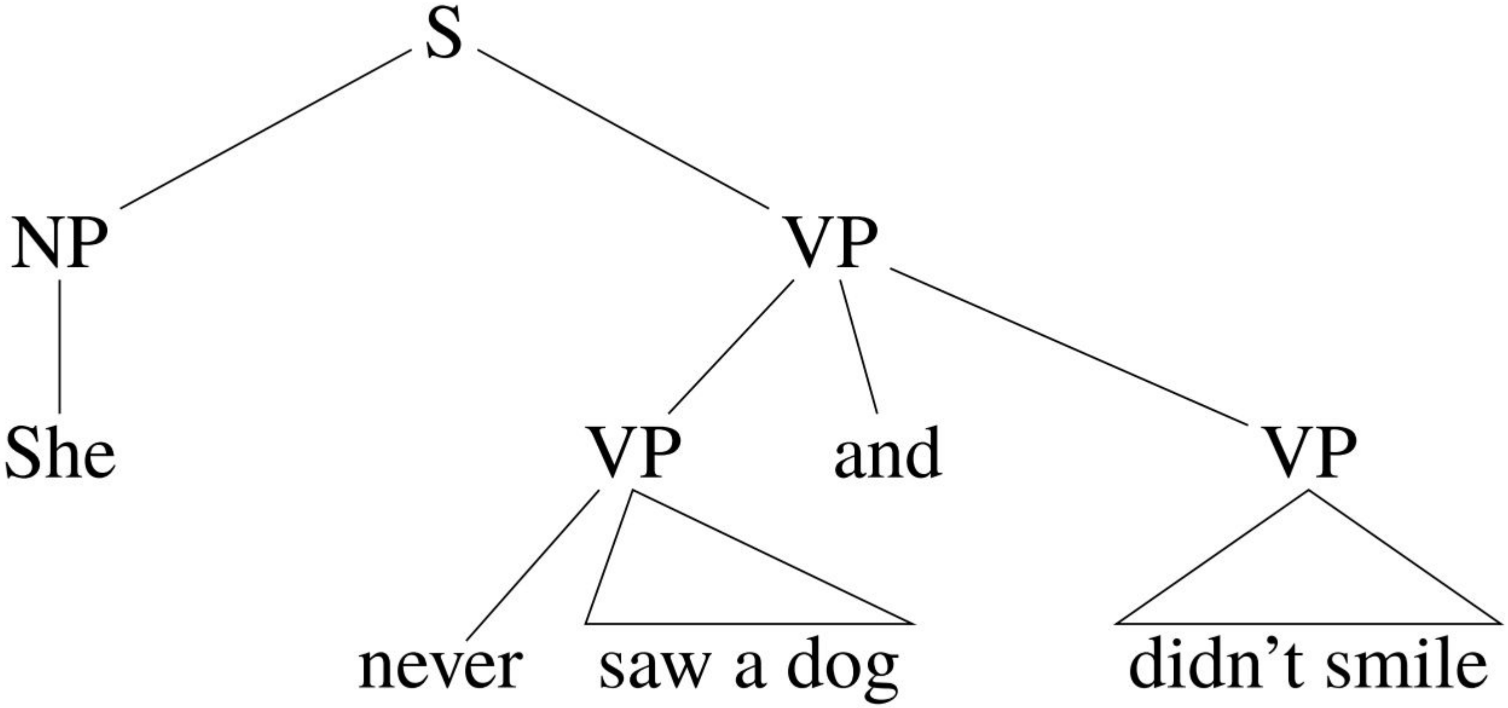
We have two common types of structural ambiguity:

- Attachment ambiguity
- Coordination ambiguity

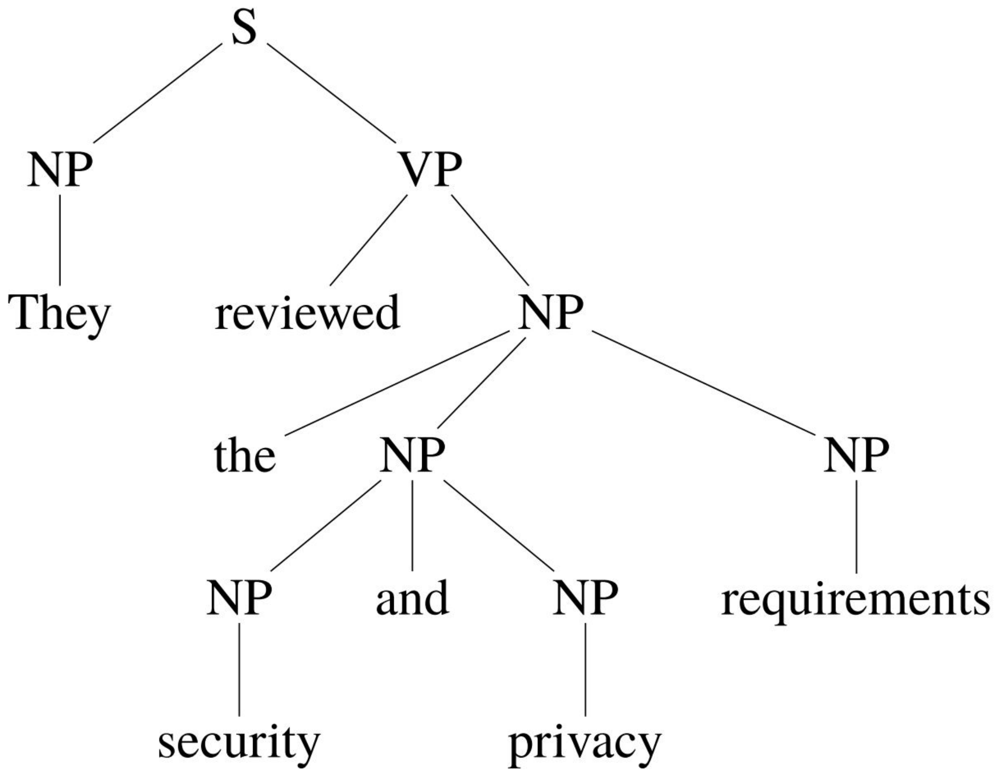
Attachment ambiguity



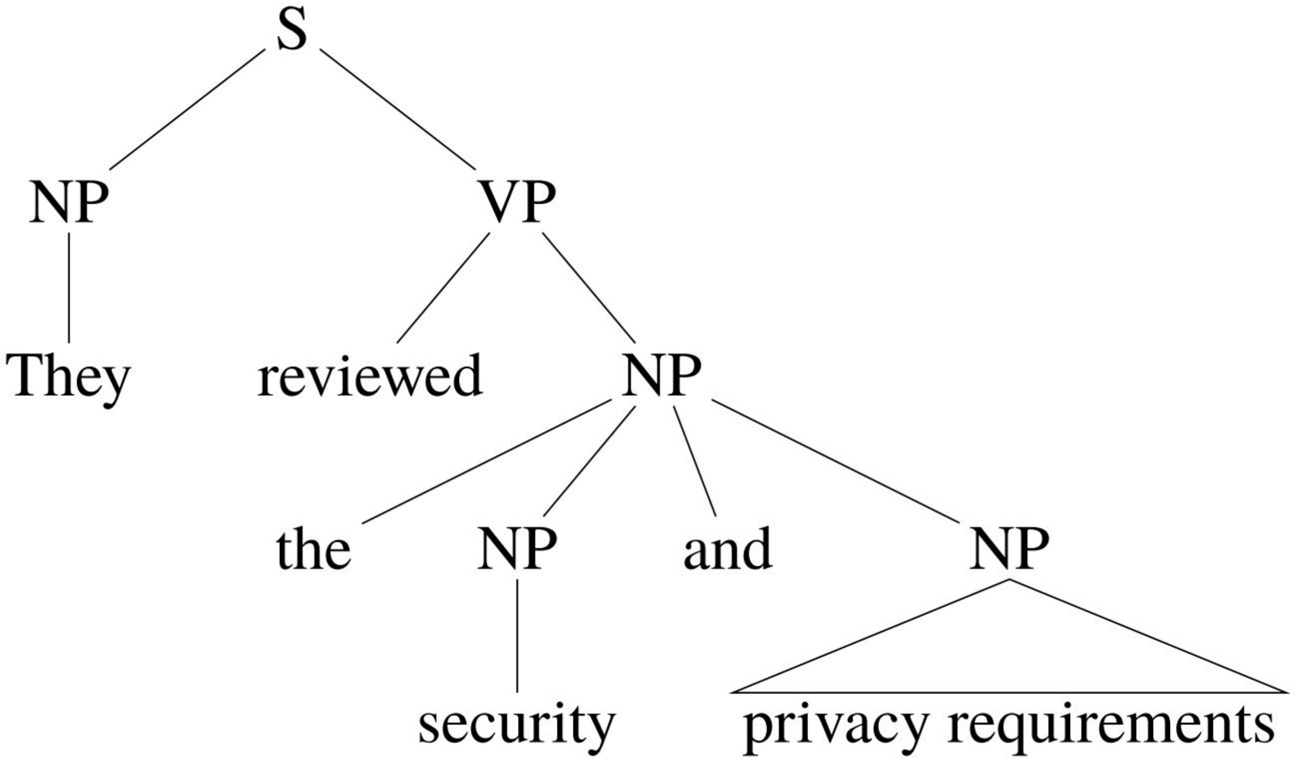
Attachment ambiguity



Coordination ambiguity



Coordination ambiguity



Practice

The following news headline is syntactically ambiguous. Identify two different possible parses of the sentence.

Lawmen From Mexico Barbecue Guests

Chomsky Normal Form (CNF)

Productions in context free grammars can have:

- Exactly one nonterminal on their left-hand side
- Any number of terminals or nonterminals on the right

This variability is not ideal for parsing.

A grammar in Chomsky normal form (CNF) allows only two types of right-hand sides:

- a single terminal
- two non-terminals

Examples:
NN → pizza
NP → DET NN
VP → to VP

Conversion to CNF

For any terminal that is not alone on the right-hand side, introduce a dummy non-terminal

VP → to VP	VP → TO VP
	TO → to

For any non-terminal that is alone on the right-hand side, substitute it everywhere

S→ VP	S→ VERB NP
VP → VERB NP	S → VERB PP
VP → VERB PP	VP → VERB NP
	VP → VERB PP

For any right-hand side with >2 non-terminals, split it

VP → VERB NP PP	VP → VERB TEMP
	TEMP → NP PP

Cocke-Kasami-Younger (CKY) algorithm

Intuition:

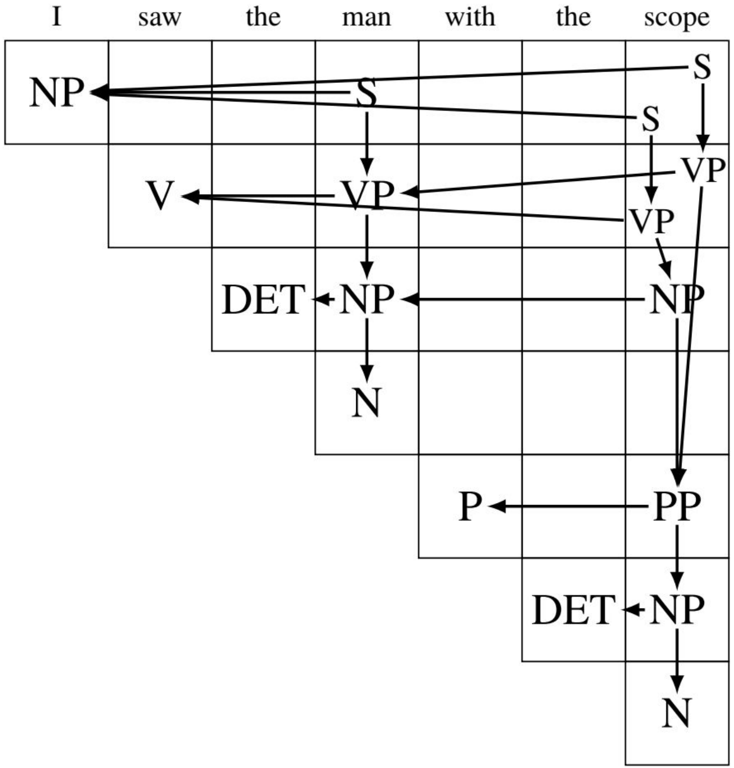
- Consider all word spans as possible constituents
    - word 1
    - words 2 to 4
    - etc.
- To see if a span could be a constituent, look for pairs of smaller constituents. E.g., for words 5 to 9, check:
    - 5 to 6 and 6 to 9
    - 5 to 7 and 7 to 9
    - etc.
- If you find a pair of constituents, see if there's a grammar rule that allows them to be combined

CKY bottom up parsing

I	saw	the	man	with	the	scope

$$\mathrm{S} \rightarrow \mathrm{NP\ VP}$$
$$\mathrm{NP} \rightarrow \mathrm{DET\ N}$$
$$\mathrm{NP} \rightarrow \mathrm{NP\ PP}$$
$$\mathrm{PP} \rightarrow \mathrm{P\ NP}$$
$$\mathrm{VP} \rightarrow \mathrm{V\ NP}$$
$$\mathrm{VP} \rightarrow \mathrm{VP\ PP}$$
$$\mathrm{DET} \rightarrow \textit{the}$$
$$\mathrm{NP} \rightarrow \textit{I}$$
$$\mathrm{N} \rightarrow \textit{man}$$
$$\mathrm{N} \rightarrow \textit{scope}$$
$$\mathrm{P} \rightarrow \textit{with}$$
$$\mathrm{V} \rightarrow \textit{saw}$$

CKY bottom up parsing



$S \rightarrow NP\ VP$

$NP \rightarrow DET\ N$

$NP \rightarrow NP\ PP$

$PP \rightarrow P\ NP$

$VP \rightarrow V\ NP$

$VP \rightarrow VP\ PP$

$DET \rightarrow$ *the*

$NP \rightarrow$ *I*

$N \rightarrow$ *man*

$N \rightarrow$ *scope*

$P \rightarrow$ *with*

$V \rightarrow$ *saw*

Ambiguity kicks in

- How can we resolve this ambiguity?
- Solution: Probabilistic context free grammar

Context free grammar

A context-free grammar consists of:

- N, a set of non-terminal symbols
- Σ, a set of terminal symbols (disjoint from N)
- R, a set of productions of the form $A \rightarrow \beta$, where
    - A is a non-terminal,
    - β is a string of symbols from $(\Sigma \cup N)*$
- S, a designated start symbol from N

Probabilistic context free grammar

A probabilistic context-free grammar consists of:

- N, a set of non-terminal symbols
- Σ, a set of terminal symbols (disjoint from N)
- R, a set of productions of the form $A \rightarrow \beta$ [p], where
    - A is a non-terminal,
    - β is a string of symbols from $(\Sigma \cup N)*$
    - p is the probability $P(\beta|A)$, also written $P(A \rightarrow \beta)$
- S, a designated start symbol from N

Example

$$
\begin{array}{rlrlrl}
\text{S} \longrightarrow \text{VP}\ [1.0] & & \text{DET} \longrightarrow \textit{that}\ [0.7] & & \text{DET} \longrightarrow \textit{my}\ [0.3] \\
\text{VP} \longrightarrow \text{VERB NP}\ [1.0] & & \text{NOUN} \longrightarrow \textit{cat}\ [0.6] & & \text{NOUN} \longrightarrow \textit{thief}\ [0.4] \\
\text{NP} \longrightarrow \text{DET NOUN}\ [1.0] & & \text{VERB} \longrightarrow \textit{stop}\ [0.1] & & \text{VERB} \longrightarrow \textit{pet}\ [0.9]
\end{array}
$$

Learning production probabilities

Learning $P(\beta|A)$ for each rule $A \longrightarrow \beta$:

1. Manually annotate many sentences with trees
2. $P(\beta|A) = \dfrac{\textsc{count}(A \longrightarrow \beta)}{\textsc{count}(A)}$

How many annotated trees are needed?

Enough so that every rule occurs many times!

Probability of a tree

The probability of a tree is calculated from the probabilities of the productions used in the tree:

$$P(T) = \prod_{(A \rightarrow \beta) \in T} P(\beta | A)$$

Example:

S $\longrightarrow$ VP [1.0]

VP $\longrightarrow$ VERB NP[1.0]

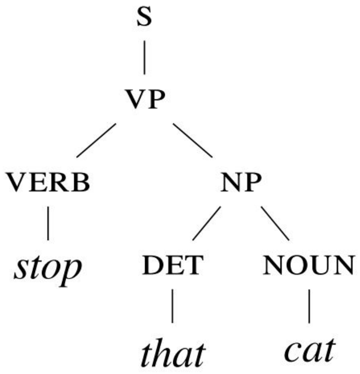
NP $\longrightarrow$ DET NOUN[1.0]

DET $\longrightarrow$ *that* [0.7]

NOUN $\longrightarrow$ *cat* [0.6]

VERB $\longrightarrow$ *stop* [0.1]

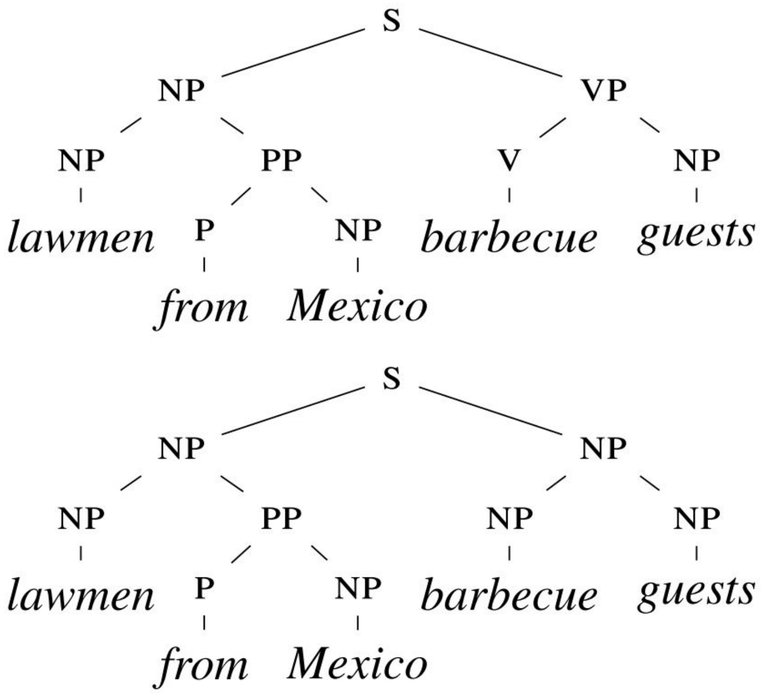
...



$$P(T) = 1.0 \cdot 1.0 \cdot 1.0 \cdot 0.7 \cdot 0.6 \cdot 0.1 = 0.042$$

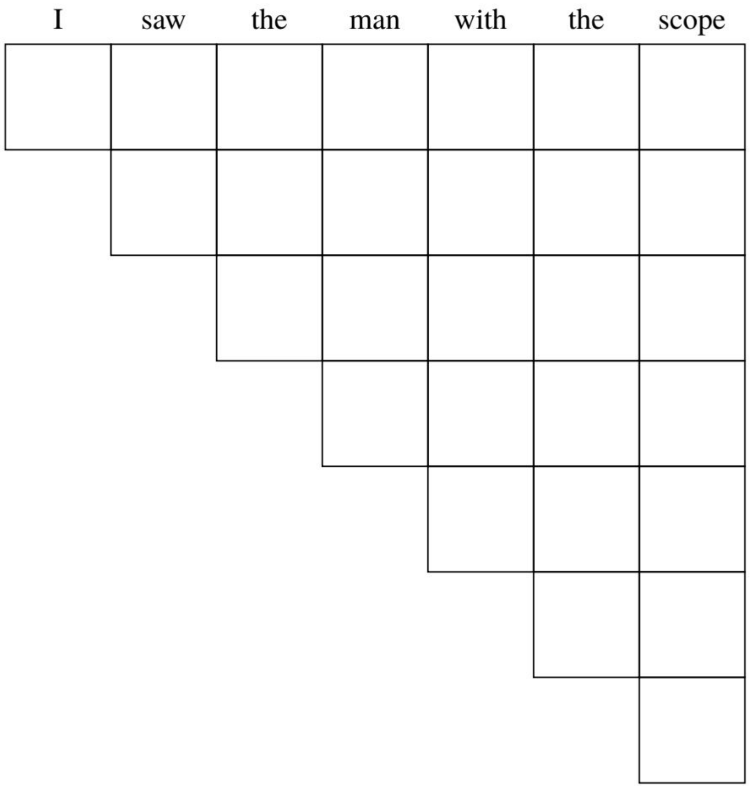
Classwork

Which parse is more probable?



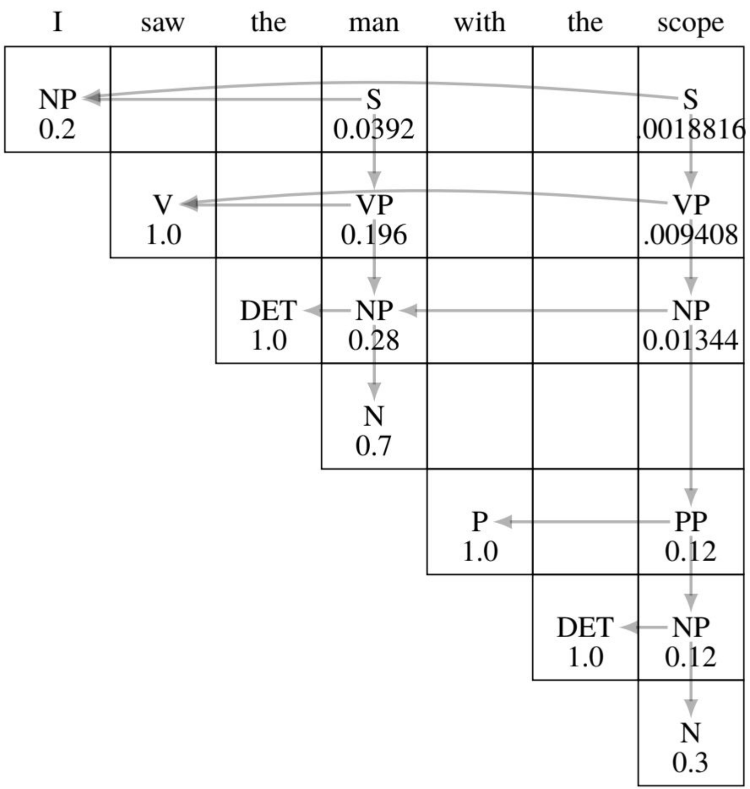
S → NP VP	[0.3]
S → NP NP	[0.7]
VP → V NP	[1.0]
PP → P NP	[1.0]
NP → NP NP	[0.2]
NP → NP PP	[0.3]
NP → *lawmen*	[0.1]
NP → *barbecue*	[0.2]
NP → *guests*	[0.1]
NP → *Mexico*	[0.1]
P → *from*	[1.0]
V → *barbecue*	[1.0]

Probabilistic CKY parsing



$$
\begin{array}{rcll}
S & \rightarrow & NP\ VP & [1.0] \\
VP & \rightarrow & V\ NP & [0.7] \\
VP & \rightarrow & VP\ PP & [0.3] \\
PP & \rightarrow & P\ NP & [1.0] \\
NP & \rightarrow & DET\ N & [0.4] \\
NP & \rightarrow & NP\ PP & [0.4] \\
NP & \rightarrow & \textit{I} & [0.2] \\
DET & \rightarrow & \textit{the} & [1.0] \\
N & \rightarrow & \textit{man} & [0.7] \\
N & \rightarrow & \textit{scope} & [0.3] \\
P & \rightarrow & \textit{with} & [1.0] \\
V & \rightarrow & \textit{saw} & [1.0]
\end{array}
$$

Probabilistic CKY parsing



$$
\begin{array}{rll}
S \longrightarrow & NP\ VP & [1.0] \\
VP \longrightarrow & V\ NP & [0.7] \\
VP \longrightarrow & VP\ PP & [0.3] \\
PP \longrightarrow & P\ NP & [1.0] \\
NP \longrightarrow & DET\ N & [0.4] \\
NP \longrightarrow & NP\ PP & [0.4] \\
NP \longrightarrow & \textit{I} & [0.2] \\
DET \longrightarrow & \textit{the} & [1.0] \\
N \longrightarrow & \textit{man} & [0.7] \\
N \longrightarrow & \textit{scope} & [0.3] \\
P \longrightarrow & \textit{with} & [1.0] \\
V \longrightarrow & \textit{saw} & [1.0]
\end{array}
$$

Dependency parsing

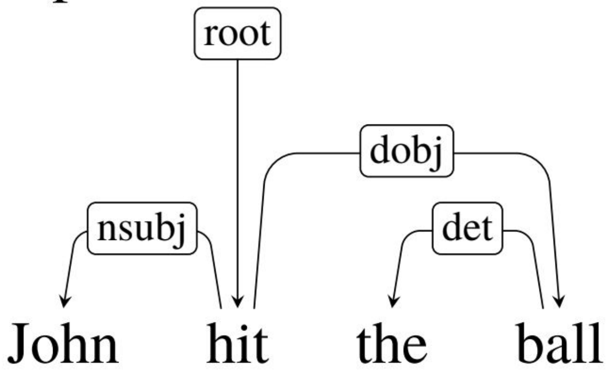
SLP chapter 18

Dependency trees

A dependency tree is a directed graph, where the nodes are words, the edges are syntactic relations, and

- There is 1 root node with no incoming arcs
- All other nodes have exactly 1 incoming arc
- There is a unique path from the root to each node

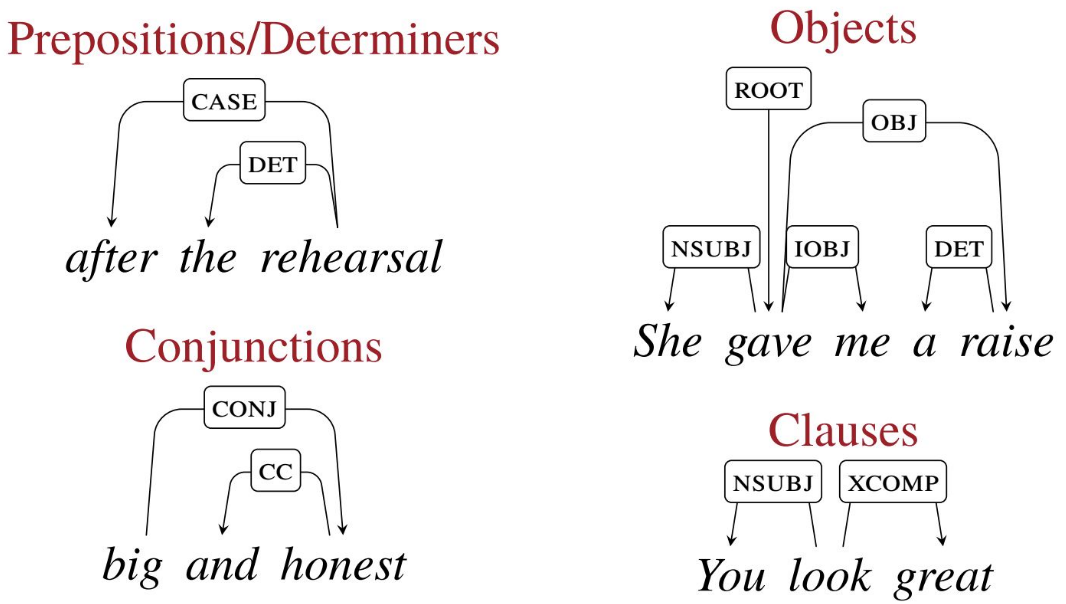
Example:



Intuition:
The head plays the primary role,
the dependent modifies that role

Dependency relations

Some examples from https://universaldependencies.org/



Example dependency relations from SLP

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
OBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

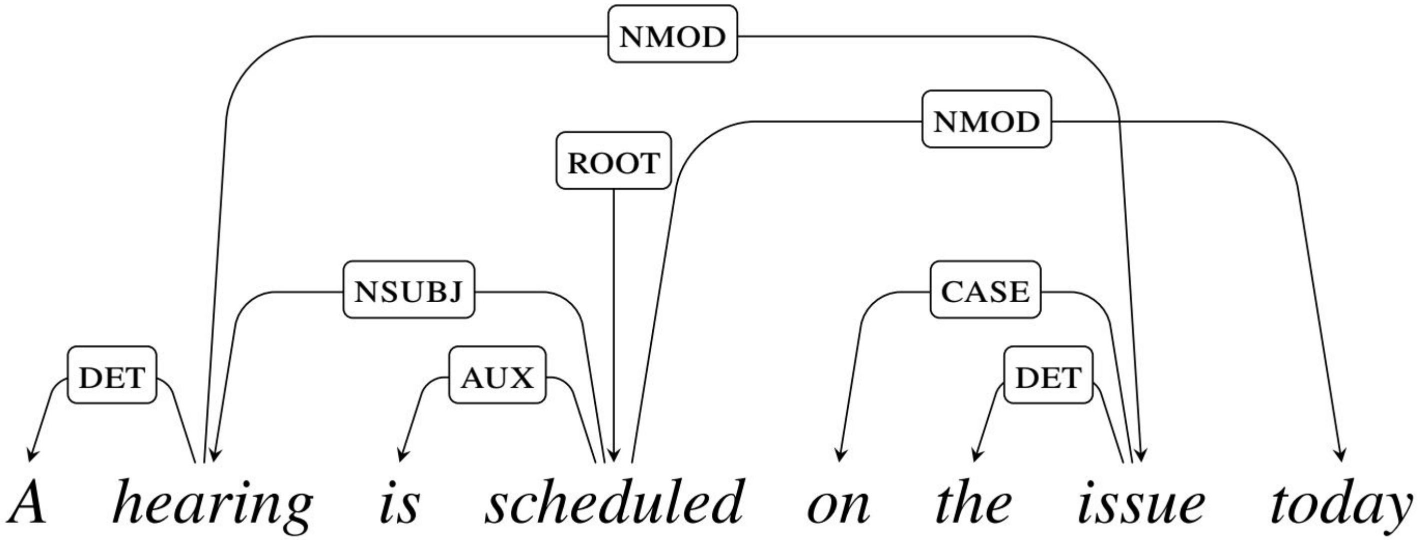
Figure 18.3 Examples of some Universal Dependency relations.

Dependency treebank

- A treebank is a *parsed text corpus that annotates syntactic or semantic sentence structure*
- Every sentence in a treebank is annotated with a parse tree
- Dependency treebanks are created by
  - having human annotators directly generate dependency structures for a given corpus, or
  - by hand-correcting the output of an automatic parser.
- The most popular dependency treebank is the universal dependency treebank
  - Open community project
  - 200 treebanks for more than 100 languages
    https://universaldependencies.org/u/dep/

Projectivity

- An arc x → y is projective if there is a path from x to every word between x and y in the text. A tree is projective if all arcs in the tree are projective.



Issues with projectivity

From SLP:

- The most widely used English dependency treebanks were automatically derived from phrase structure treebanks through the use of head-finding rules. The trees generated in such a fashion will always be projective, and hence will be incorrect when non-projective examples like this one are encountered
- There are computational limitations to the most widely used families of parsing algorithms