NUS National University of Singapore | School of Computing

Trees

CS4248 Natural Language Processing

Week 09

Anab Maulana BARIK, Tianyang ZHANG and Min-Yen KAN

9

Recap of Week 08

Recurrent Neural Networks: Modeling sequences, NN style

Conditional Language Models: LMs with inputs

- Encoder–Decoder: When Conditional LMs are implemented NN style

Solving the encoding bottleneck: Attention Mechanism

Searching more effectively: Beam Search Decoding

Week 09 Agenda

Context-Free Grammar (CFG)

Chomsky Normal Form (CNF)

Syntactic Parsing

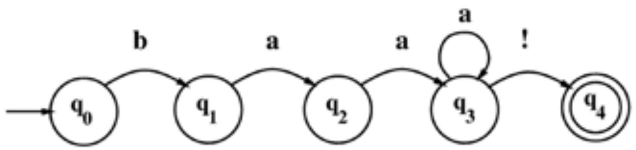
Statistical Parsing

Context Free Grammars (CFG)

Recap: Regular and Context Free Languages

From Week 02: Equivalence among Regex, Regular Languages and Finite State Automata (FSA).

A regex is an equivalent to an FSA.

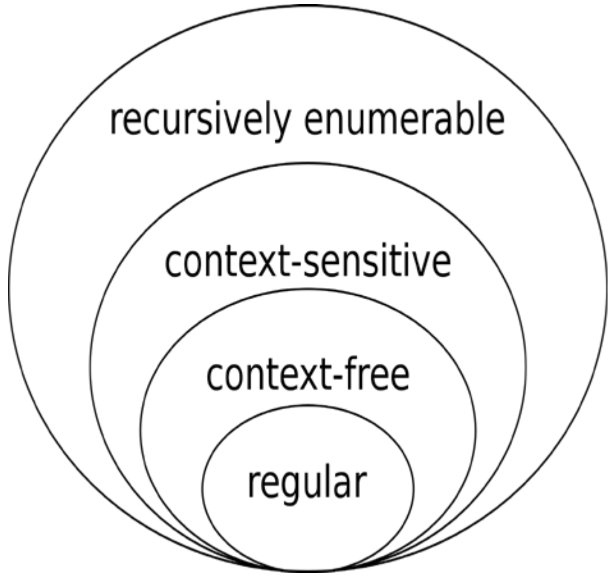


RE: /baa+!/

baa!
baaa!
baaaa!
baaaaa!
...

	Input		
State	b	a	!
0	1	∅	∅
1	∅	2	∅
2	∅	3	∅
3	∅	3	4
4:	∅	∅	∅

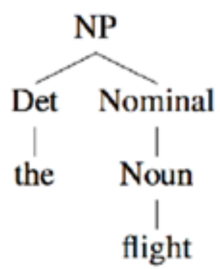
state-transition table



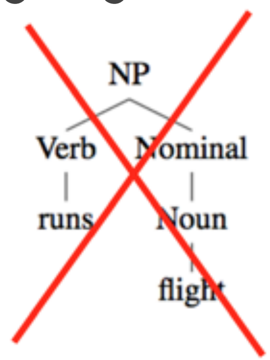
Slides adapted from Hwee Tou Ng (NUS). Picture from Wikipedia.

Context Free Grammar

A CFG gives a formal way to define what meaningful constituents are and exactly how a constituent is formed out of other constituents (or words).
It defines the valid structures in a language.



NP → Det Nominal



NP → Verb Nominal

Slide Credits: David Bamman (UCB)

Definition of CFG

A context-free grammar defines how symbols in a language combine to form valid structures

NP	→	Det Nominal
NP	→	ProperNoun
Nominal	→	Noun | Nominal Noun
Det	→	a | the
Noun	→	flight

non-terminals

lexicon/ terminals

Slide Credits: David Bamman (UCB)

Definition of CFG

N	Finite set of non-terminal symbols	NP, VP, S
Σ	Finite alphabet of terminal symbols	the, dog, a
R	Set of production rules, each $A \rightarrow \beta$ $\beta \in (\Sigma, N)$	S → NP VP Noun → dog
S	Start symbol	

What part of this specification makes this context-free?

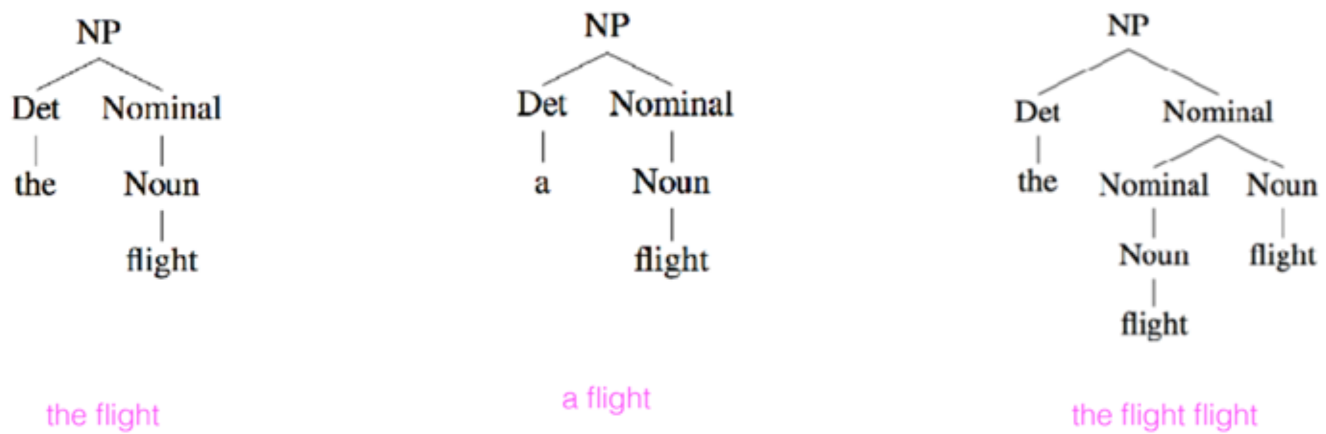
Slide Credits: David Bamman (UCB)

Infinite strings from finite productions

- *This is the house*
- *This is the house that Jack built*
- *This is the cat that lives in the house that Jack built*
- *This is the dog that chased the cat that lives in the house that Jack built*
- *This is the flea that bit the dog that chased the cat that lives in the house the Jack built*
- *This is the virus that infected the flea that bit the dog that chased the cat that lives in the house that Jack built*

Definition of Derivation

Given a CFG, a derivation is the sequence of productions used to generate a string of words (e.g., a sentence), often visualized as a parse tree.



Slide Credits: David Bamman (UCB)

Constituents

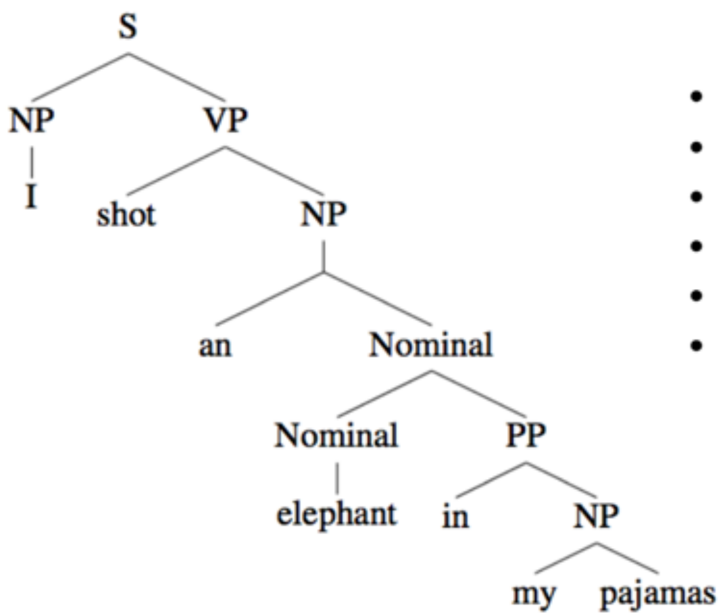
Constituents are group of words that behave as a single unit.

Linguists characterize constituents in a number of ways, including:

- where they occur (e.g., “NPs can occur before verbs”)
- where they can move in variations of a sentence
  - On March 19th, I’d like to fly from Atlanta to Denver
  - I’d like to fly on March 19th from Atlanta to Denver
  - I’d like to fly from Atlanta to Denver on March 19th
- what parts can move and what parts can’t
  - X On March I’d like to fly 19th from Atlanta to Denver
- what they can be conjoined with
  - I’d like to fly from Atlanta to Denver on March 17th and in the morning

Adapted from Dan Jurafsky (Stanford)

Definition of Constituents



Every internal node is a phrase

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

Each phrase could be replaced by another of the same type of constituent

Slide Credits: David Bamman (UCB)

Ambiguity, Revisited

Why is ambiguity a problem in NLP?

Picture Credits: https://examples.yourdictionary.com/reference/examples/examples-of-ambiguity.html

Ambiguity

There are multiple ways to interpret a sentence

Structural Ambiguity: When a grammar can assign more than one parse to a sentence.

Ambiguity

There are multiple ways to interpret a sentence

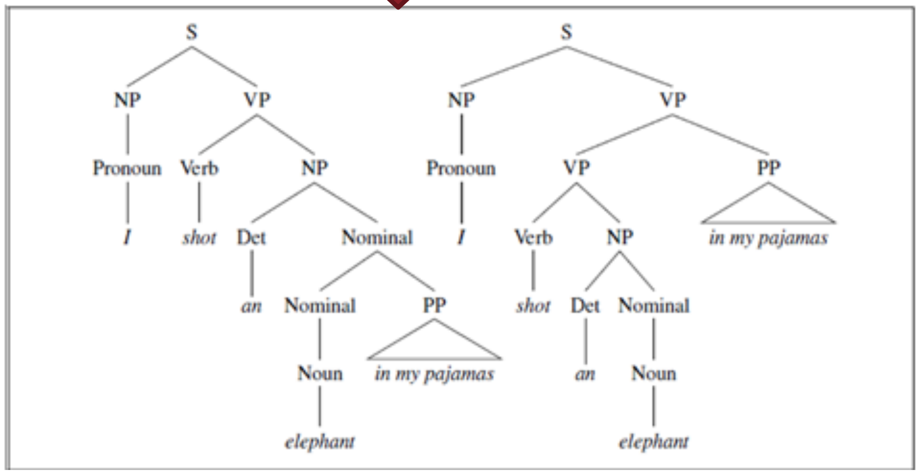
Structural Ambiguity: When a grammar can assign more than one parse to a sentence.

Why would this become a problem?

Structural Ambiguity

Which one is correct?

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid the \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	



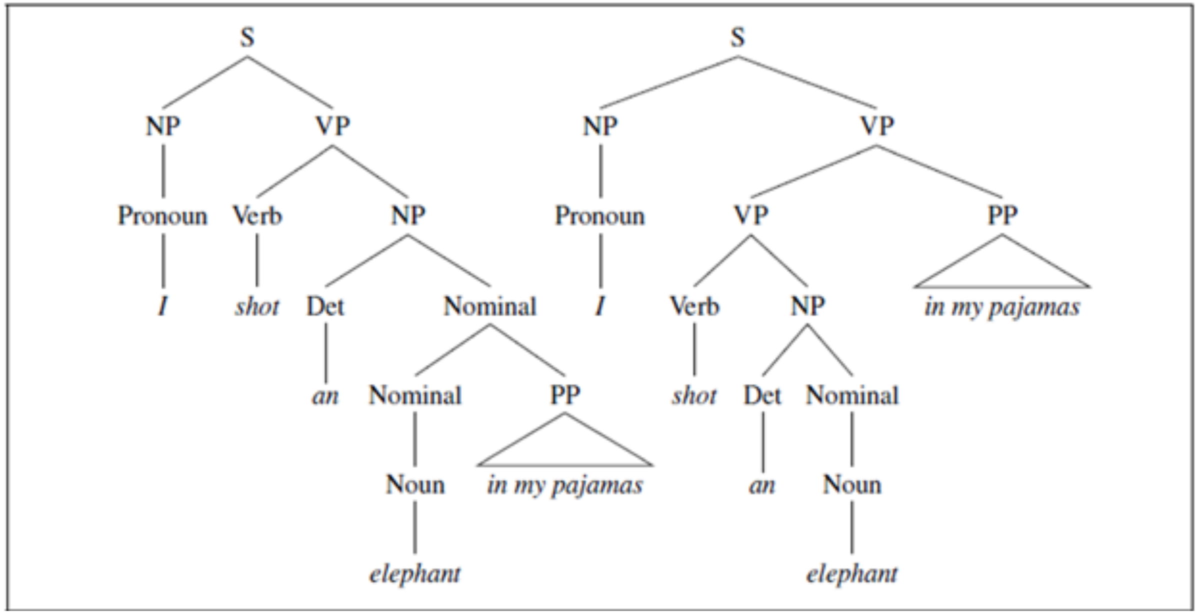
Slide adapted from Dan Jurafsky (Stanford)

Structural Ambiguity

Two common kinds of structural ambiguity

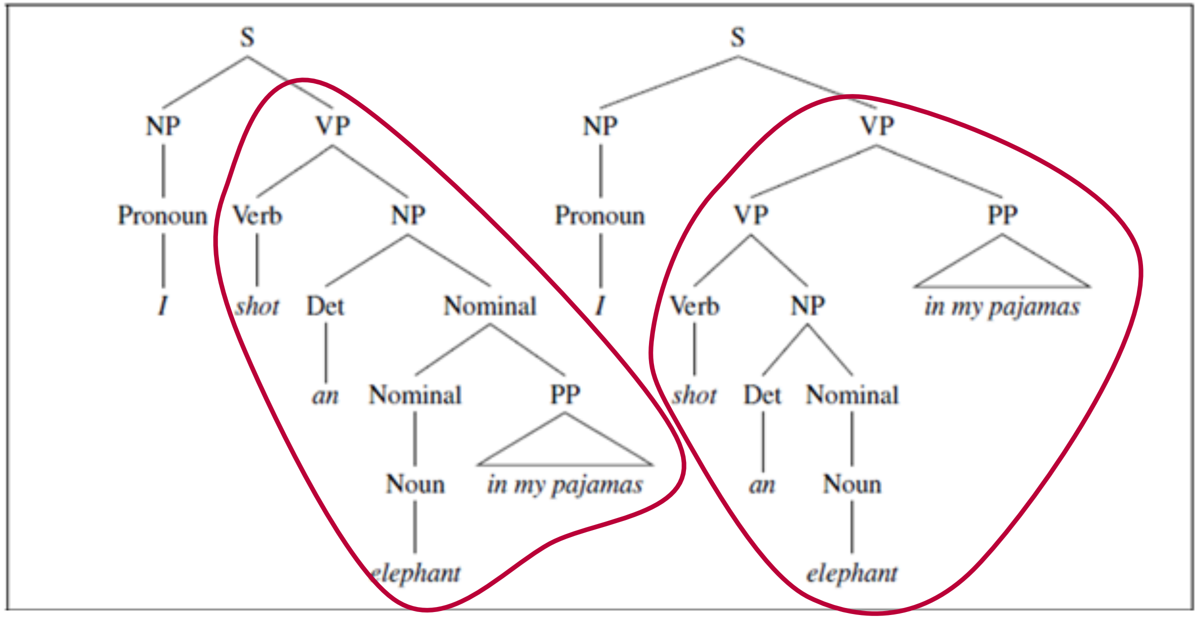
1. Attachment Ambiguity
   - a particular constituent can be attached to the parse tree at more than one place
2. Coordination Ambiguity
   - phrases can be conjoined by a conjunction like *"and"*

Attachment Ambiguity



Slide adapted from Dan Jurafsky (Stanford)

Attachment Ambiguity Cont.



Relating "*in my pajamas*" to "*an elephant*" or modifying "*shot*"

Slide adapted from Dan Jurafsky (Stanford)

How many interpretations?

"the best roti prata and laksa"

Slide adapted from Dan Jurafsky (Stanford)

Coordination Ambiguity

"the best roti prata and laksa"

1. [the best [[roti prata] and [laksa]]
2. [[the best [roti prata]] and [laksa]]

Slide adapted from Dan Jurafsky (Stanford)

Solving Ambiguity

The fact that there are many grammatically correct parses but which are unreasonable semantically becomes a problem.

How can we solve this problem?

Slide adapted from Dan Jurafsky (Stanford)

Solving Ambiguity

The fact that there are many grammatically correct parses but which are unreasonable semantically becomes a problem.

How can we solve this problem?

1. Syntactic Parsing: Extract all possible parses for a sentence
2. Syntactic Disambiguation: Score all parses and return the best parse

Slide adapted from Dan Jurafsky (Stanford)

How do we evaluate parses?

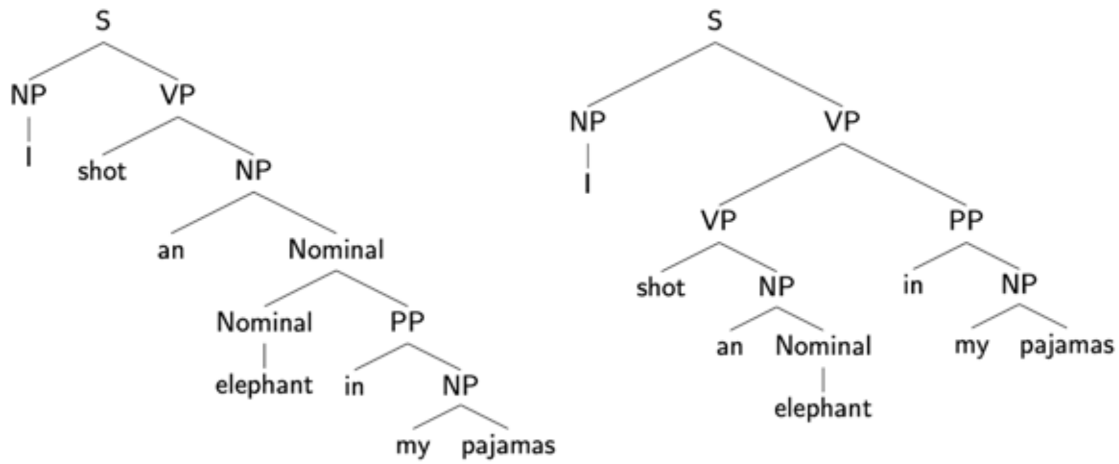
Represent a parse tree as a collection of tuples $\langle\langle l_1, i_1, j_1\rangle,$ $\langle l_2, i_2, j_2\rangle, \ldots, \langle l_n, i_n, j_n\rangle$, where

- l_k is the non-terminal labeling the k^{th} phrase
- i_k is the index of the first word in the k^{th} phrase
- j_k is the index of the last word in the k^{th} phrase

Convert gold-standard tree and system hypothesized tree into this representation, then estimate precision, recall, and F1.

Slide adapted from Dan Jurafsky (Stanford)

Evaluation Example



$$\underbrace{\left\langle \begin{array}{c} \langle \mathsf{NP}, 3, 7\rangle, \\ \langle \mathsf{Nominal}, 4, 7\rangle \end{array} \right\rangle}_{\text{only in left tree}} \underbrace{\left\langle \begin{array}{c} \langle \mathsf{NP}, 1, 1\rangle \\ \langle \mathsf{S}, 1, 7\rangle, \langle \mathsf{VP}, 2, 7\rangle, \\ \langle \mathsf{PP}, 5, 7\rangle, \langle \mathsf{NP}, 6, 7\rangle \\ \langle \mathsf{Nominal}, 4, 4\rangle \end{array} \right\rangle}_{\text{in both trees}} \underbrace{\left\langle \begin{array}{c} \langle \mathsf{VP}, 2, 4\rangle, \\ \langle \mathsf{NP}, 3, 4\rangle \end{array} \right\rangle}_{\text{only in right tree}}$$

Slide adapted from Dan Jurafsky (Stanford)

Chomsky Normal Form

Normalizing Context-Free Grammars

Slide Credits: David Bamman (UCB)

Context Free Grammar

N	Finite set of non-terminal symbols	NP, VP, S
Σ	Finite alphabet of terminal symbols	the, dog, a
R	Set of production rules, each $A \rightarrow \beta$ $\beta \in (\Sigma, N)$	NP → DT JJ NN Noun → dog
S	Start symbol	

Slide Credits: David Bamman (UCB)

Chomsky Normal Form

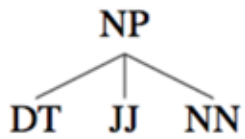
N	Finite set of non-terminal symbols	NP, VP, S
Σ	Finite alphabet of terminal symbols	the, dog, a
R	Set of production rules, each $A \rightarrow \beta$ β = single terminal (from Σ) or two non-terminals (from N)	$S \rightarrow NP\ VP$ $Noun \rightarrow dog$
S	Start symbol	

Slide Credits: David Bamman (UCB)

CNF and CFG

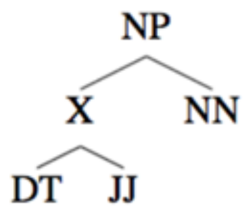
Any CFG can be converted into a weakly equivalent CNF grammar (recognizing the same set of sentences as in the original grammar *but* differing in their derivation).

NP → DT JJ NN



NP → X NN

X → DT JJ



Slide Credits: David Bamman (UCB)

CFG to CNF: CFG

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	NN
Nominal	→	NNS
Nominal	→	PRP
PP	→	IN NP
NP	→	DT NN
NP	→	Nominal
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
NN	→	elephant
NNS	→	pajamas
PRP	→	I
PRP$	→	my
IN	→	in

I shot an elephant in my pajamas

Slide Credits: David Bamman (UCB)

CFG to CNF: CNF

NUS National University of Singapore | School of Computing

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I shot an elephant in my pajamas

Slide Credits: David Bamman (UCB)

Syntactic Parsing

Cocke–Younger–Kasami (CYK) Chart Parsing

a.k.a.

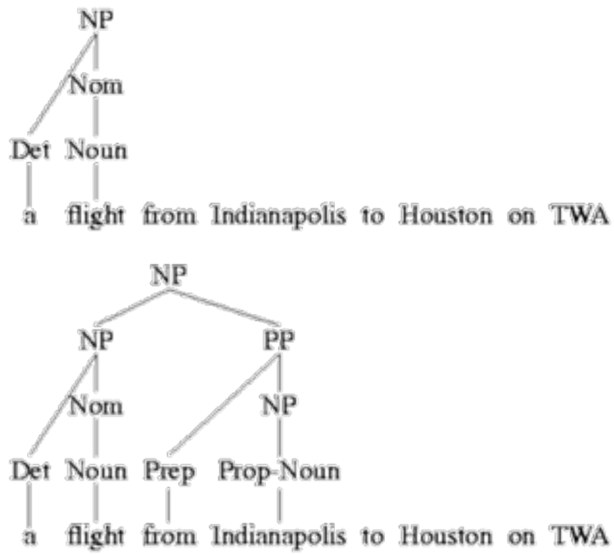
Dynamic Programming, 3rd encounter

Parsing

NUS National University of Singapore | School of Computing

One means to extract a sentential parse tree is by repeatedly parsing the sentence, left to right.

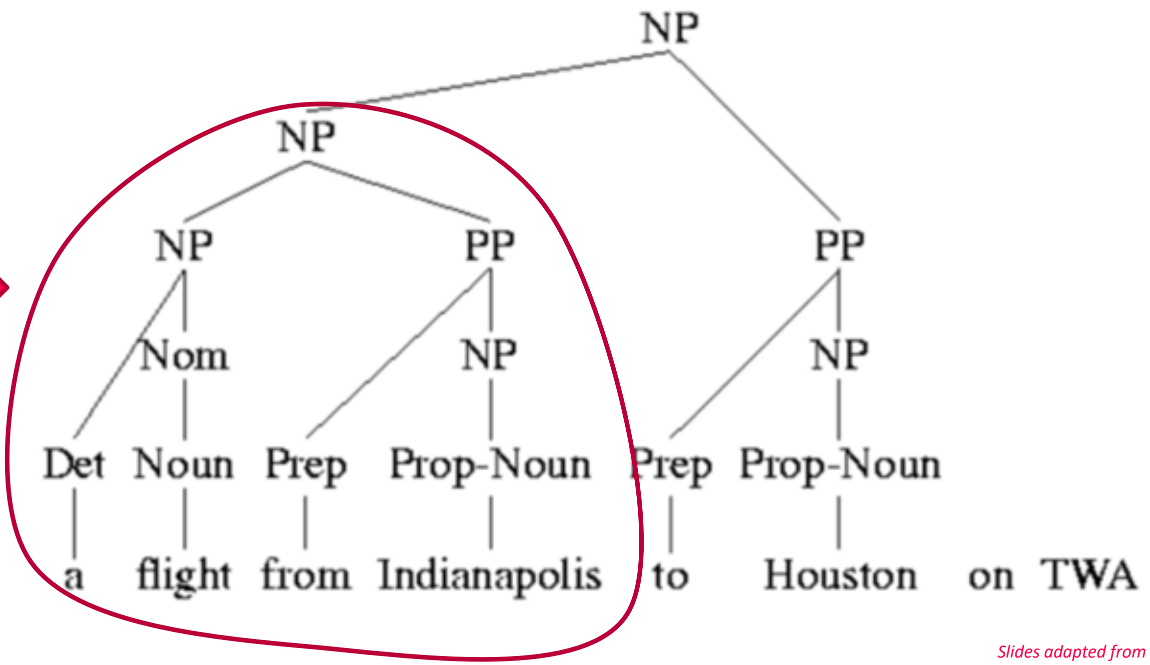
Any problems?



Slides adapted from Prof. Hwee Tou Ng (NUS)

Parsing of subtrees

Recalculation of the same subtrees from the previous steps

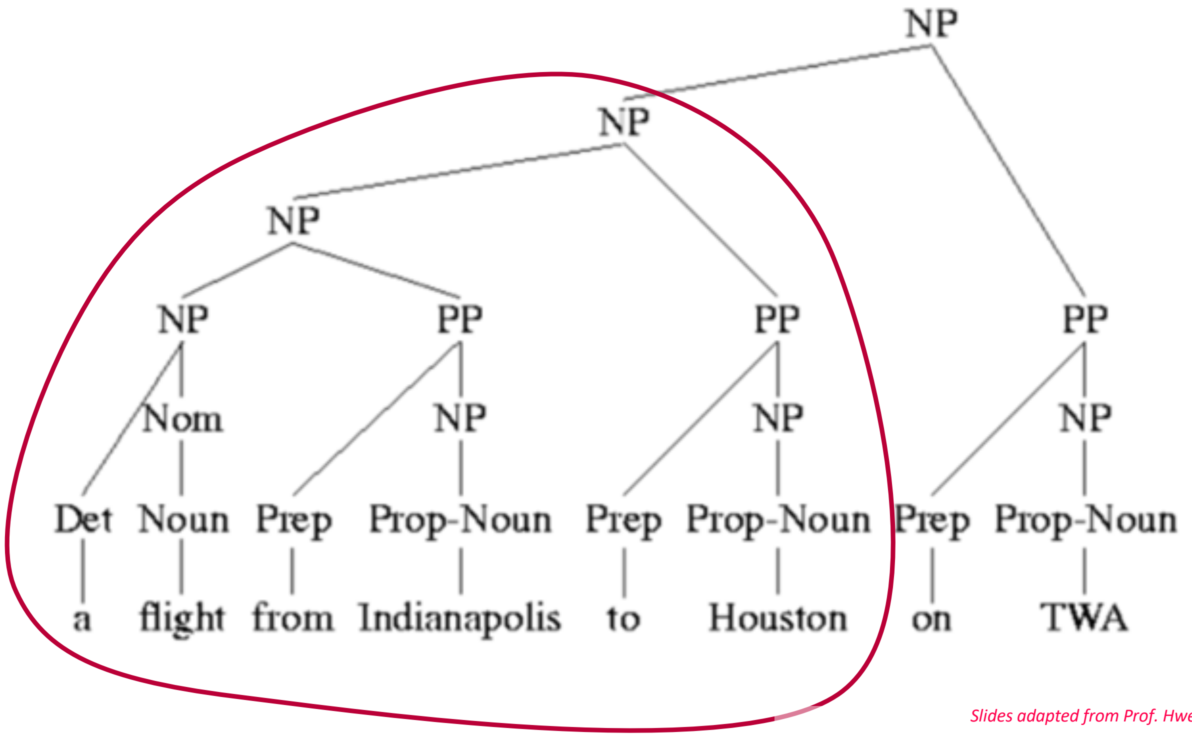


Slides adapted from Prof. Hwee Tou Ng (NUS)

Parsing of subtrees

Recalculate the same subtrees from the previous steps.

Need a more efficient way!



Slides adapted from Prof. Hwee Tou Ng (NUS)

CYK Parsing

Stands for its 3 inventors: Cocke–Younger–Kasami (CYK)

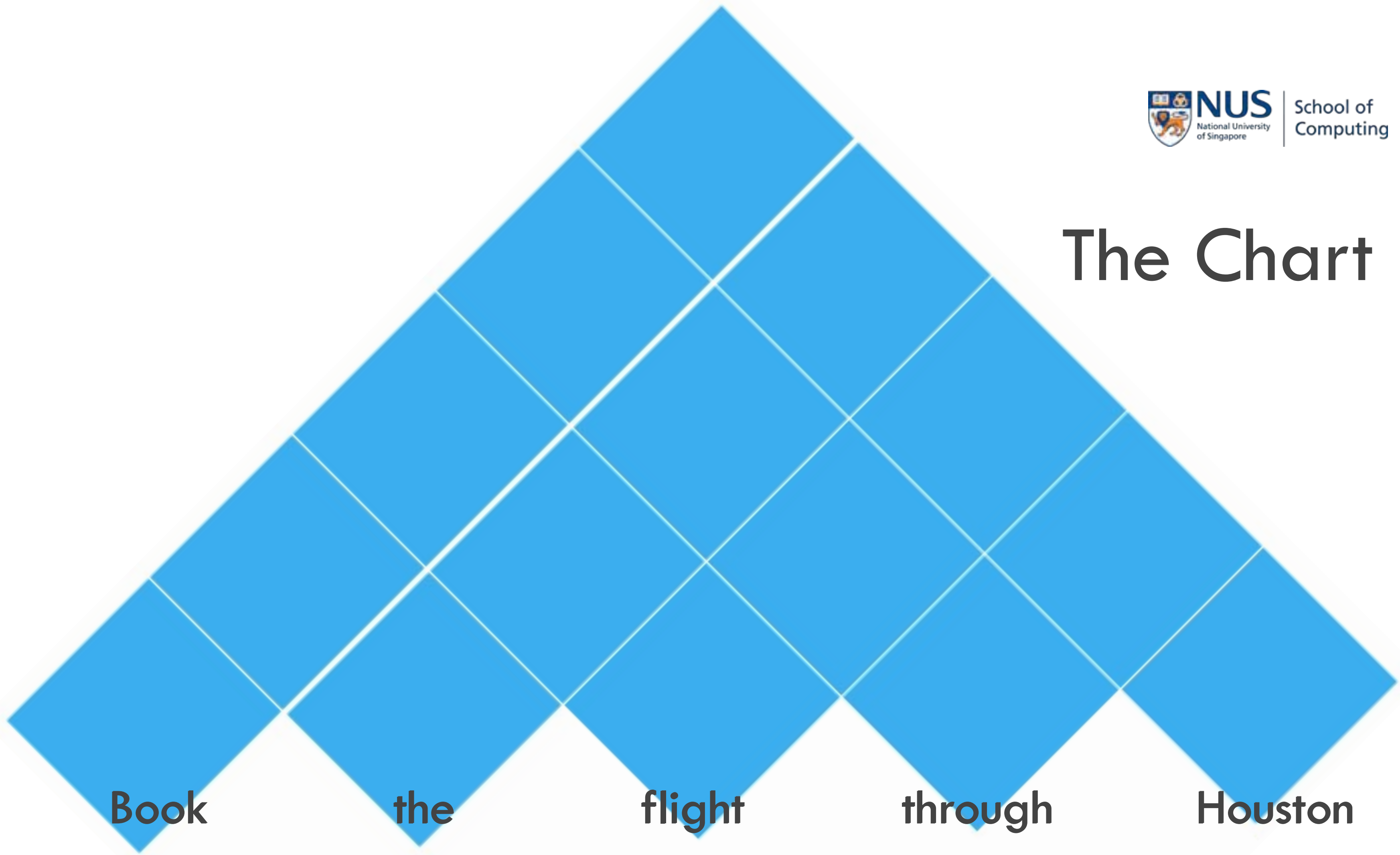
Bottom-Up Dynamic Programming approach to handling redundancy when computing the parse trees.

It requires the CFG to be in CNF:

- The grammar is ε (epsilon)-free
- Either 2 non-terminal symbols or 1 terminal symbol on the RHS

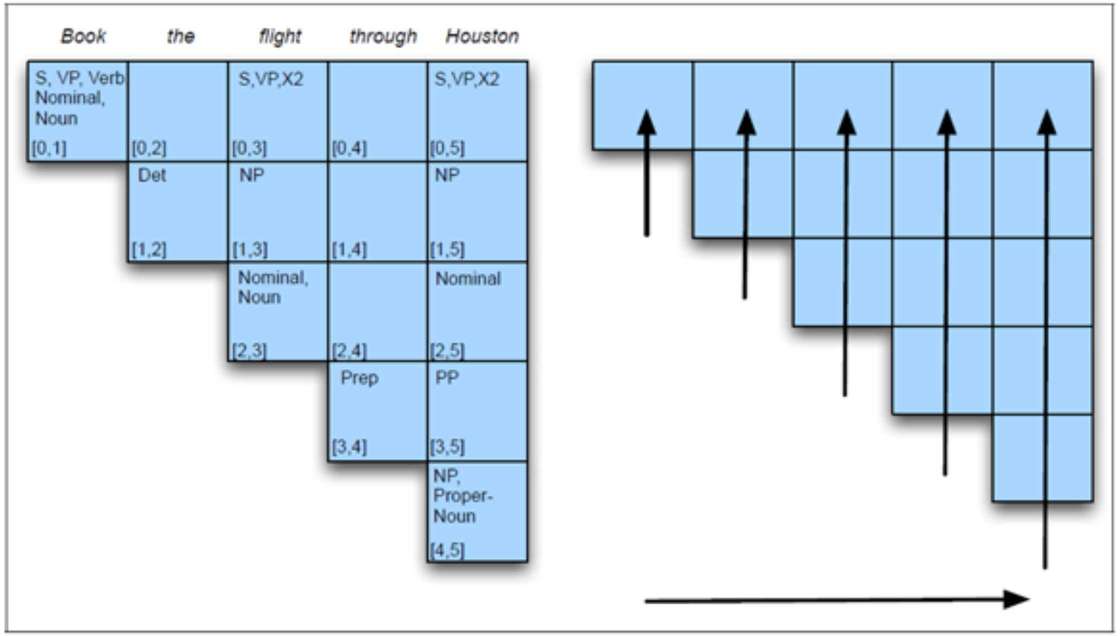
Slides adapted from Prof. Hwee Tou Ng (NUS)

The Chart



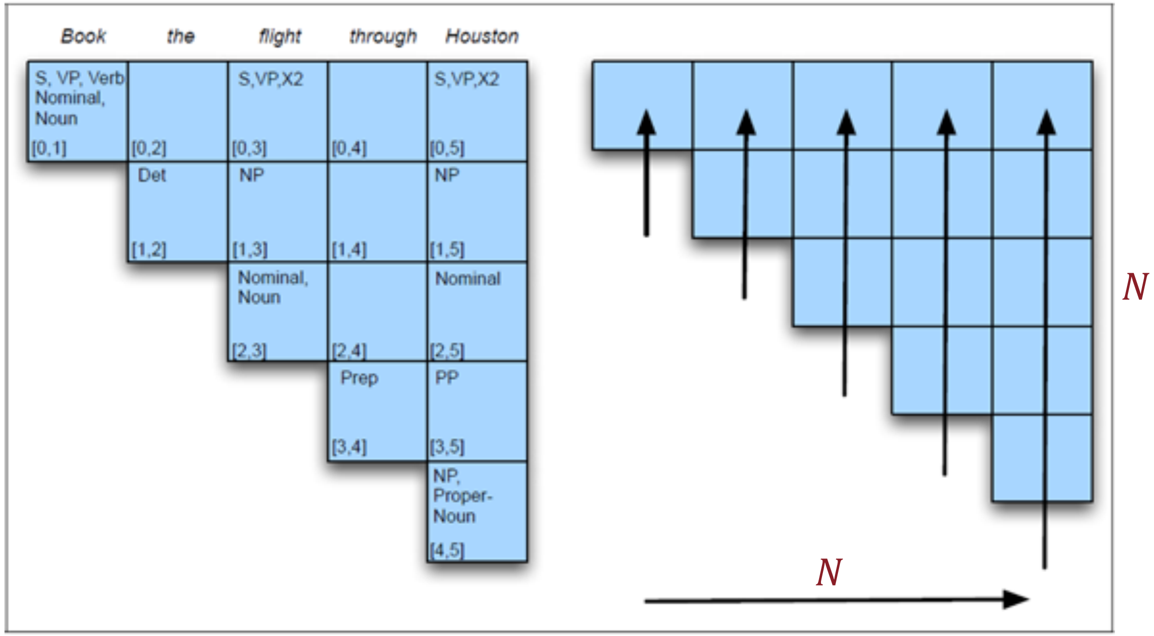
CYK Parsing

Completing the parse table in a bottom up manner



CYK Parsing

Completing the parse table in a bottom up manner

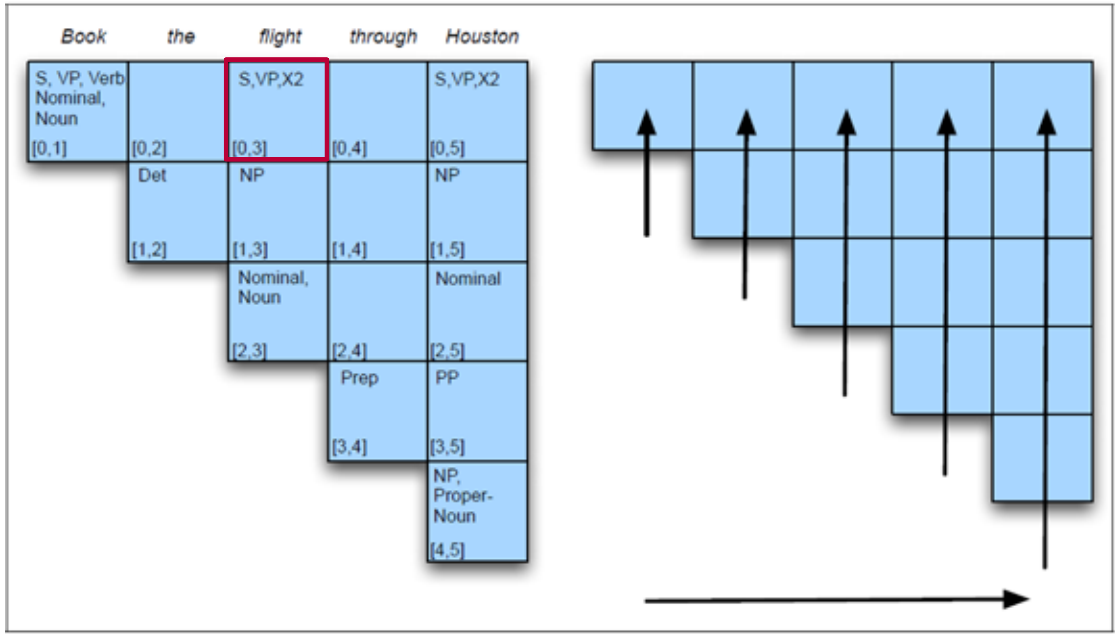


Size $= N \times N$

N = length of the sentence

CYK Parsing

Completing the parse table in a bottom up manner

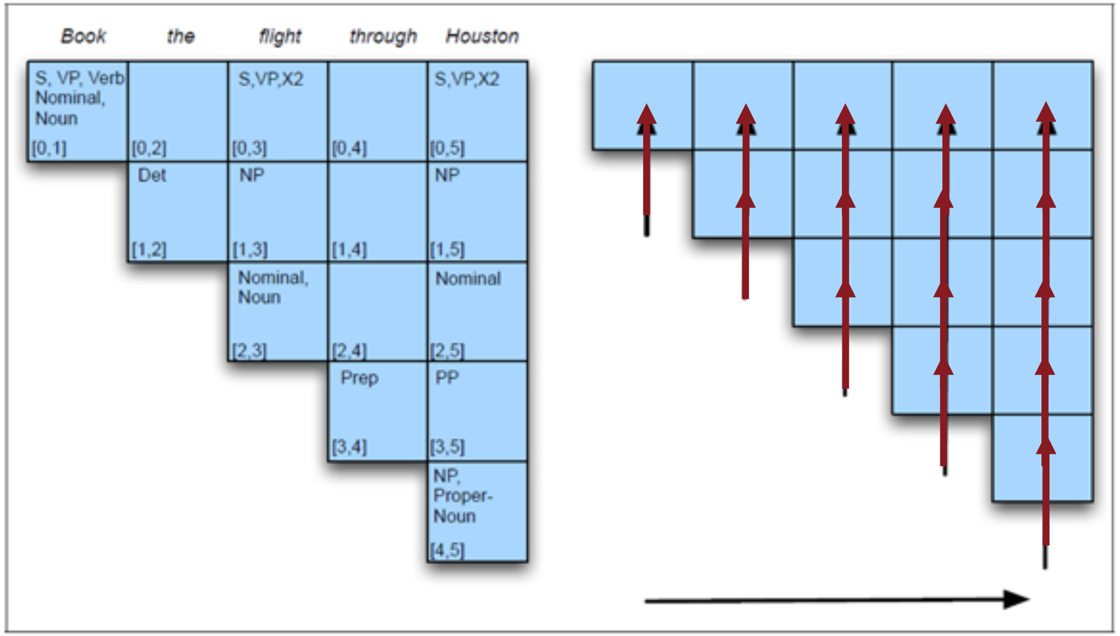


Each cell represent all the possible parses for span [i, j) (using a 0 index).

E.g. [0,3] = all the possible parses for span [0, 3) “book the flight”

CYK Parsing

Completing the parse table in a bottom up manner

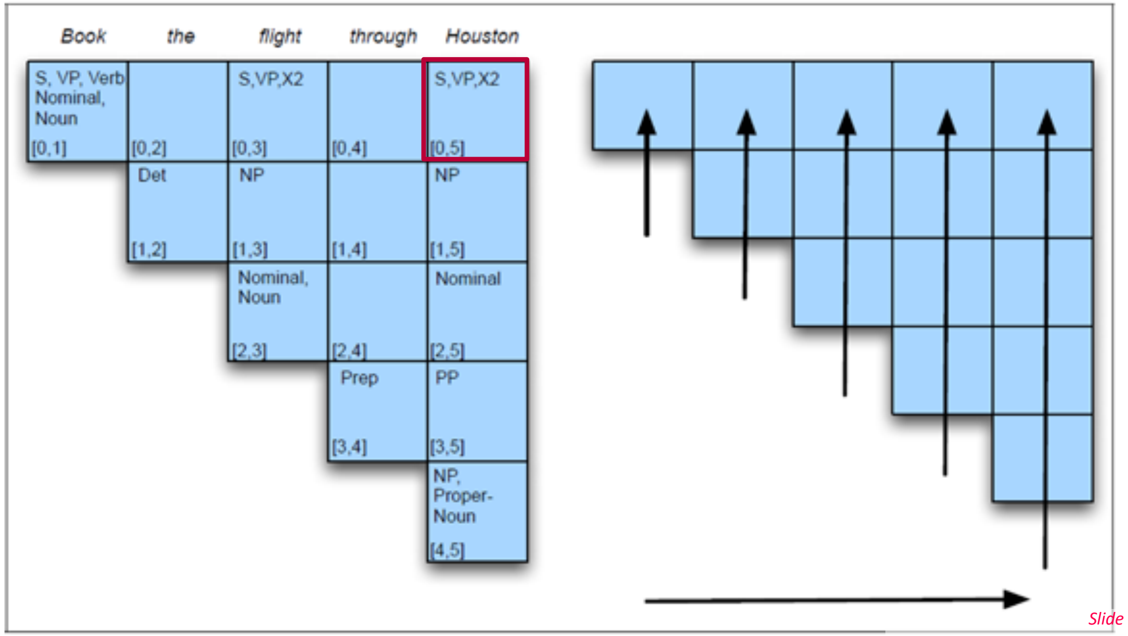


CYK fills the table from the bottom to top, from left to right.

Starts from [4,5], then followed by [3, 4], [3, 5] until [0,5]

CYK Parsing

Completing the parse table in bottom up-manner.



The result = all the possible parses are stored at cell $[0, N]$

Slide adapted from Prof. Hwee Tou Ng (NUS)

CYK Algorithm

```
function CKY-PARSE(words, grammar) returns table

  for j ← from 1 to LENGTH(words) do
      for all {A | A → words[j] ∈ grammar}
            table[j−1, j] ← table[j−1, j] ∪ A
      for i ← from j−2 down to 0 do
          for k ← i+1 to j−1 do
              for all {A | A → BC ∈ grammar and B ∈ table[i,k] and C ∈ table[k,j]}
                    table[i,j] ← table[i,j] ∪ A
```

CYK Algorithm Cont.

function CKY-PARSE(*words, grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**

$\boxed{\textbf{for all } \{A \mid A \rightarrow words[j] \in grammar\} \quad table[j-1, j] \leftarrow table[j-1, j] \cup A}$

Given the CFG:

Nominal → *flight*

Noun → *flight*

Base cases: filling the cells by adding all the possible parses for each word.

Nominal, Noun

[2,3]

Book the flight through Houston

CYK Algorithm Cont.

Recursive cases: fill all the other cells by adding all the possible combinations of 2 subtrees.

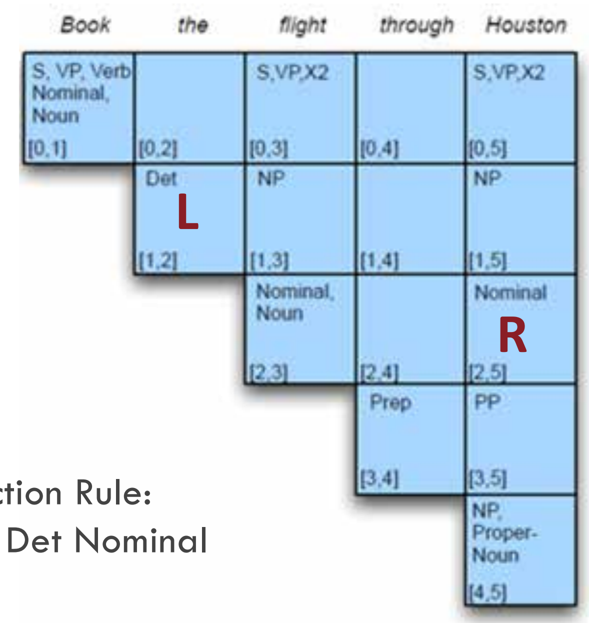
for $i \leftarrow$ **from** $j-2$ **down to** 0 **do**

  for $k \leftarrow i+1$ **to** $j-1$ **do**

    for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$

      $table[i,j] \leftarrow table[i,j] \cup A$

CYK Algorithm (Filling [1,5], Try 1)



Production Rule:
NP → Det Nominal

[1,5] = "*the flight through Houston*"

Combine "*the*" with "*flight through Houston*"

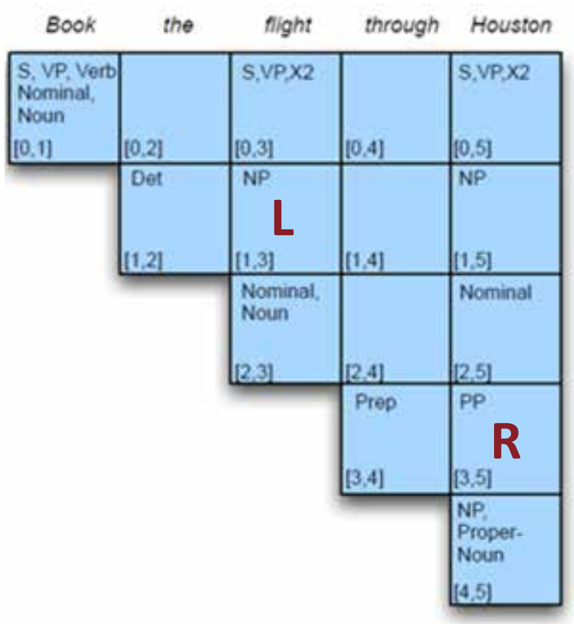
[1,2] = [Det]

[2,5] = [Nominal]

= [NP]

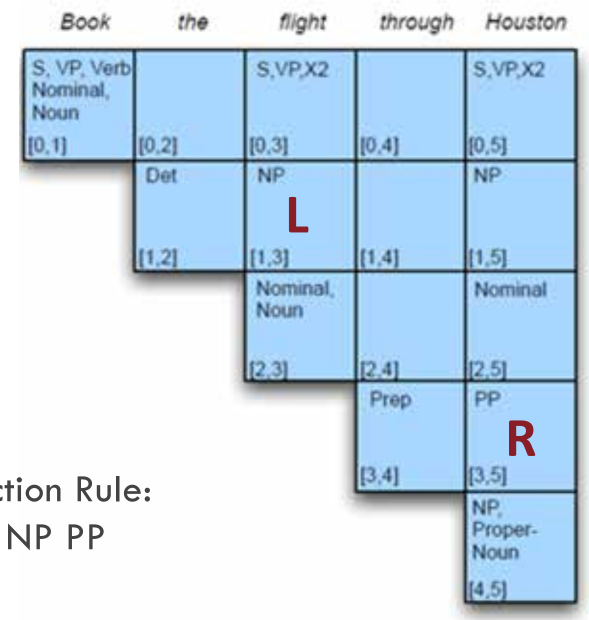
Slides adapted from Prof. Hwee Tou Ng (NUS)

CYK Algorithm (Filling [1,5], Try 2)



Slides adapted from Prof. Hwee Tou Ng (NUS)

CYK Algorithm (Filling [1,5], Try 2)



Production Rule:
NP → NP PP

[1,5] = "*the flight through Houston*"

Combine "*the flight*" with "*through Houston*"

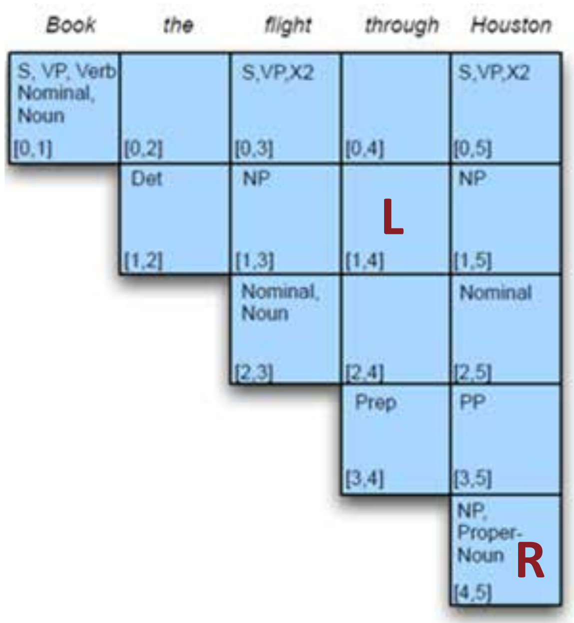
[1,3] = [NP]

[3,5] = [PP]

= [NP]

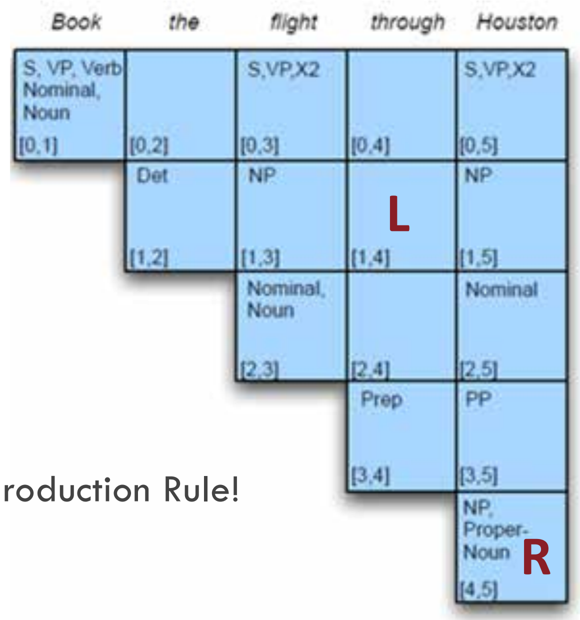
Slides adapted from Prof. Hwee Tou Ng (NUS)

CYK Algorithm (Filling [1,5], Try 3)



Slides adapted from Prof. Hwee Tou Ng (NUS)

CYK Algorithm (Filling [1,5], Try 3)



X No Production Rule!

[1,5] = "*the flight through Houston*"

Combine "*the flight through*" with "*Houston*"

[1,4] = []

[4,5] = [NP, Proper-Noun]

= []

Slide adapted from Prof. Hwee Tou Ng (NUS)

CYK Summary

Another encounter with dynamic programming to reuse computations.

Cell [0,N] contains all the possible parses for a given input

While resultant parses are grammatical, some are unlikely

We want to know how to get the most reasonable parses

Statistical Parsing

Upgrading CKY to account for probable parses

Statistical Parsing

Resolve structural ambiguity by choosing the most probable parse

Definition of PCFG

Probabilistic context-free grammar: each production is also associated with a probability.

for a given parse tree T for sentence S comprised of n rules from R (each $A \rightarrow \beta$):

$$P(T, S) = \prod_{i}^{n} P(\beta | A)$$

Slide Credits: David Bamman (UCB)

Definition of PCFG

N	Finite set of non-terminal symbols	NP, VP, S
Σ	Finite alphabet of terminal symbols	the, dog, a
R	Set of production rules, each $A \rightarrow \beta$ [p] $p = P(\beta \mid A)$	$S \rightarrow NP\ VP$ $Noun \rightarrow dog$
S	Start symbol	

Slide Credits: David Bamman (UCB)

CFG and PCFG

A CFG tells us whether a sentence is in the language it defines.

A PCFG gives us a mechanism for assigning scores (here, probabilities) to different parses for the same sentence.

Slide Credits: David Bamman (UCB)

Probabilistic Context-Free Grammar

The sum of the probabilities with the same LHS rule **must** equal to unity.

$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.05] | the [.80] | a	[.15]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$	[.10]
$S \rightarrow VP$	[.05]	$Noun \rightarrow flights$	[.50]
$NP \rightarrow Det\ Nom$	[.20]	$Noun \rightarrow meal$	[.40]
$NP \rightarrow Proper\text{-}Noun$	[.35]	$Verb \rightarrow book$	[.30]
$NP \rightarrow Nom$	[.05]	$Verb \rightarrow include$	[.30]
$NP \rightarrow Pronoun$	[.40]	$Verb \rightarrow want$	[.40]
$Nom \rightarrow Noun$	[.75]	$Aux \rightarrow can$	[.40]
$Nom \rightarrow Noun\ Nom$	[.20]	$Aux \rightarrow does$	[.30]
$Nom \rightarrow Proper\text{-}Noun\ Nom$	[.05]	$Aux \rightarrow do$	[.30]
$VP \rightarrow Verb$	[.55]	$Proper\text{-}Noun \rightarrow TWA$	[.40]
$VP \rightarrow Verb\ NP$	[.40]	$Proper\text{-}Noun \rightarrow Denver$	[~~.40~~] .60
$VP \rightarrow Verb\ NP\ NP$	[.05]	$Pronoun \rightarrow you$ [.40] | I [.60]	

Slides adapted from Prof. Hwee Tou Ng (NUS)

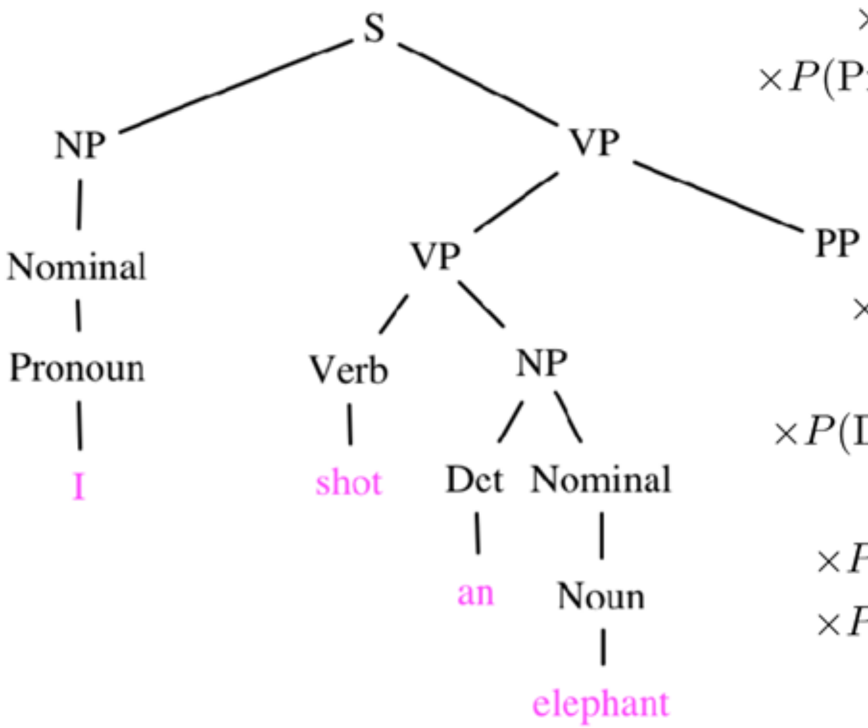
Estimating PCFG Probabilities

$$\sum_{\beta} P(A \rightarrow \beta) = 1 \quad \Rightarrow \quad \sum_{\beta} P(\beta | A) = 1$$

$$\sum_{\beta} P(\beta | A) = \frac{C(A \rightarrow \beta)}{\sum_{\gamma} C(A \rightarrow \gamma)} \quad \Rightarrow \quad \sum_{\beta} P(\beta | A) = \frac{C(A \rightarrow \beta)}{C(A)}$$

Slide Credits: David Bamman (UCB)

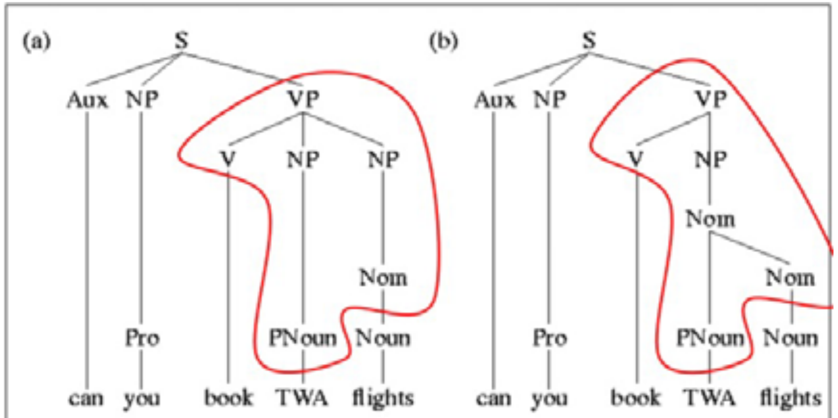
PCFG Example



$$
\begin{aligned}
&P(\text{NP VP} \mid \text{S}) \\
&\times P(\text{Nominal} \mid \text{NP}) \\
&\times P(\text{Pronoun} \mid \text{Nominal}) \\
&\times P(\text{I} \mid \text{Pronoun}) \\
&\times P(\text{VP PP} \mid \text{VP}) \\
&\times P(\text{Verb NP} \mid \text{VP}) \\
&\times P(\text{shot} \mid \text{Verb}) \\
&\times P(\text{Det Nominal} \mid \text{NP}) \\
&\times P(\text{an} \mid \text{Det}) \\
&\times P(\text{Noun} \mid \text{Nominal}) \\
&\times P(\text{elephant} \mid \text{Noun})
\end{aligned}
$$

Slide Credits: David Bamman (UCB)

Comparing Possible Parses



Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP NP	.05
NP	→	Nom	.05
NP	→	PNoun	.35
Nom	→	Noun	.75
Aux	→	Can	.40
~~NP~~	~~→~~	~~Pro~~	~~.40~~
Pro	→	you	.40
Verb	→	book	.30
PNoun	→	TWA	.40
Noun	→	flights	.50

Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP	.40
NP	→	Nom	.05
Nom	→	PNoun Nom	.05
Nom	→	Noun	.75
Aux	→	Can	.40
~~NP~~	~~→~~	~~Pro~~	~~.40~~
Pro	→	you	.40
Verb	→	book	.30
Pnoun	→	TWA	.40
Noun	→	flights	.50

$$P(T) = \prod_{n \in T} P(r(n))$$

$$P(T_a) \\ = 0.15 * 0.4 * 0.05 * 0.05 * 0.35 * 0.75 \\ * 0.40 * 0.40 * 0.30 * 0.40 * 0.5 \\ = 3.78 * 10^{-7}$$

$$P(T_b) \\ = 0.15 * 0.4 * 0.4 * 0.05 * 0.05 * 0.75 \\ * 0.40 * 0.40 * 0.30 * 0.40 * 0.5 \\ = 4.32 * 10^{-7}$$

Slides adapted from Prof. Hwee Tou Ng (NUS)

Probabilistic CYK parsing

Standard CYK

- $Table[i,j]$ = all possible parses for span $[i,j)$

- $Table[0,N]$ = all possible parses for the given input sentence

Probabilistic CYK

- $Table[i,j,A]$ = the highest score for span $[i,j)$ resulting in constituent A

- $Table[0,N,S]$ = the highest score for a parse for the given input sentence

Slides adapted from Prof. Hwee Tou Ng (NUS)

Probabilistic CYK parsing

```
function Probabilistic-CKY(words, grammar)
returns most probable parse and its probability

for j ← from 1 to Length(words) do
   for all { A | A → words[j]  ∈ grammar }
      table[j – 1, j, A] ← P(A → words[j])
   for i ← from j – 2 downto 0 do
      for k ← i + 1 to j – 1 do
         for all { A | A → BC ∈ grammar and
                           table[i, k, B] > 0 and table[k, j, C] > 0 }
            if (table[i, j, A]  < P(A → BC) × table[i, k, B] × table[k, j, C]) then
               table[i, j, A] ← P(A → BC) × table[i, k, B] × table[k, j, C]
               back[i, j, A] ← { k, B, C }
return Build-Tree(back[0, Length(words), S]),  table[0, Length(words), S]
```

Slides adapted from Prof. Hwee Tou Ng (NUS)

Probabilistic CYK parsing

function Probabilistic-CKY(words, grammar)
returns most probable parse and its probability

for j ← **from** 1 **to** Length(words) **do**
  for all { A | A → words[j] ∈ grammar }
    $\boxed{\text{table}[j-1, j, A] \leftarrow P(A \rightarrow \text{words}[j])}$

Base cases: filling the cells by assigning with the probability based on a particular rule

Given CFG

Noun → *flight* (0.5)

[2, 3, Noun] = 0,5

Slides adapted from Prof. Hwee Tou Ng (NUS)

Probabilistic CYK parsing

A. Recursive case: assigning
each cells with the highest score

B. Backtracking purposes:
storing the best parses

```
for i ← from j – 2 downto 0 do
  for k ← i + 1 to j – 1 do
    for all { A | A → BC ∈ grammar and
                    table[i, k, B] > 0 and table[k, j, C] > 0 }
      if (table[i, j, A]  < P(A → BC) × table[i, k, B] × table[k, j, C]) then
        table[i, j, A] ← P(A → BC) × table[i, k, B] × table[k, j, C]
        back[i, j, A] ← { k, B, C }
return Build-Tree(back[0, Length(words), S]),  table[0, Length(words), S]
```

Slides adapted from Prof. Hwee Tou Ng (NUS)

Probabilistic CYK parsing

A. Recursive case: assigning
each cells with the highest score

B. Backtracking purposes:
storing the best parses

```
for i ← from j – 2 downto 0 do
  for k ← i + 1 to j – 1 do
    for all { A | A → BC ∈ grammar and
                  table[i, k, B] > 0 and table[k, j, C] > 0 }
      if (table[i, j, A]  < P(A → BC) × table[i, k, B] × table[k, j, C]) then
        table[i, j, A] ← P(A → BC) × table[i, k, B] × table[k, j, C]
        back[i, j, A] ← { k, B, C }
return Build-Tree(back[0, Length(words), S]),  table[0, Length(words), S]
```

Slides adapted from Prof. Hwee Tou Ng (NUS)

From Sequences to Trees

Parsing assigns structure (trees) to sequences of natural language.

Grammars give acceptability, and probabilistic ones help resolve structural ambiguity to find probable interpretations

CKY parsing memoizes basic building block parses to assemble larger blocks, leading to a sentence parse

There are modern neural forms of parsing that outperform the traditional ones we've presented today