Department of Linguistics and Translation
香港城市大學
City University of Hong Kong

Computational Linguistics LT3233

Jixing Li

Lecture 5: Parsing

Slides adapted from Julia Hockenmaier

Lecture plan

- Top-down, bottom-up, left-corner parsing
- CKY parsing
- Short break (15 mins)
- Hands-on exercises

Top-down parsing

CFG:

S → NP VP
NP → DT N
DT → the
N → dog
VP → VB
VB → laughs

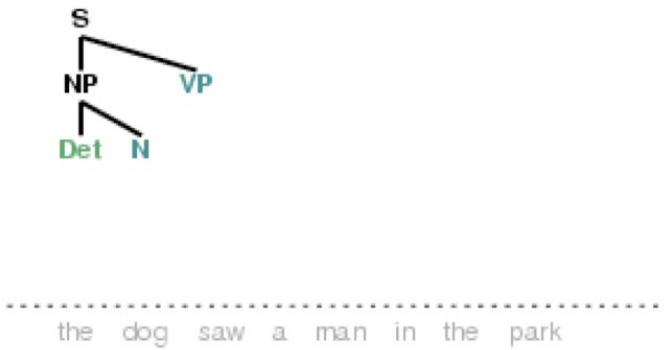
Input:	**Stack**	**Operation**	
'the dog laughs'	S	**expand**	S → NP VP
'the dog laughs'	NP VP	**expand**	NP → DT N
'the dog laughs'	DT N VP	**expand**	DT → the
~~*'the*~~ *dog laughs'*	~~the~~ N VP	**scan**	the
'dog laughs'	N VP	**expand**	N → dog
~~*'dog*~~ *laughs'*	~~dog~~ VP	**scan**	dog
'laughs'	VP	**expand**	VP → VB
'laughs'	VB	**expand**	VB → laughs
~~*'laughs'*~~	~~laughs~~	**scan**	laughs
[]	[]		

Recursively expanding the tree downward

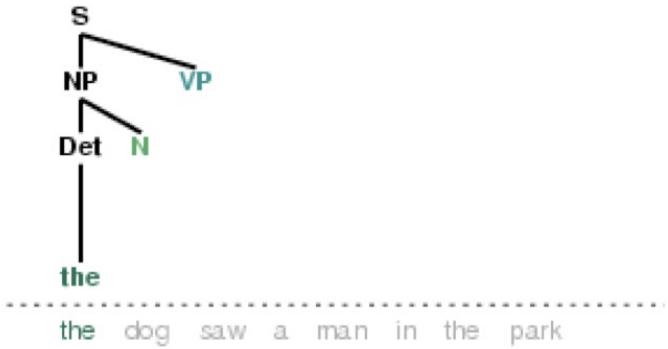
nltk.app.rdparser_app.app() **Recursive-descent parsing**

1. Initial stage

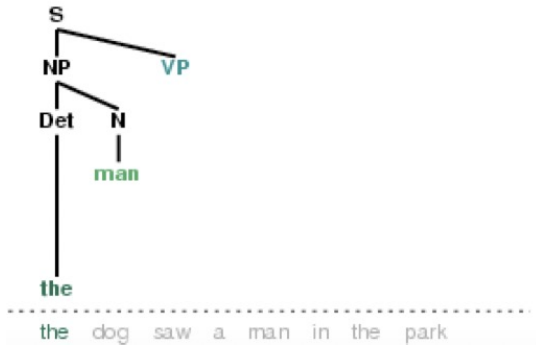
2. Second production



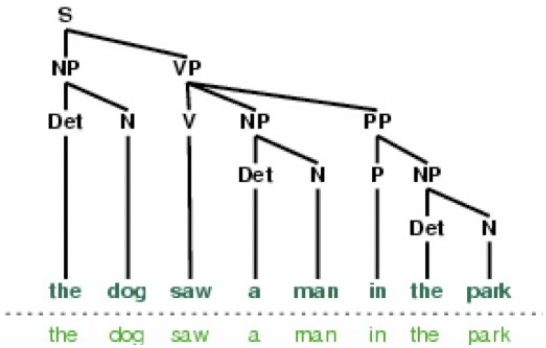
3. Matching *the*



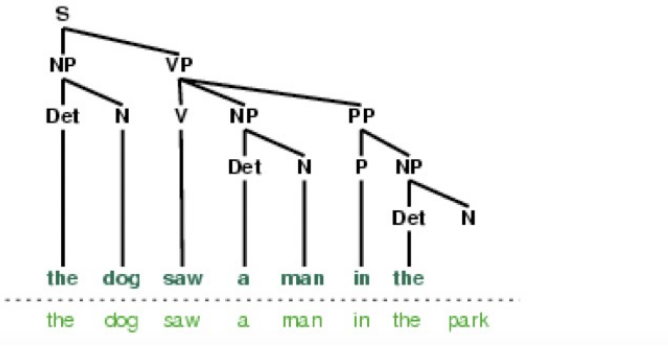
4. Cannot match *man*



5. Completed parse



6. Backtracking



Bottom-up parsing → the shift reduce parser

CFG:

S → NP VP
NP → DT N
DT → the
N → dog
VP → VB
VB → laughs

Input:	**Stack**	**Operation**	
‘~~*the*~~ *dog laughs’	the	**shift**	the
‘*dog laughs’	DT	**reduce**	DT → the
‘~~*dog*~~ *laughs’	DT dog	**shift**	dog
‘*laughs’	DT N	**reduce**	N → dog
‘*laughs’	NP	**reduce**	NP → DT N
‘~~*laughs’~~	NP laughs	**shift**	laughs
[]	NP VB	**reduce**	VB → laughs
[]	NP VP	**reduce**	VP → VB
[]	S	**reduce**	S → NP VP

Building trees from bottom-up

nltk.app.srparser_app.app()

1. Initial state

Stack	Remaining Text
	the dog saw a man in the park

2. After one shift

Stack	Remaining Text
the	dog saw a man in the park

3. After reduce shift reduce

Stack	Remaining Text
Det N the dog	saw a man in the park

4. After recognizing the second NP

Stack	Remaining Text
NP V NP in Det N saw Det N the dog a man	the park

5. After building a complex NP

Stack	Remaining Text
NP V NP Det N saw NP PP the dog Det N P NP a man in Det N the park	

6. Built a complete parse tree

Stack	Remaining Text
S NP VP Det N V NP the dog saw NP PP Det N P NP a man in Det N the park	

Left-corner parsing

- Top-down parsing: missing some important information provided by the words
- Bottom-up parsing: can sometimes end up in dead ends without top-down information
- Left-corner parsing: a mix of these two strategies.

CFG:

S → NP VP

NP → DT N | PropN

DT → the

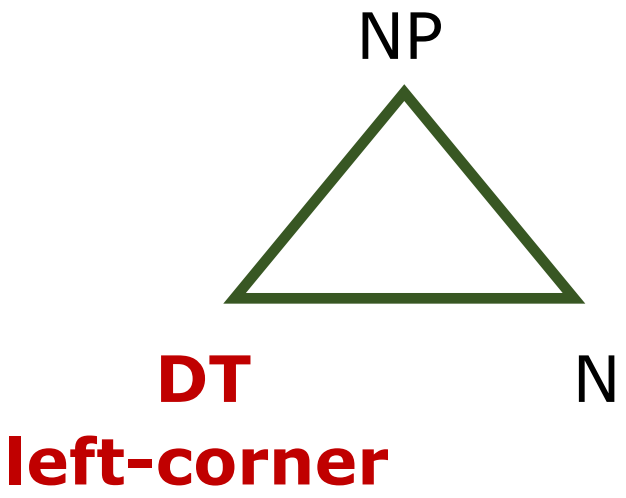
N → dog

VP → VB

VB → laughs

PropN → Sue

'The dog laughs'



Left-corner parsing

CFG:

S → NP VP

NP → DT N

DT → the

N → dog

VP → VB

VB → laughs

Input:	**Stack**	**Operation**	
~~*'the*~~ *dog laughs'*	S	**shift**	the
'dog laughs'	(the) S	**project**	DT → the
'dog laughs'	(DT) S	**project**	NP → DT N
'dog laughs'	N (NP) S	**project**	S → NP VP
~~*'dog*~~ *laughs'*	N VP	**shift**	dog
'laughs'	(dog) N VP	**project**	N → dog
~~*'laughs'*~~	VP	**shift**	laughs
[]	(laughs) VP	**project**	VB → laughs
[]	(VB) VP	**project**	VP → VB
[]	[]		

Practice

CFG:

S → NP VP

NP → PropN

PropN → John | Mary

VP → VB NP

VB → loves

Input:

'John loves Mary'

Top-down

expand	S → NP VP
expand	NP → PropN
expand	PropN → John
scan	John
expand	VP → VB NP
expand	VB → loves
scan	loves
expand	NP → PropN
expand	PropN → Mary
scan	Mary

Bottom-up

shift	John
reduce	PropN → John
reduce	NP → PropN
shift	loves
reduce	VB → loves
shift	Mary
reduce	PropN → Mary
reduce	NP → PropN
reduce	VP → VB NP
reduce	S → NP VP

Left-corner

shift	John
project	PropN → John
project	NP → PropN
project	S → NP VP
shift	loves
project	VB → loves
project	VP → VB NP
shift	Mary
project	PropN → Mary
project	NP → PropN

CKY parsing

The **Cocke-Kasami-Younger (CKY)** algorithm, the most widely used **dynamic-programming** based approach to parsing → **Chart parsing**

A dynamic programming approach breaks down a problem into sub-problems and stores the solutions to sub-problems.

In the case of syntactic parsing, these sub-problems represent parse trees for all the constituents detected in the input.

CKY algorithm

Bottom-up parsing:

start with the words

Dynamic programming:

save the results in a table/chart

re-use these results in finding larger constituents

Presumes a CFG in Chomsky Normal Form:

Rules are all either **A → B C** or **A → a**

(**A,B,C** are non-terminals and **a** is a terminal)

CKY algorithm

1. Create the chart

An *nxn* upper triangular matrix for a sentence with n words, each cell chart[i][j] corresponds to the substring $w_i...w_j$

2. Fill in the chart

Working from left to right, bottom to top

3. Extract the parse trees from the S in chart[0][n].

CFG in CNF:

S → NP VP

NP → DT N

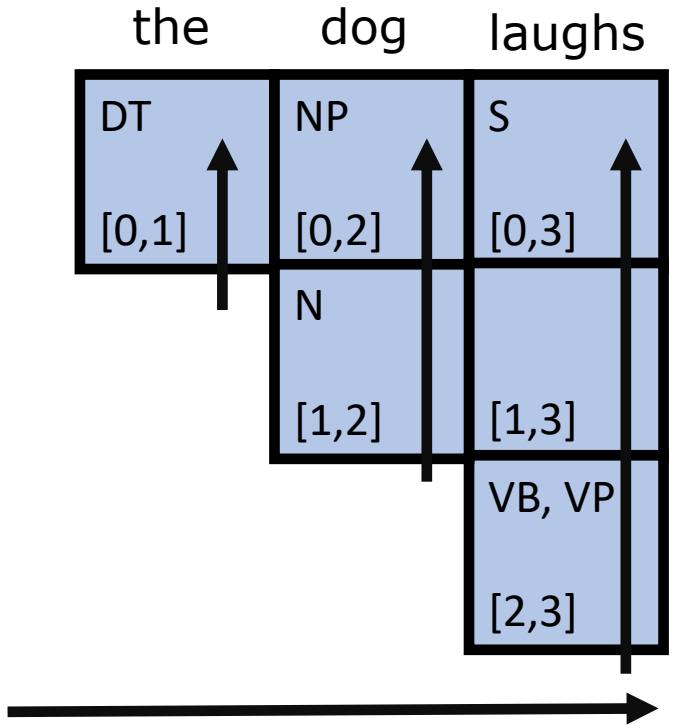
DT → the

N → dog

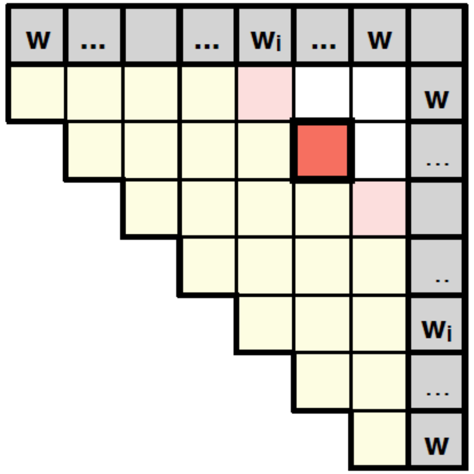
~~VP → VB~~

VP → laughs

VB → laughs



CKY: filling one cell

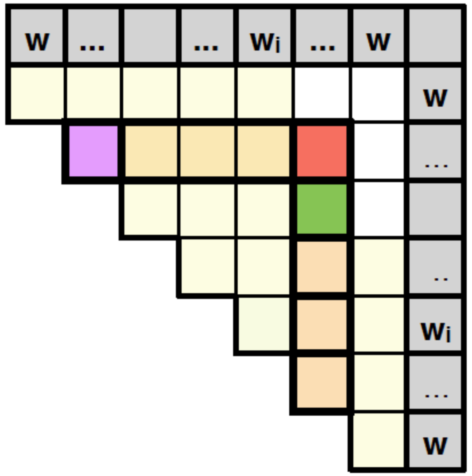


chart[2][6]:

w_1 $\mathbf{w_2}$ $\mathbf{w_3}$ $\mathbf{w_4}$ $\mathbf{w_5}$ $\mathbf{w_6}$ w_7

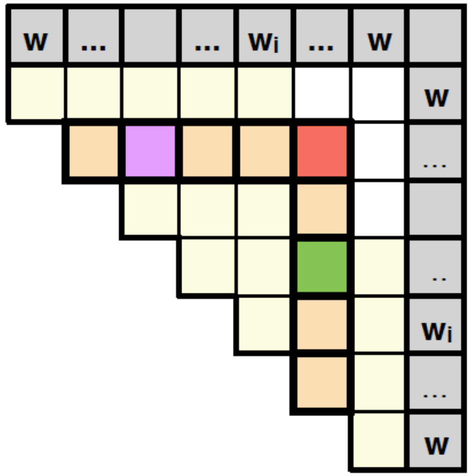
chart[2][6]:

w_1 $w_2 w_3 w_4 w_5 w_6$ w_7



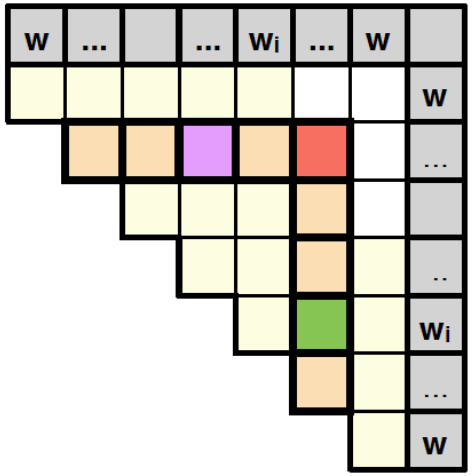
chart[2][6]:

w_1 $w_2 w_3 w_4 w_5 w_6$ w_7



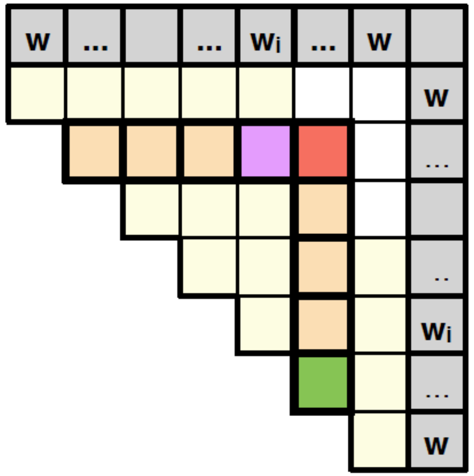
chart[2][6]:

w_1 $w_2 w_3 w_4 w_5 w_6$ w_7



chart[2][6]:

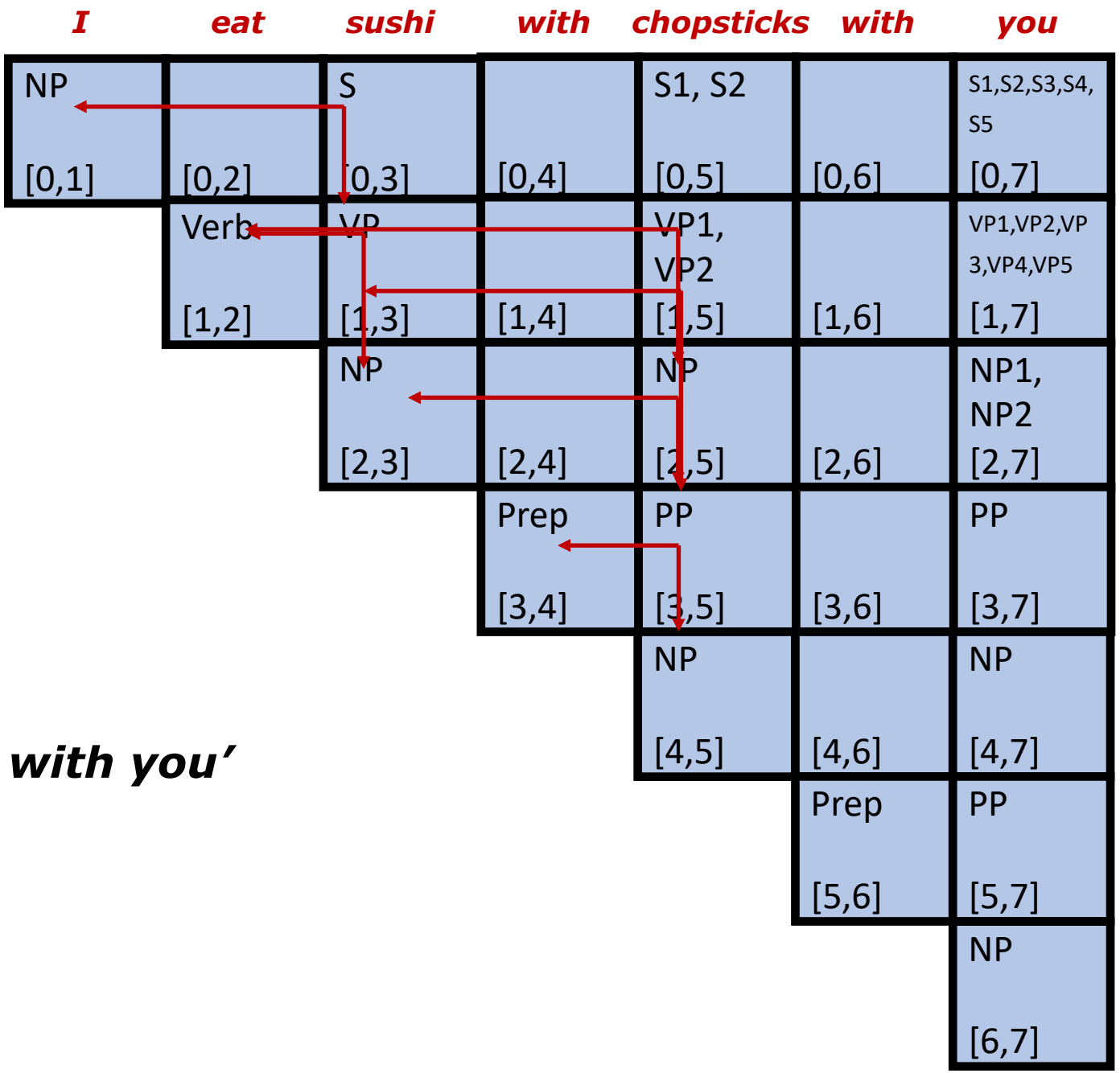
w_1 $w_2 w_3 w_4 w_5 w_6$ w_7



Practice

CFG in CNF:

S → NP VP	NP → you
NP → NP PP	VP → VP PP
NP → sushi	VP → Verb NP
NP → I	Verb → eat
NP → chopsticks	PP → Prep NP
	Prep → with



'I eat sushi with chopsticks with you'

To do

- Do HW4
- Optional reading: **NLTK** Ch8:4; **SLP** Ch13.2; **Stabler** Ch2,4,5