CE4045 CZ4045 SC4002
Natural Language Processing

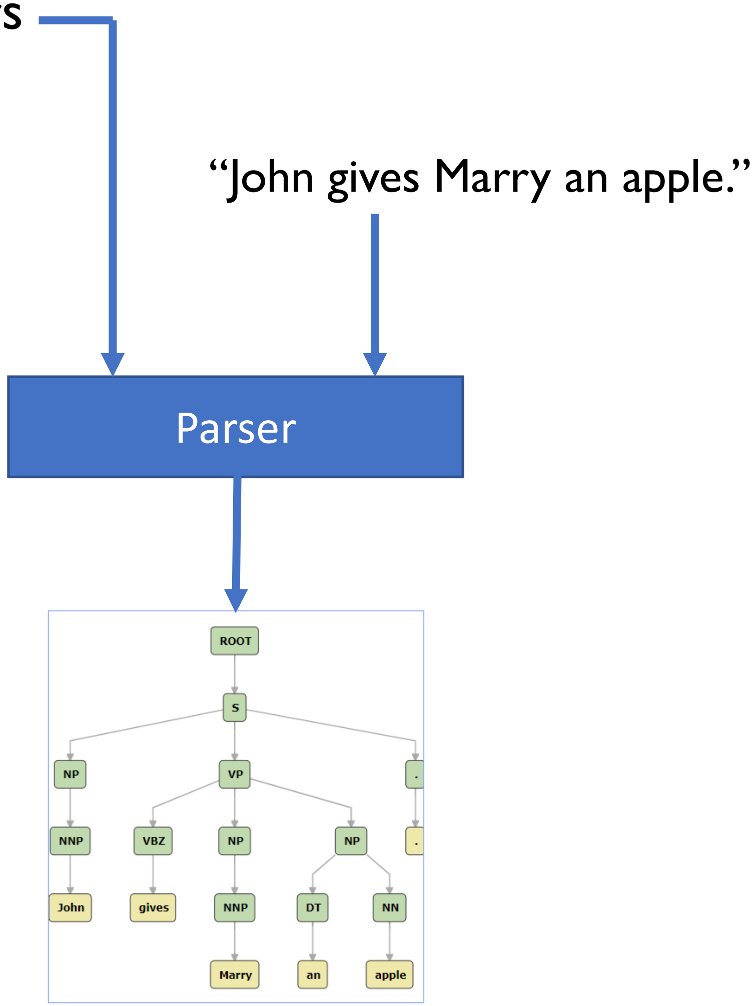
Constituency Grammars and Parsing

Dr. Sun Aixin

Constituency Grammars and Constituency Parsing

- Constituency Grammars

- Parsing



Syntax vs Semantic

- The word **syntax** comes from the Greek *sýntaxis*, meaning "setting out together or arrangement",
- In our context: syntax refers to **the way words are arranged together**.
  - There are certain probabilities between words, e.g., N-gram model
  - Words can offer be replaced by words under the same POS tags, like: "I have a green apple" and "I have a red apple"
  - A formal way to describe syntax? → Grammar
- **Context-free grammars** are the backbone of many formal models of the syntax of natural language (and computer languages).
  - Applications: grammar checking, semantic interpretation, dialogue understanding, and machine translation
- There are other forms for grammars, e.g., combinatory categorial grammar (CCG), and syntactic dependency.

Constituency

- Syntactic constituency is the idea that **groups of words** can **behave as single units**, or constituents
- Example constituency: **Noun Phrase**, a sequence of words surrounding at least one noun
  - 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
- Another example: **prepositional phrase**
  - The whole prepositional phrase behaves as a single unit, and can be placed at different places 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*

Context-Free Grammar (CFG)

- CFG is also called **Phrase-Structure** Grammars, and the formalism is equivalent to **Backus-Naur Form** (BNF)
- A context-free grammar consists of
  - A set of **rules** or **productions**, each of which expresses the ways that symbols of the language can be grouped and ordered together,
  - and a **lexicon** of words and symbols
  - The symbols in a CFG are divided into two classes
    - **Terminal**: The symbols that correspond to words in the language
    - **Non-terminals**: The symbols that express ***abstractions*** over these terminals

NP → *Det Nominal*
NP → *ProperNoun*
Nominal → *Noun* | *Nominal Noun*

These rules express that a noun phrase (**NP**) can be composed of either a *ProperNoun* or a *determiner* (Det) followed by a *Nominal*.

A *Nominal* consists of one or more Nouns

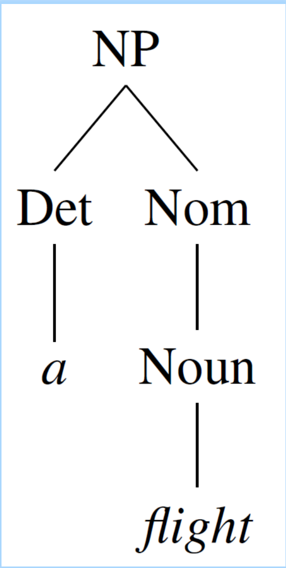
Det → *a* | *the*
Noun → *flight*

"a, the, and flight" are **terminals;** "*NP, Det, Nominal, ProperNoun* and *Noun*", are **non-terminals**

Context-Free Grammar (CFG)

NP → *Det Nominal*
Noun → *flight*

- Each rule has an arrow (→)
  - To the left: **a single non-terminal**,
  - To the right: **an** **ordered** **list** of **one or more terminals and non-terminals**.
  - The non-terminal associated with each word is its POS.
- A CFG can be viewed (or used) in two ways
  - **Generation**: Start with *NP*, we have "*NP* → *Det Nominal*"; for *Det*, we can have "the", for *Nominal*, we can have "*Nominal* → *Noun*", "*Noun* → *flight*". As the result, we generate the string "the flight"
  - **Derivation**: Given a string of words "a flight", we derive its structure using CFG rules, with a sequence of rule expansions.



- The formal language defined by a CFG is the set of strings that are derivable from the designated **start symbol.**
  - Each grammar must have **one** designated start symbol, often called *S*
  - In out tasks, *S* is usually interpreted as the "sentence" node.

Example rules

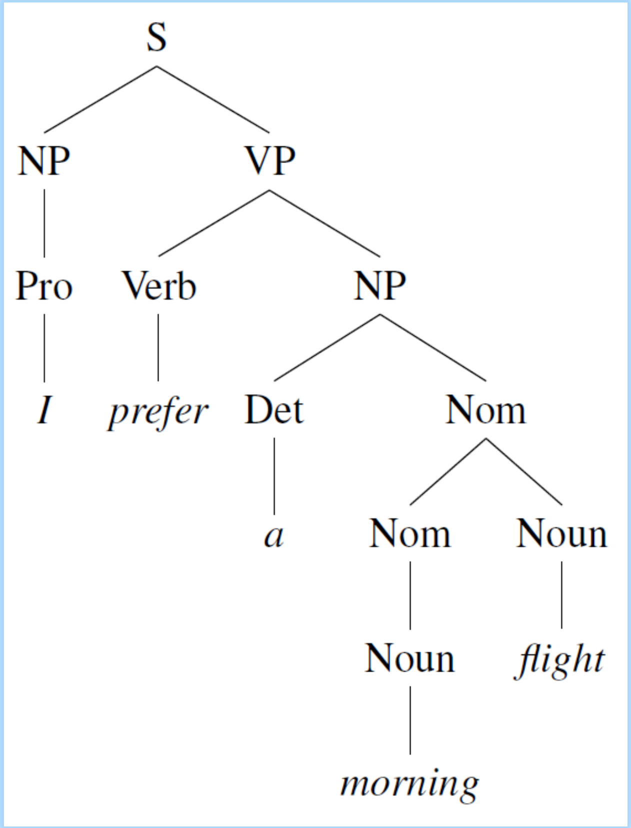
- $S \rightarrow NP\ VP$ — I prefer a morning flight
- $VP \rightarrow Verb\ NP$ — prefer a morning flight
- $VP \rightarrow Verb\ NP\ PP$ — leave Boston in the morning
- $VP \rightarrow Verb\ PP$ — leaving on Thursday
- $PP \rightarrow Preposition\ NP$ — from Los Angeles

- More example prepositional phrases
  - **to** Seattle
  - **in** Minneapolis
  - **on** Wednesday
  - **in** the evening
  - **on** the ninth of July
  - **on** these flights
  - **about** the ground transportation in Chicago
  - **of** the round trip flight on United Airlines
  - **of** the AP fifty seven flight
  - **with** a stopover in Nashville

A more complete example: L_0 grammar

Grammar Rules	Examples
$S \rightarrow NP\ VP$	I + want a morning flight
$NP \rightarrow Pronoun$	I
$\quad \mid Proper\text{-}Noun$	Los Angeles
$\quad \mid Det\ Nominal$	a + flight
$Nominal \rightarrow Nominal\ Noun$	morning + flight
$\quad \mid Noun$	flights
$VP \rightarrow Verb$	do
$\quad \mid Verb\ NP$	want + a flight
$\quad \mid Verb\ NP\ PP$	leave + Boston + in the morning
$\quad \mid Verb\ PP$	leaving + on Thursday
$PP \rightarrow Preposition\ NP$	from + Los Angeles

Noun → *flights* | *flight* | *breeze* | *trip* | *morning*
Verb → *is* | *prefer* | *like* | *need* | *want* | *fly* | *do*
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* | *in*
Conjunction → *and* | *or* | *but*



bracketed representation

$[_S\ [_{NP}\ [_{Pro}\ \text{I}]]\ [_{VP}\ [_V\ \text{prefer}]\ [_{NP}\ [_{Det}\ \text{a}]\ [_{Nom}\ [_N\ \text{morning}]\ [_{Nom}\ [_N\ \text{flight}]]]]]]$

Sentence-Level Constructions

For your information

- Declarative structure *S → NP VP*
  - "I prefer a morning flight."
- Imperative structure *S→ VP*
  - "List all flights between five and seven p.m."
- Yes-no question structure *S → Aux NP VP*
  - "Do any of these flights have stops?"
- Sentences with **wh-**word (who, whose, when, where, what, which, how, why)
  - wh-subject-question structure *S → Wh-NP VP*
    - The **wh-**word is the subject, like declarative structure
    - "What airlines fly from Burbank to Denver?"
  - wh-non-subject-question *S → Wh-NP Aux NP VP*
    - The **wh-**phrase is not the subject of the sentence
    - "What flights do you have from Burbank to Tacoma Washington?"

More grammar rules

For your information

- Refer to SLP3, Chapter 12, Section 12.3
  - The noun phrases
  - The verb phrases
    - **subcategorization** frame: the way a verb taking complements

Frame	Verb	Example
$\emptyset$	eat, sleep	I ate
NP	prefer, find, leave	Find [$_{NP}$ the flight from Pittsburgh to Boston]
$NP\ NP$	show, give	Show [$_{NP}$ me] [$_{NP}$ airlines with flights from Pittsburgh]
$PP_{\text{from}}\ PP_{\text{to}}$	fly, travel	I would like to fly [$_{PP}$ from Boston] [$_{PP}$ to Philadelphia]
$NP\ PP_{\text{with}}$	help, load	Can you help [$_{NP}$ me] [$_{PP}$ with a flight]
$VPto$	prefer, want, need	I would prefer [$_{VPto}$ to go by United Airlines]
S	mean	Does this mean [$_{S}$ AA has a hub in Boston]

Treebanks

- **Sufficiently** robust grammars consisting of context-free grammar rules can be used to assign a parse tree to any sentence.
  - build a corpus where every sentence in the collection is paired with a corresponding parse tree.
  - Such a syntactically annotated corpus is called a treebank
  - Typically build using Parsers to automatically parse each sentence, followed hand-corrections by humans (linguists)
- Example Treebanks
  - The Penn Treebank Project for constituency parsing
  - The Universal Dependencies Project for dependency parsing
- From Treebanks, we can derive grammars of a language

$$
\begin{aligned}
VP &\rightarrow VBD\ PP \\
VP &\rightarrow VBD\ PP\ PP \\
VP &\rightarrow VBD\ PP\ PP\ PP \\
VP &\rightarrow VBD\ PP\ PP\ PP\ PP \\
VP &\rightarrow VB\ ADVP\ PP \\
VP &\rightarrow VB\ PP\ ADVP \\
VP &\rightarrow ADVP\ VB\ PP
\end{aligned}
$$

Combinatory Categorial Grammar (CCG)

For your information

- **Categories** are either atomic elements or single-argument functions that return a category, when provided with a desired category as argument
  - Categories X, Y
  - Function X/Y seeks a Y to its right and returns a value of X;
  - Function X\Y seeks a Y to its left and returns a value of X;

$$X/Y \quad Y \Rightarrow X$$
$$Y \quad X\backslash Y \Rightarrow X$$

- **Lexicon** consists of assignments of categories to words
  - Example:

$$\begin{array}{rc} \mathit{flight}: & N \\ \mathit{Miami}: & NP \\ \mathit{cancel}: & (S\backslash NP)/NP \end{array}$$

- **Parsing**

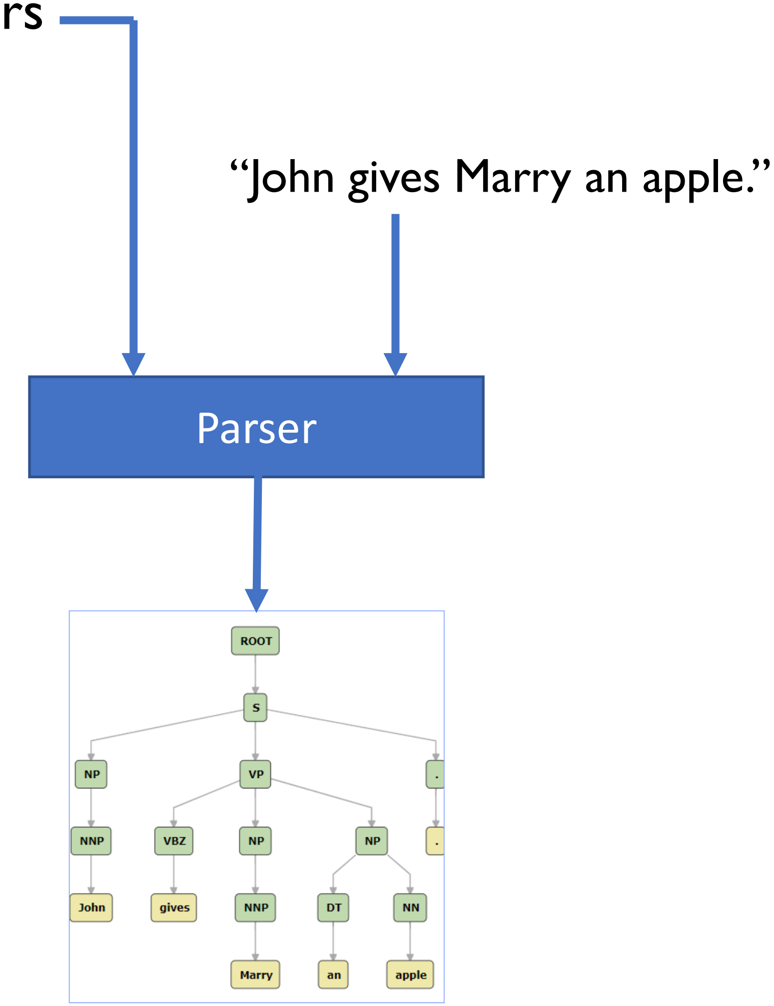
$$\begin{array}{ccc} \mathit{United} & \mathit{serves} & \mathit{Miami} \\ \hline \mathrm{NP} & (\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP} & \mathrm{NP} \\ & \multicolumn{2}{c}{\overline{\qquad\qquad\qquad}>} \\ & \multicolumn{2}{c}{\mathrm{S}\backslash\mathrm{NP}} \\ \multicolumn{3}{c}{\overline{\qquad\qquad\qquad\qquad\qquad}<} \\ \multicolumn{3}{c}{\mathrm{S}} \end{array}$$

Summary

- Syntax vs Semantics
- Constituency
- Context-free grammar
- Reference
  - Chapter 12 https://web.stanford.edu/~jurafsky/slp3/

Constituency Grammars and Constituency Parsing

- Constituency Grammars



- Parsing

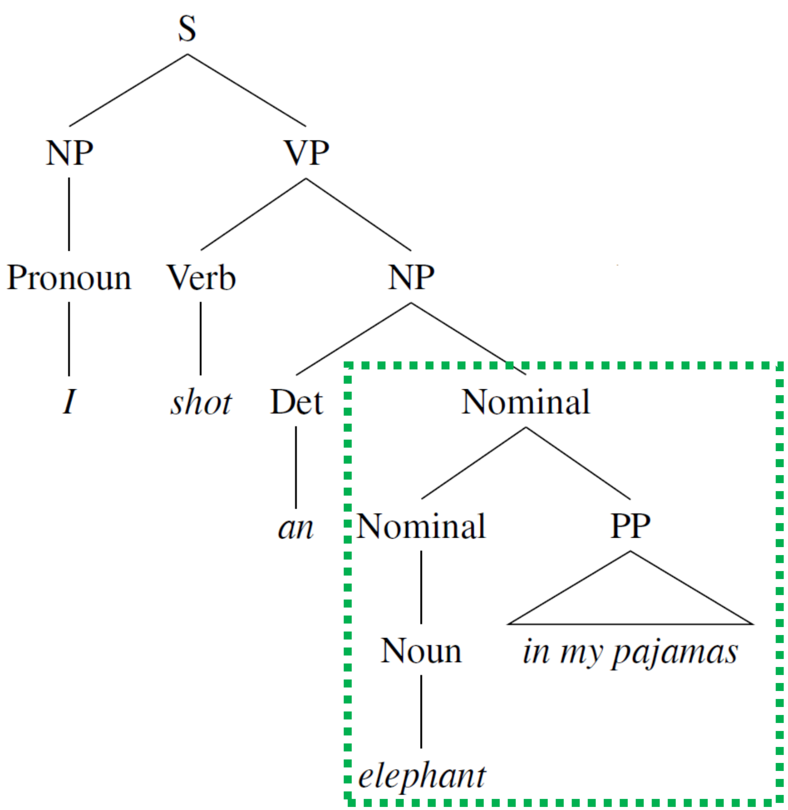
Constituency Parsing

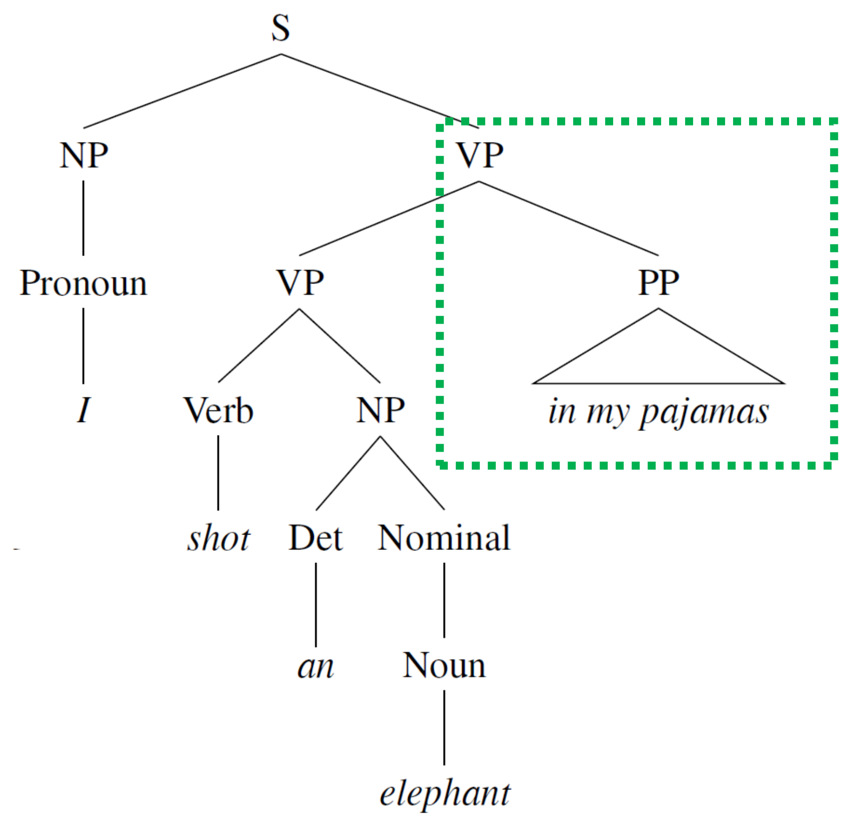
- Syntactic parsing is the task of assigning a syntactic structure to a sentence.
  - We need a **grammar**
  - We need a **parser**

- Parse trees can be used in many applications
  - **Grammar checking**: sentence that cannot be parsed may have grammatical errors
  - **Semantic analysis**: parse tree serves as an intermediate stage of representation for understanding the meaning of a sentence
  - Other applications like question answering and information extraction

- Key challenge here: **structural ambiguity**
  - occurs when the grammar can assign more than one parse to a sentence.

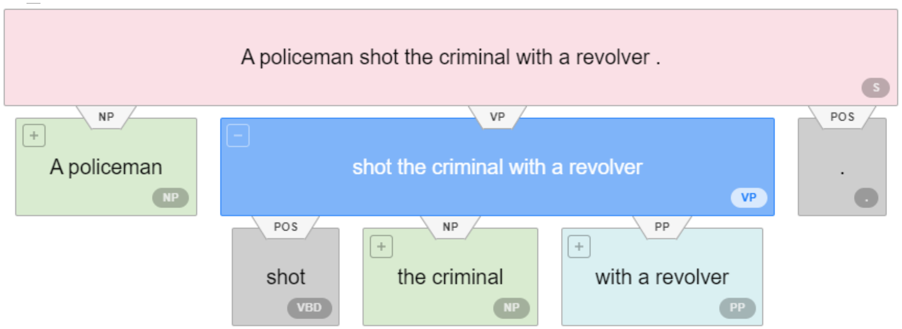
"I shot an elephant in my pajamas"





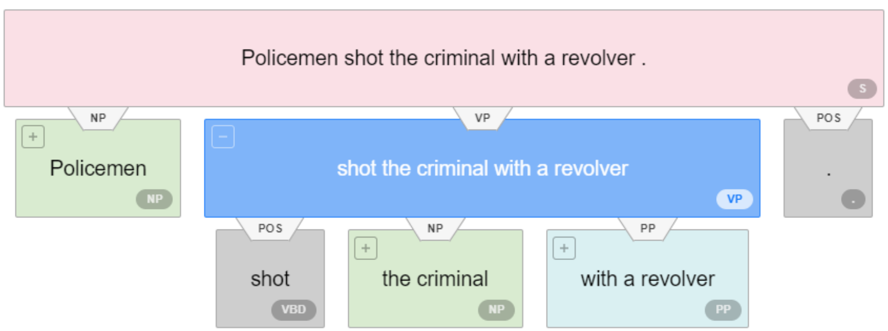
Structural ambiguity: Attachment ambiguity

- A policeman shot the criminal with a revolver.

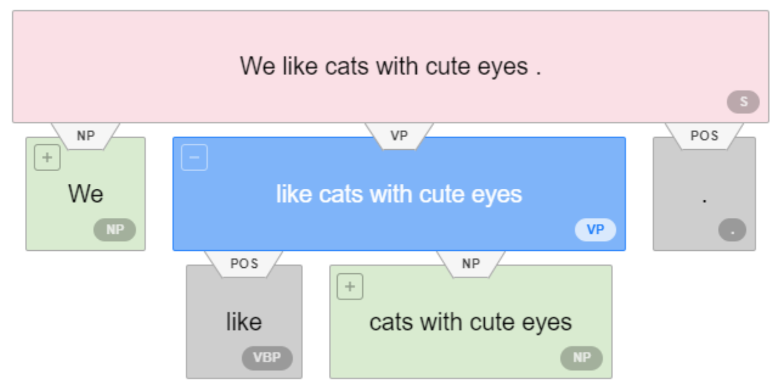


Attachment ambiguity: if a particular constituent can be attached to the parse tree at more than one place

- Policemen shot the criminal with a revolver.



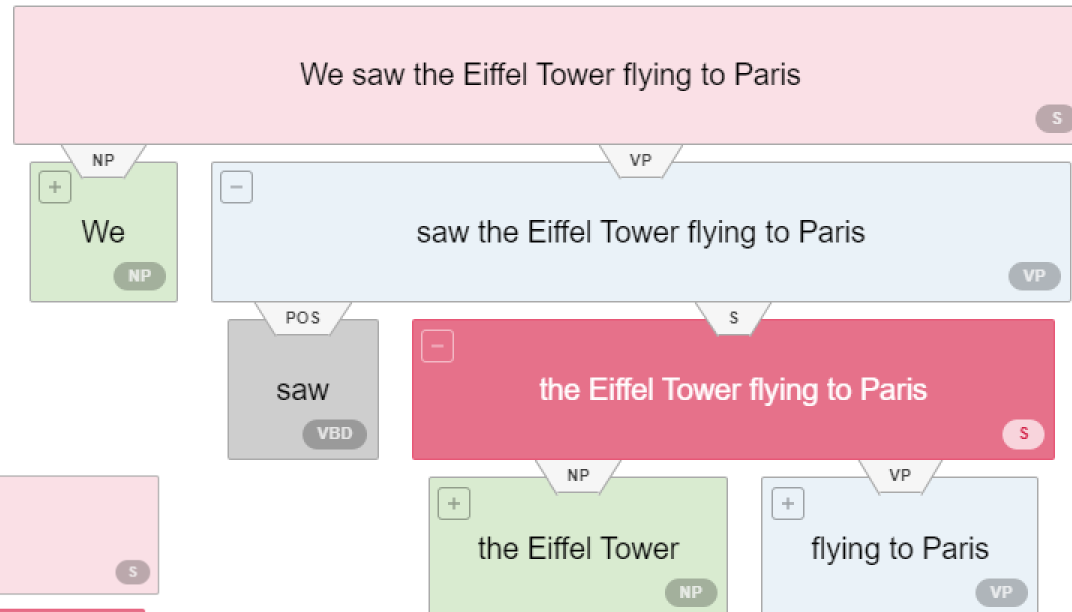
- We like cats with cute eyes.



https://demo.allennlp.org/constituency-parsing

Structural ambiguity: Attachment ambiguity

- We saw the Eiffel Tower flying to Paris
  - Who fly to Paris?



- Flying to Paris, we saw the Eiffel Tower.

Structural ambiguity: coordination ambiguity

- **Coordination ambiguity** occurs in some phrases with a conjunction word like "and".

- Example: $NP \rightarrow NP\ and\ NP$

"old men and women"	"dogs in house and cats"
[old men] and [women]	[dogs in house] and [cats]
[old] [men and women]	[dogs in] [house and cats]

- There are many grammatically correct but semantically unreasonable parses for naturally occurring sentences.
- Such ambiguity affect all parsers.

Parsing: A Dynamic Programming Approach

- A dynamic programming approach systematically fills in a table of solutions to sub-problems.
  - The complete table has the solution to all the sub-problems needed to solve the problem as a whole.
  - Edit distance, the Viterbi algorithm for HMM decoding

- In syntactic parsing, these sub-problems represent **parse trees for all the constituents** detected in the input.
  - Once a constituent (e.g., an NP) has been discovered in a segment of the input, we can record it and make it available for use in any subsequent derivation (e.g., a VP is derived from a Verb and a NP detected earlier). $VP \rightarrow$ Verb NP

- We next introduce the Cocke-Kasami-Younger (CKY) algorithm
  - The most widely used dynamic-programming based approach to parsing.
  - It can be extended with neural methods to handle the ambiguity issue.

The CKY algorithm

- CKY algorithm requires grammars to be in Chomsky Normal Form (CNF).
  - CNF rules can only be in two forms: $A \rightarrow B\ C$ or $A \rightarrow w$.
  - That is, the right-hand side of each rule must expand either to **two non-terminals** or to **a single terminal**.

- Any CFG can be converted into a corresponding equivalent CNF grammar
  - Rules that mix terminals with non-terminals on the right-hand side
    - e.g., *INF-VP* → *to VP*. Create a dummy non-terminal *TO*
    - *INF-VP* → *to VP* becomes *INF-VP* → *TO VP* and *TO* → *to*
  - Rules that have a single non-terminal on the right-hand side
    - e.g., *S*→ *VP*. Rewrite the right-hand side and expand *VP* with all its corresponding rules. *S*→ *VP* becomes *S*→ *Verb NP*, *S*→ *Verb NP PP*, and …
  - Rules that the length of the right-hand side is greater than 2
    - e.g., *S*→ *Verb NP PP*. Create a dummy non-terminal *X1*. *S*→ *Verb NP PP* becomes *S*→*X1 NP*, *X1*→*Verb NP*

An example CFG grammar in its CNF form

$\mathscr{L}_1$ Grammar	$\mathscr{L}_1$ in CNF
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

CNF rules can only be in two forms: ***A → B C*** or ***A → w***.

Each non-terminal node above the POS level in a parse tree will have **exactly two** daughters

That is: a non-terminal node can be derived from **exactly TWO constituents** (that can be derived earlier).

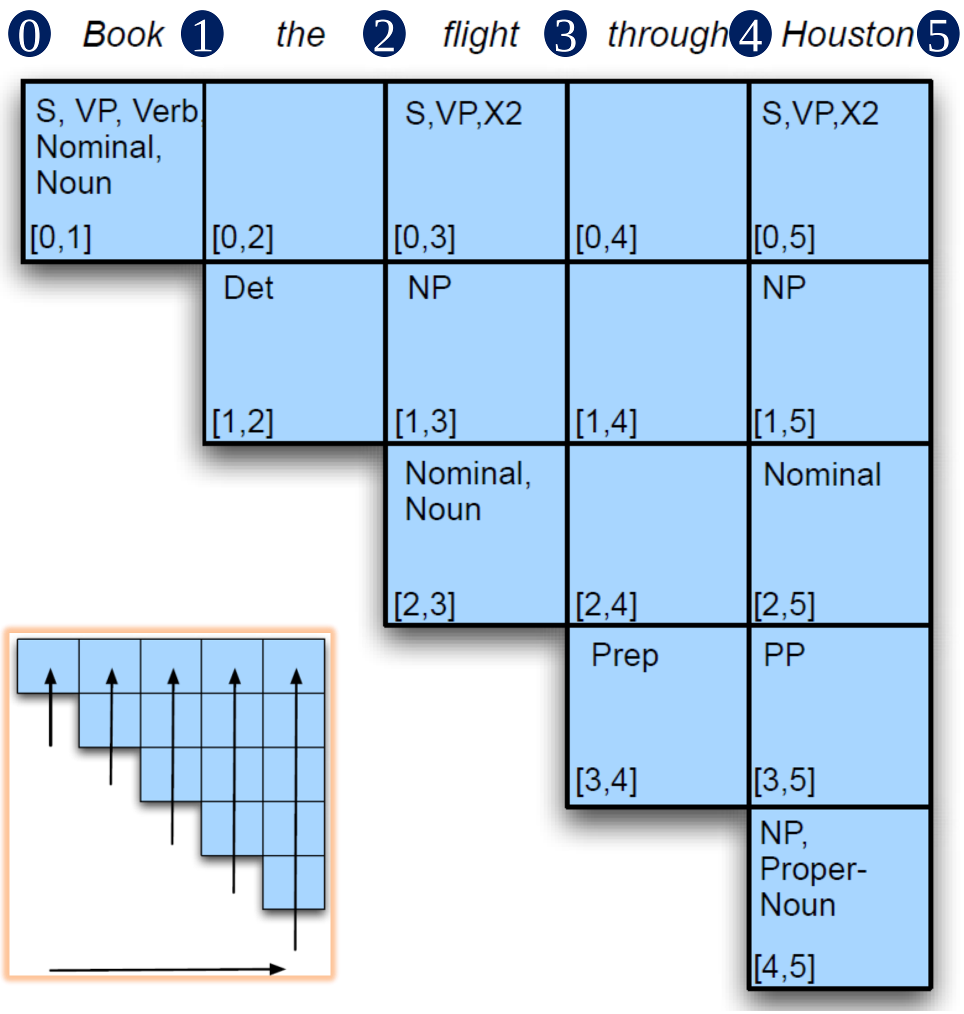
The CKY algorithm

- For a sentence of a length n words, we work with the upper-triangular portion of an $(n+1) \times (n+1)$ matrix
- The indexes (staring with 0) point at the **gaps** between the input words
- Each cell $[i, j]$ contains **the set of non-terminals** that represent **all the constituents** that **span positions i through j** of the input
- Example:
  - Input sentence: *Book the flight through Houston*
  - Indexes inserted at the gaps between words

    ⓿ Book ❶ the ❷ flight ❸ through ❹ Huston ❺

  - Cell $[0, 1]$ contains all constituents that can be assigned to "Book", e.g., *Noun*, *Verb*, *S*…
  - Cell $[1, 3]$ contains all constituents that can be assigned to "the flight", e.g., *NP*

The CKY algorithm



- Each cell $[i, j]$ contains **all the constituents** that **the text span** $[i, j]$ can be assigned
- Starting with cell $[0, 1]$, we fill cell $[1, 2]$, then cell $[0,2]$ …
- CNF rules can only be in two forms: ***A → B C*** or ***A → w***.
  - To assign cell $[0,2]$, we check all possible combinations of cell $[0,1]$ and cell$[1,2]$
  - To assign cell $[1, 5]$, we have

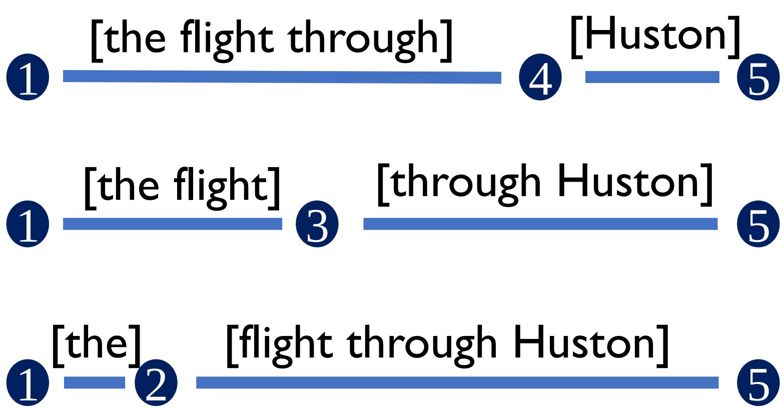
    $[1, 4]$ + $[4, 5]$

    $[1, 3]$ + $[3, 5]$

    $[1, 2]$ + $[2, 5]$

Example for cell [*1*, *5*]

0 Book 1 the 2 flight 3 through 4 Huston 5



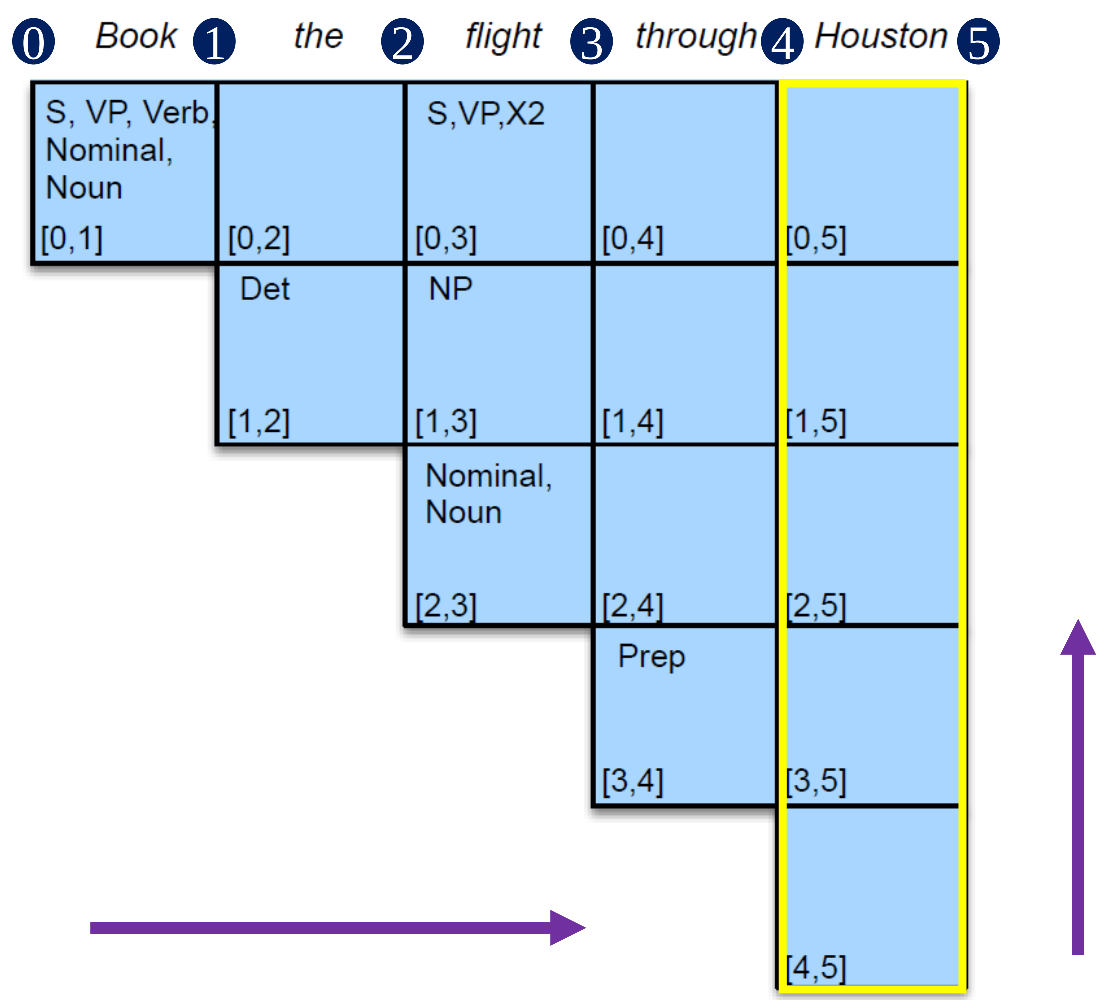
Book	the	flight	through	Houston
S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

- For every cell [i, j] covering two or more words, there is a k, such that we have cell [I, k] and cell [k, j] already computed, where $i < k < j$

The CKY algorithm, working on the last column

$\mathscr{L}_1$ in CNF

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$



The CKY algorithm, Cell [4, 5]

$\mathscr{L}_1$ in CNF

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$\underline{NP \rightarrow TWA \mid Houston}$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

0 *Book* 1 *the* 2 *flight* 3 *through* 4 *Houston* 5

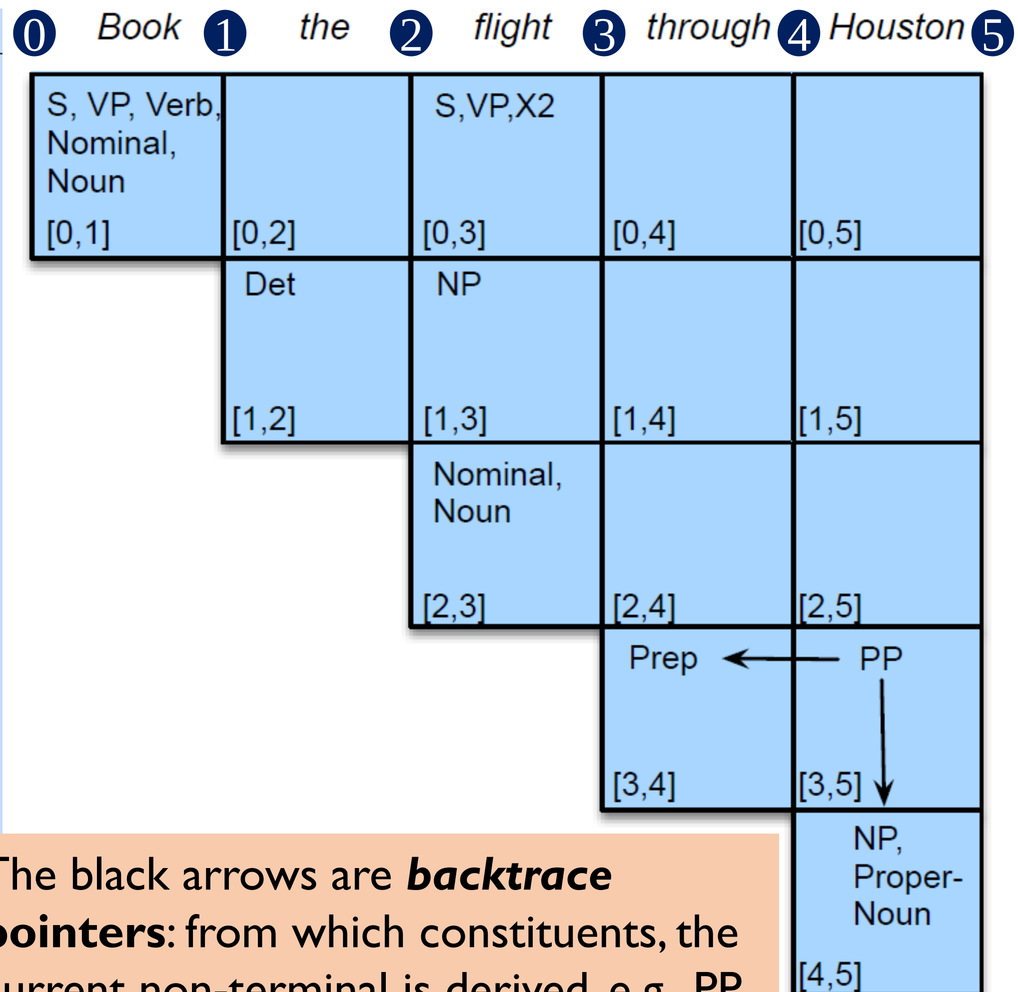
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]
	Det [1,2]	NP [1,3]	[1,4]	[1,5]
		Nominal, Noun [2,3]	[2,4]	[2,5]
			Prep [3,4]	[3,5]
				NP, Proper-Noun [4,5]

POS tags are not shown in the *L1* grammar

The CKY algorithm, Cell [3, 5]

$\mathscr{L}_1$ in CNF

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

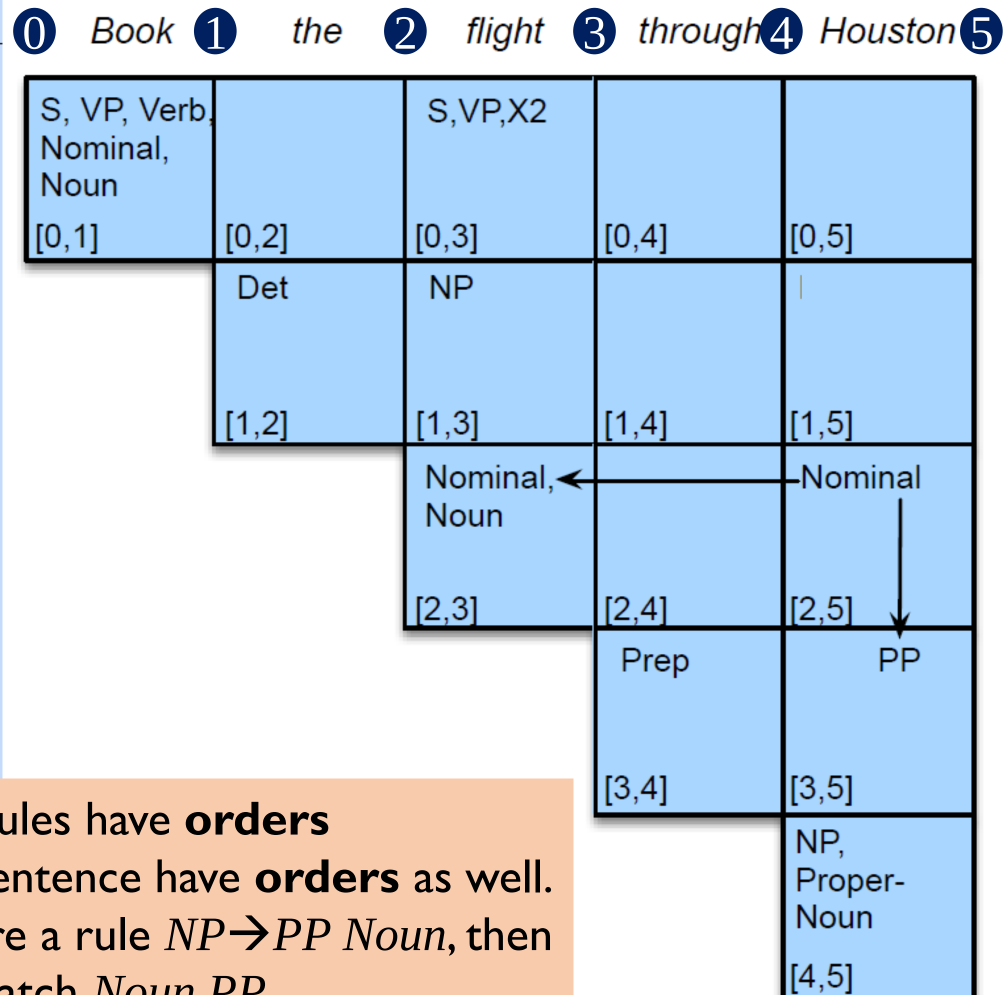


The black arrows are ***backtrace* pointers**: from which constituents, the current non-terminal is derived, e.g., PP

The CKY algorithm, Cell [2, 5]

$\mathscr{L}_1$ in CNF

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

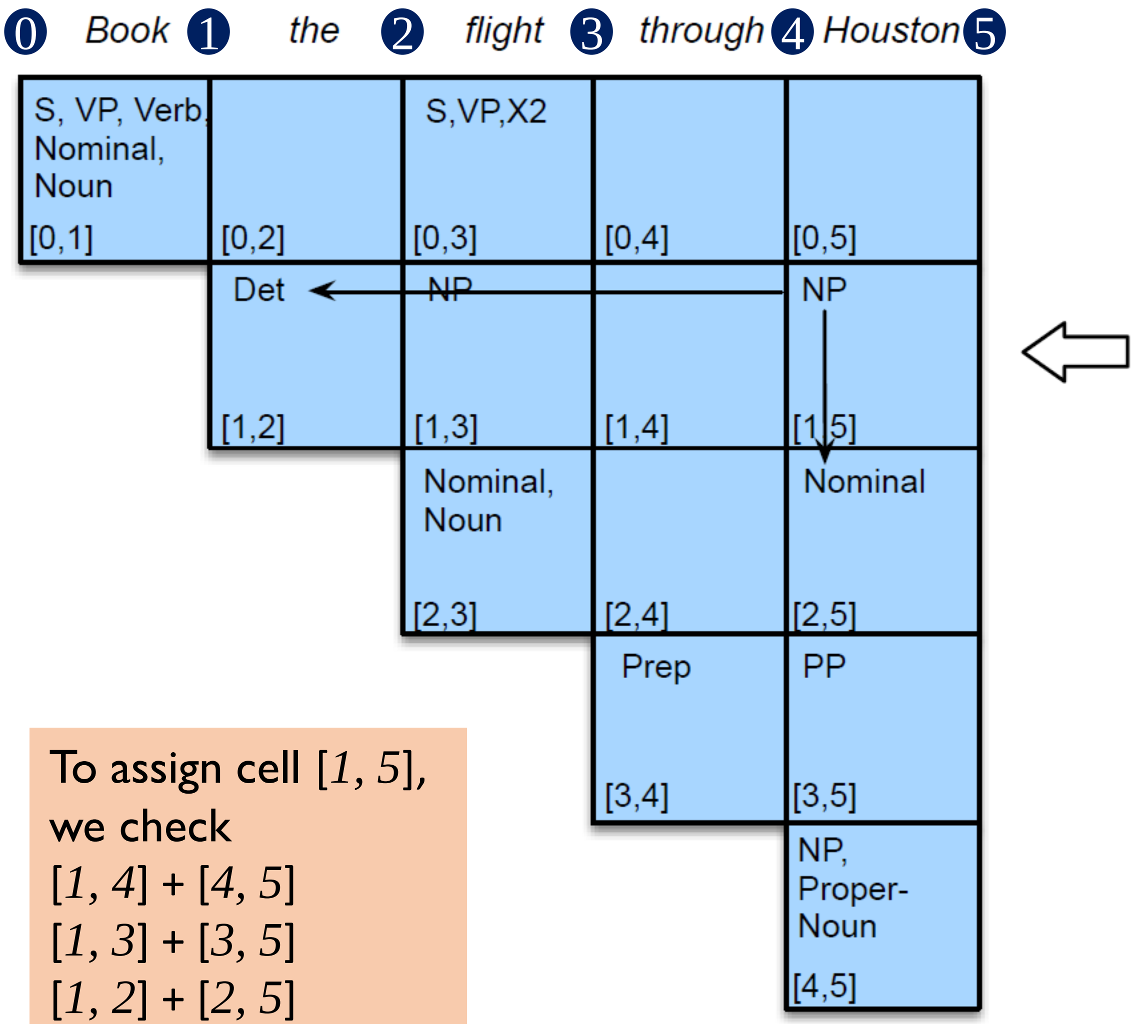


Grammar rules have **orders**
Words in sentence have **orders** as well.
If there were a rule *NP→PP Noun*, then it cannot match *Noun PP*.

The CKY algorithm, Cell [1, 5]

$\mathscr{L}_1$ in CNF

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$\underline{NP \rightarrow Det\ Nominal}$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

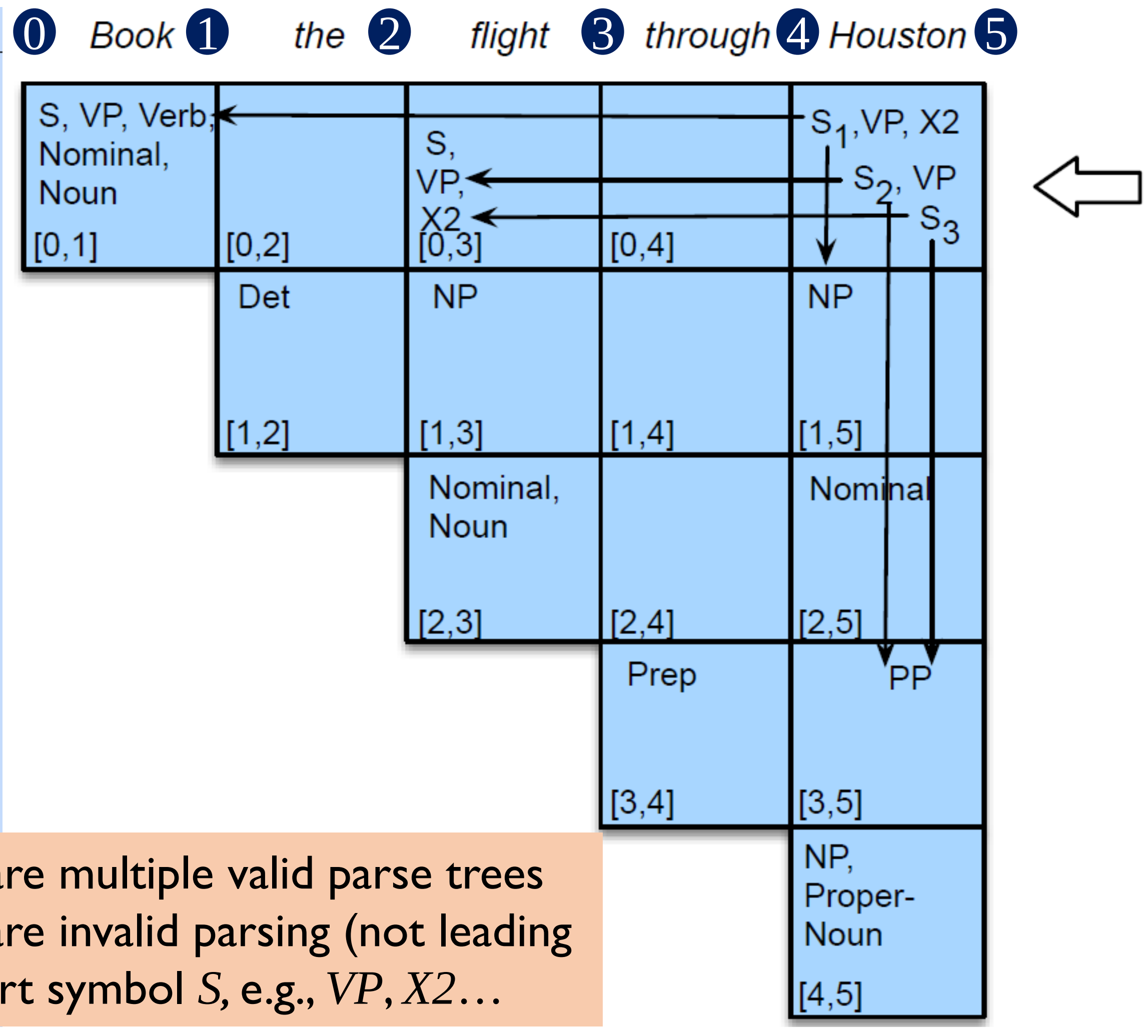


To assign cell [*1*, *5*], we check
[*1*, *4*] + [*4*, *5*]
[*1*, *3*] + [*3*, *5*]
[*1*, *2*] + [*2*, *5*]

The CKY algorithm, Cell [0, 5]

$\mathscr{L}_1$ in CNF

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
S_1 $\underline{S \rightarrow Verb\ NP}$
S_3 $\underline{S \rightarrow X2\ PP}$
$S \rightarrow Verb\ PP$
S_2 $\underline{S \rightarrow VP\ PP}$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$



There are multiple valid parse trees
There are invalid parsing (not leading to a start symbol S, e.g., VP, $X2$…

The CKY algorithm

- A parse tree can be derived by following backtrace pointers
  - There are possibly multiple valid parse trees for a sentence
  - Each cell in the parsing table may have multiple choices as well

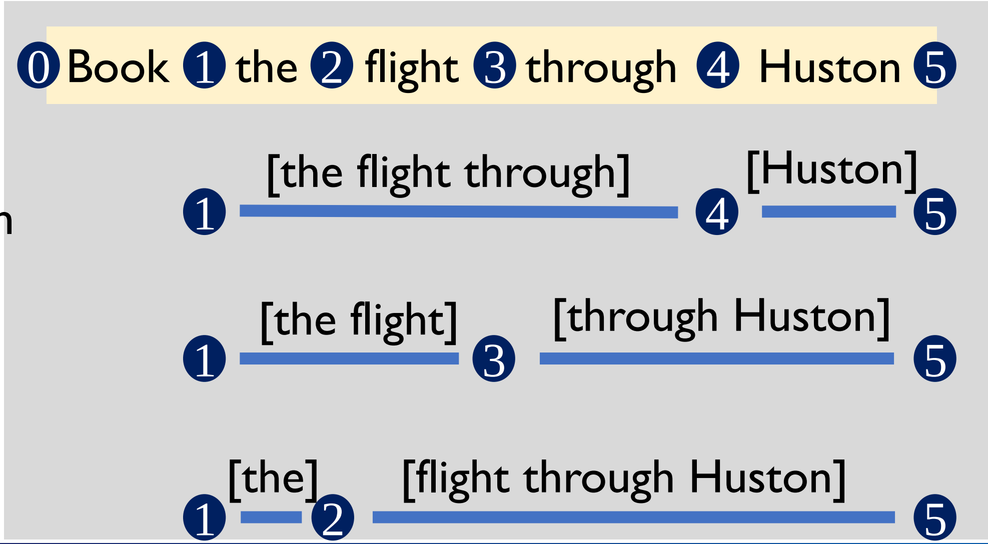
- There is a conversion process to map the tree to follow the original grammar, not the CNF version

- Returning every parse for a sentence may not be useful, since there may be an exponential number of parses
  - We need to retrieve only the best parse for a sentence

Extending the CKY algorithm

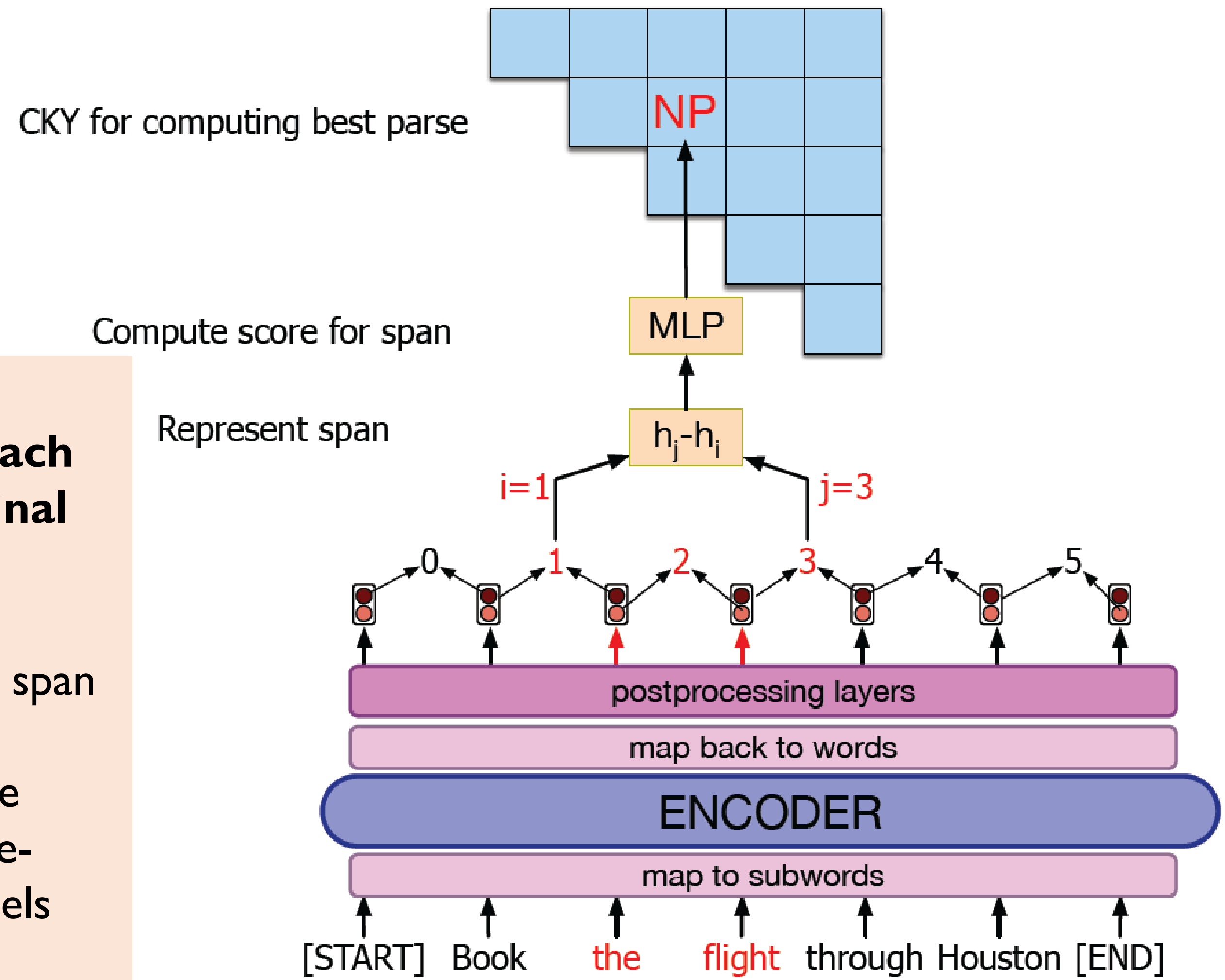
For Your Information

- CKY algorithm enumerates all the possible parse trees for a sentence,
  - It does not disambiguate among the possible parses
  - Which parse is the best parse?
- We introduce a simple neural extension of the CKY algorithm.
  - Known as span-based constituency parsing, or neural CKY
  - Train a neural classifier to assign a score to each constituent,
  - Then use a modified version of CKY to combine these constituent scores to find the best-scoring parse tree.
- Each cell corresponds a text span
  - We compute a score for each span with non-terminal symbol label l with a classifier
  - $score\ (i, j, l)$
  - E.g., $score\ (1, 3, NP)$



Computing span label scores

For Your Information



- As the classifier, MLP outputs a score for **each possible non-terminal**
- $h_j - h_i$ is the hidden representation of the span
- Subword encoding are commonly used in pre-trained language models like BERT

Integrating Span Scores into a Parse

- A parse tree T is represented as a set of such labeled spans
  - All spans in a T cover the whole input sentence, e.g., [0, 2, $L1$] [2, 5, $L2$], [5, 6, $L3$] or [0, 4, $L1$] [4, 6, $L2$], for a sentence with 5 words.
  - The best T is the parse tree with highest scores $s(T) = \sum_{(i,j,l)\in T} s(i,j,l)$

- A variant of the CKY algorithm to find the best parse
  - The score of the best subtree spanning (i, j) is $s_{best}(i, j)$
  - For span of length one: $s_{best}(i, i+1) = \max_{l} s(i, i+1, l)$
  - Other spans (i, j) is computed in recursive manner

$$s_{best}(i,j) = \max_{l} s(i,j,l) + \max_{k}[s_{best}(i,k) + s_{best}(k,j)]$$

- The parser is using the max label for span (i, j) plus the max labels for spans (i, k) and (k, j) **without** checking whether they are valid in grammar.
  - The neural model seems to learn these kinds of contextual constraints during its mapping from spans to non-terminals.

Evaluating Parsers

- The standard tool for evaluating parsers that assign a single parse tree to a sentence is the **PARSEVAL** metrics
  - The PARSEVAL metric measures how much the **constituents** (e.g., NP, VP, PP) in the hypothesis parse tree look like the constituents in a reference parse.
  - PARSEVAL thus requires a human-labeled reference (or "gold standard") parse tree for each sentence in the test set

- A **constituent** in a hypothesis parse C_h of a sentence s is labeled correct if there is a constituent in the reference parse C_r with the same **starting point**, **ending point**, and **non-terminal symbol**, e.g., "the flight" is a NP.

$$\textbf{labeled recall:} = \frac{\text{\# of correct constituents in hypothesis parse of } s}{\text{\# of correct constituents in reference parse of } s}$$

$$\textbf{labeled precision:} = \frac{\text{\# of correct constituents in hypothesis parse of } s}{\text{\# of total constituents in hypothesis parse of } s}$$

$$\mathrm{F}_1 = \frac{2PR}{P+R}$$

Partial or Shallow Parsing

- Many language processing tasks do not require complex, complete parse trees for all inputs.
  - A partial parse, or shallow parse, of input sentences may be sufficient, e.g., information extraction systems only need to identify and classify the segments in a text that are likely to contain valuable information.
  - Example partial parsing is **chunking**

[$_{NP}$ The morning flight] [$_{PP}$ from] [$_{NP}$ Denver] [$_{VP}$ has arrived.]

- Chunking is the process of identifying and classifying the *flat*, *non-overlapping segments* of a sentence that constitute the basic non-recursive phrases corresponding to: *noun phrases*, *verb phrases*, *adjective phrases*, and *prepositional phrases*.
  - **Segmenting:** finding the non-overlapping extents of the chunks
  - **Labeling**: assigning the correct tag to the discovered chunks
  - Chunking can be formulated as a **sequence labeling** tasks, with BIO tagging scheme.

Summary

- Structural ambiguity
- Parsing with CKY algorithm
- Evaluating parsers
- Partial or Shallow Parsing
- References
  - Chapter 13 https://web.stanford.edu/~jurafsky/slp3/

What can we do?

- Given a sentence, we can have its parse tree with the help from a parser
- We are able to traverse the parse tree to obtain various subtrees, corresponding to different segments of the sentence
- We can also compare the structural similarity between two sentences based on their parse trees.