RICE SOCIAL SCIENCES
Department of Linguistics

parsing

LING 430 Computational Linguistics Fall 2026
lecture 10b
Oct 29 2026

Siyu Liang
siyu.liang@rice.edu

logistics

- **HW 5 due tonight, 11:59 pm**

last time

- named entities annotation
  - BIO tagging
- named entities recognition
- evaluating NER
- annotation
  - inter-annotator agreement
  - Cohen’s kappa

agenda

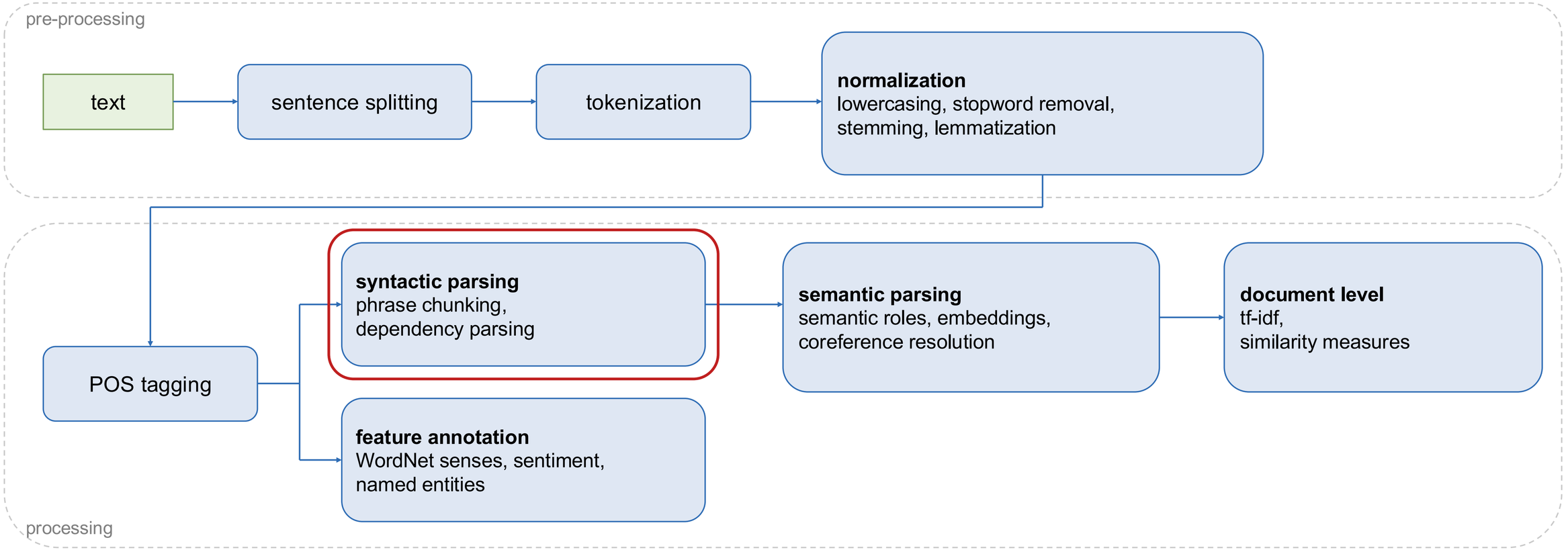
- constituency
- context-free grammars
- treebanks
- CKY parsing
- ambiguity

today's notebook

class10b.ipynb

- course website → week 10 → activity
  - **File → Save a copy in Drive**

the NLP pipeline



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

constituency

constituency

- what do these have in common?
  - *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*

constituency

- they are all **noun phrases**: a sequence of words surrounding at least one noun
- how do we know the words group together?
  - they all appear in the same environments, for example before a verb
    - *three parties from Brooklyn* arrive…
    - *the Broadway coppers* love…
    - *they* sit
- the words inside them do not: **from* arrive…, **the* is…
- a **constituent** is a group of words that behaves as a single unit

constituency

- why is a POS tagger not enough?
  - *The rat ate the malt. The cat killed the rat. The dog chased the cat.*
  - *The rat the cat the dog chased killed ate the malt.*
- a POS tagger sees DT NN DT NN DT NN VBD VBD VBD DT NN and is perfectly happy
- to check that three nouns get three verbs we need a sentence inside a sentence: S → NP S VP

context-free grammars

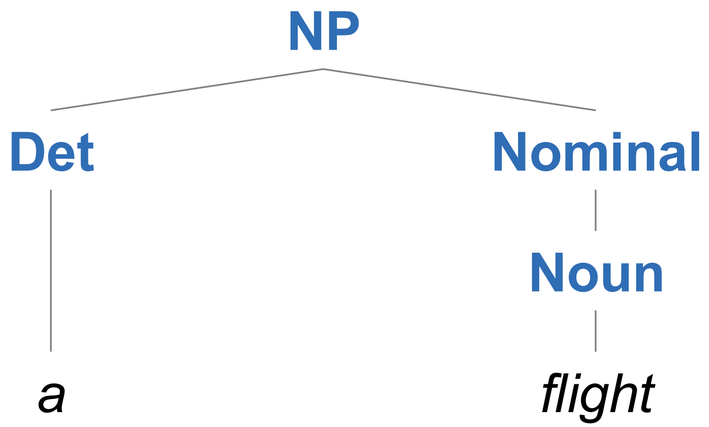
context-free grammars

- we want to say “noun phrases can occur before verbs”
- a **context-free grammar** (CFG) is a set of **rules** and a **lexicon**
  - NP → Det Nominal
  - NP → ProperNoun
  - Nominal → Noun | Nominal Noun
- the lexicon is rules too
  - Det → *a*, Det → *the*, Noun → *flight*
- **terminals** are the words
- **non-terminals** are the abstractions over them: NP, Nominal, Det

context-free grammars

derivation

- start from NP
- rewrite NP as Det Nominal
- rewrite Nominal as Noun
- rewrite the parts of speech as *a flight*
- this sequence of rewrites is a **derivation**; drawn with the root at the top, it is a **parse tree**
- NP **dominates** everything below it



context-free grammars

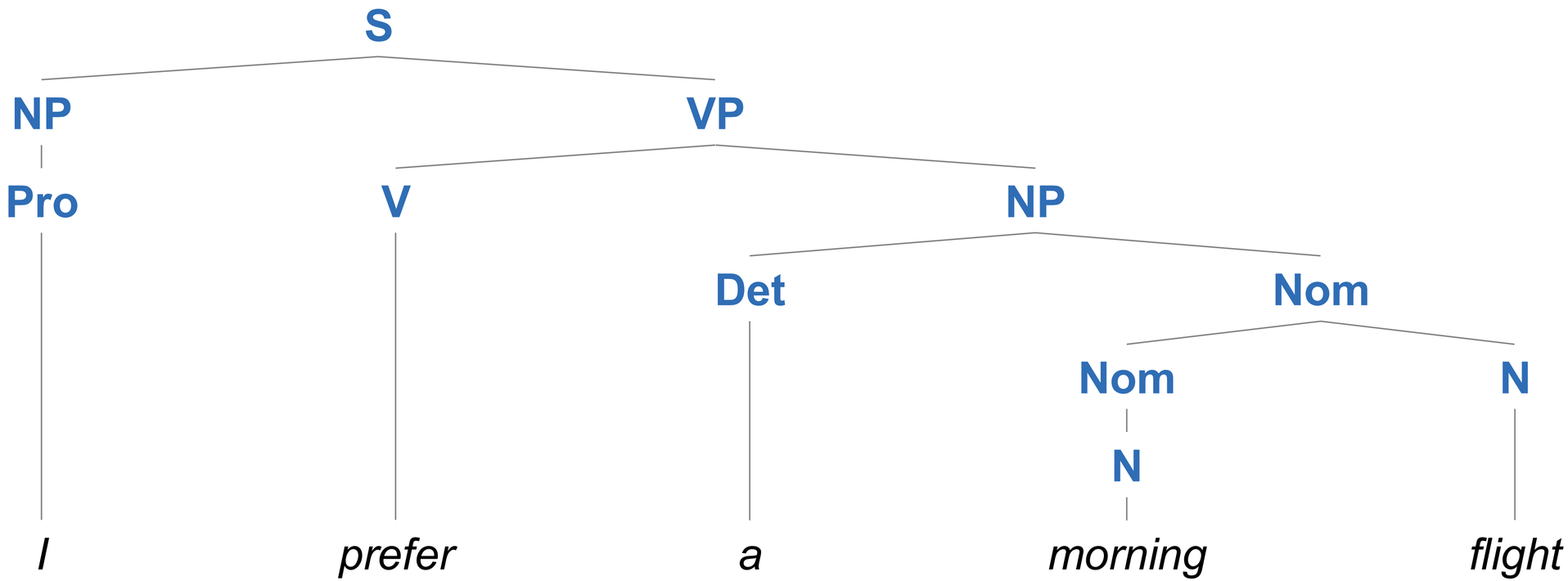
the grammar L0

- a few more rules before we can derive whole sentences
- S is the **start symbol**

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

context-free grammars

- I prefer a morning flight



context-free grammars

bracketed notation

- the same tree can be written on one line
  - `[S [NP [Pro I]] [VP [V prefer] [NP [Det a] [Nom [Nom [N morning]] [N flight]]]]]`
- a sentence the grammar can derive is **grammatical**
  - one it cannot derive is **ungrammatical**
- it is a simplified model of how natural languages really work
- this use of formal languages to model natural languages is called **generative grammar**

context-free grammars

the grammar L1

- let's expand L0
- add a few more rules:
  - S → Aux NP VP (*does United include a meal*), Nominal → Nominal PP (*the flight through Houston*), VP → VP PP (*book the flight through Houston*)

left side	right sides
S	NP VP | Aux NP VP | VP
NP	Pronoun | Proper-Noun | Det Nominal
Nominal	Noun | Nominal Noun | Nominal PP
VP	Verb | Verb NP | Verb NP PP | Verb PP | VP PP
PP	Preposition NP

context-free grammars

the lexicon of L1

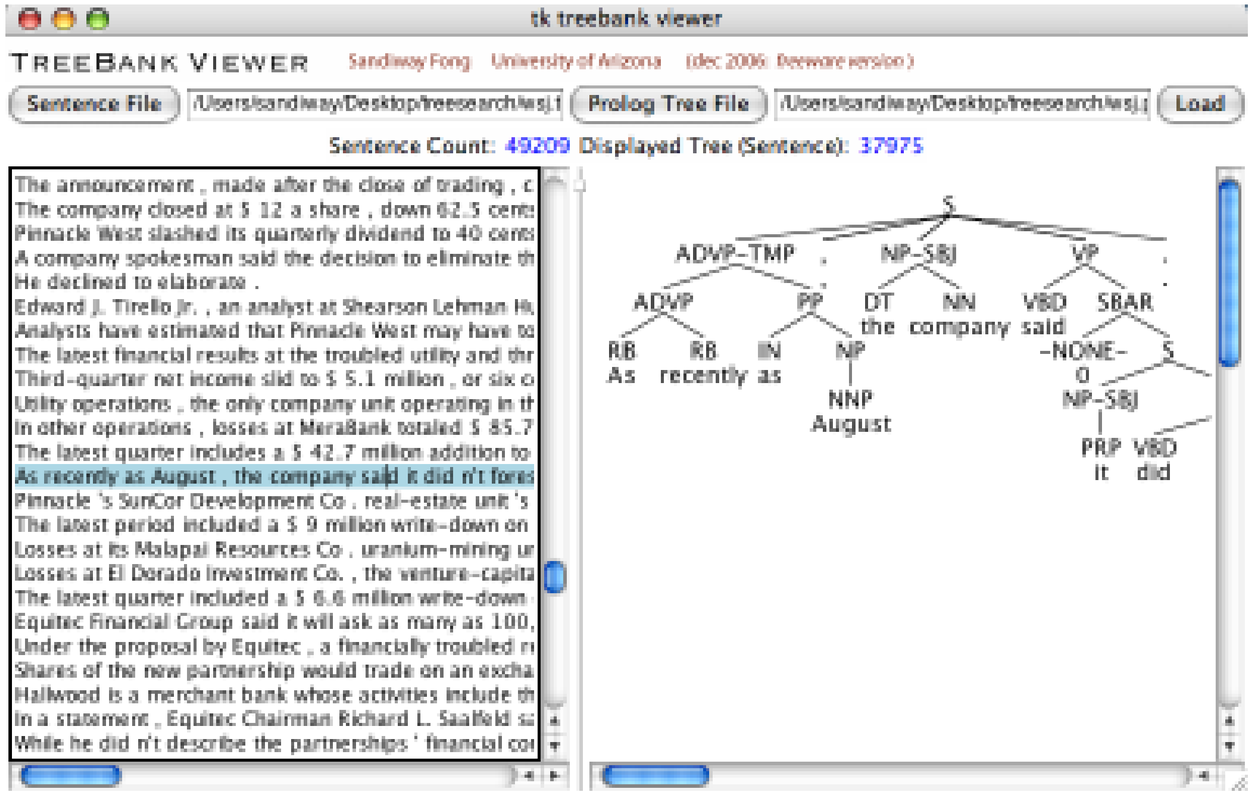
part of speech	words
Det	*that* | *this* | *the* | *a*
Noun	*book* | *flight* | *meal* | *money*
Verb	*book* | *include* | *prefer*
Pronoun	*I* | *she* | *me*
Proper-Noun	*Houston* | *United*
Aux	*does*
Preposition	*from* | *to* | *on* | *near* | *through*

context-free grammars

- can we write a grammar of English by hand, rule by rule?
- a plausible rule: verbs with a prefix take an object, not a PP
  - outrun, outlive, underpay (Carlson and Roeper 1980)
- then a corpus turns up I want to overindulge in you (Sampson 2007)
- “all grammars leak” (Sapir 1921)
- so we take the rules from data

treebanks

- a **treebank** is a corpus in which every sentence is annotated with a parse tree
- made by running a parser over each sentence, then having linguists hand-correct the result
- trees are stored in a parenthesized notation, like our bracketed notation



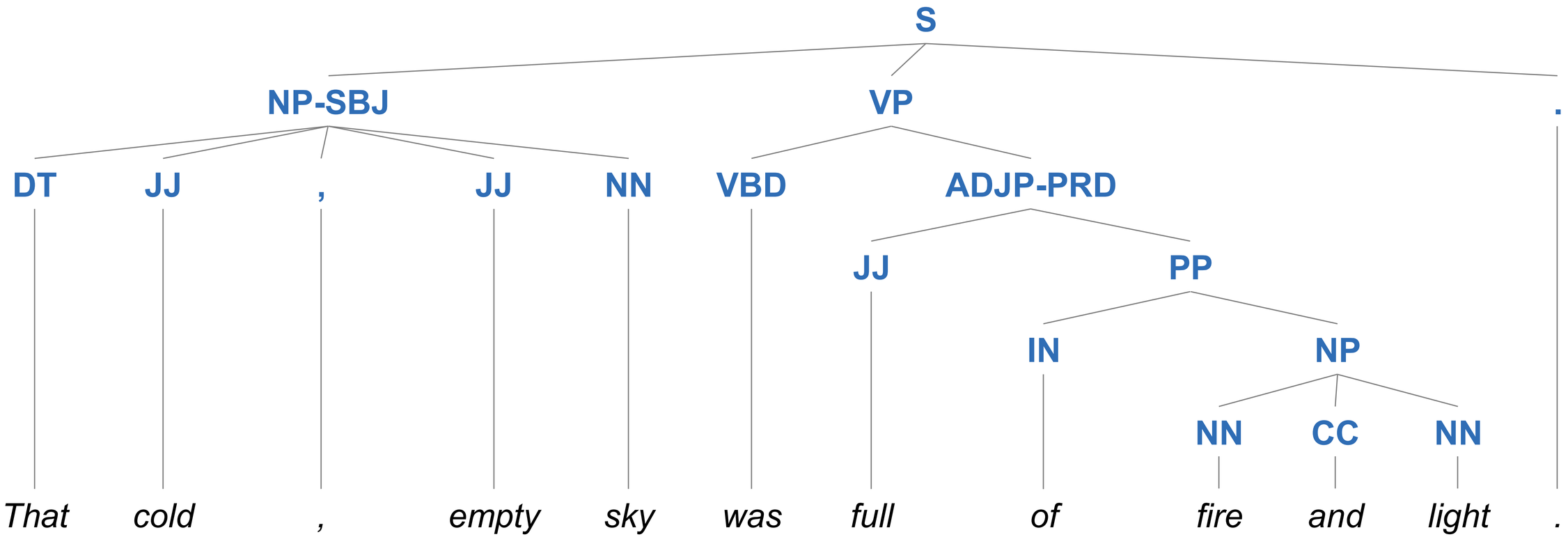
treebanks

a Brown corpus sentence

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
             (CC and)
             (NN light) )))))
   (. .) ))
```

treebanks

- -SBJ and -PRD mark the grammatical function: subject, predicate



treebanks

- which rules can we infer from this tree?

rule	e.g.
S → NP VP .	*That cold, empty sky was full of fire and light.*
NP → DT JJ , JJ NN	*That cold, empty sky*
VP → VBD ADJP	*was full of fire and light*
ADJP → JJ PP	*full of fire and light*
PP → IN NP	*of fire and light*
NP → NN CC NN	*fire and light*
DT → *that*	

treebanks

a very flat grammar

- how many rules does the Penn Treebank have for expanding VP?
  - about 4,500!
- a separate rule for every number of prepositional phrases
  - VP → VBD PP
  - VP → VBD PP PP
  - VP → VBD PP PP PP
- and for every arrangement of the verb's arguments: VP → VB ADVP PP, VP → VB PP ADVP, VP → ADVP VB PP
- counting how often each rule is used turns the treebank into a probabilistic grammar, which is how a parser chooses between parses

CKY parsing

CKY parsing

- **syntactic parsing**: mapping from a string of words to its parse tree
- with L1, what structure does *Book the flight through Houston* get?
- trying every derivation from S is hopeless
  - the number of trees explodes with the length of the sentence 🤯

CKY parsing

- so we have dynamic programming again!
- the subproblems are the constituents inside the sentence
  - the best solution for subproblem is part of the best solution for the entire problem
- the **CKY algorithm**, after Cocke, Kasami and Younger, who each found it independently in the 1960s
  - using a table of the constituents in the sentence, filled from short spans to long, for a grammar in a special form

CKY parsing

Chomsky normal form

- CKY wants every rule to have exactly two daughters 👯♀
  - the Penn Treebank NP had five daughters
  - can we have that without changing the language?
- **Chomsky normal form** (CNF)
  - every rule is A → B C or A → a, two non-terminals or one word
  - so the trees are **binary branching**
- two grammars are **weakly equivalent** if they generate the same string; **strongly equivalent** if they also assign the same structures
- any CFG converts into a weakly equivalent CNF grammar
  - A → B C D becomes A → B X and X → C D

CKY parsing

converting to CNF

- a word mixed into a rule
  - INF-VP → to VP becomes INF-VP → TO VP and TO → *to*
- a **unit production**, one non-terminal on the right
  - S → VP
  - replace it with everything it leads to: S → VP, VP → Verb, Verb → *book* give S → *book*
- a rule longer than two
  - S → Aux NP VP becomes S → X1 VP and X1 → Aux NP
- rules already in the right form are copied unchanged

CKY parsing

L1 in Chomsky normal form

- converting L1 to CNF

L1	in CNF
S → Aux NP VP	S → X1 VP, X1 → Aux NP
S → VP	S → *book* | *include* | *prefer*
	S → Verb NP | X2 PP | Verb PP | VP PP
NP → Pronoun	NP → *I* | *she* | *me*
NP → Proper-Noun	NP → *United* | *Houston*
Nominal → Noun	Nominal → *book* | *flight* | *meal* | *money*
VP → Verb	VP → *book* | *include* | *prefer*
VP → Verb NP PP	VP → X2 PP, X2 → Verb NP

CKY parsing

- all the rules we parse with

left side	right sides
S	NP VP | X1 VP | Verb NP | X2 PP | Verb PP | VP PP
X1	Aux NP
X2	Verb NP
NP	Det Nom
Nom	Nom Noun | Nom PP
VP	Verb NP | X2 PP | Verb PP | VP PP
PP	Prep NP

CKY parsing

the table

- number the gaps: $_0$ Book $_1$ the $_2$ flight $_3$ through $_4$ Houston $_5$
- cell [i, j] holds the constituents from gap *i* to gap *j*

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

CKY parsing

1. unit productions

$_0$ Book $_1$ the $_2$ flight $_3$ through $_4$ Houston $_5$

- apply unit productions from the lexicon across the diagonal

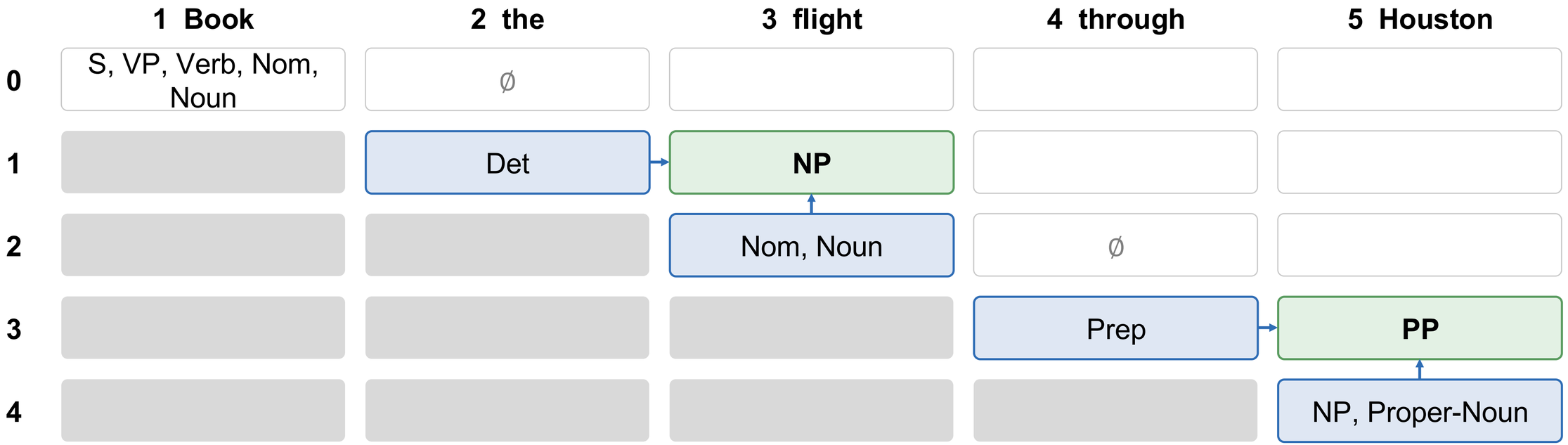
	1 Book	2 the	3 flight	4 through	5 Houston
0	S, VP, Verb, Nom, Noun				
1		Det			
2			Nom, Noun		
3				Prep	
4					NP, Proper-Noun

CKY parsing

2. moving up and right

$_0$ Book $_1$ the $_2$ flight $_3$ through $_4$ Houston $_5$

- NP → Det Nom, PP → Prep NP
- no rule ends in Det or Prep, so [0, 2] and [2, 4] stay empty

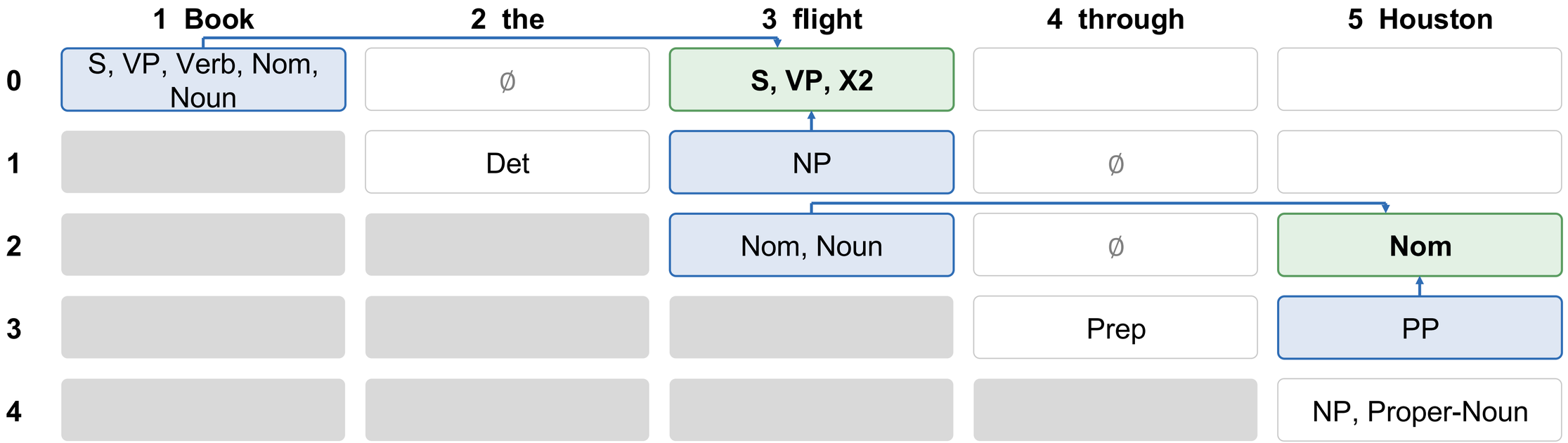


CKY parsing

3. continue

$_0$ Book $_1$ the $_2$ flight $_3$ through $_4$ Houston $_5$

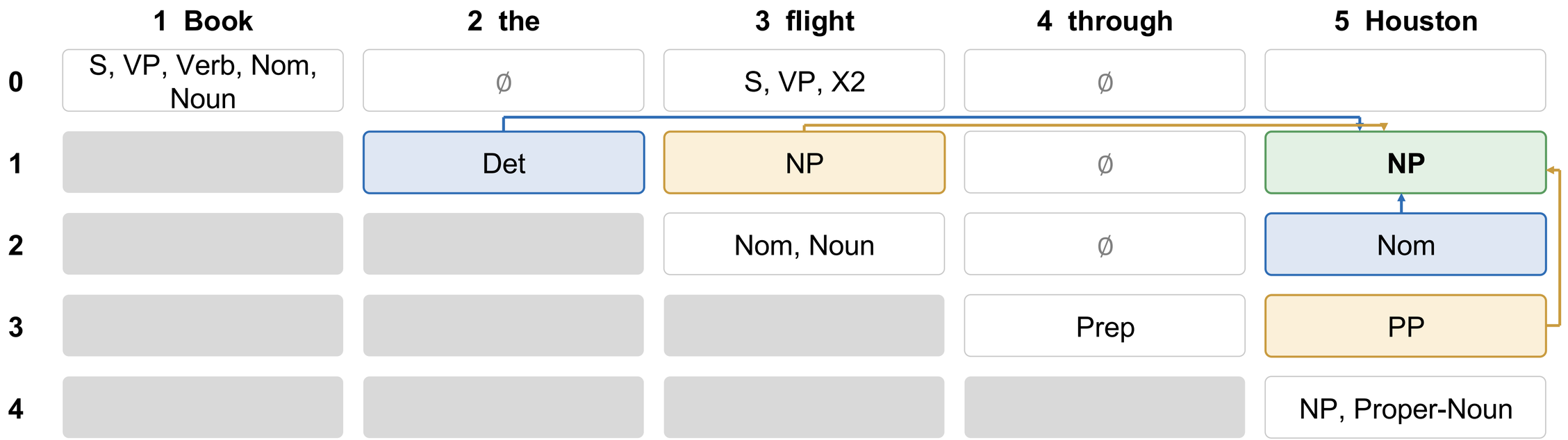
- [0, 3]: Verb + NP gives VP, S and X2
- [2, 5]: Nom + PP gives Nom



CKY parsing

4. continue

$_0$ Book $_1$ the $_2$ flight $_3$ through $_4$ Houston $_5$



CKY parsing

5. continue

$_0$ Book $_1$ the $_2$ flight $_3$ through $_4$ Houston $_5$

- four options: split at 1, Verb + NP; at 2, ∅ + Nom; at 3, VP or X2 + PP; at 4, ∅ + NP

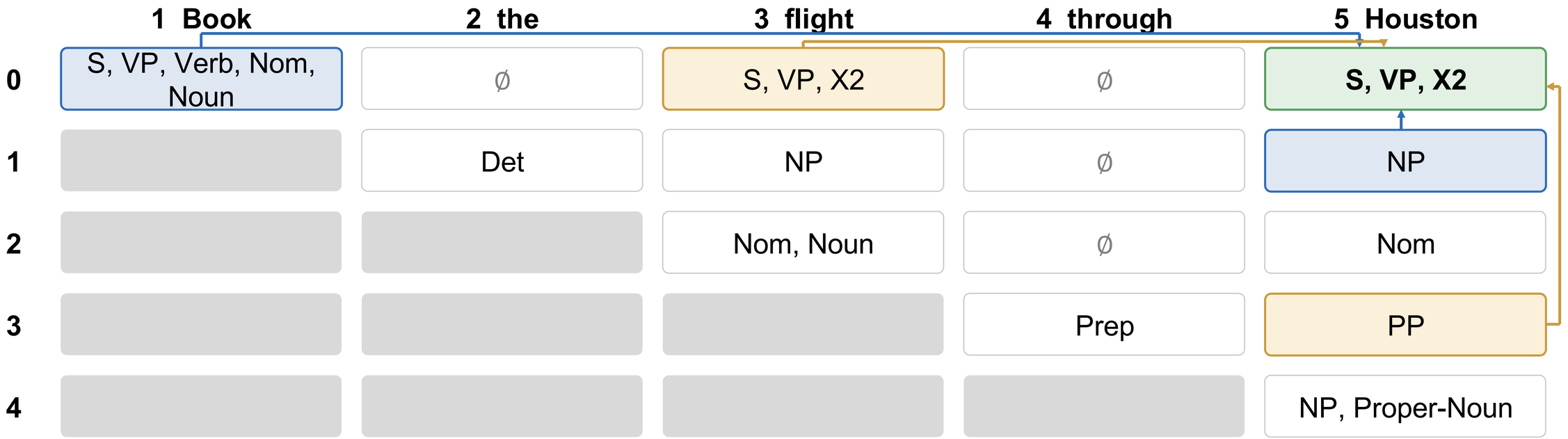
	1 Book	2 the	3 flight	4 through	5 Houston
0	S, VP, Verb, Nom, Noun	∅	S, VP, X2	∅	?
1		Det	NP	∅	NP
2			Nom, Noun	∅	Nom
3				Prep	PP
4					NP, Proper-Noun

CKY parsing

5. finding the final split

$_0$ Book $_1$ the $_2$ flight $_3$ through $_4$ Houston $_5$

- split at 1, Verb + NP gives VP, S and X2
- split at 3, VP + PP and X2 + PP give VP and S



CKY parsing

debrief

- CKY as a recognizer
  - is there a parse for it? i.e. can we find a way to fill the top right cell?
- a parser also keeps, with each entry, pointers to the two cells it came from

S in [0, 5]	from	reading
S1	VP [0, 3] + PP [3, 5]	book the flight, and do the booking through Houston
S2	Verb [0, 1] + NP [1, 5]	a command to book the flight that goes through Houston
S3	X2 [0, 3] + PP [3, 5]	the old three-part rule Verb NP PP; reads like S1

CKY parsing

ambiguity

- S1 and S2 differ in where the PP attaches
  - to the verb phrase: how to book, through Houston
  - inside the noun phrase: which flight, the one through Houston
- the grammar allows both, so the sentence has two meanings
- but nothing in the table says which was meant
- CKY finds every parse; it does not pick one
- a sentence with more than one parse is **structurally ambiguous**

ambiguity

ambiguity



ambiguity

- where is the elephant?
  - *One morning I shot an elephant in my pajamas. How he got into my pajamas I don't know.*
  - Groucho Marx, *Animal Crackers*, 1930

ambiguity

structural ambiguity

- last week we talked about part-of-speech ambiguity
  - one word with several tags
- **structural ambiguity**: the grammar assigns more than one parse to a sentence
- *in my pajamas* can be part of the NP headed by *elephant*, or part of the VP headed by *shot*
- L1 has a rule for each: Nominal → Nominal PP and VP → VP PP
- the most frequent problem syntactic parsers face

activity 1

part 1: draw the trees

- in pairs, draw both parse trees of *I shot an elephant in my pajamas* with the grammar at the right
- which rule attaches the PP in the funny reading?

rule	lexicon
S → NP VP	Pronoun → *I*
NP → Pronoun | Det Nom	Verb → *shot*
Nom → Noun | Nom PP	Det → *an* | *my*
VP → Verb NP | VP PP	Noun → *elephant* | *pajamas*
PP → Prep NP	Prep → *in*

activity 1

part 2: run the parser

- in the notebook, parse the sentence with the nltk grammar
- how many trees do you get, and which one comes first?
- then add a word to the lexicon and parse a sentence of your own

ambiguity

attachment ambiguity

- **attachment ambiguity**: a constituent can be attached to the tree at more than one place
- the elephant is a case of **PP-attachment ambiguity**
  - the PP can go into the NP or into the VP
- adverbial phrases do it too
  - *We saw the Eiffel Tower flying to Paris.*

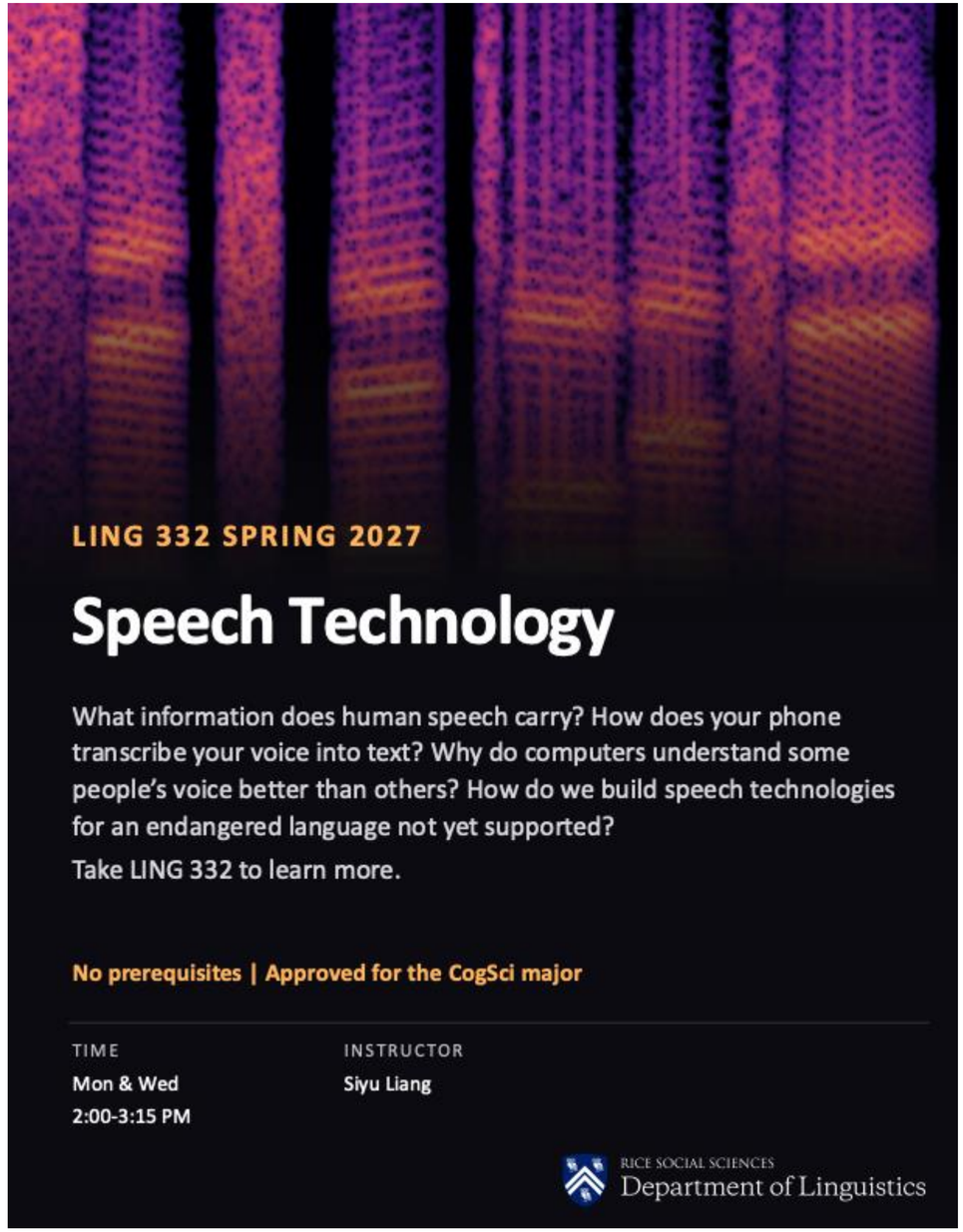
ambiguity

coordination ambiguity

- **coordination ambiguity**
  - e.g. what does *and* join?
  - *old men and women*: [old [men and women]], the women are old too, or [[old men] and women]
  - [nationwide [television and radio]] or [[nationwide television] and radio]?
- a parser like CKY finds every grammatical parse
  - choosing one is **syntactic disambiguation**

announcements

new courses!



announcements

new courses!



next time

- Tuesday, Nov 3
  - TF-IDF and vector semantics
- **HW 5 due tonight, 11:59 pm**