Syntactic Parsing

Natalie Parde

UIC CS 421

We’ve learned all about the general building blocks of NLP.

- How can we use these tools to make sense of language?
- Popular category of tasks: syntactic parsing
- **Syntactic parsing:** The process of automatically recognizing and assigning syntactic (grammatical) roles to the constituents within sentences

Why is syntactic parsing useful?

- Lots of reasons! For example:
  - Grammar checking
  - Downstream applications
    - Question answering
    - Information extraction

This Week's Topics

Tuesday

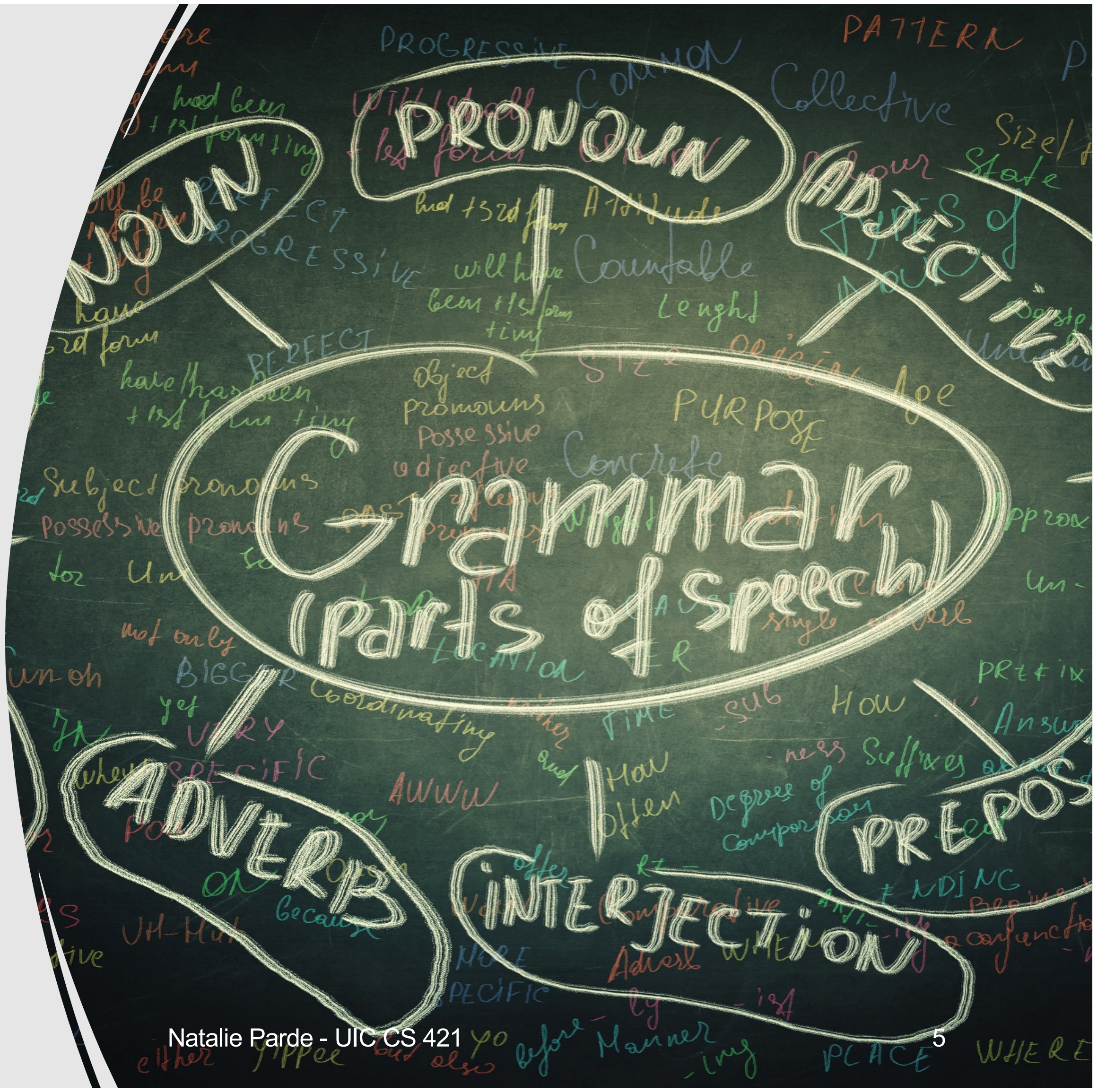
Parts of Speech
POS Tagsets
POS Tagging
Context-Free Grammars

Thursday

Hierarchical Parsing
Dynamic Programming Parsing Algorithms
Probabilistic CKY
Lexicalized Grammars

What is part-of-speech (POS) tagging?

- The process of automatically assigning grammatical word classes to individual tokens in text.
- Traditional (broad) categories:
  - noun
  - verb
  - adjective
  - adverb
  - preposition
  - article
  - interjection
  - pronoun
  - conjunction
- Sometimes also referred to as **lexical categories**, **word classes**, or **morphological classes**



POS Tagging

- Can be very challenging!
- Words often have more than one valid part of speech tag
  - Today's faculty meeting went really **well**! = adverb
  - Do you think the undergrads are **well**? = adjective
  - **Well**, did you see the latest response to your email? = interjection
  - Jurafsky and Martin's book is a **well** of information. = noun
  - Laughter began to **well** up inside her at, as always, a highly inconvenient time. = verb
- Our goal in those cases is to determine the *best* POS tag for a particular instance of a word.

Why is POS tagging useful?

- First step of many pipelined NLP tasks:
  - Speech synthesis
  - Constituency parsing
  - Dependency parsing
  - And many more!
- For approaches that don't require a modular pipeline, offers an avenue for interpretable linguistic analysis

POS Tag Categories

Each POS type falls into one of two larger classes:

- Open
- Closed

Open class:

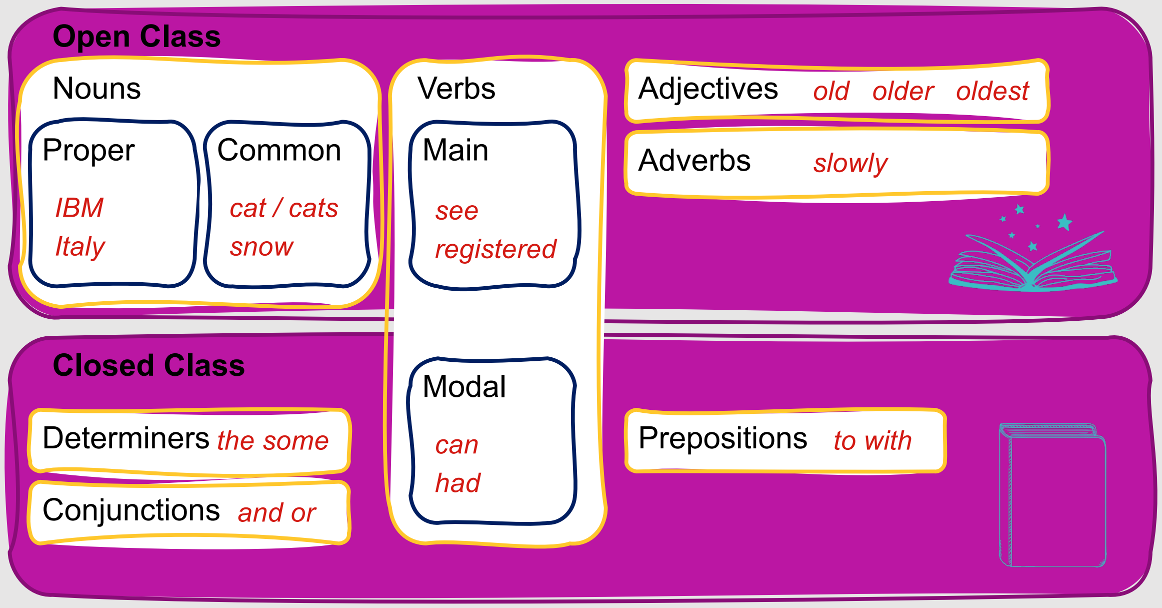
- New members can be created at any time
- In English:
  - Nouns, verbs, adjectives, and adverbs
- Many (but not all!) languages have these four classes

Closed class:

- A small, fixed membership …new members cannot be created spontaneously
- Usually function words
- In English:
  - Prepositions and auxiliaries (may, can, been, etc.)

Finer-Grained POS Classes

- Broader POS classes often have smaller subclasses
  - Noun:
    - Proper (Illinois)
    - Common (state)
  - Verb:
    - Main (tweet)
    - Modal (had)
- Some subclasses of a broad part of speech might be open, while others are closed



This Week's Topics

Tuesday

- Parts of Speech
- ☆ POS Tagsets
- POS Tagging
- Context-Free Grammars

Thursday

- Hierarchical Parsing
- Dynamic Programming Parsing Algorithms
- Probabilistic CKY
- Lexicalized Grammars

POS Tagsets

When determining which POS tag to assign to a word, we first need to decide which **tagset** we will use

Tagset: A finite set of POS tags, where each tag defines a distinct grammatical role

Can range from very coarse to very fine

Penn Treebank Tagset

- **Most common POS tagset**
- 36 POS tags + 12 other tags (punctuation and currency)
- Used when developing the Penn Treebank, a corpus created at the University of Pennsylvania containing more than 4.5 million words of American English
- Link to documentation: https://catalog.ldc.upenn.edu/docs/LDC95T7/cl93.html

Penn Treebank Tagset

CC	Coordinating Conjunction	**NNS**	Noun, plural	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Determiner	**NNPS**	Proper noun, plural	**VB**	Verb, base form
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive ending	**VBG**	Verb, gerund or present participle
IN	Preposition or subordinating conjunction	**PRP**	Personal pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present
JJS	Adjective, superlative	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singular or mass	**SYM**	Symbol	**WRB**	Wh-adverb

What do some of these distinctions mean?

cities

Chicago

Chicagos

city

CC	Coordinating Conjunction	**NNS**	Noun, plural	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Deter	**NNPS**	Proper noun, plural	**VB**	Verb, base form
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive ending	**VBG**	Verb, gerund or present participle
IN	Preposition or subordinating conjunction	**PRP**	Personal pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present
JJS	Adjective, superlative	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singular or mass	**SYM**	Symbol	**WRB**	Wh-adverb

What do some of these distinctions mean?

CC	Coordinating Conjunction	**NNS**	Noun, plural	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Determiner	**NNPS**	Proper noun, plural	**VB**	Verb, base form
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive ending	**VBG**	Verb, gerund or present participle
IN	Preposition or subordinating conjunction	**PRP**	Personal pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present
JJS	Adjective, superlative	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singular or mass	**SYM**	Symbol	**WRB**	Wh-adverb

eat

ate

eating

eaten

eat

eats

should

What do some of these distinctions mean?

CC	Coordinating Conjunction	**NNS**	Noun, plural	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Determiner	**NNPS**	Proper noun, plural	**VB**	Verb, base form
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive ending	**VBG**	Verb, gerund or present participle
IN	Preposition or subordinating conjunction	**PRP**	Personal pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present
JJS	Adjective, superlative	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singular or mass	**SYM**	Symbol	**WRB**	Wh-adverb

weird

weirder

weirdest

What do some of these distinctions mean?

CC	Coordinating Conjunction	**NNS**	Noun, plural	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Determiner	**NNPS**	Proper noun, plural	**VB**	Verb, base form
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive ending	**VBG**	Verb, gerund or present participle
IN	Preposition or subordinating conjunction	calmly	Personal pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present
JJS	Adjective, calmer ve	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singul calmest	**SYM**	Symbol	**WRB**	Wh-adverb

As a general (but not perfect!) rule….

CC	Coordinating Conjunction	**NNS**	Noun, plural	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Determiner	**NNPS**	Proper noun, plural	**VB**	Verb, base form
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive ending	**VBG**	Ve[illegible]d or present pa[illegible]
IN	Preposition or subordinating conjunction	**PRP**	Personal pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present
JJS	Adjective, superlative	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singular or mass	**SYM**	Symbol	**WRB**	Wh-adverb

Closed Class

As a general (but not perfect!) rule….

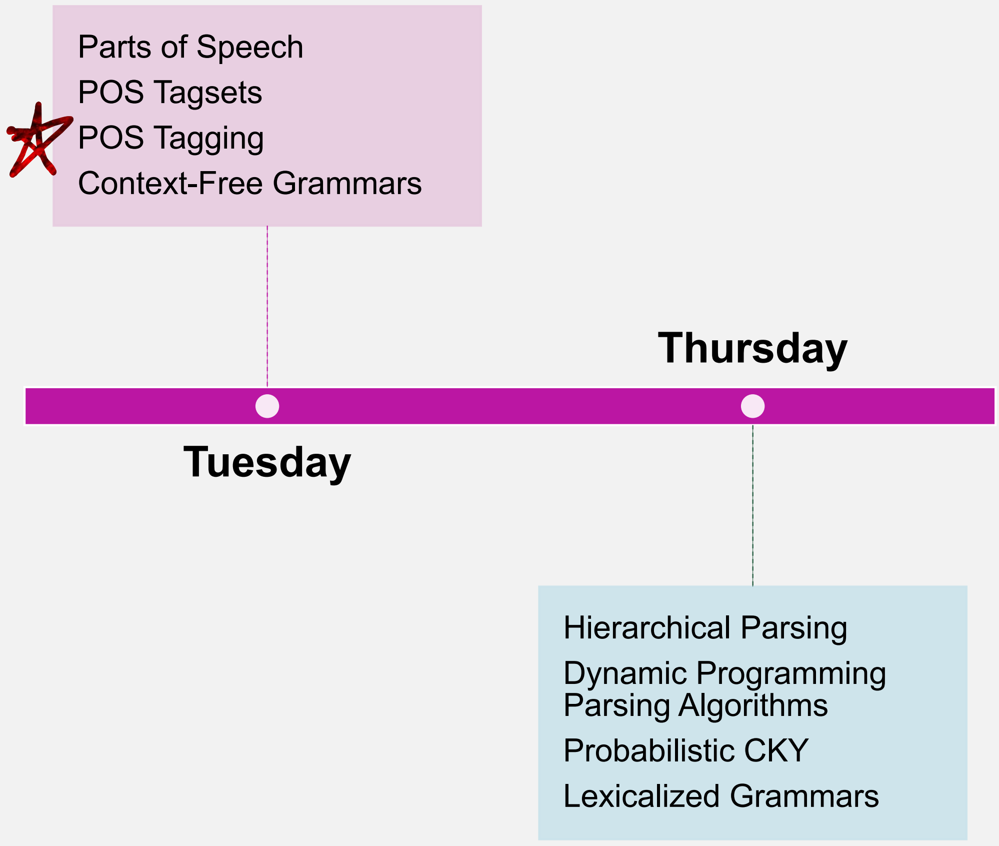
CC	Coordinating Conjunction	**NNS**	Noun, plural	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Determiner	**NNPS**	Proper noun, plural	**VB**	Verb, base form
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive [illegible]	**VBG**	Verb, gerund or present participle
IN	Preposition or subordinating conjunction	**PRP**	[illegible] pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present
JJS	Adjective, superlative	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singular or mass	**SYM**	Symbol	**WRB**	Wh-adverb

Open Class

Other Popular POS Tagsets

- Brown Corpus
  - ~1 million words of American English text
  - 82 (!) POS tags
- C5 Tagset
  - Text from the British National Corpus
  - 61 POS tags
- C7 Tagset
  - Text from the British National Corpus
  - 146 (!!) POS tags

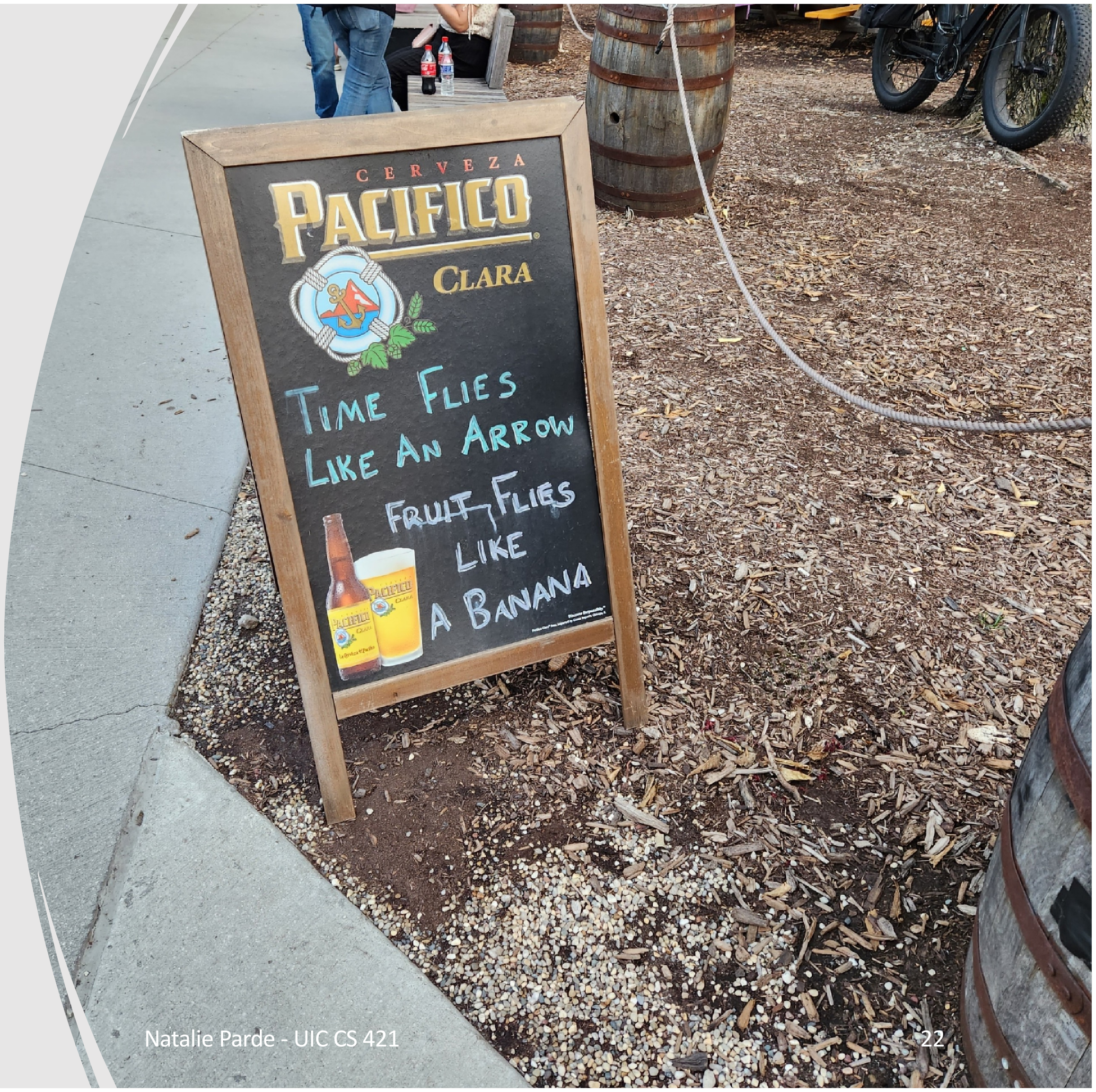
This Week's Topics



Parts of Speech
POS Tagsets
★ POS Tagging
Context-Free Grammars

Hierarchical Parsing
Dynamic Programming Parsing Algorithms
Probabilistic CKY
Lexicalized Grammars

So …how can we assign POS tags?



So …how can we assign POS tags?

Time	flies	like	an	arrow;	fruit	flies	like	a	banana

CC	Coordinating Conjunction	**NNS**	Noun, plural	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Determiner	**NNPS**	Proper noun, plural	**VB**	Verb, base form
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive ending	**VBG**	Verb, gerund or present participle
IN	Preposition or subordinating conjunction	**PRP**	Personal pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present
JJS	Adjective, superlative	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singular or mass	**SYM**	Symbol	**WRB**	Wh-adverb

So …how can we assign POS tags?

Time	flies	like	an	arrow	fruit	flies	like	a	banana
NN									

CC	Coordinating Conjunction	NNS	Noun, plural	TO	to
CD	Cardinal Number	NNP	Proper noun, singular	UH	Interjection
DT	Determiner	NNPS	Proper noun, plural	VB	Verb, base form ?
EX	Existential *there*	PDT	Predeterminer	VBD	Verb, past tense
FW	Foreign word	POS	Possessive ending	VBG	Verb, gerund or present participle
IN	Preposition or subordinating conjunction	PRP	Personal pronoun	VBN	Verb, past participle
JJ	Adjective	PRP$	Possessive pronoun	VBP	Verb, non-3rd person singular present
JJR	Adjective, comparative	RB	Adverb	VBZ	Verb, 3rd person singular present
JJS	Adjective, superlative	RBR	Adverb, comparative	WDT	Wh-determiner
LS	List item marker	RBS	Adverb, superlative	WP	Wh-pronoun
MD	Modal	RP	Particle	WP$	Possessive wh-pronoun
NN	Noun, singular or mass ?	SYM	Symbol	WRB	Wh-adverb

So …how can we assign POS tags?

Time	flies	like	an	arrow	fruit	flies	like	a	banana
NN	VBZ								

CC	Coordinating Conjunction	**NNS**	Noun, plural ?	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Determiner	**NNPS**	Proper noun, plural	**VB**	Verb, base form ?
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive ending	**VBG**	Verb, gerund or present participle
IN	Preposition or subordinating conjunction	**PRP**	Personal pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present ?
JJS	Adjective, superlative	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singular or mass ?	**SYM**	Symbol	**WRB**	Wh-adverb

So …how can we assign POS tags?

Time	flies	like	an	arrow	fruit	flies	like	a	banana
NN	VBZ	IN							

CC	Coordinating Conjunction	**NNS**	Noun, plural ?	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Determiner	**NNPS**	Proper noun, plural	**VB**	Verb, base form ??
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive ending	**VBG**	Verb, gerund or present participle
IN	Preposition or subordinating conjunction ?	**PRP**	Personal pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present ?
JJS	Adjective, superlative	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singular or mass ?	**SYM**	Symbol	**WRB**	Wh-adverb

So …how can we assign POS tags?

Time	flies	like	an	arrow	fruit	flies	like	a	banana
NN	VBZ	IN	DT						

CC	Coordinating Conjunction	NNS	Noun, plural ?	TO	to
CD	Cardinal Number	NNP	Proper noun, singular	UH	Interjection
DT	Determiner 🙂	NNPS	Proper noun, plural	VB	Verb, base form ??
EX	Existential *there*	PDT	Predeterminer	VBD	Verb, past tense
FW	Foreign word	POS	Possessive ending	VBG	Verb, gerund or present participle
IN	Preposition or subordinating conjunction ?	PRP	Personal pronoun	VBN	Verb, past participle
JJ	Adjective	PRP$	Possessive pronoun	VBP	Verb, non-3rd person singular present
JJR	Adjective, comparative	RB	Adverb	VBZ	Verb, 3rd person singular present ?
JJS	Adjective, superlative	RBR	Adverb, comparative	WDT	Wh-determiner
LS	List item marker	RBS	Adverb, superlative	WP	Wh-pronoun
MD	Modal	RP	Particle	WP$	Possessive wh-pronoun
NN	Noun, singular or mass ?	SYM	Symbol	WRB	Wh-adverb

So …how can we assign POS tags?

Time	flies	like	an	arrow	fruit	flies	like	a	banana
NN	VBZ	IN	DT	NN	NN				

CC	Coordinating Conjunction	NNS	Noun, plural ?	TO	to
CD	Cardinal Number	NNP	Proper noun, singular	UH	Interjection
DT	Determiner :)	NNPS	Proper noun, plural	VB	Verb, base form ? ?
EX	Existential *there*	PDT	Predeterminer	VBD	Verb, past tense
FW	Foreign word	POS	Possessive ending	VBG	Verb, gerund or present participle
IN	Preposition or subordinating conjunction ?	PRP	Personal pronoun	VBN	Verb, past participle
JJ	Adjective	PRP$	Possessive pronoun	VBP	Verb, non-3rd person singular present
JJR	Adjective, comparative	RB	Adverb	VBZ	Verb, 3rd person singular present ?
JJS	Adjective, superlative	RBR	Adverb, comparative	WDT	Wh-determiner
LS	List item marker	RBS	Adverb, superlative	WP	Wh-pronoun
MD	Modal	RP	Particle	WP$	Possessive wh-pronoun
NN	Noun, singular or mass ?	SYM	Symbol	WRB	Wh-adverb

So …how can we assign POS tags?

Time	flies	like	an	arrow	fruit	flies	like	a	banana
NN	VBZ	IN	DT	NN	NN	NNS			

CC	Coordinating Conjunction	**NNS**	Noun, plural ??	**TO**	to
CD	Cardinal Number	**NNP**	Proper noun, singular	**UH**	Interjection
DT	Determiner	**NNPS**	Proper noun, plural	**VB**	Verb, base form ??
EX	Existential *there*	**PDT**	Predeterminer	**VBD**	Verb, past tense
FW	Foreign word	**POS**	Possessive ending	**VBG**	Verb, gerund or present participle
IN	Preposition or subordinating conjunction ?	**PRP**	Personal pronoun	**VBN**	Verb, past participle
JJ	Adjective	**PRP$**	Possessive pronoun	**VBP**	Verb, non-3rd person singular present
JJR	Adjective, comparative	**RB**	Adverb	**VBZ**	Verb, 3rd person singular present ??
JJS	Adjective, superlative	**RBR**	Adverb, comparative	**WDT**	Wh-determiner
LS	List item marker	**RBS**	Adverb, superlative	**WP**	Wh-pronoun
MD	Modal	**RP**	Particle	**WP$**	Possessive wh-pronoun
NN	Noun, singular or mass ?	**SYM**	Symbol	**WRB**	Wh-adverb

So …how can we assign POS tags?

Time	flies	like	an	arrow	fruit	flies	like	a	banana
NN	VBZ	IN	DT	NN	NN	NNS	VBZ		

CC	Coordinating Conjunction	NNS	Noun, plural ??	TO	to
CD	Cardinal Number	NNP	Proper noun, singular	UH	Interjection
DT	Determiner :)	NNPS	Proper noun, plural	VB	Verb, base form ??
EX	Existential *there*	PDT	Predeterminer	VBD	Verb, past tense
FW	Foreign word	POS	Possessive ending	VBG	Verb, gerund or present participle
IN	Preposition or subordinating conjunction ??	PRP	Personal pronoun	VBN	Verb, past participle
JJ	Adjective	PRP$	Possessive pronoun	VBP	Verb, non-3rd person singular present
JJR	Adjective, comparative	RB	Adverb	VBZ	Verb, 3rd person singular present ???
JJS	Adjective, superlative	RBR	Adverb, comparative	WDT	Wh-determiner
LS	List item marker	RBS	Adverb, superlative	WP	Wh-pronoun
MD	Modal	RP	Particle	WP$	Possessive wh-pronoun
NN	Noun, singular or mass ?	SYM	Symbol	WRB	Wh-adverb

So …how can we assign POS tags?

Time	flies	like	an	arrow	fruit	flies	like	a	banana
NN	VBZ	IN	DT	NN	NN	NNS	VBZ	DT	

CC	Coordinating Conjunction	NNS	Noun, plural ??	TO	to
CD	Cardinal Number	NNP	Proper noun, singular	UH	Interjection
DT	Determiner	NNPS	Proper noun, plural	VB	Verb, base form ??
EX	Existential *there*	PDT	Predeterminer	VBD	Verb, past tense
FW	Foreign word	POS	Possessive ending	VBG	Verb, gerund or present participle
IN	Preposition or subordinating conjunction ??	PRP	Personal pronoun	VBN	Verb, past participle
JJ	Adjective	PRP$	Possessive pronoun	VBP	Verb, non-3rd person singular present
JJR	Adjective, comparative	RB	Adverb	VBZ	Verb, 3rd person singular present ???
JJS	Adjective, superlative	RBR	Adverb, comparative	WDT	Wh-determiner
LS	List item marker	RBS	Adverb, superlative	WP	Wh-pronoun
MD	Modal	RP	Particle	WP$	Possessive wh-pronoun
NN	Noun, singular or mass ?	SYM	Symbol	WRB	Wh-adverb

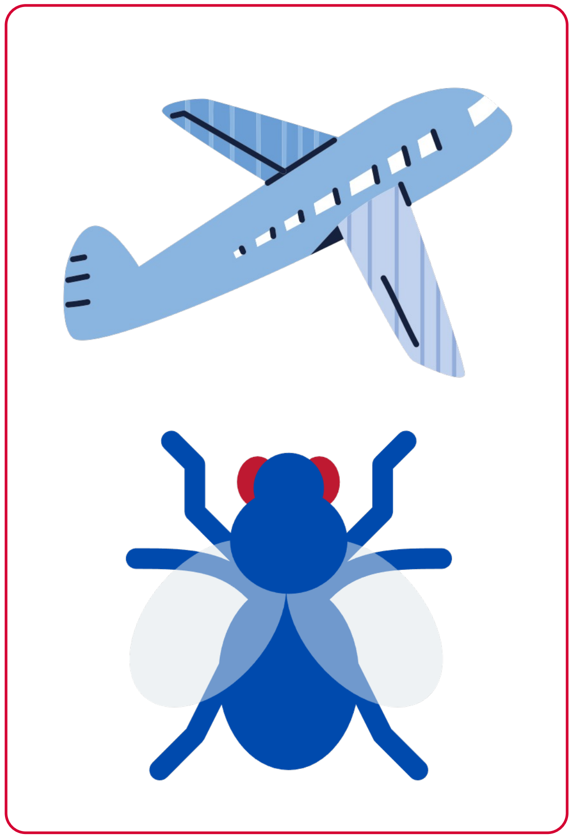
So …how can we assign POS tags?

Time	flies	like	an	arrow	fruit	flies	like	a	banana
NN	VBZ	IN	DT	NN	NN	NNS	VBZ	DT	NN

CC	Coordinating Conjunction	NNS	Noun, plural ??	TO	to
CD	Cardinal Number	NNP	Proper noun, singular	UH	Interjection
DT	Determiner :)	NNPS	Proper noun, plural	VB	Verb, base form ??
EX	Existential *there*	PDT	Predeterminer	VBD	Verb, past tense
FW	Foreign word	POS	Possessive ending	VBG	Verb, gerund or present participle
IN	Preposition or subordinating conjunction ??	PRP	Personal pronoun	VBN	Verb, past participle
JJ	Adjective	PRP$	Possessive pronoun	VBP	Verb, non-3rd person singular present
JJR	Adjective, comparative	RB	Adverb	VBZ	Verb, 3rd person singular present ???
JJS	Adjective, superlative	RBR	Adverb, comparative	WDT	Wh-determiner
LS	List item marker	RBS	Adverb, superlative	WP	Wh-pronoun
MD	Modal	RP	Particle	WP$	Possessive wh-pronoun
NN	Noun, singular or mass ?	SYM	Symbol	WRB	Wh-adverb

Ambiguity is a big issue for POS taggers!

- Many words have multiple senses
  - **time** = noun, verb
  - **flies** = noun, verb
  - **like** = verb, preposition
- Brown Corpus: Approximately 11% of word types have multiple valid part of speech labels, and many words with multiple valid POS labels are very common words
- Overall, ~40% of word *tokens* are instances of ambiguous word *types*



How do POS taggers work?

- Numerous ways to predict POS tags:
  - Rule-based
  - Statistical
    - HMMs
    - Neural sequence modeling

Rule-Based POS Tagging

- Start with a dictionary, and assign all relevant tags to the words in that dictionary
- Manually design rules to selectively remove invalid tags for test instances in context
- Keep the remaining correct tag for each word

Example Rule-Based Approach

she	promised	to	back	the	bill
PRP	VBN	TO	VB	DT	NN
	VBD		JJ		VB
			RB		
			NN		

Example Rule-Based Approach

Eliminate VBN if VBD is an option when VBN|VBD follows “<start> PRP”

she	promised	to	back	the	bill
PRP	~~VBN~~	TO	VB	DT	NN
	VBD		JJ		VB
			RB		
			NN		

Example Rule-Based Approach

she	promised	to	back	the	bill
PRP	~~VBN~~	TO	VB	DT	NN
	VBD		~~JJ~~		~~VB~~
			~~RB~~		
			~~NN~~		

Rule-based POS taggers are an adequate baseline, but….

- Like all rule-based methods, they are time-consuming to build, difficult to update or generalize to new domains, and may miss important patterns
- Simple alternative to rule-based POS tagging?
  - **Statistical POS Tagging:** POS taggers that make decisions based on learned knowledge of POS tag distribution in a training corpus
    - *the* is usually tagged as DT
    - Words with uppercase letters are more likely to be tagged NNP or NNPS
    - Words starting with the prefix *un-* may be tagged JJ
    - Words ending with the suffix *–ly* may be tagged RB

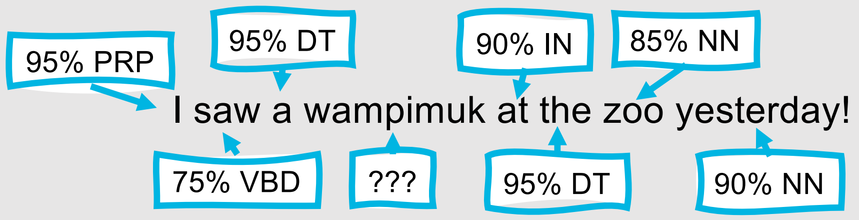
Example Statistical POS Tagger

- Using a training corpus, determine the most frequent tag for each word
- Assign POS tags to new words based on those frequencies
- Assign NN to new words for which there is no information from the training corpus

I saw a wampimuk at the zoo yesterday!

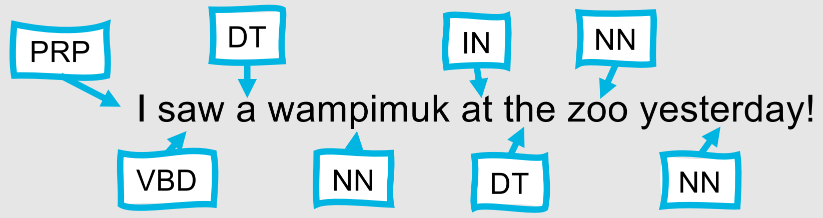
Example Statistical POS Tagger

- Using a training corpus, determine the most frequent tag for each word
- Assign POS tags to new words based on those frequencies
- Assign NN to new words for which there is no information from the training corpus



Example Statistical POS Tagger

- Using a training corpus, determine the most frequent tag for each word
- Assign POS tags to new words based on those frequencies
- Assign NN to new words for which there is no information from the training corpus



Experiments show that this approach achieves ~90% accuracy!

Bigram HMM POS Tagger

- We can improve upon the previous approach using HMMs
- To determine the tag *t_i* for a single word *w_i*:
  - $t_i = \underset{t_j \in \{t_0, t_1, \ldots, t_{t-1}\}}{\operatorname{argmax}} P(t_j | t_{i-1}) P(w_i | t_j)$
- This means we need to be able to compute two probabilities:
  - The probability that the tag is t_j given that the previous tag is t_{i-1}
    - $P(t_j | t_{i-1})$
  - The probability that the word is w_i given that the tag is t_j
    - $P(w_i | t_j)$
- We can compute both of these from corpora like the Penn Treebank or the Brown Corpus
- Then, we can find the most optimal sequence of tags using the Viterbi algorithm!

Example: Bigram HMM Tagger

Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR

Proper noun, singular

Verb, 3rd person singular present

Verb, past participle

Infinitive to

Adverbial noun

- Given two possible sequences of tags from the Brown Corpus tagset for the following sentence, what is the best way to tag the word “fly”?

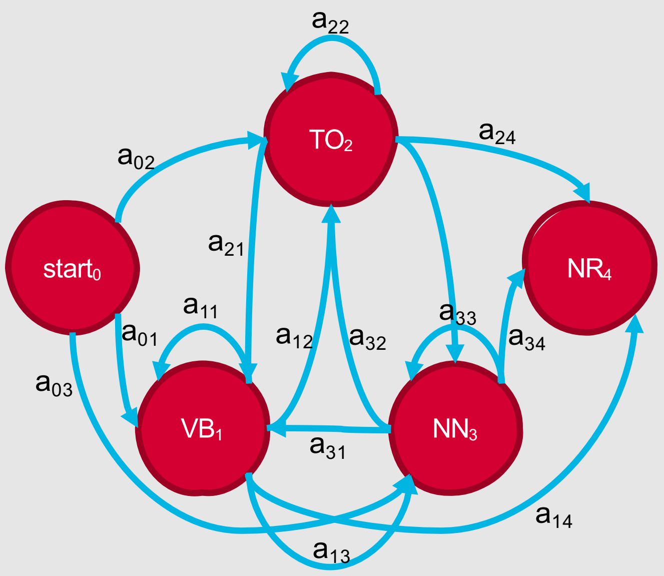
Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR

- Since we're creating a bigram HMM tagger and focusing on the word "fly," we only need to be concerned with the subsequence "to fly tomorrow"
  - For simplicity when decoding, we'll assume that:
    - The first word in the subsequence for sure has label TO (v_0(TO) = 1.0)
    - The word "tomorrow" for sure has label NR (P("tomorrow"|NR) = 1.0)

Example: Bigram HMM Tagger

Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR

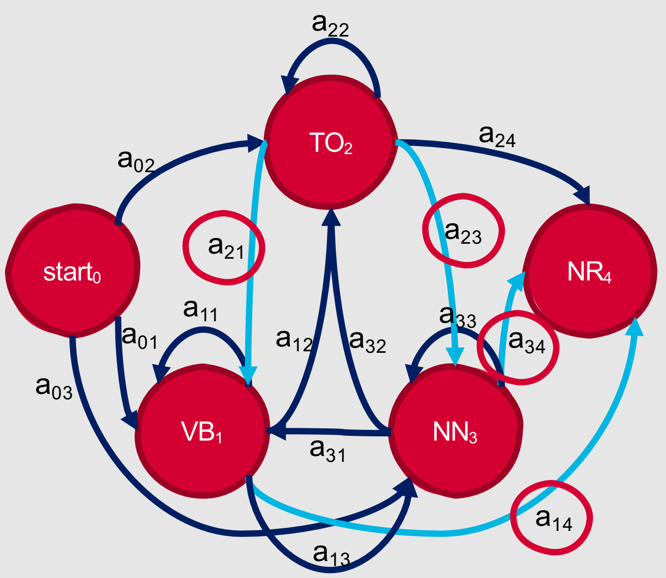
We have the following HMM sample:



Example: Bigram HMM Tagger

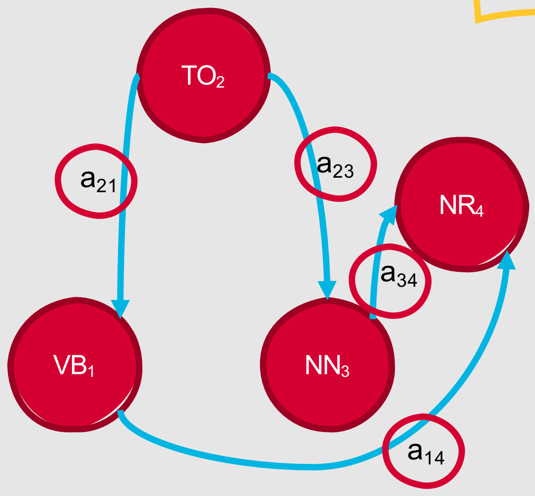
Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR

The specific transition probabilities we are interested in are:



Example: Bigram HMM Tagger

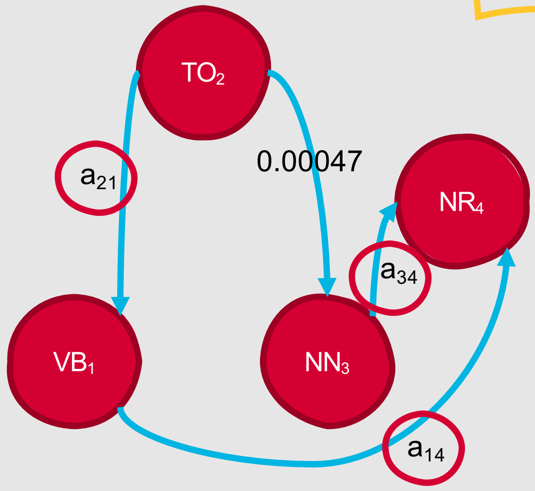
Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR



- We can estimate the transition probabilities for a_{21}, a_{23}, a_{34}, and a_{14} using frequency counts from the Brown Corpus
- $P(t_i|t_{i-1}) = \frac{c(t_{i-1}t_i)}{c(t_{i-1})}$

Example: Bigram HMM Tagger

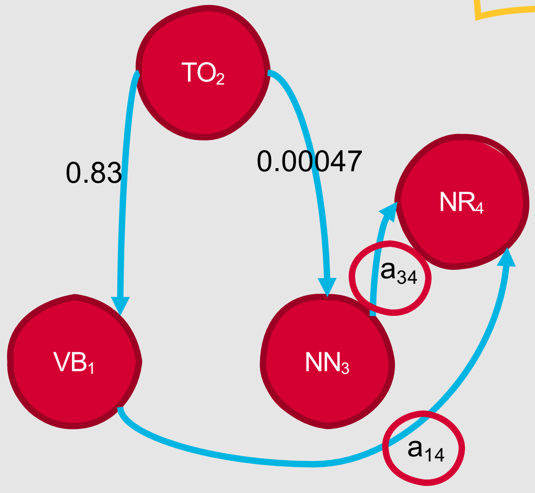
Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR



- We can estimate the transition probabilities for a_{21}, a_{23}, a_{34}, and a_{14} using frequency counts from the Brown Corpus
- $P(t_i|t_{i-1}) = \frac{c(t_{i-1}t_i)}{c(t_{i-1})}$
- So, P(NN|TO) = C(TO NN) / C(TO) = 0.00047

Example: Bigram HMM Tagger

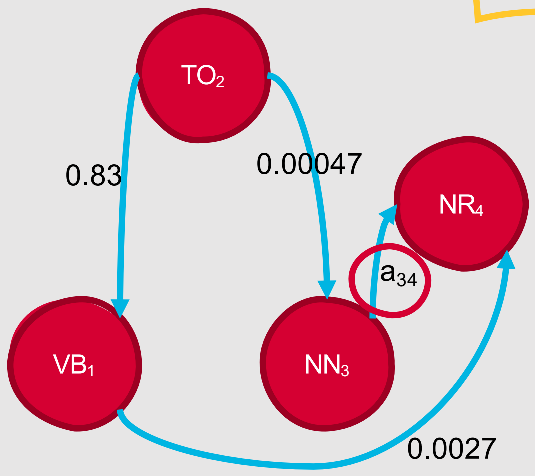
Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR



- We can estimate the transition probabilities for a_{21}, a_{23}, a_{34}, and a_{14} using frequency counts from the Brown Corpus
- $P(t_i|t_{i-1}) = \frac{c(t_{i-1}t_i)}{c(t_{i-1})}$
- So, P(NN|TO) = C(TO NN) / C(TO) = 0.00047
- Likewise, P(VB|TO) = C(TO VB) / C(TO) = 0.83

Example: Bigram HMM Tagger

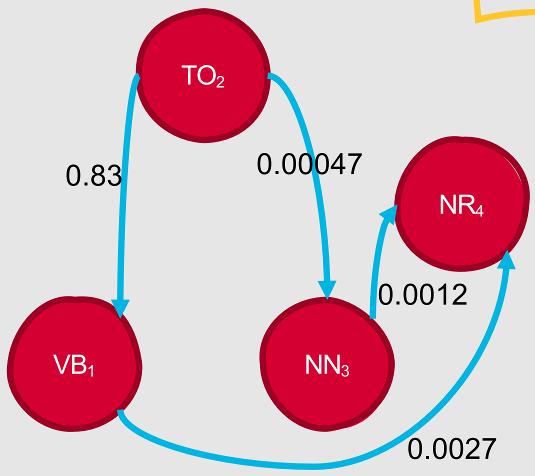
Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR



- We can estimate the transition probabilities for a_{21}, a_{23}, a_{34}, and a_{14} using frequency counts from the Brown Corpus
- $P(t_i|t_{i-1}) = \frac{c(t_{i-1}t_i)}{c(t_{i-1})}$
- So, P(NN|TO) = C(TO NN) / C(TO) = 0.00047
- Likewise, P(VB|TO) = C(TO VB) / C(TO) = 0.83
- P(NR|VB) = C(VB NR) / C(VB) = 0.0027

Example: Bigram HMM Tagger

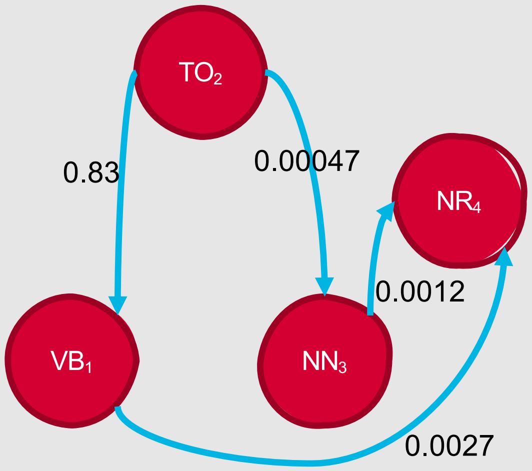
Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR



- We can estimate the transition probabilities for a_{21}, a_{23}, a_{34}, and a_{14} using frequency counts from the Brown Corpus
- $P(t_i|t_{i-1}) = \frac{c(t_{i-1}t_i)}{c(t_{i-1})}$
- So, P(NN|TO) = C(TO NN) / C(TO) = 0.00047
- Likewise, P(VB|TO) = C(TO VB) / C(TO) = 0.83
- P(NR|VB) = C(VB NR) / C(VB) = 0.0027
- Finally, P(NR|NN) = C(NN NR) / C(NN) = 0.0012

Example: Bigram HMM Tagger

Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR

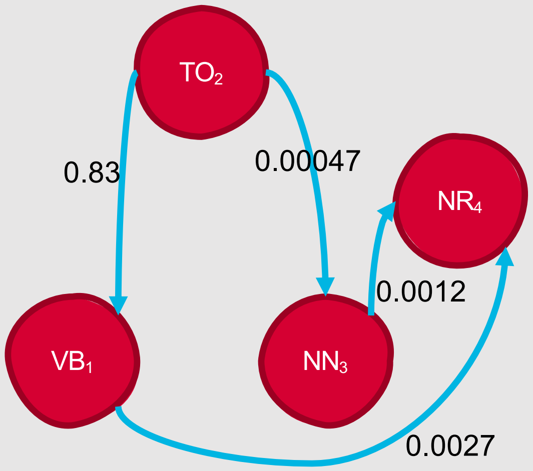


	fly
VB	
NN	

- We have our transition probabilities …what now?
- Observation likelihoods!
- We can also estimate these using frequency counts from the Brown Corpus
- $P(w_i|t_i) = \frac{c(w_i,t_i)}{c(t_i)}$
- Since we're trying to decide the best tag for "fly," we need to compute both P(fly|VB) and P(fly|NN)

Example: Bigram HMM Tagger

Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR

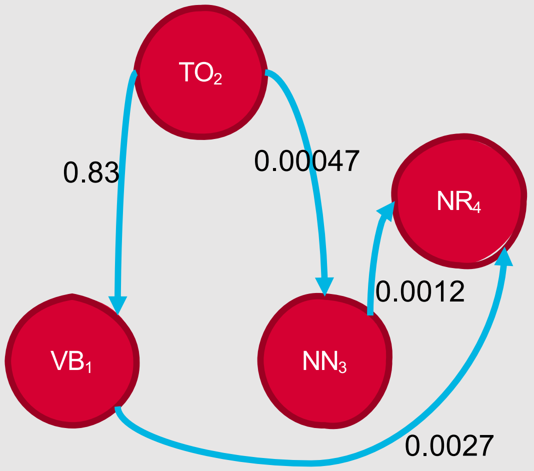


	fly
VB	0.00012
NN	

- We have our transition probabilities …what now?
- Observation likelihoods!
- We can also estimate these using frequency counts from the Brown Corpus
- $P(w_i|t_i) = \frac{c(w_i,t_i)}{c(t_i)}$
- Since we're trying to decide the best tag for "fly," we need to compute both P(fly|VB) and P(fly|NN)
- P(fly|VB) = C(fly, VB) / C(VB) = 0.00012

Example: Bigram HMM Tagger

Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR

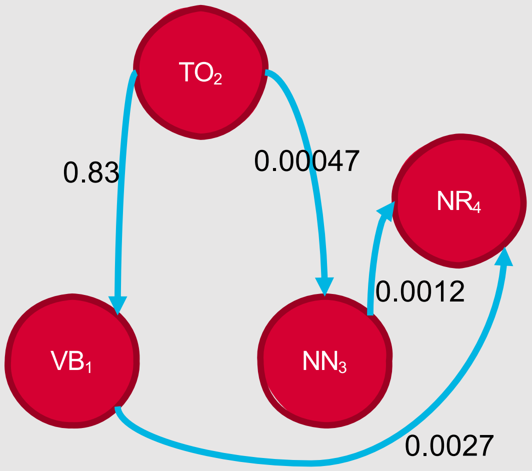


	fly
VB	0.00012
NN	0.00057

- We have our transition probabilities …what now?
- Observation likelihoods!
- We can also estimate these using frequency counts from the Brown Corpus
- $P(w_i|t_i) = \frac{c(w_i,t_i)}{c(t_i)}$
- Since we're trying to decide the best tag for "fly," we need to compute both P(fly|VB) and P(fly|NN)
- P(fly|VB) = C(fly, VB) / C(VB) = 0.00012
- P(fly|NN) = C(fly, NN) / C(NN) = 0.00057

Example: Bigram HMM Tagger

Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	TO	VB	NR
NNP	VBZ	VBN	TO	NN	NR

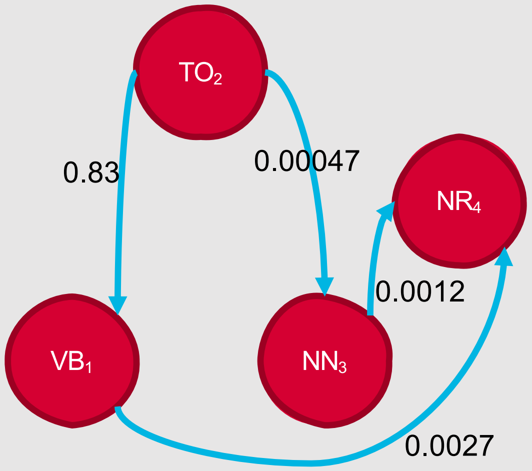


	fly
VB	0.00012
NN	0.00057

- Now, to decide how to tag "fly," we can consider our two possible sequences:
  - to (TO) fly (VB) tomorrow (NR)
  - to (TO) fly (NN) tomorrow (NR)
- We will select the tag that maximizes the probability:
  - $P(t_i|TO)P(NR|t_i)P(fly|t_i)$
- We determine that:
  - P(VB|TO)P(NR|VB)P(fly|VB) = 0.83 * 0.0027 * 0.00012 = 0.00000027
  - P(NN|TO)P(NR|NN)P(fly|NN) = 0.00047 * 0.0012 * 0.00057 = 0.00000000032

Example: Bigram HMM Tagger

Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	**TO**	**VB**	**NR**
NNP	VBZ	VBN	TO	NN	NR



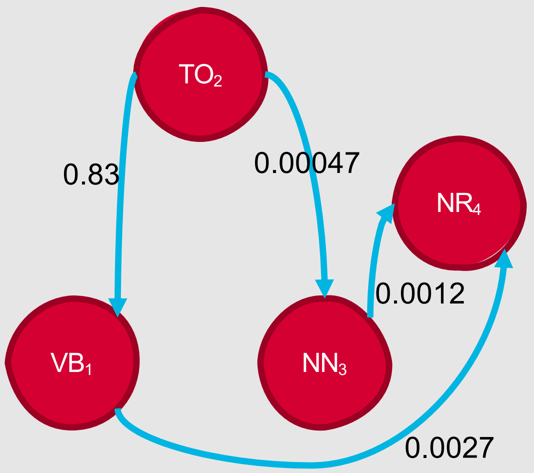
	fly
VB	0.00012
NN	0.00057

- Now, to decide how to tag "fly," we can consider our two possible sequences:
  - to (TO) fly (VB) tomorrow (NR)
  - to (TO) fly (NN) tomorrow (NR)
- We will select the tag that maximizes the probability:
  - $P(t_i|TO)P(NR|t_i)P(fly|t_i)$
- We determine that:
  - **P(VB|TO)P(NR|VB)P(fly|VB) = 0.83 * 0.0027 * 0.00012 = 0.00000027**
    - Optimal sequence!
  - P(NN|TO)P(NR|NN)P(fly|NN) = 0.00047 * 0.0012 * 0.00057 = 0.00000000032

Example: Bigram HMM Tagger

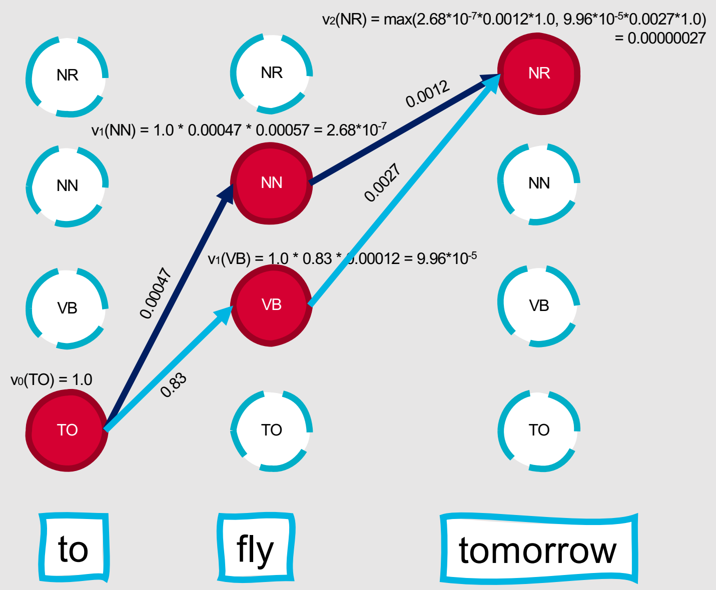
Example: Bigram HMM Tagger

Superman	is	expected	to	fly	tomorrow
NNP	VBZ	VBN	**TO**	**VB**	**NR**
NNP	VBZ	VBN	TO	NN	NR



	fly
VB	0.00012
NN	0.00057

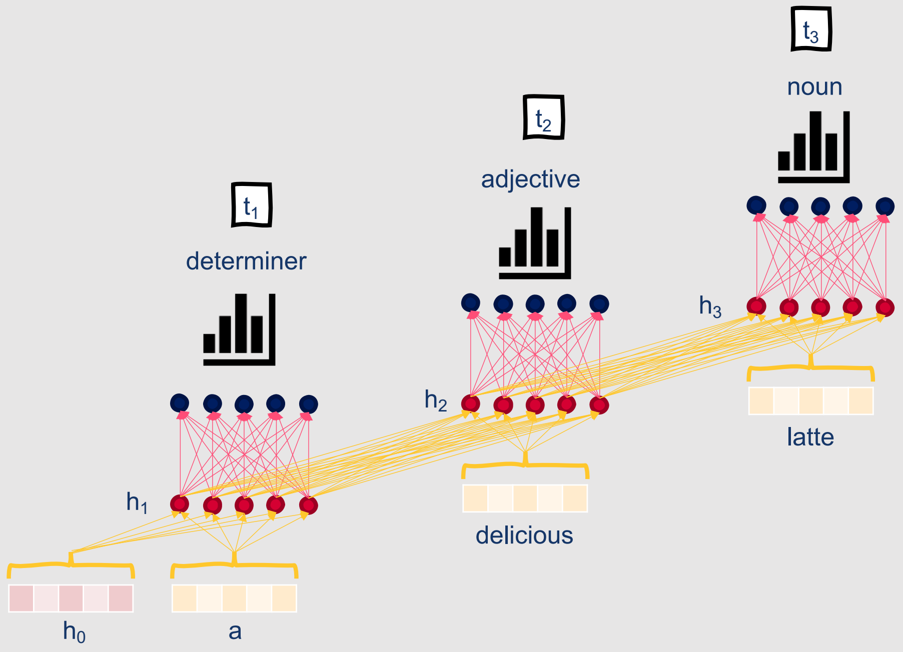
- Visualized in a Viterbi trellis, this would look like:



$v_2(NR) = \max(2.68*10^{-7}*0.0012*1.0, 9.96*10^{-5}*0.0027*1.0) = 0.00000027$
NR
NR
NR
0.0012
$v_1(NN) = 1.0 * 0.00047 * 0.00057 = 2.68*10^{-7}$
NN
NN
NN
0.0027
$v_1(VB) = 1.0 * 0.83 * 0.00012 = 9.96*10^{-5}$
0.00047
VB
VB
VB
$v_0(TO) = 1.0$
0.83
TO
TO
TO
to
fly
tomorrow

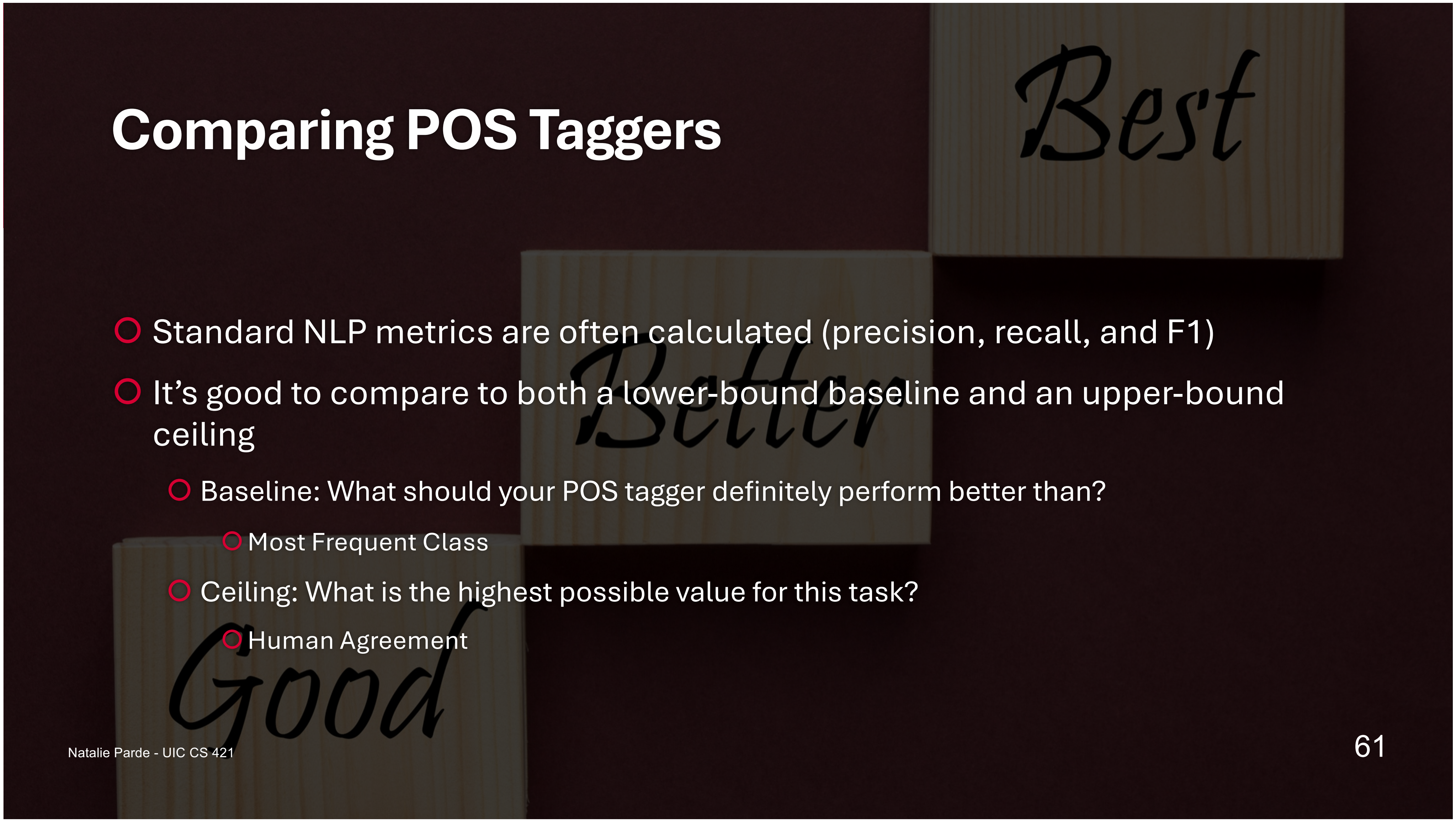
We can also perform POS tagging using neural sequence modeling.

- Use a sequential or pretrained neural network architecture
  - Recurrent neural networks
  - Transformers
- Predict a label for each item in the input sequence
  - If using a subword vocabulary, you will need to merge the labels predicted for all subwords in a word



How can POS taggers handle unknown words?

- New words are continually added to language, so it is likely that a POS tagger will encounter words not found in its training corpus
- Easy baseline approach: **Assume that unknown words are nouns**
- More sophisticated approach: **Assume that unknown words have a probability distribution similar to other words occurring only once in the training corpus**, and make an (informed) random choice
- Even more sophisticated approach: **Use morphological information** to choose the POS tag (for example, words ending with “ed” tend to be tagged VBN)



Comparing POS Taggers

- Standard NLP metrics are often calculated (precision, recall, and F1)
- It's good to compare to both a lower-bound baseline and an upper-bound ceiling
  - Baseline: What should your POS tagger definitely perform better than?
    - Most Frequent Class
  - Ceiling: What is the highest possible value for this task?
    - Human Agreement

This Week's Topics

Tuesday

Parts of Speech
POS Tagsets
POS Tagging
Context-Free Grammars

Thursday

Hierarchical Parsing
Dynamic Programming Parsing Algorithms
Probabilistic CKY
Lexicalized Grammars

POS tags are one way to formalize language structure.

- Constituency grammars are another!
- Constituency grammars define language using a lexicon and a set of rules to break sentences into hierarchical parts
- They provide the necessary structure to answer important questions:
  - What are the **constituents** (groups of words that behave as a single unit or phrase) in this sentence?
  - What are the **grammatical relations** between these constituents?
  - Which words are **dependent** upon one another?
- **Constituency grammars model sentences as recursive generating processes**
  - Usually, this is done using a tree structure

Grammar Formalisms vs. Specific Grammars

- **Grammar Formalisms:** A precise way to define and describe the structure of independent sentences.
- **Specific Grammars:** Implementations (according to a specific formalism) for a particular language
  - English, Arabic, Mandarin, or Hindi
- In general, our specific grammars are close but imperfect ways to formalize a language
  - For example: There are an infinite number of possible English sentences, but our specific grammar for English needs to be finite

It’s all about finding the right balance!

Overgeneration:

Love NLP class my so much that don’t care about being it early morning in!

Did get the you email guy that that from class said he forward to you would?

Well, there just happened.

English:

I love my NLP class so much that I don’t even care about it being in early morning!

Did you get the email that that guy from class said he would forward to you?

Well, that just happened.

Undergeneration:

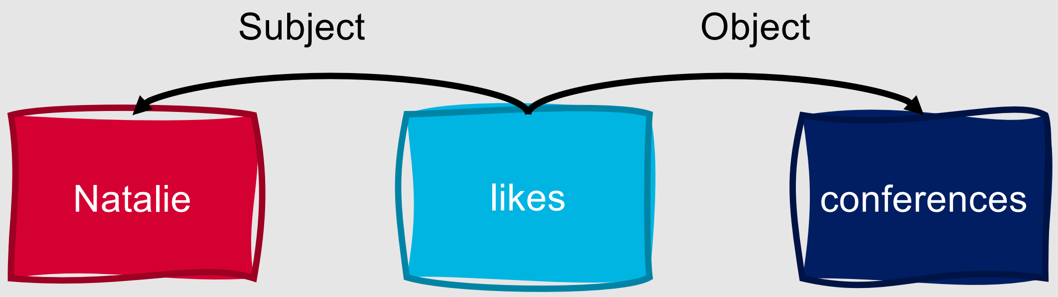
I love my class!

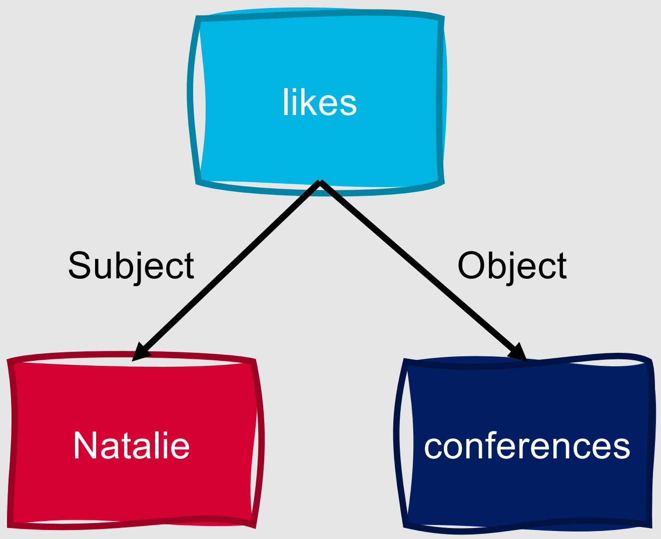
Did you get his email?

What happened?

Visually, we can represent grammars in many ways.

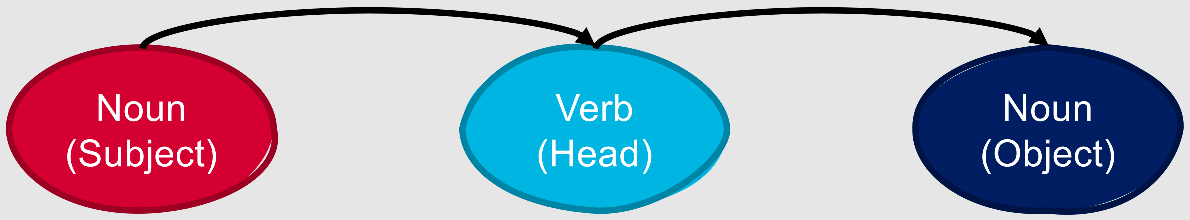
As a dependency graph:





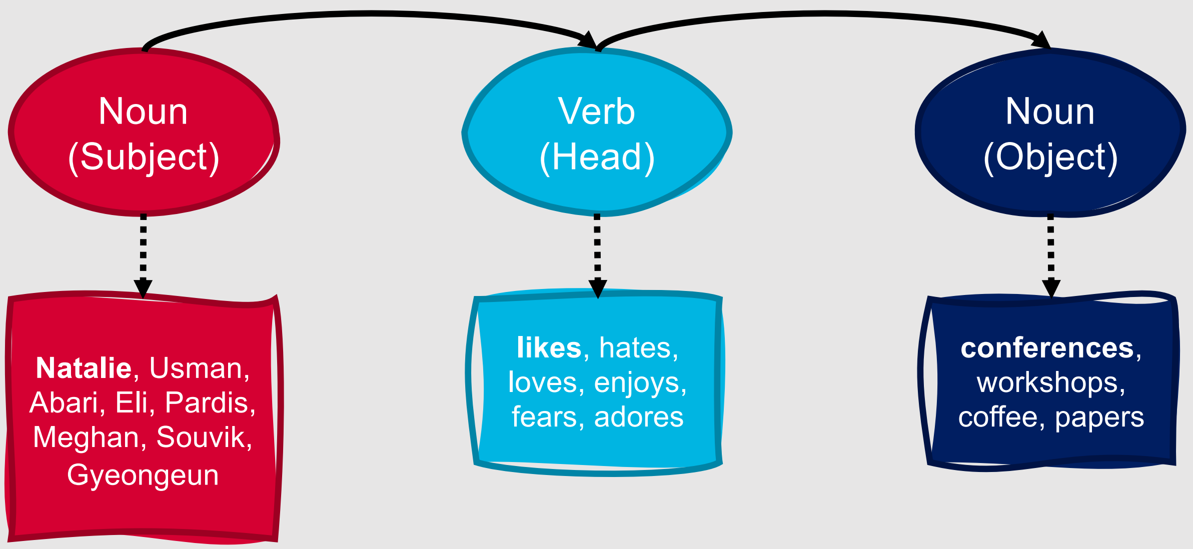
Visually, we can represent grammars in many ways.

As a finite state automaton:



Visually, we can represent grammars in many ways.

As a hidden Markov model:

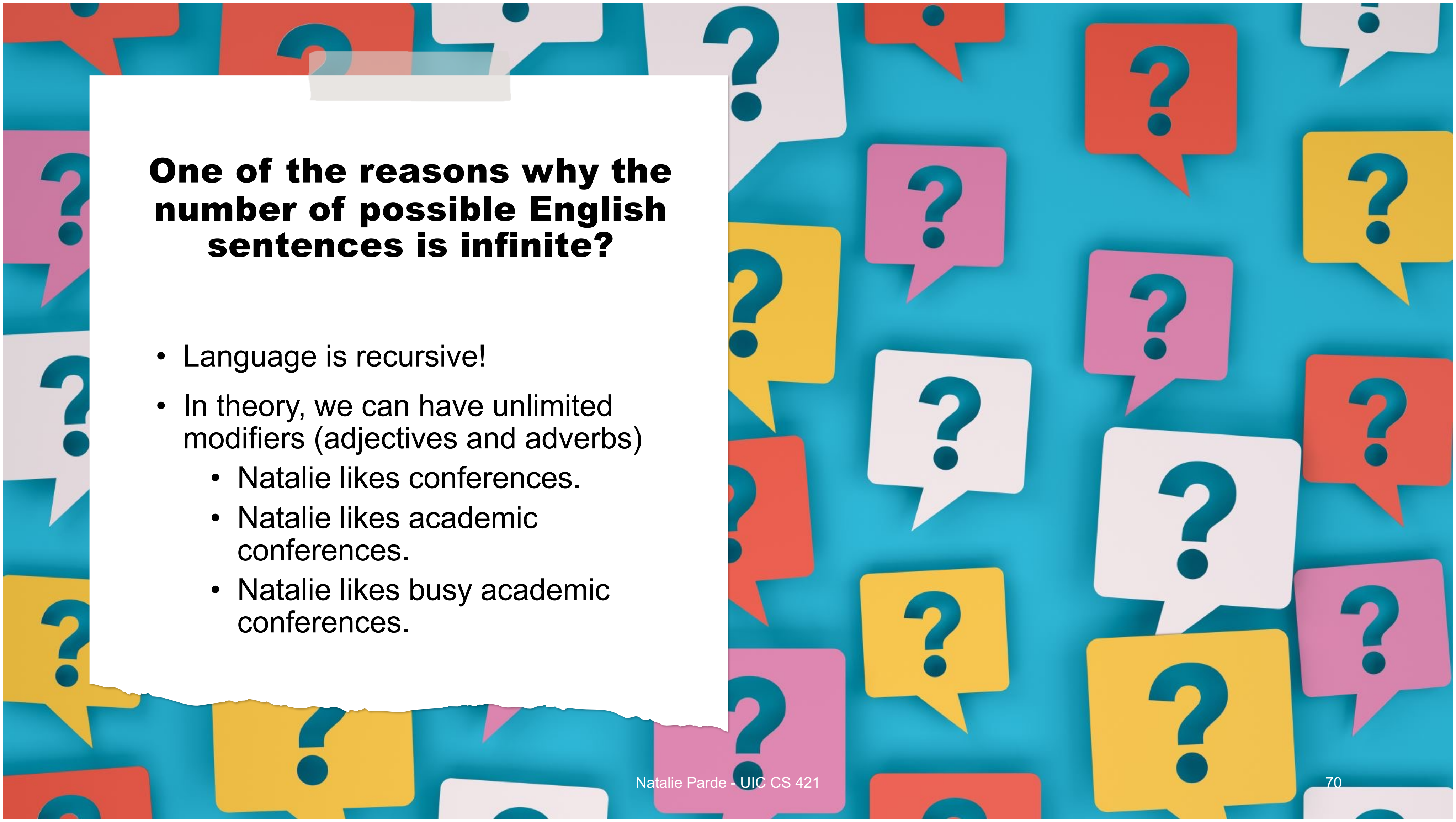

Noun (Subject)
Verb (Head)
Noun (Object)
Natalie, Usman, Abari, Eli, Pardis, Meghan, Souvik, Gyeongeun
likes, hates, loves, enjoys, fears, adores
conferences, workshops, coffee, papers

Different types of words accept different types of arguments.

- **Subcategorization:** Syntactic constraints on the set of arguments that a group of words will accept.
  - **Intransitive verbs** accept only subjects
    - Sleep, arrive
  - **Transitive verbs** accept a subject and a direct object
    - Eat, drink
  - **Ditransitive verbs** accept a subject, a direct object, and an indirect object
    - Give, make

Natalie likes conferences. 🙂

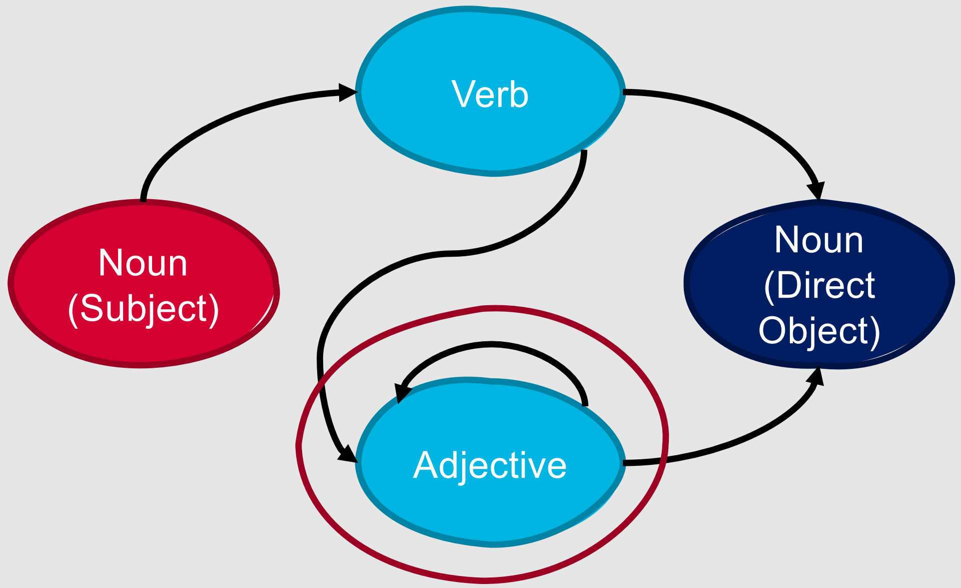
Natalie drinks conferences. 🤨



One of the reasons why the number of possible English sentences is infinite?

- Language is recursive!
- In theory, we can have unlimited modifiers (adjectives and adverbs)
  - Natalie likes conferences.
  - Natalie likes academic conferences.
  - Natalie likes busy academic conferences.

We can easily model *simple* cases of recursion in a finite state model.

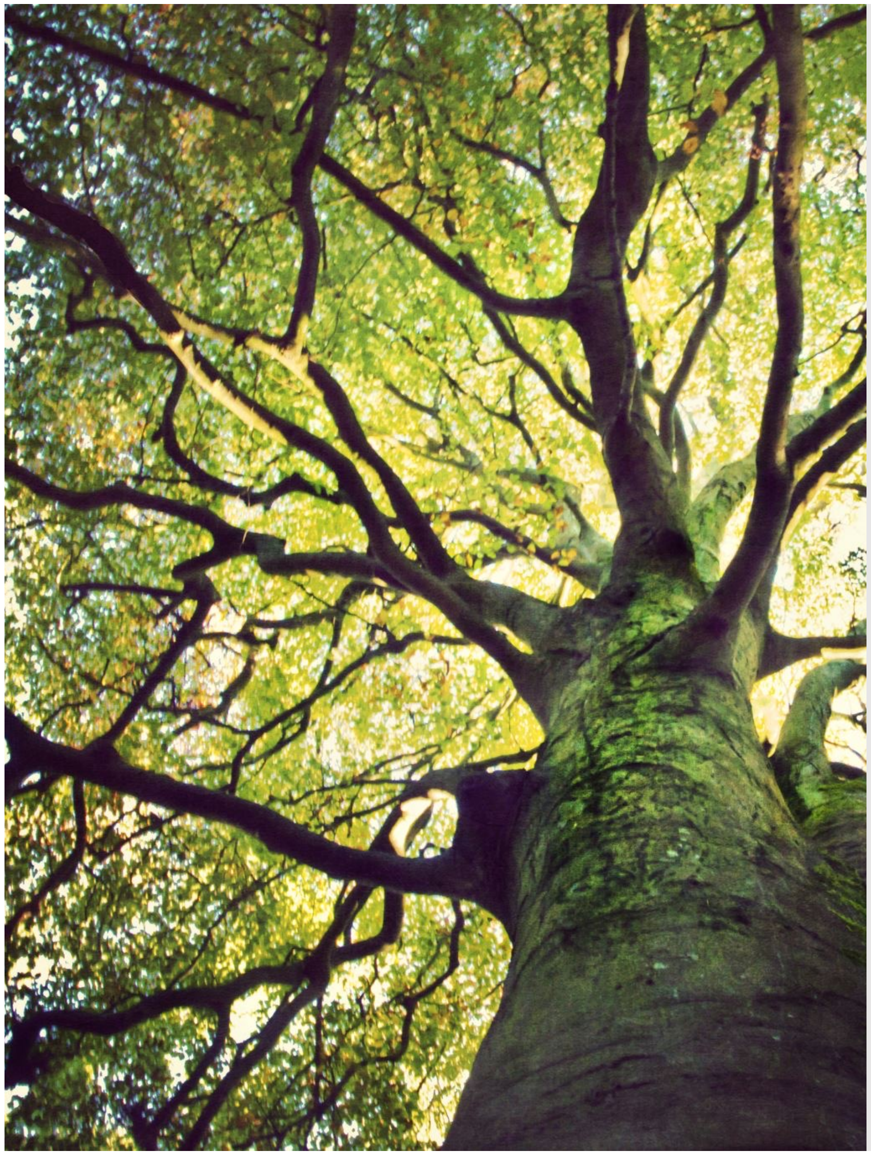


However, recursion in sentences can also be more complex.

- FSAs can model recursion, but they can't model hierarchical structure or handle issues like **attachment ambiguity**

Natalie likes conferences in either Europe or Asia. ← Natalie **likes conferences in** Europe **or** Asia.

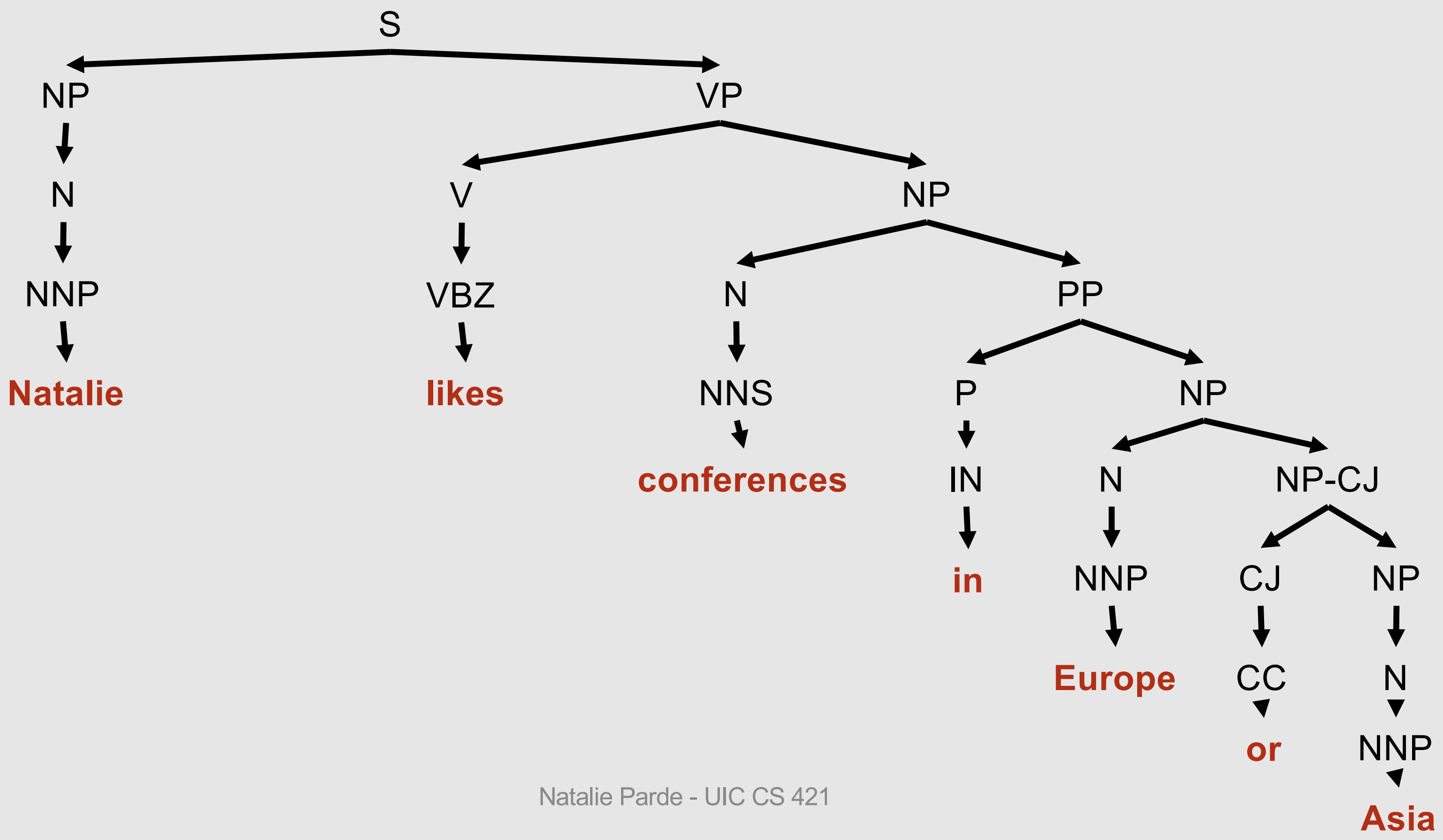
Natalie **likes** conferences in Europe **or** Asia. → Natalie likes two things: Asia, or conferences in Europe.



Hierarchical trees to the rescue!

- Words in a sentence can be grouped into phrases (**constituents**) using a hierarchical structure
- Formal trees will usually have **internal (non-terminal) nodes** and **outer (terminal) leaves**
- **Nodes: Elements of sentence structure**
  - Constituent type
  - POS type
- **Leaves: Surface wordforms**
- The nodes and leaves are connected to one another by **branches**

What does this look like?



We use context-free grammars to define these hierarchical trees.

- **Context-Free Grammar (CFG):** A system to define constituent structure in regular languages, defined by productions that indicate which strings can be generated.
  - **Production:** Rules expressing the allowable combinations of symbols (e.g., POS types) that can form a constituent
  - Productions can be **hierarchically embedded**
    - Noun Phrase (NP) → Determiner Nominal
    - Nominal → Noun | Nominal Noun
- Why is it called context-free?
  - A subtree can be replaced by a production rule independent of the greater context (other nodes in the hierarchy) in which it occurs.
- Also called **Phrase-Structure Grammars**

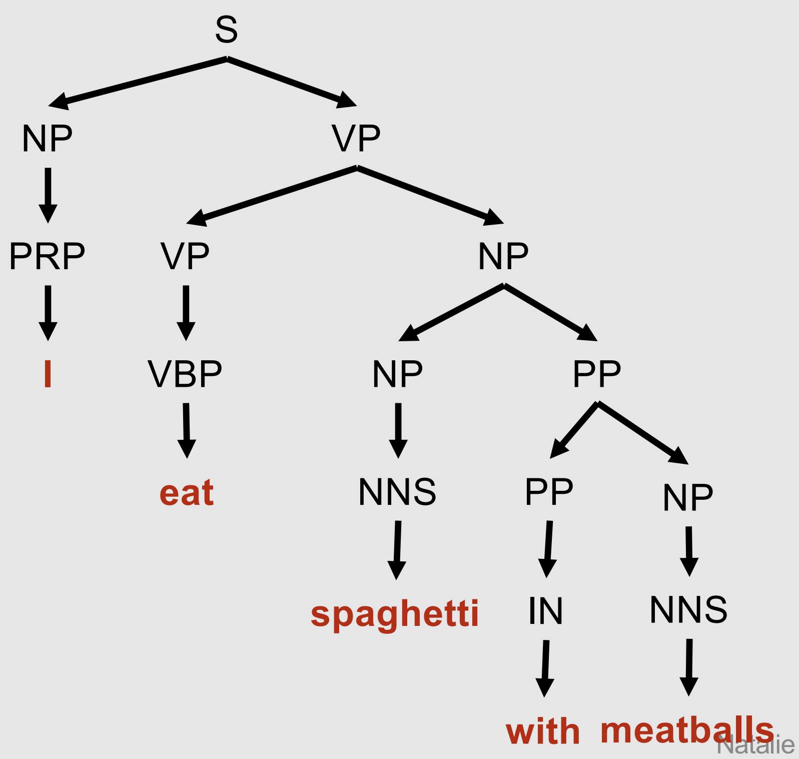
Formal Definition

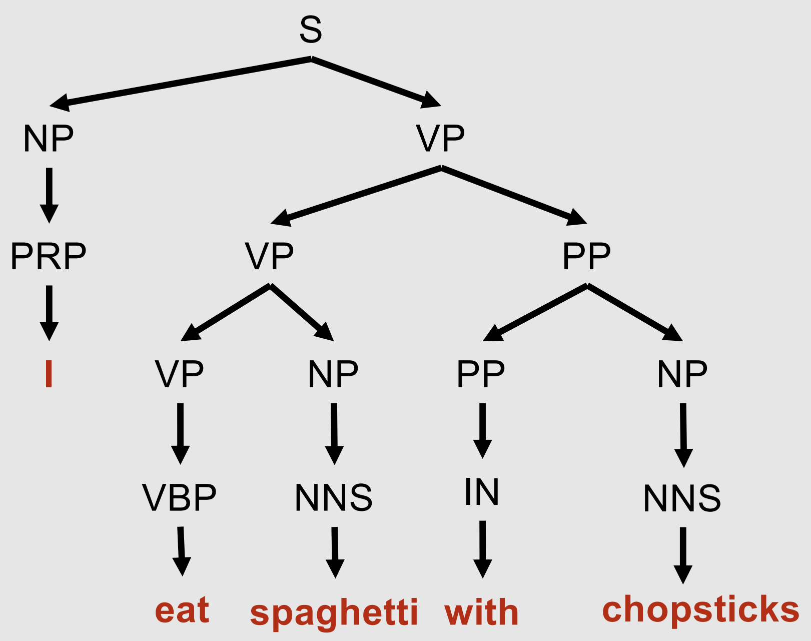
- A CFG is a 4-tuple $\langle \boldsymbol{N}, \boldsymbol{\Sigma}, \boldsymbol{R}, \boldsymbol{S} \rangle$ consisting of:
  - A set of non-terminal nodes ***N***
    - ***N*** = {S, NP, VP, PP, N, V, …}
  - A set of terminal nodes (leaves) $\boldsymbol{\Sigma}$
    - $\boldsymbol{\Sigma}$ = {time, flies, like, an, arrow, …}
  - A set of rules ***R***
  - A start symbol ***S*** ∈ ***N***
- How to check for **grammatical correctness**?
  - Any sentences for which the CFG can construct a tree (all words in the sentence must be reachable as leaf nodes) are accepted by the CFG.

Production rules determine how constituents can be combined.

- **Constituent:** A group of words that behaves as a single unit.
  - **Constituents can be substituted with one another** in the context of the greater sentence
  - **A constituent can move around** within the context of the sentence
  - **A constituent can be used to answer a question** about the sentence
- Constituents contain **heads** and **dependents**
  - **Head:** The most informative word in the constituent
  - **Dependent:** The other word that contributes to the overall meaning
- Dependents can be arguments or adjuncts
  - Arguments are **obligatory**
  - Adjuncts are **optional**

The structure of constituents in a tree corresponds to their meaning.





Case Example

- Draw a constituent tree for the sentence:
  - **Time flies like an arrow.**

Production Rules	
S ! NP VP	PP ! P NP
NP ! DET N	PP ! P
NP ! N	P ! like
NP ! N N	V ! flies | like
VP ! VP PP	DET ! a | an
VP ! V NP	N ! time | fruit | flies | arrow | banana
VP ! V	

Case Example

Production Rules	
S ! NP VP	PP ! P NP
NP ! DET N	PP ! P
NP ! N	P ! like
NP ! N N	V ! flies | like
VP ! VP PP	DET ! a | an
VP ! V NP	N ! time | fruit | flies | arrow | banana
VP ! V	

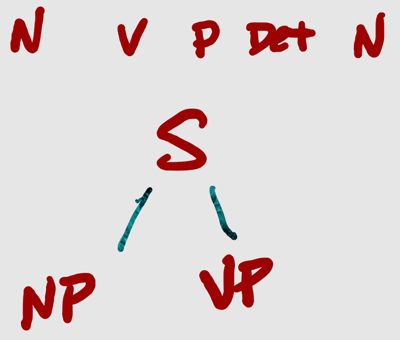
Time flies like an arrow

N V P Det N

Case Example

Production Rules	
S ! NP VP	PP ! P NP
NP ! DET N	PP ! P
NP ! N	P ! like
NP ! N N	V ! flies | like
VP ! VP PP	DET ! a | an
VP ! V NP	N ! time | fruit | flies | arrow | banana
VP ! V	

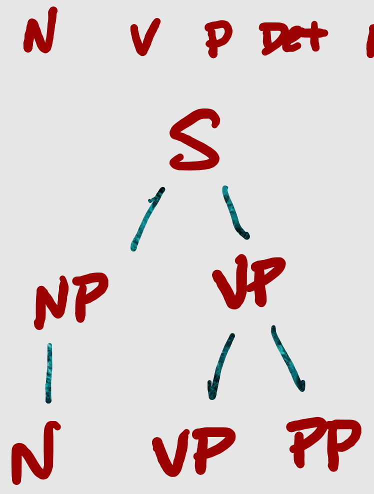
Time flies like an arrow



Case Example

Production Rules	
S ! NP VP	PP ! P NP
NP ! DET N	PP ! P
NP ! N	P ! like
NP ! N N	V ! flies | like
VP ! VP PP	DET ! a | an
VP ! V NP	N ! time | fruit | flies | arrow | banana
VP ! V	

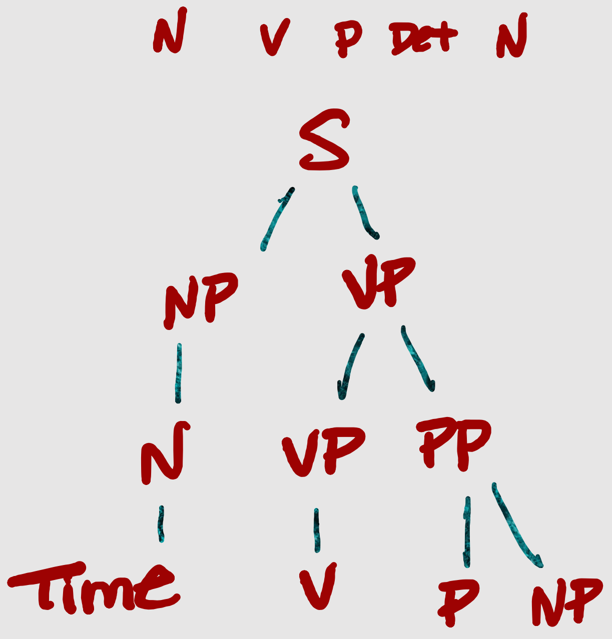
Time flies like an arrow



Case Example

Production Rules	
S ! NP VP	PP ! P NP
NP ! DET N	PP ! P
NP ! N	P ! like
NP ! N N	V ! flies | like
VP ! VP PP	DET ! a | an
VP ! V NP	N ! time | fruit | flies | arrow | banana
VP ! V	

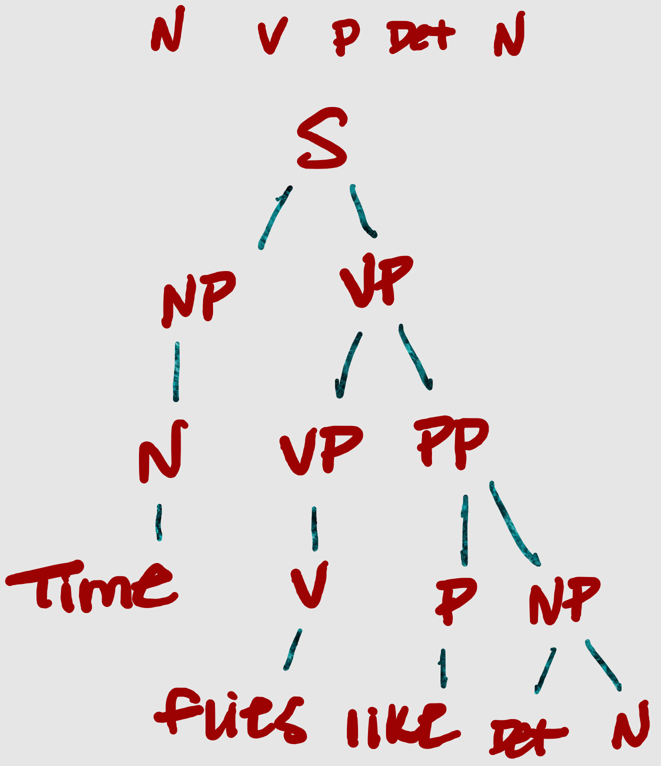
Time flies like an arrow



Case Example

Production Rules	
S ! NP VP	PP ! P NP
NP ! DET N	PP ! P
NP ! N	P ! like
NP ! N N	V ! flies | like
VP ! VP PP	DET ! a | an
VP ! V NP	N ! time | fruit | flies | arrow | banana
VP ! V	

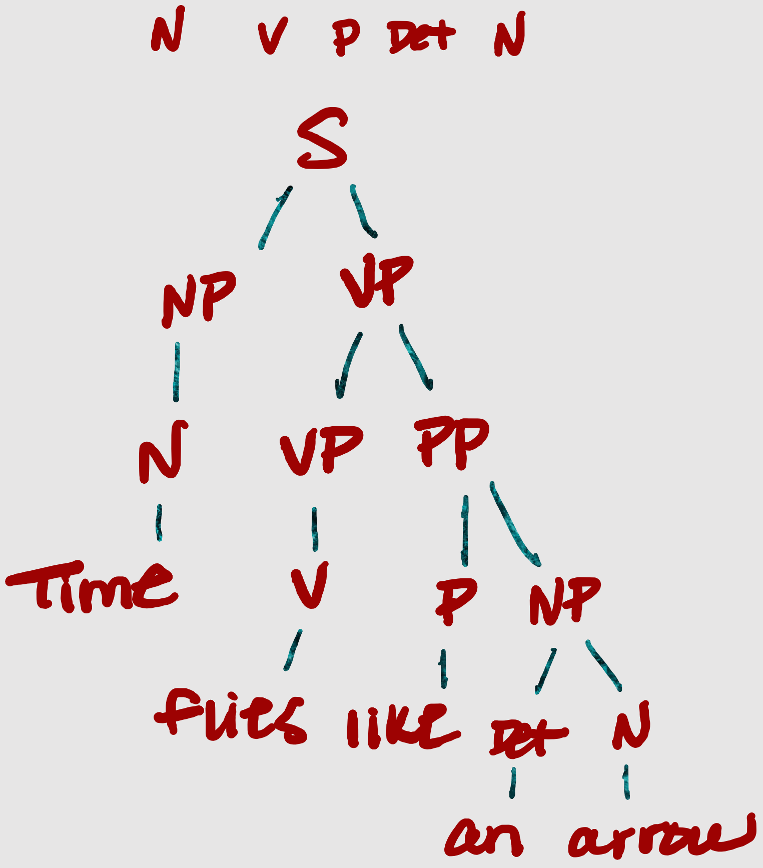
Time flies like an arrow



Case Example

Production Rules	
S ! NP VP	PP ! P NP
NP ! DET N	PP ! P
NP ! N	P ! like
NP ! N N	V ! flies | like
VP ! VP PP	DET ! a | an
VP ! V NP	N ! time | fruit | flies | arrow | banana
VP ! V	

Time flies like an arrow



Refresher: Typical CFG Constituents (English)

Noun phrases (NPs)

- Simple:
  - **She** talks. (**pronoun**)
  - **Natalie** talks. (**proper noun**)
  - **A person** talks. (**determiner** + **common noun**)
- Complex:
  - **A professorial person** talks. (**determiner** + **adjective** + **common noun**)
  - **The person at the lectern** talks. (**noun phrase (determiner + common noun)** + **prepositional phrase**)
  - **The person who teaches NLP** talks. (**noun phrase (determiner + common noun)** + **relative clause**)

Visualized as production rules:

- NP → Pronoun
- NP → Proper Noun
- NP → Determiner Common Noun
- NP → Determiner Adjective Common Noun
- NP → NP PP
- NP → NP RelClause
- Pronoun → {she}
- Determiner → {a}
- Proper Noun → {Natalie}
- Common Noun → {person}
- Adjective → {professorial}

Refresher: Typical CFG Constituents (English)

Verb Phrases (VPs)

- She **drinks**. (**verb**)
- She **drinks** **tea**. (**verb** + **noun phrase**)
- She **drinks tea** **from a mug**. (**verb phrase** + **prepositional phrase**)
- Visualized as production rules:
  - VP → V
  - VP → V NP
  - VP → V NP PP
  - VP → VP PP
  - V → {drinks}

We can also capture subcategorization this way!

- She **drinks**. (**verb**)
- She **drinks** **tea**. (**verb** + **noun phrase**)
- She **gives** **him** **tea**. (**verb phrase** + **noun phrase** + **noun phrase**)
- Visualized as production rules:
  - VP → $V_{intransitive}$
  - VP → $V_{transitive}$ NP
  - VP → $V_{ditransitive}$ NP NP
  - $V_{intransitive}$ → {drinks, talks}
  - $V_{transitive}$ → {drinks}
  - $V_{ditransitive}$ → {gives}

To comprehensively cover English grammar, more complex production rules are necessary.

- We want to prevent against grammatical incorrectness:
  - She drinks tea. 🙂
  - I drinks tea. 🤨
  - They drinks tea. 🤨
- We can do this by establishing different production rules for different tenses or other phenomena:
  - Present Tense: She drinks tea.
  - Simple Past Tense: She drank tea.
  - Past Perfect Tense: She has drunk tea.
  - Future Perfect Tense: She will have drunk tea.
  - Passive: The tea was drunk by her.
  - Progressive: She will be drinking tea.

- $VP \rightarrow V_{have}\ VP_{pastPart}$
- $VP \rightarrow V_{be}\ VP_{pass}$
- $VP_{pastPart} \rightarrow V_{pastPart}\ NP$
- $VP_{pass} \rightarrow V_{pastPart}\ PP$
- $V_{have} \rightarrow \{has\}$
- $V_{pastPart} \rightarrow \{drunk\}$
- etc….

Refresher: Typical CFG Constituents (English)

- Production rules can also recursively include sentences
  - She drinks tea. (noun phrase + verb phrase)
  - Sometimes, she drinks tea. (adverbial phrase + sentence)
  - In England, she drinks tea. (prepositional phrase + sentence)
- Visualized as production rules:
  - S → NP VP
  - S → AdvP S
  - S → PP S
- They can include coordinating conjunctions:
  - **She drinks tea** and **he drinks coffee**.
  - **Natalie** and **her mom** drink tea.
  - She **drinks tea** and **eats cake**.
  - Production Rules:
    - S → S conj S
    - NP → NP conj NP
    - VP → VP conj VP
- They can use relative clauses to add extra information to noun phrases:
  - Subject: She had a poodle **that drank my tea**.
    - We cannot drop the relative pronoun and keep the same meaning
  - Object: I'd really been enjoying the tea **that her poodle drank**.
    - We can drop the relative pronoun and the sentence still works

Summary: Part-of-Speech Tagging and Constituency Grammars

- **POS tagging** is the process of automatically assigning grammatical word classes (parts of speech) to individual tokens
- The most common POS tagset is the **Penn Treebank** tagset
- **Ambiguity** is common in natural language, and is a major issue that **POS taggers** must address
- **Constituency grammars** describe a language's syntactic structure
- **Constituents**, a core component of constituency grammars, are groups of words that function as a single unit
- There are many ways to represent constituency grammars, but the most common way is by using **trees**
- Constituency grammars can generate any sentences belonging to their language using (potentially recursive) combinations of **production rules**

This Week's Topics

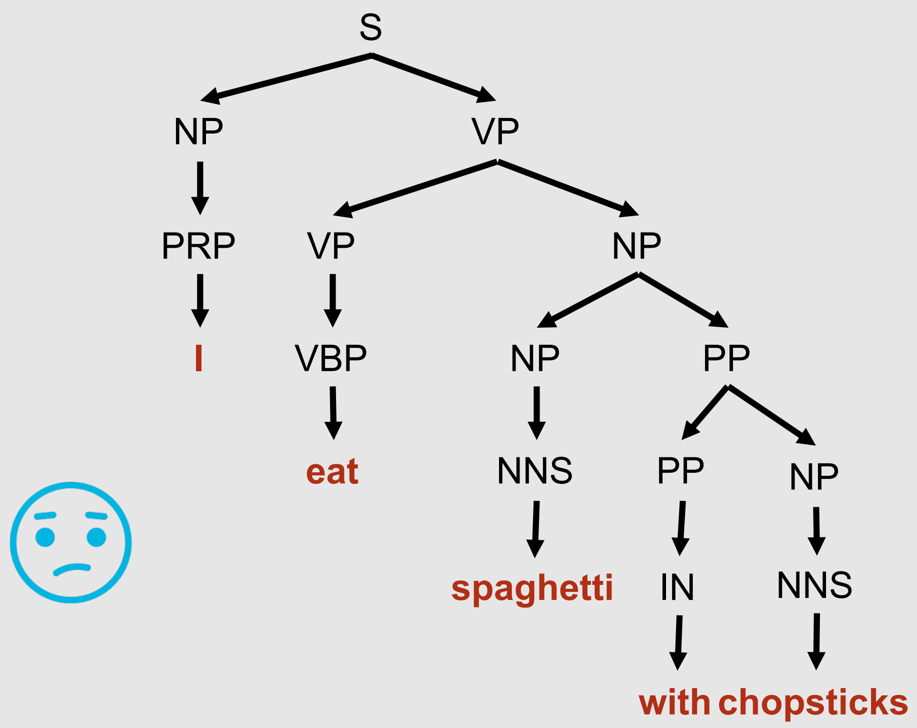
Parts of Speech
POS Tagsets
POS Tagging
Context-Free Grammars

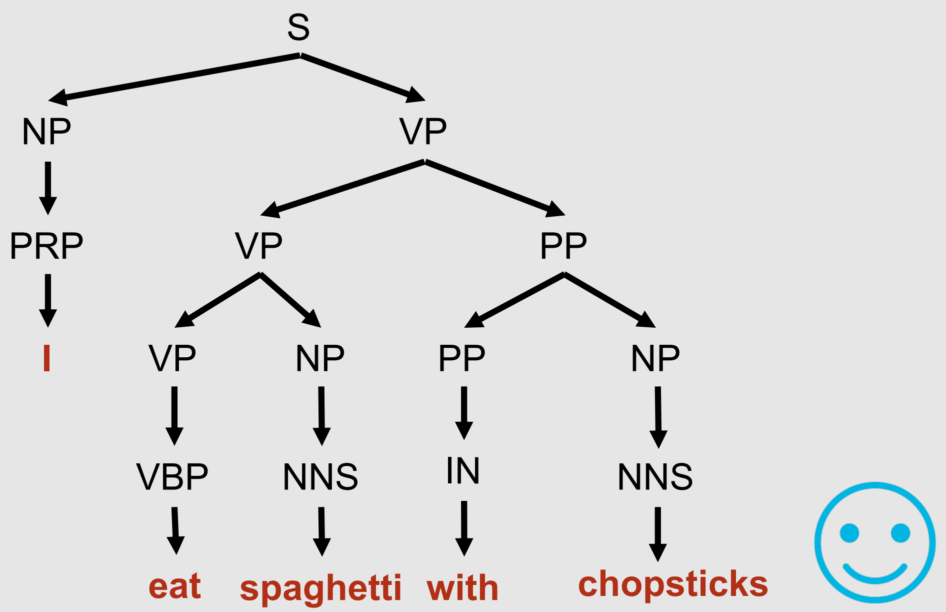
✓

Tuesday

Thursday

★ Hierarchical Parsing
Dynamic Programming Parsing Algorithms
Probabilistic CKY
Lexicalized Grammars





Remember, language is ambiguous!

Input sentences may have many possible parses

There are also many ways to generate parse trees.

Top-Down Parsing:	Bottom-Up Parsing:
Goal-driven Builds parse tree from the start symbol down to the terminal nodes	Data-driven Builds parse tree from the terminal nodes up to the start symbol

Top-Down Parsing

- Assume that the input can be derived by the designated start symbol ***S***
- Find the tops of all trees that can start with ***S***
    - Look for all production rules with ***S*** on the left-hand side
- Find the tops of all trees that can start with those constituents
- (Repeat recursively until terminal nodes are reached)
- Trees whose leaves fail to match all words in the input sentence can be rejected, leaving behind trees that represent successful parses

Top-Down Parsing: Example

Input Sentence:

Book that flight.

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

Lexicon:

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

$$\begin{tikzcd} & S \arrow[dl] \arrow[dr] & \\ NP & & VP \end{tikzcd}$$

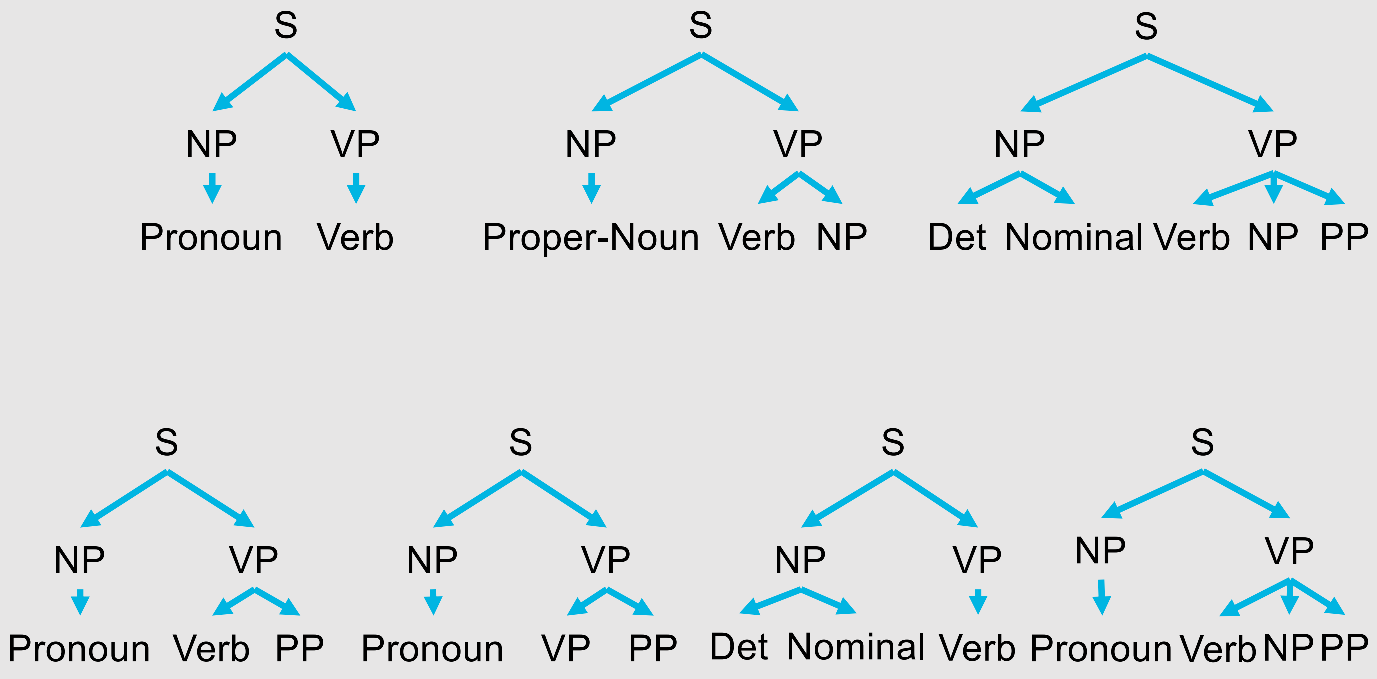
$$\begin{tikzcd} & S \arrow[dl] \arrow[d] \arrow[dr] & \\ Aux & NP & VP \end{tikzcd}$$

$$\begin{tikzcd} S \arrow[d] \\ VP \end{tikzcd}$$

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

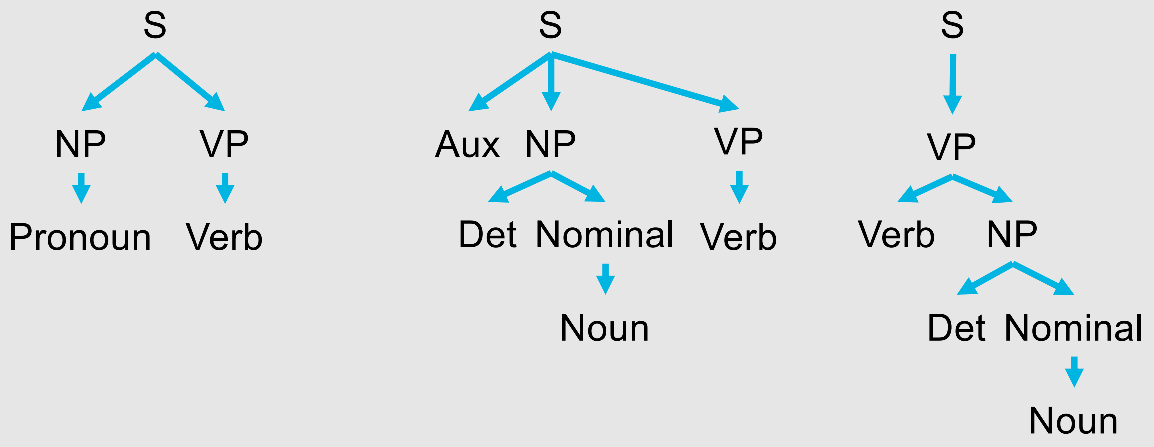


…and many more!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

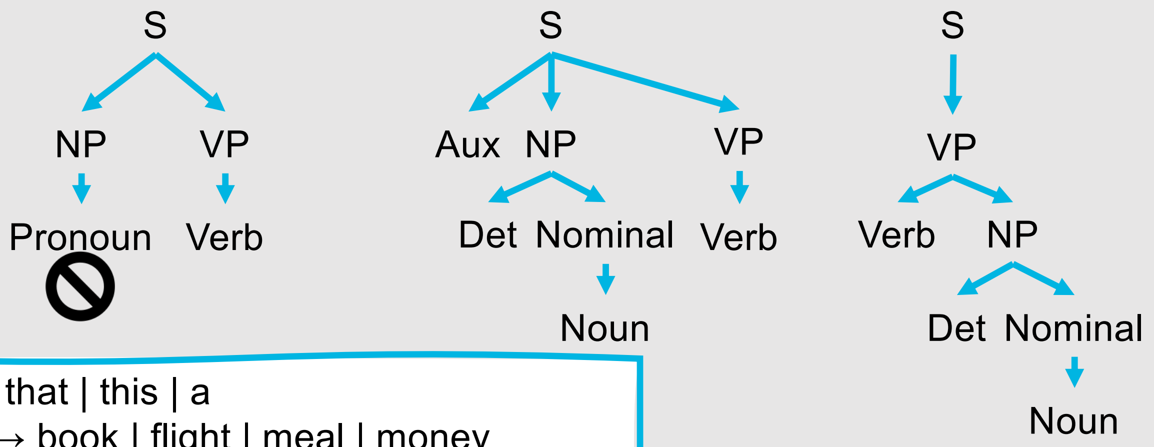


…and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



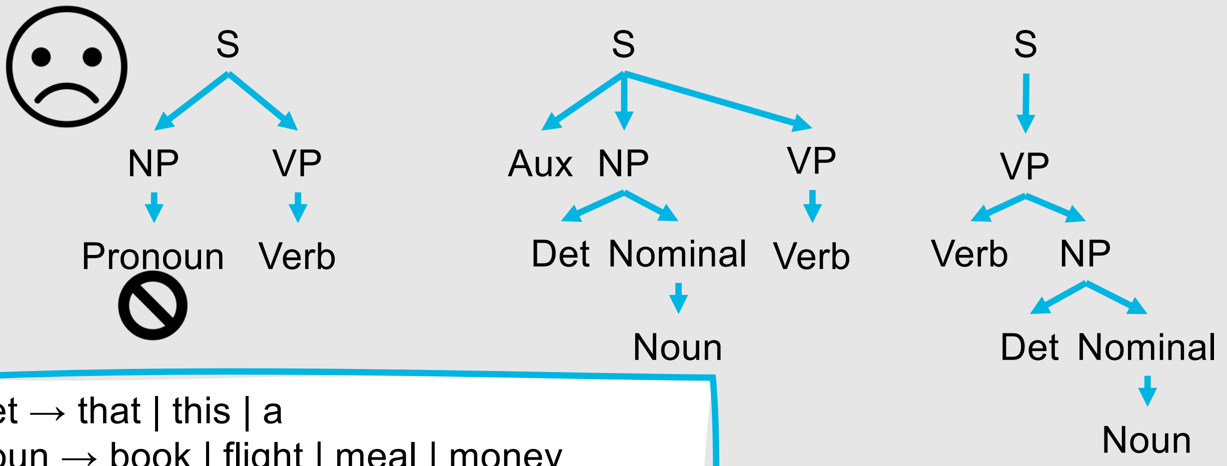
Det → that | this | a
Noun → book | flight | meal | money
Verb → **book** | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

...and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



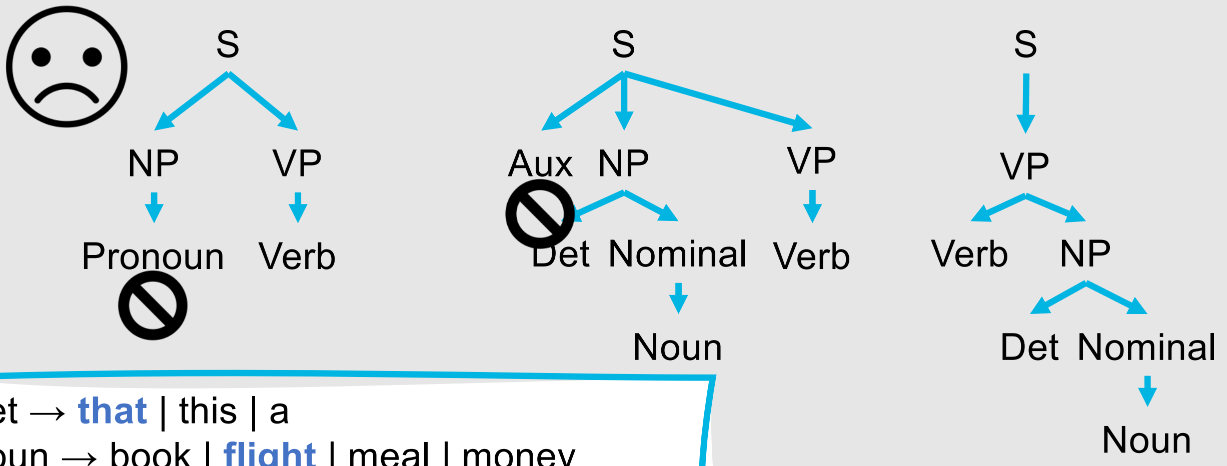
Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

…and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



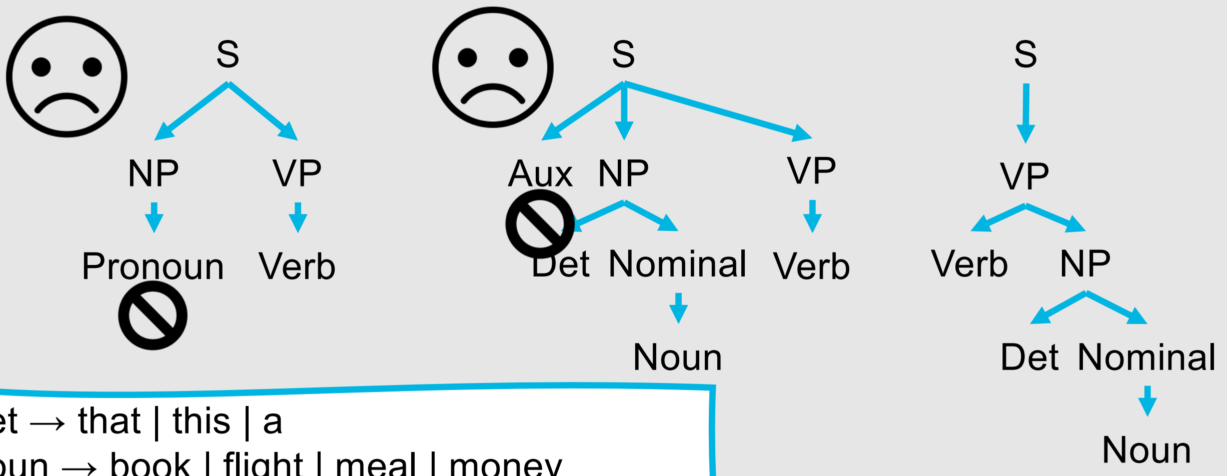
Det → **that** | this | a
Noun → book | **flight** | meal | money
Verb → **book** | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

…and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



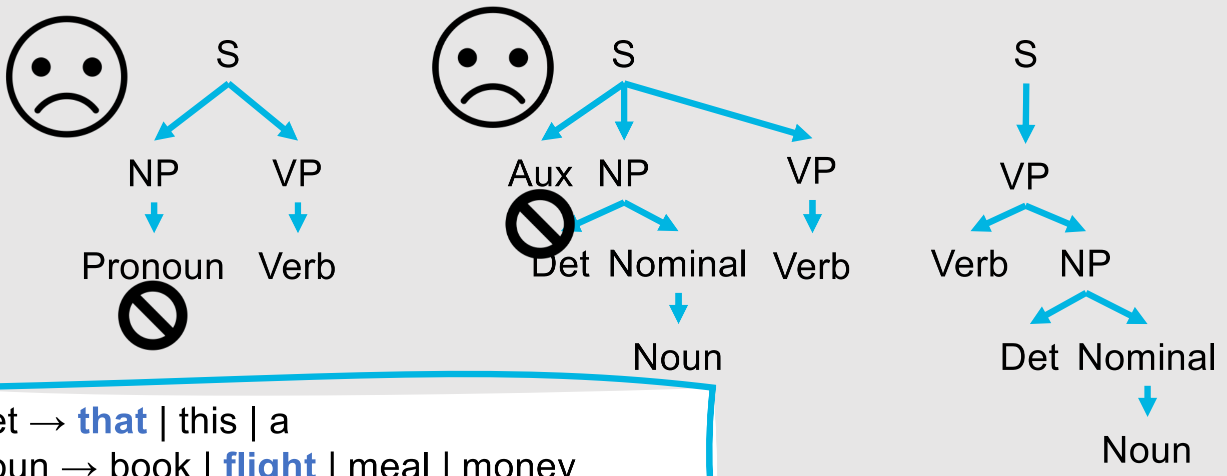
Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

…and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



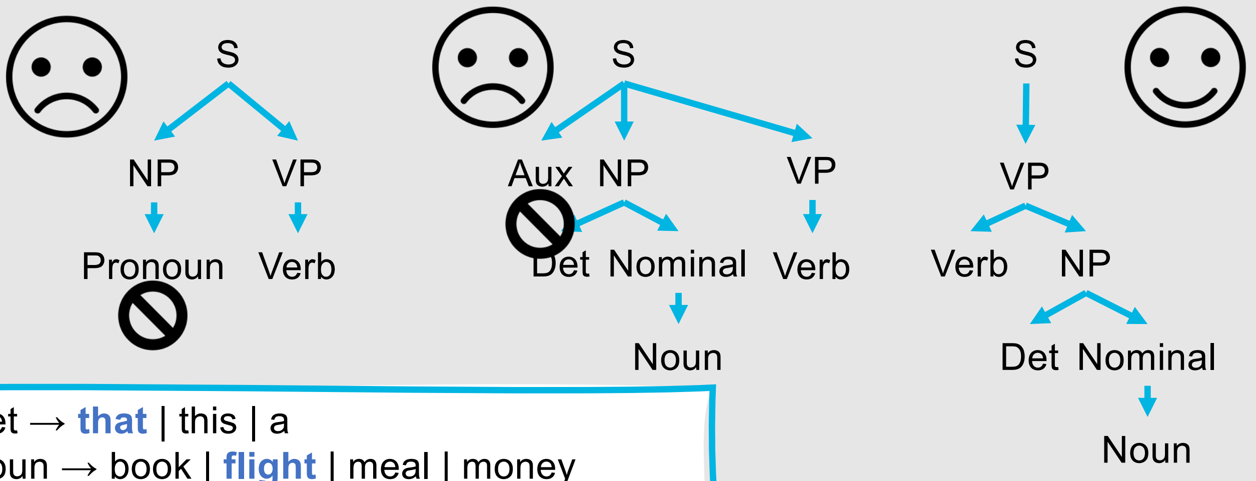
Det → **that** | this | a
Noun → book | **flight** | meal | money
Verb → **book** | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

…and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Det → **that** | this | a
Noun → book | **flight** | meal | money
Verb → **book** | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

…and many, many more not shown!

Bottom-Up Parsing

- Used in the earliest known parsing algorithm
- Starts with the words in the input sentence, and tries to build trees from those words up by applying rules from the grammar one at a time
  - Looks for places in the in-progress parse where the righthand side of a production rule might fit
- Success = parser builds a tree rooted in the start symbol ***S*** that covers all of the input words

Bottom-Up Parsing: Example

Input Sentence:

Book that flight.

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

Lexicon:

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

Bottom-Up Parsing: Example

Book that flight.

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

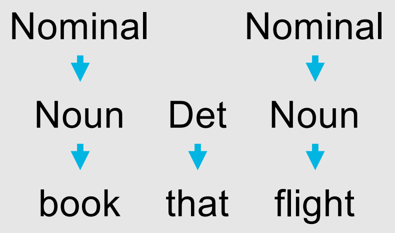
$$\begin{tikzcd}
\text{Noun} \arrow[d] & \text{Det} \arrow[d] & \text{Noun} \arrow[d] \\
\text{book} & \text{that} & \text{flight}
\end{tikzcd}$$

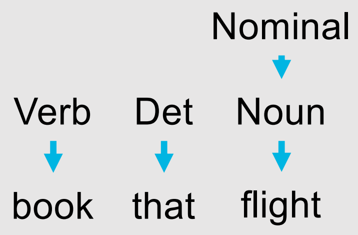
$$\begin{tikzcd}
\text{Verb} \arrow[d] & \text{Det} \arrow[d] & \text{Noun} \arrow[d] \\
\text{book} & \text{that} & \text{flight}
\end{tikzcd}$$

Bottom-Up Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

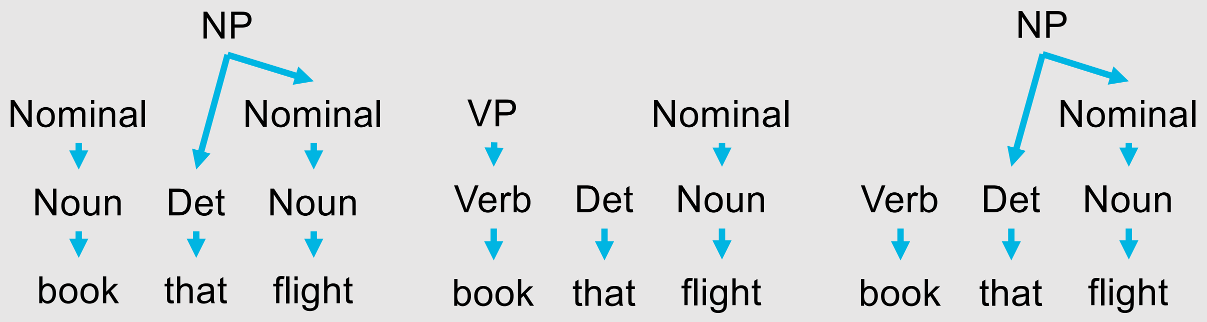




Bottom-Up Parsing: Example

Book that flight.

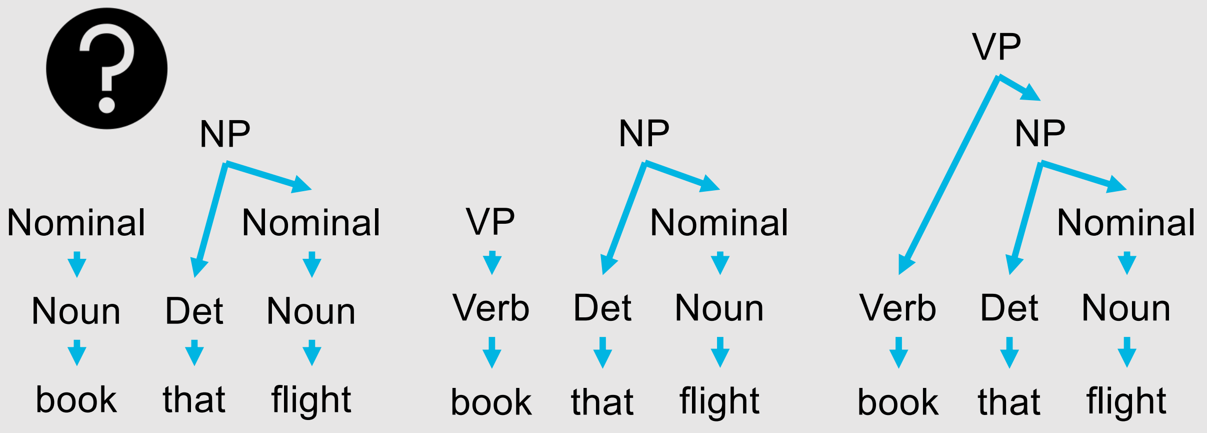
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Bottom-Up Parsing: Example

Book that flight.

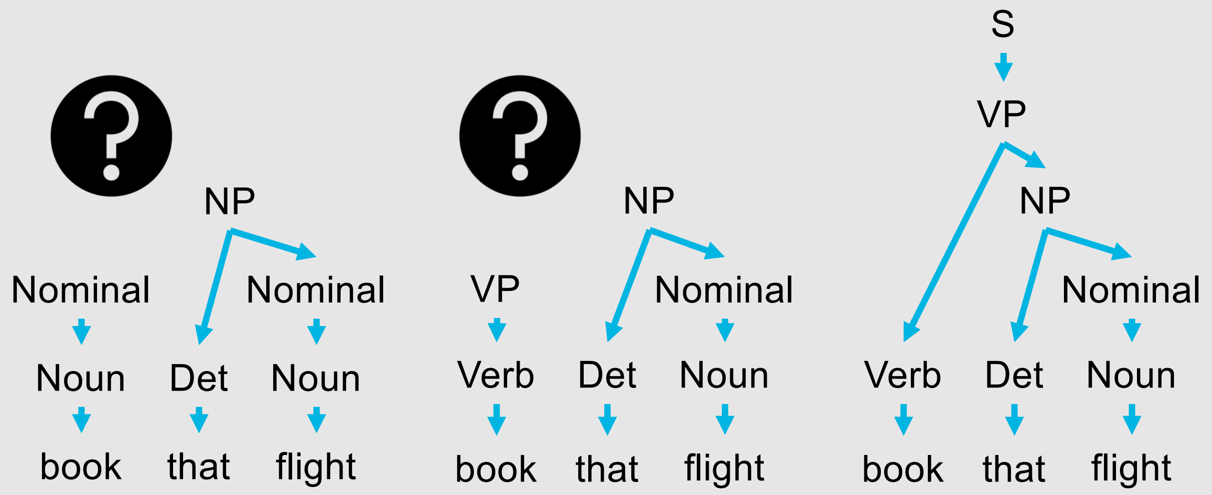
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Bottom-Up Parsing: Example

Book that flight.

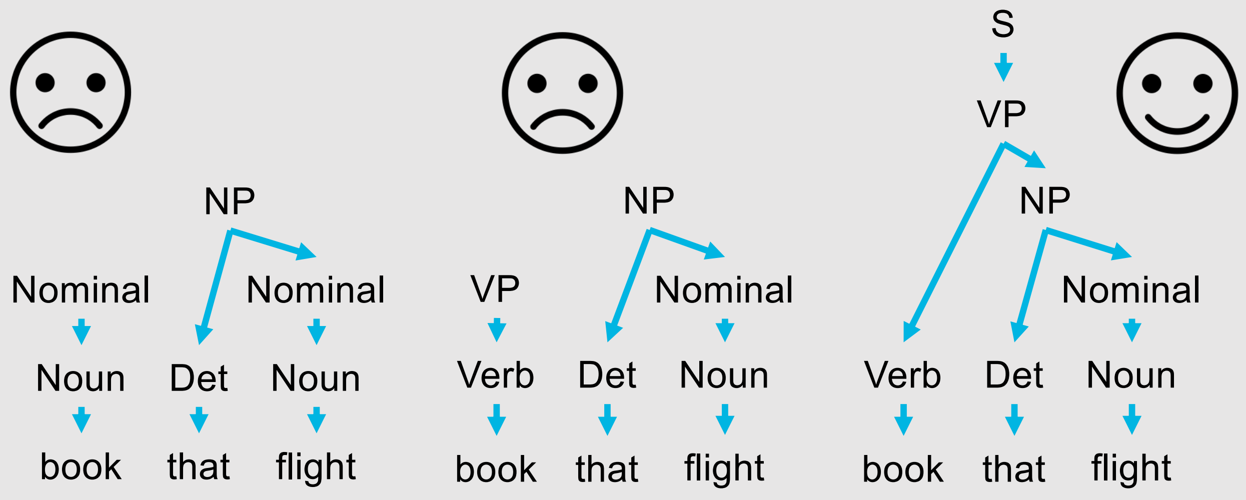
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Bottom-Up Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Top-Down vs. Bottom-Up Parsing

Top-Down Parsing

- Pros:
  - Never wastes time exploring invalid trees
- Cons:
  - Spends considerable effort on trees that are not consistent with the input

Bottom-Up Parsing

- Pros:
  - Never suggests trees that are inconsistent with the input
- Cons:
  - Generates many trees and subtrees that cannot result in a valid sentence (according to production rules specified by the grammar)

This Week's Topics

Tuesday

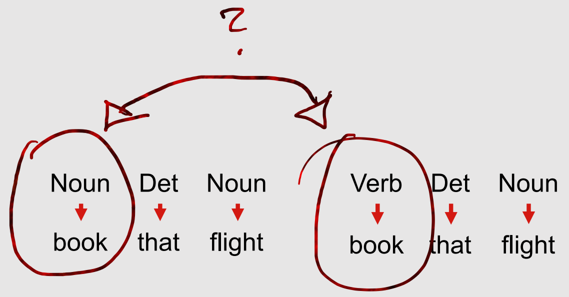
Parts of Speech
POS Tagsets
POS Tagging
Context-Free Grammars

Thursday

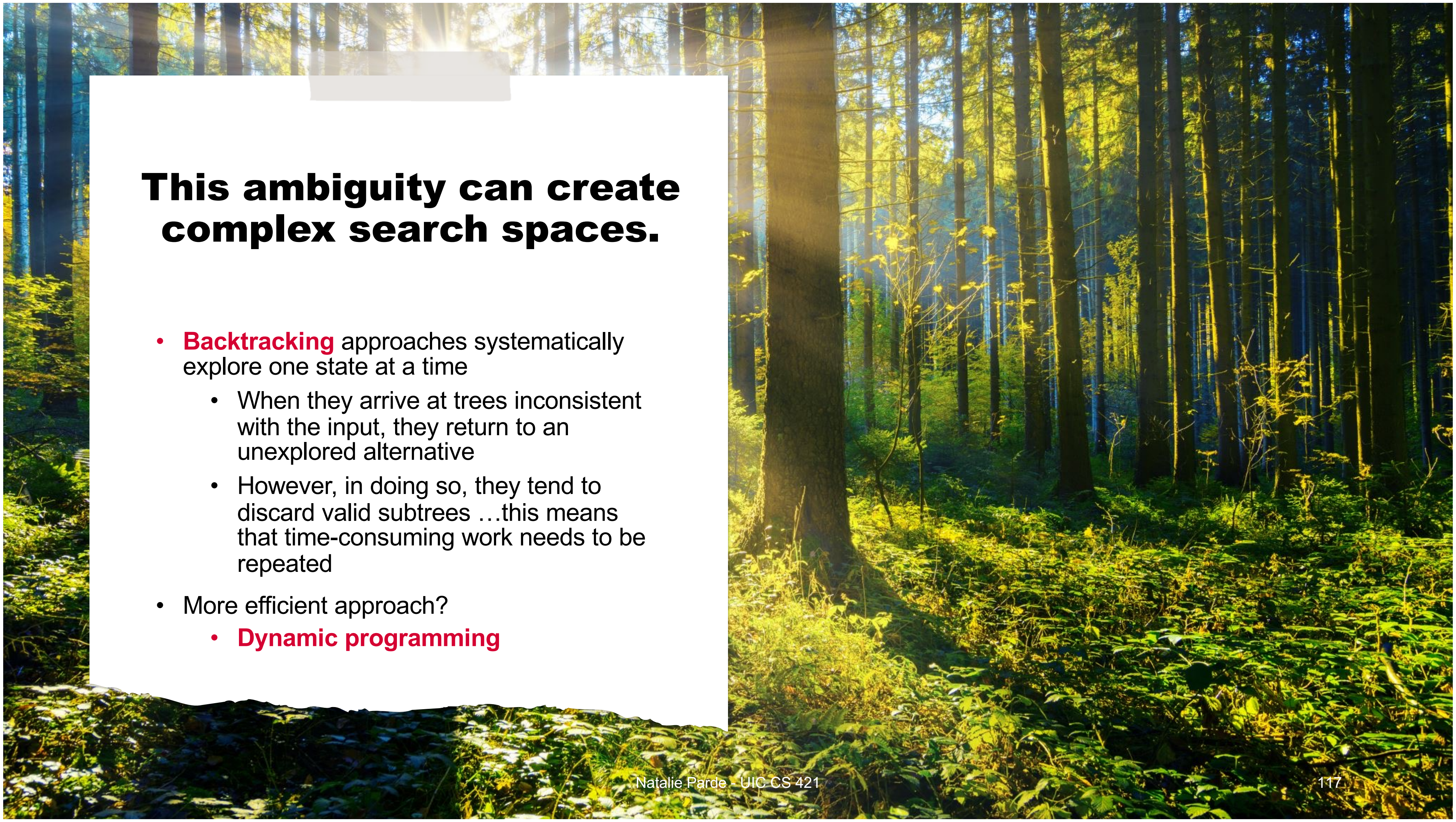
Hierarchical Parsing
☆ Dynamic Programming Parsing Algorithms
Probabilistic CKY
Lexicalized Grammars

Many forms of ambiguity can arise during syntactic parsing!

- **Structural Ambiguity:** Grammar allows for more than one possible parse for a given sentence
  - **Attachment Ambiguity:** Constituent can be attached to a parse tree at more than one place
    - I eat spaghetti *with chopsticks.*
  - **Coordination Ambiguity:** Different sets of phrases can be conjoined by a conjunction
    - I grabbed a muffin from the table marked "nut-free scones *and* muffins," hoping I'd parsed the sign correctly.
- **Local Ambiguity:** Word may be interpreted multiple ways



- Det → that | this | a
- Noun → **book** | flight | meal | money
- Verb → **book** | include | prefer
- Pronoun → I | she | me
- Proper-Noun → Houston | NWA
- Aux → does
- Preposition → from | to | on | near | through



This ambiguity can create complex search spaces.

- **Backtracking** approaches systematically explore one state at a time
  - When they arrive at trees inconsistent with the input, they return to an unexplored alternative
  - However, in doing so, they tend to discard valid subtrees …this means that time-consuming work needs to be repeated
- More efficient approach?
  - **Dynamic programming**

Dynamic Programming Parsing Methods

- Widely used methods:
  - Cocke-Kasami-Younger (**CKY**) algorithm
    - Bottom-up approach
  - **Earley** algorithm
    - Top-down approach

CKY Algorithm

- One of the earliest recognition and parsing algorithms
- Standard version can only recognize CFGs in **Chomsky Normal Form** (CNF)
  - Grammars are restricted to production rules of the form:
    - A → B C
    - A → w
  - This means that the righthand side of each rule must expand to either two non-terminals or a single terminal
  - Any CFG can be converted to a corresponding CNF grammar that accepts exactly the same set of strings as the original grammar!

How does this conversion work?

- Three situations we need to address:
    1. Production rules that mix terminals and non-terminals on the righthand side
    2. Production rules that have a single non-terminal on the righthand side (**unit productions**)
    3. Production rules that have more than two non-terminals on the righthand side
- Situation #1: **Introduce a dummy non-terminal that covers only the original terminal**
    - INF-VP → to VP could be replaced with INF-VP → TO VP and TO → to
- Situation #2: **Replace the non-terminals with the non-unit production rules to which they eventually lead**
    - A → B and B → w could be replaced with A → w
- Situation #3: **Introduce new non-terminals that spread longer sequences over multiple rules**
    - A → B C D could be replaced with A → B X1 and X1 → C D

Original	CNF
S → NP VP	S → NP VP
S → AdjP NP VP	S → X1 VP
	X1 → AdjP NP
S → VP	S → book | include | prefer

CKY Algorithm

- With the grammar in CNF, each non-terminal node above the POS level of the parse tree will have exactly two children
- Thus, a two-dimensional matrix can encode the tree structure
- Each cell [*i*,*j*] contains a set of non-terminals that represent all constituents spanning positions *i* through *j* of the input
    - Cell that represents the entire input resides in position [0,*n*]

CKY Algorithm

- Non-terminal entries: For each constituent [*i*,*j*], there is a position, *k*, where the constituent can be split into two parts such that $i < k < j$
  - [*i*,*k*] must lie to the left of [*i*,*j*] somewhere along row *i*, and [*k*,*j*] must lie beneath it along column *j*
- To fill in the parse table, we proceed in a bottom-up fashion so when we fill a cell [*i*,*j*], the cells containing the parts that could contribute to this entry have already been filled

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	🟥	🟥	🟥	🟥	🟥
1		🟥	🟥	🟥	🟥
2			🟥	🟥	🟥
3				🟥	🟥
4					🟥

CKY Algorithm: Example

Det → that | this | a | the
Noun → **book** | flight | meal | money
Verb → **book** | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → **book** | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → **book** | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → **book** | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1					
2					
3					
4					

CKY Algorithm: Example

Det → that | this | a | **the**
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det			
2					
3					
4					

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | **flight** | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | **flight** | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det			
2			Noun, Nominal		
3					
4					

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | **through**

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det			
2			Noun, Nominal		
3				Prep.	
4					

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

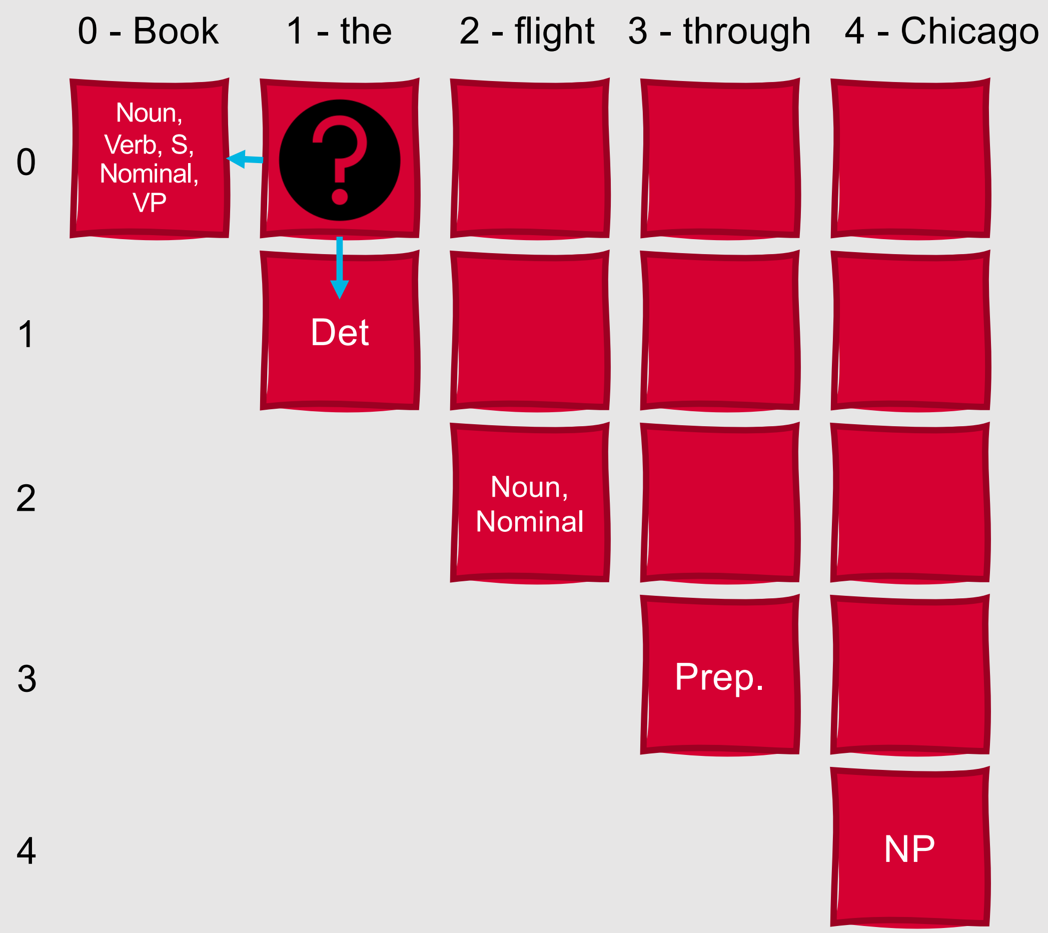
S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → **Chicago** | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det			
2			Noun, Nominal		
3				Prep.	
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

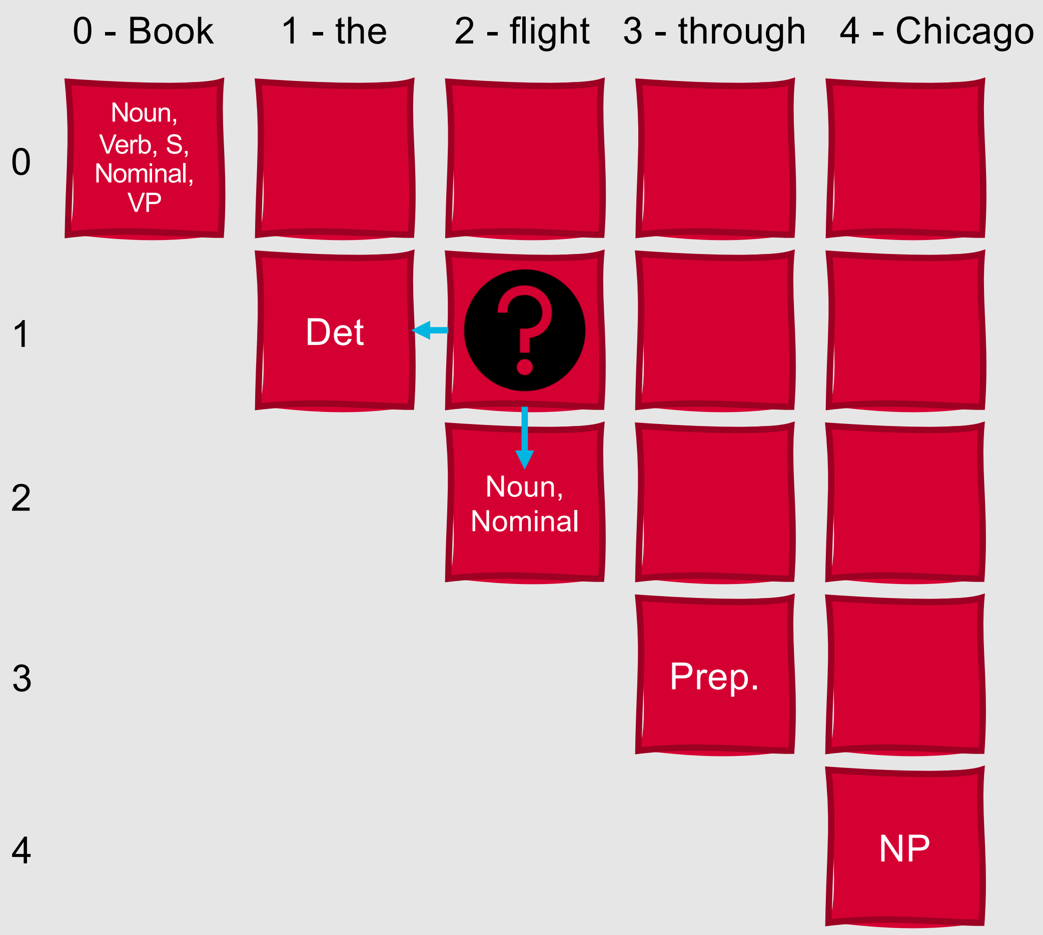


	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP	?			
1		Det			
2			Noun, Nominal		
3				Prep.	
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det	?		
2			Noun, Nominal		
3				Prep.	
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

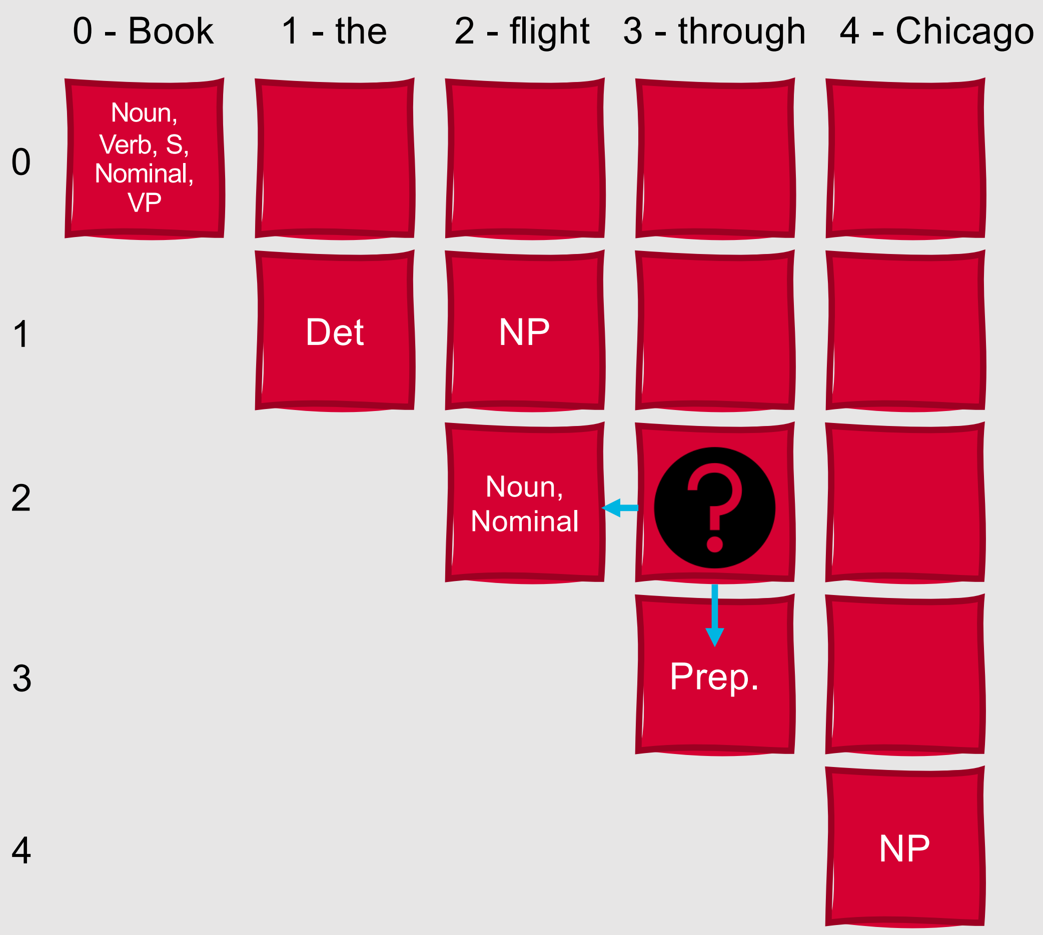
S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det	NP		
2			Noun, Nominal		
3				Prep.	
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

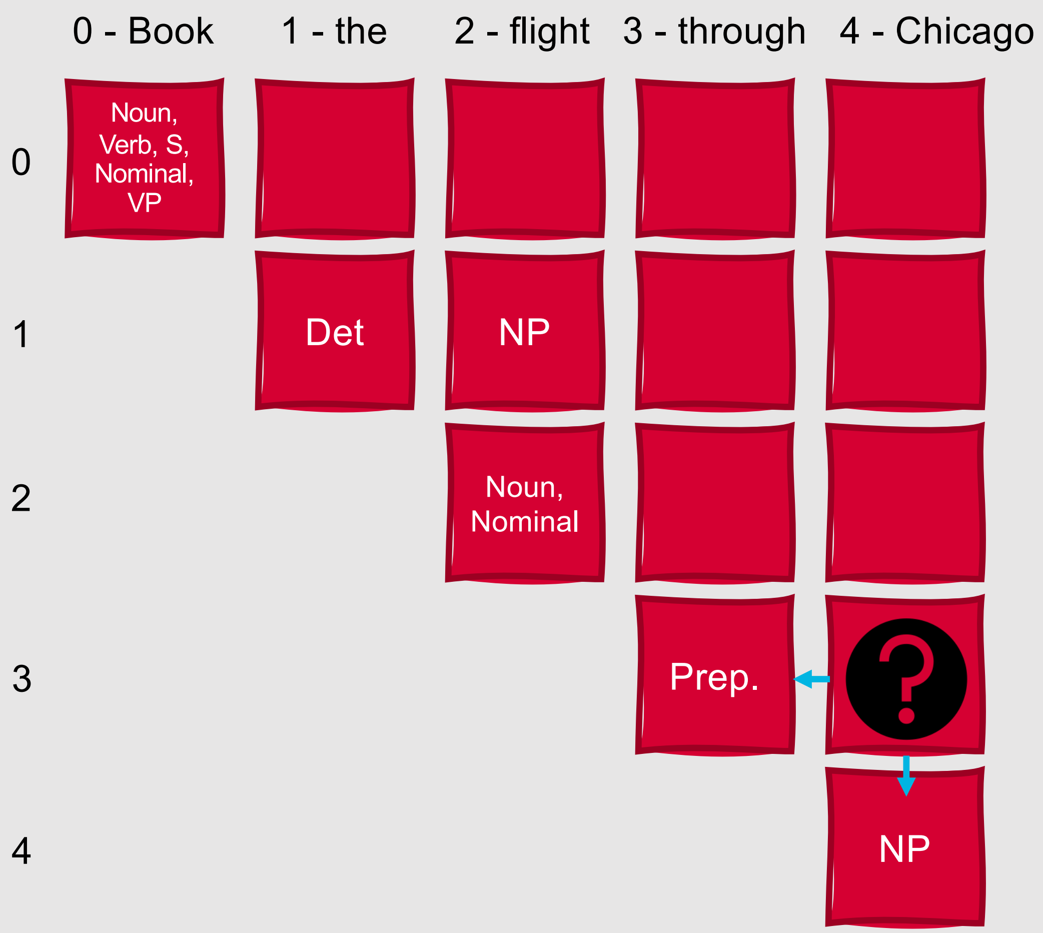


	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det	NP		
2			Noun, Nominal	?	
3				Prep.	
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det	NP		
2			Noun, Nominal		
3				Prep.	?
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

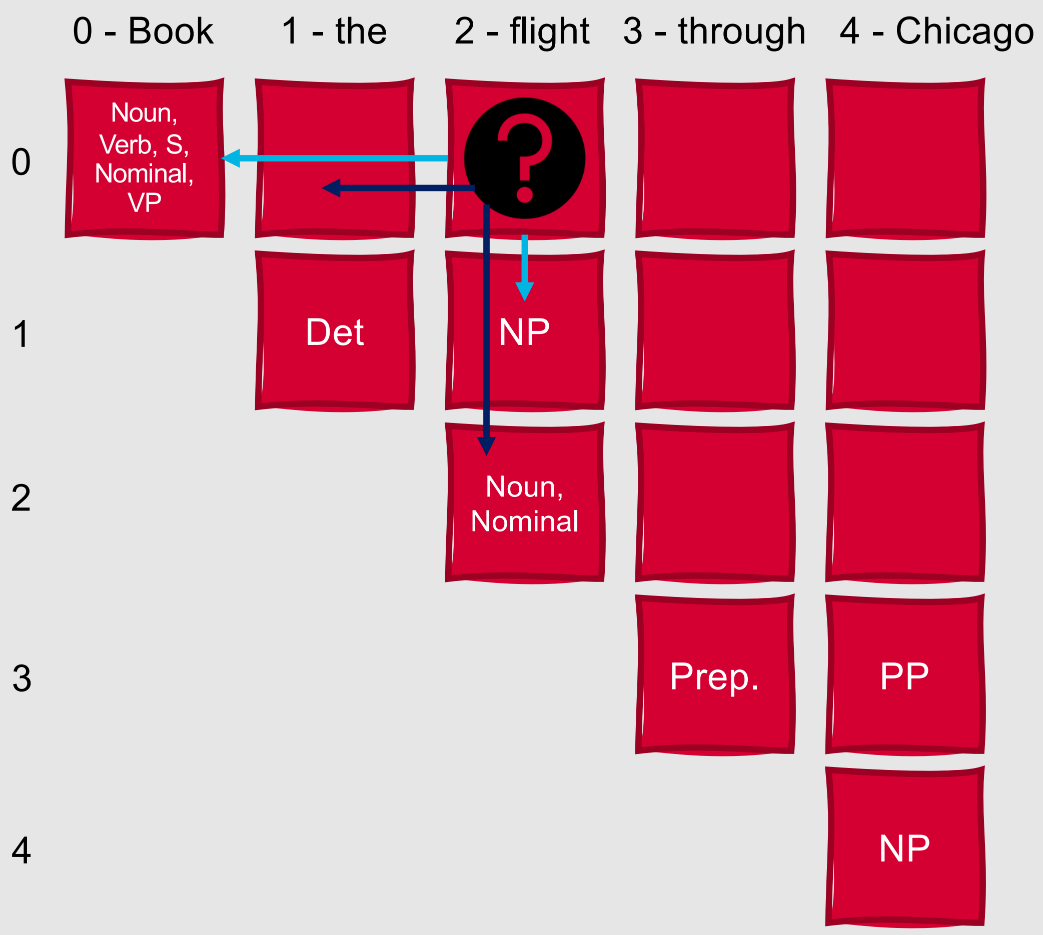
S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det	NP		
2			Noun, Nominal		
3				Prep.	PP
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

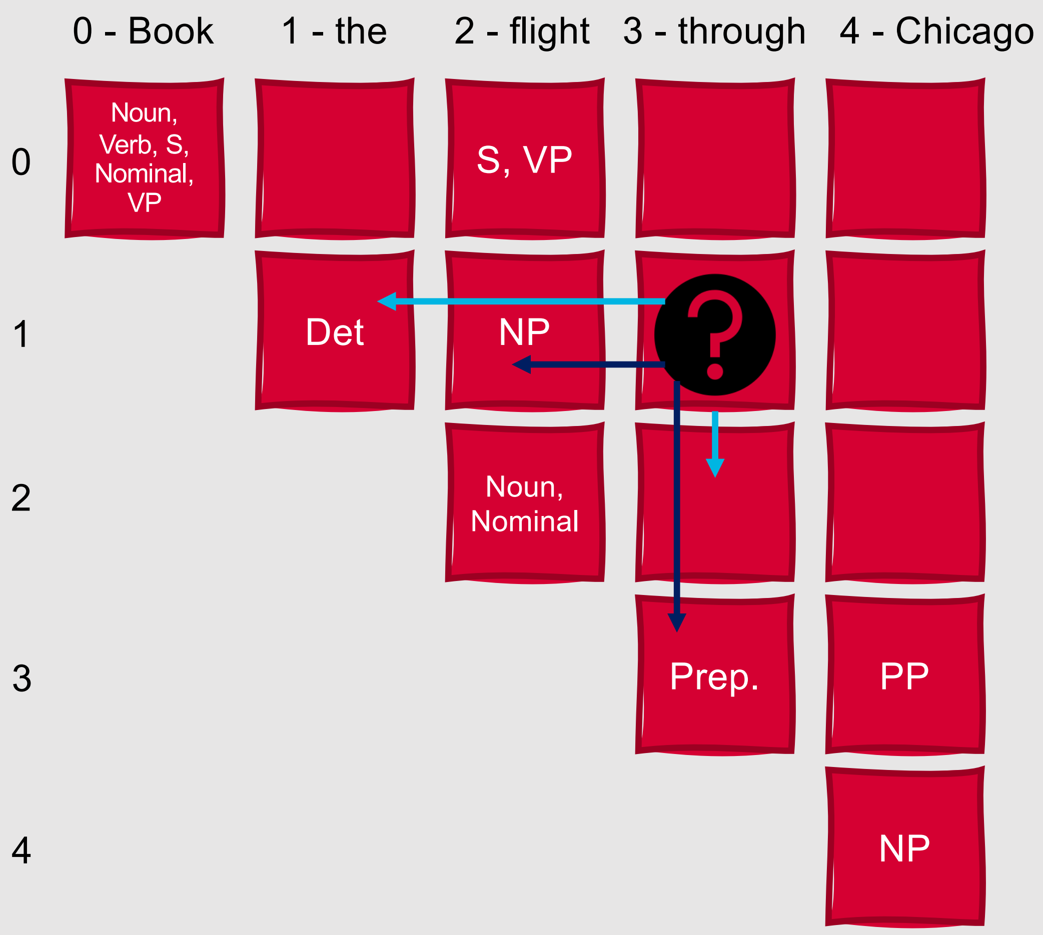
S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		
1		Det	NP		
2			Noun, Nominal		
3				Prep.	PP
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

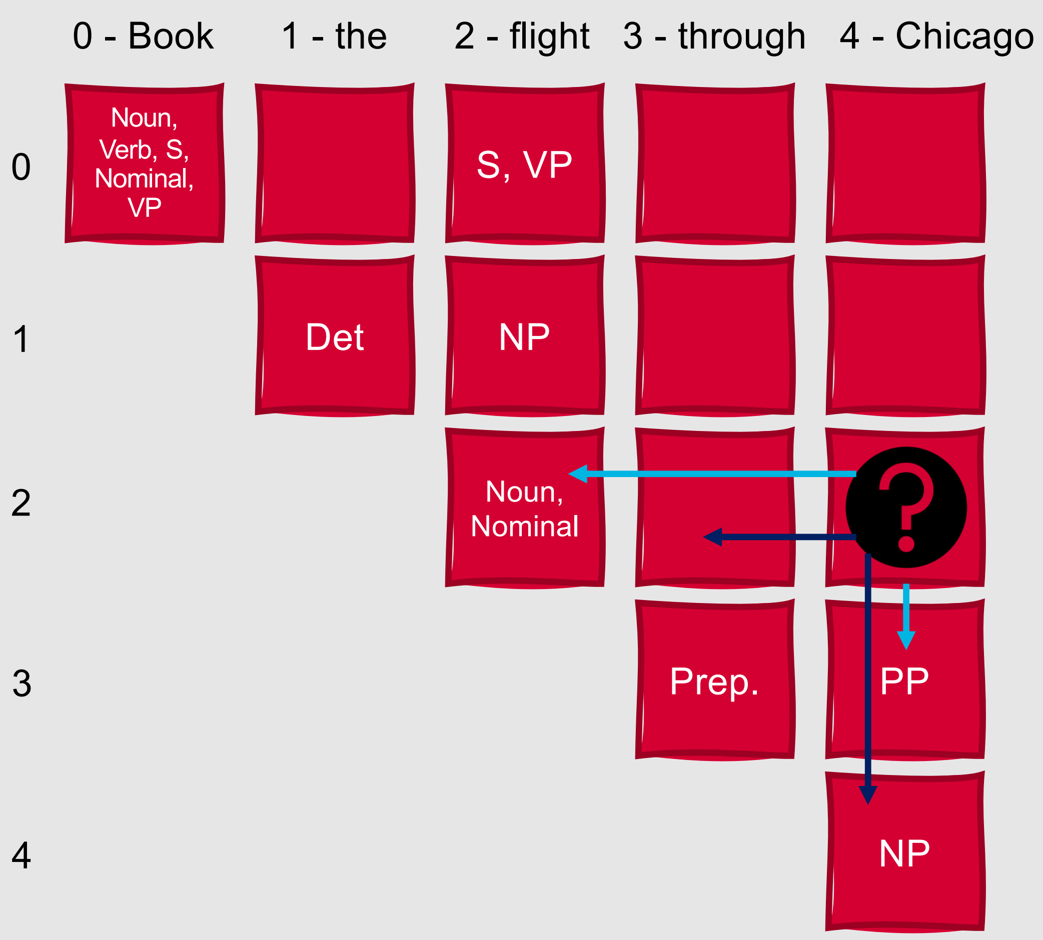
S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		
1		Det	NP		
2			Noun, Nominal		?
3				Prep.	PP
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

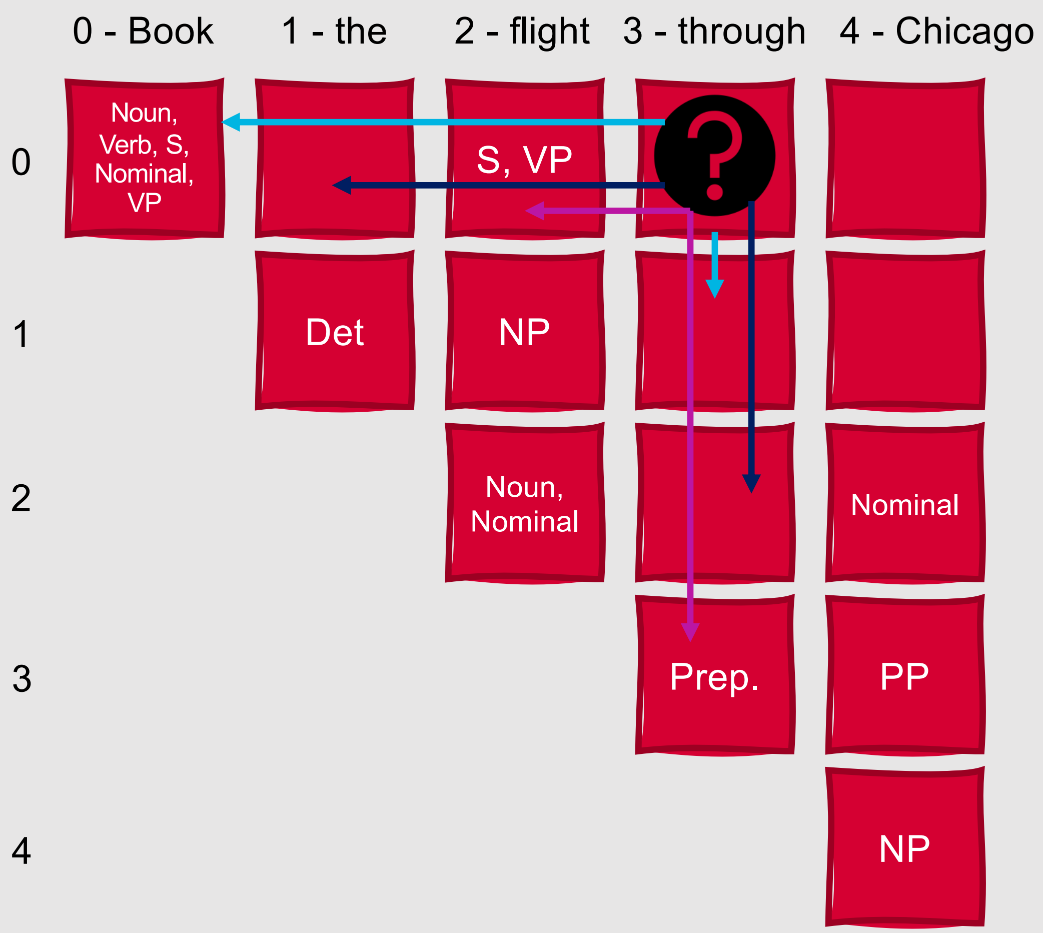
S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		
1		Det	NP		
2			Noun, Nominal		Nominal
3				Prep.	PP
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

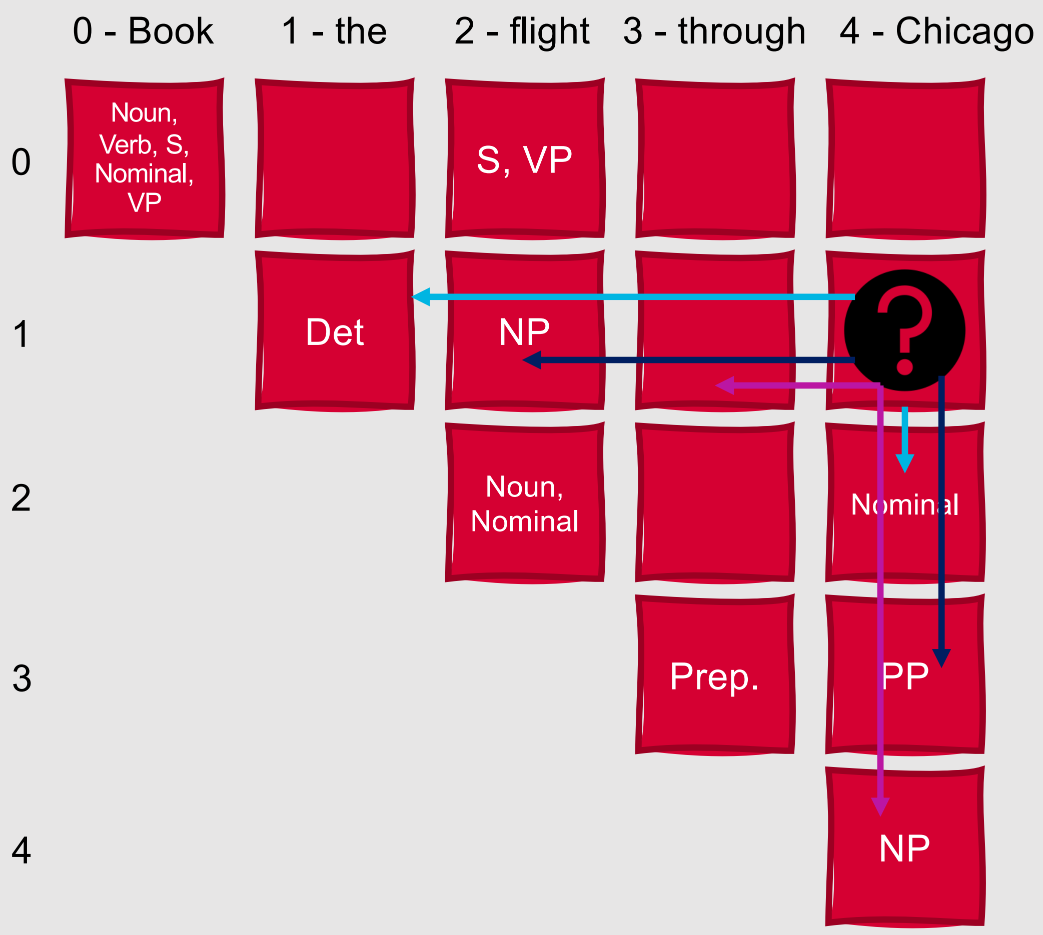
S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

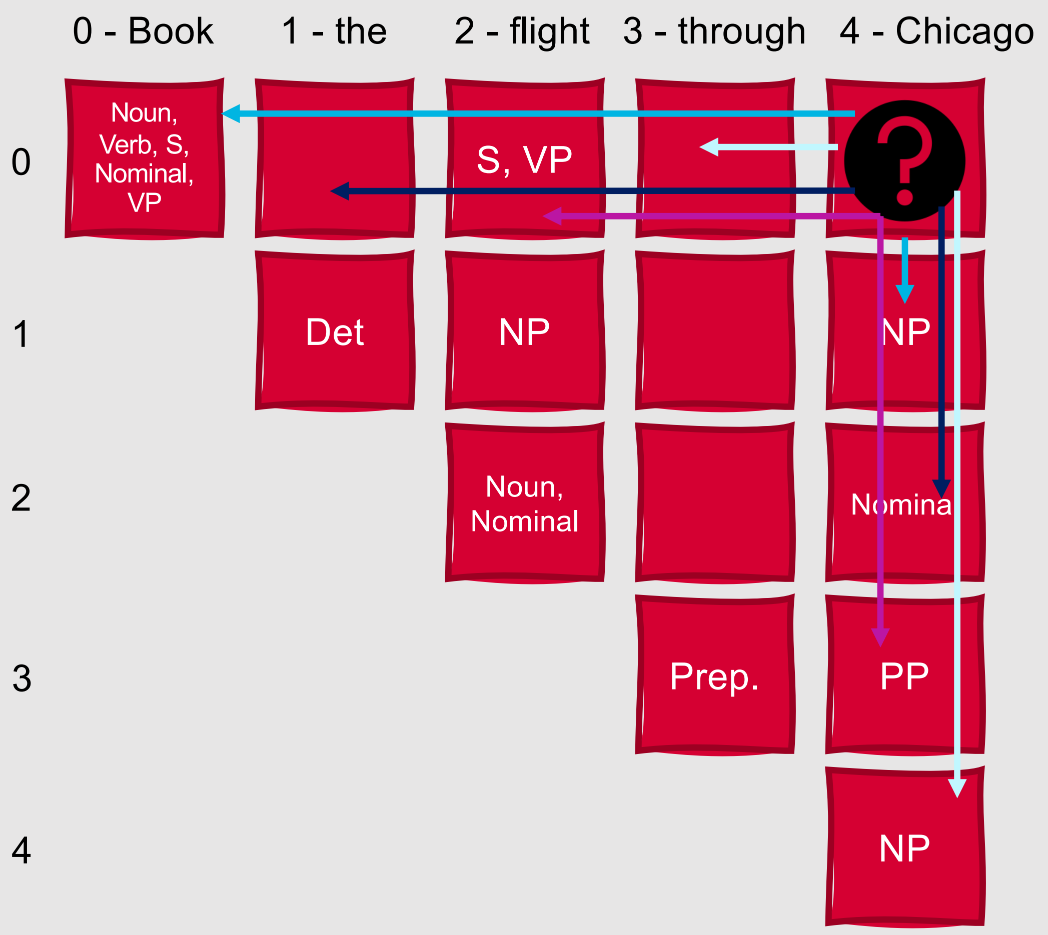
S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		
1		Det	NP		NP
2			Noun, Nominal		Nominal
3				Prep.	PP
4					NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

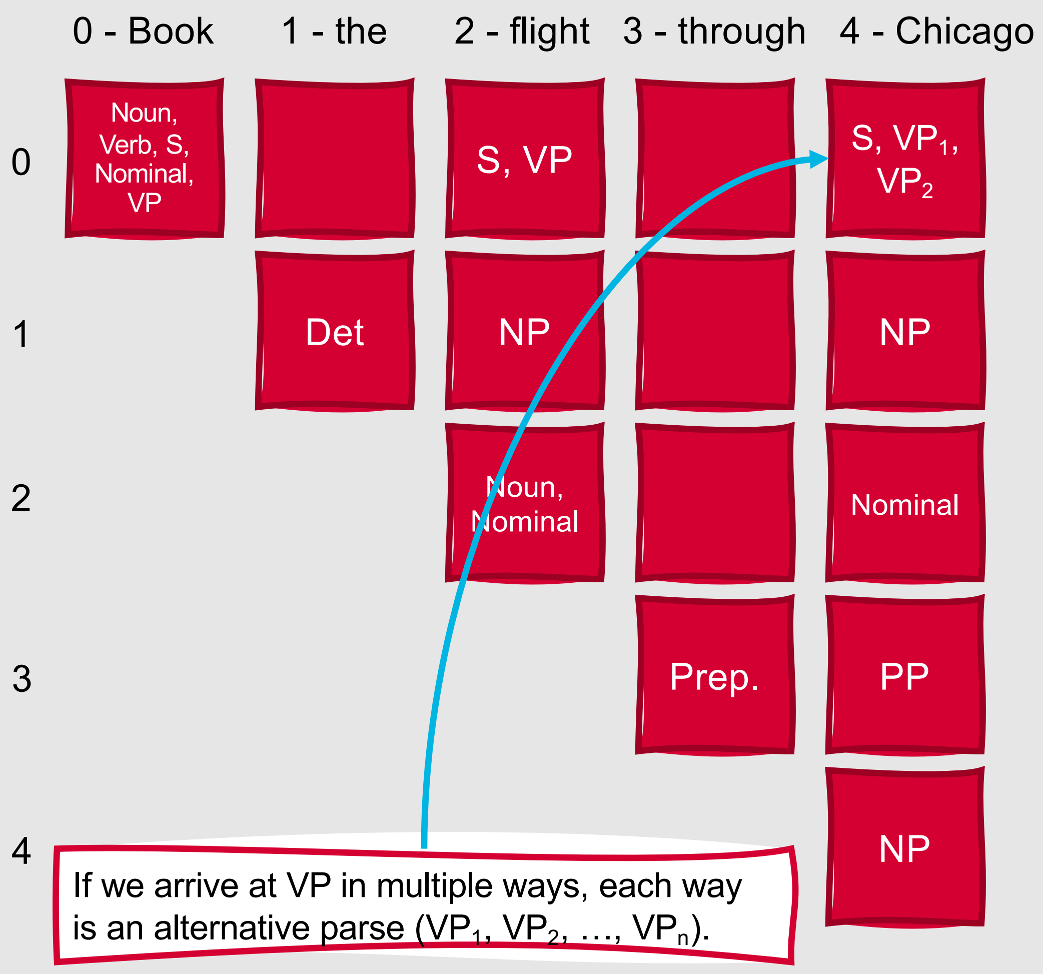
S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		S, VP_1, VP_2
1		Det	NP		NP
2			Noun, Nominal		Nominal
3				Prep.	PP
4					NP

If we arrive at VP in multiple ways, each way is an alternative parse (VP_1, VP_2, …, VP_n).

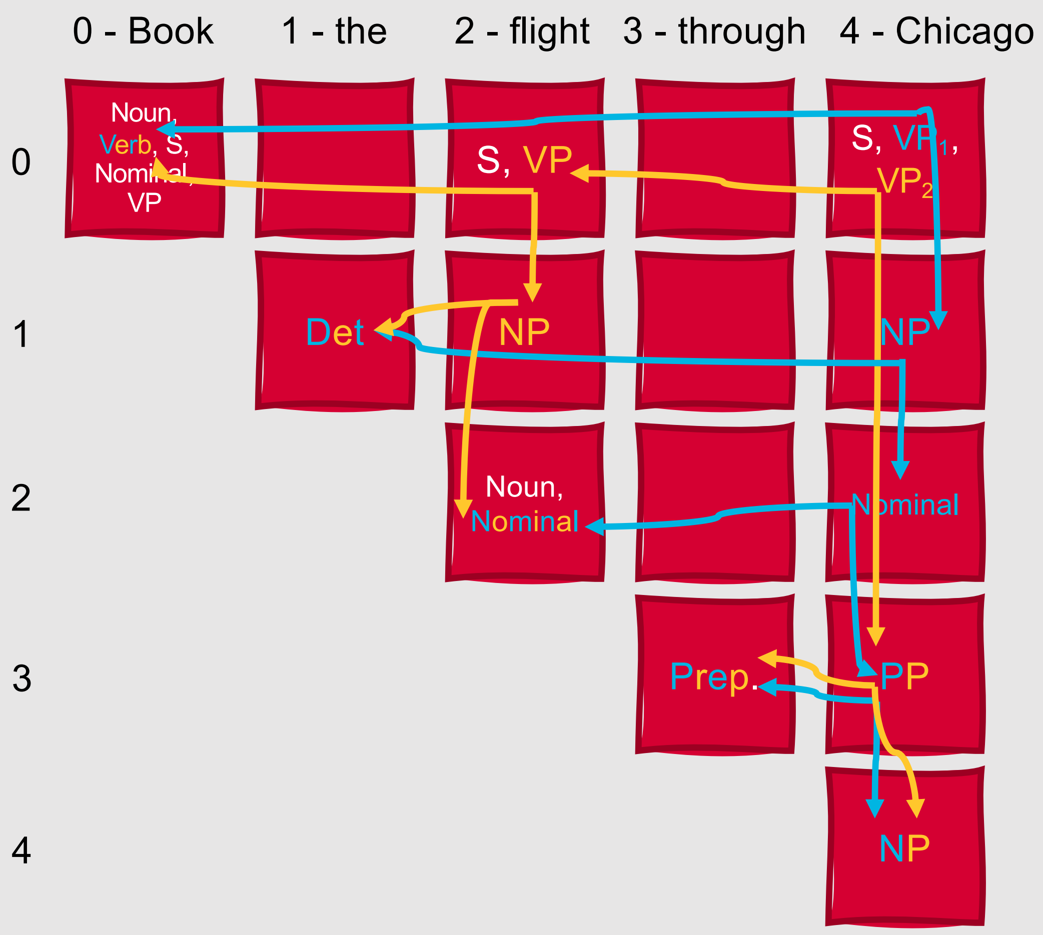
CKY Algorithm

- In the previous example, we **recognized** a valid that this sentence was valid according to our grammar by finding an S in cell [0,n]
- To return all possible parses, we need to also pair each non-terminal with pointers to the table entries from which it was derived
- Then, we can choose a non-terminal and recursively retrieve its component constituents from the table
- Complexity of this algorithm:
  - Time complexity: $O(n^3)$
  - Space complexity: $O(n^2)$

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Preposition → from | to | on | near | through

S → NP VP
S → book | include | prefer
S → Verb NP
NP → I | she | me
NP → Chicago | Dallas
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Earley Parsing

- Top-down dynamic parsing approach
- Table is length *n*+1, where *n* is equivalent to the number of words
- Table entries contain three types of information:
  - A single grammar rule
  - Information about the progress made in completing that rule
    - A • within the righthand side of a state's grammar rule indicates the progress made towards recognizing it
  - The position of the in-progress rule with respect to the input
    - Represented by two numbers, indicating (1) where the state begins, and (2) where its dot lies

Example States

- Input: Book that flight.
- S → • VP, [0,0]
  - Top-down prediction for this particular kind of S
  - First 0: Constituent predicted by this state should begin at the start of the input
  - Second 0: Dot lies at the start of the input as well
- NP → Det • Nominal, [1,2]
  - NP begins at position 1
  - Det has been successfully parsed
  - Nominal is expected next
- VP → V NP •, [0,3]
  - Successful discovery of a tree corresponding to a VP that spans the entire input

Earley Algorithm

- An Earley parser moves through the *n*+1 sets of states in a chart in order
- At each step, one of three operators is applied to each state depending on its status
  - Predictor
  - Scanner
  - Completer
- States can be added to the chart, but are never removed
- The algorithm never backtracks
- The presence of $S \rightarrow \alpha \bullet, [0,n]$ indicates a successful parse

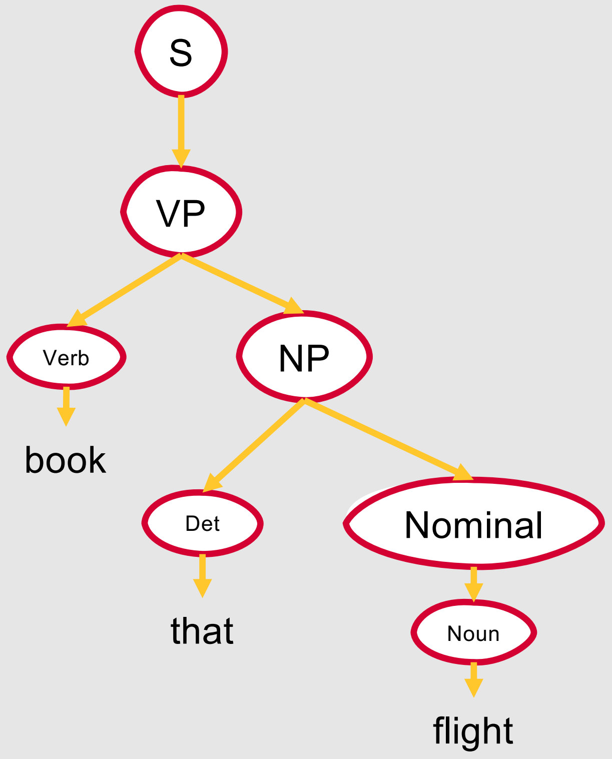
Which states participate in the final parse?

- We can retrieve parse trees by adding a field to store information about the completed states that generated constituents
  - Have the Completer add a pointer to the previous state(s) that it completed
  - Then, when an S is found in the final chart, just follow pointers backward

Chart	State	Rule	Start, End	Added By (Backward Pointer)
0	S0	$\gamma \rightarrow \bullet$ S	0, 0	Start State
0	S1	S → • NP VP	0, 0	Predictor
0	S2	S → • VP	0, 0	Predictor
0	S3	NP → • Det Nominal	0, 0	Predictor
0	S4	VP → • Verb	0, 0	Predictor
0	S5	VP → • Verb NP	0, 0	Predictor
1	S6	Verb → book •	0, 1	Scanner
1	S7	VP → Verb •	0, 1	Completer
1	S8	VP → Verb • NP	0, 1	Completer
1	S9	S → VP •	0, 1	Completer
1	S10	NP → • Det Nominal	1, 1	Predictor
2	S11	Det → that •	1, 2	Scanner
2	S12	NP → Det • Nominal	1, 2	Completer
2	S13	Nominal → • Noun	2, 2	Predictor
3	S14	Noun → flight •	2, 3	Scanner
3	S15	Nominal → Noun •	2, 3	Completer (S14)
3	S16	NP → Det Nominal •	1, 3	Completer (S11, S15)
3	S17	VP → Verb NP •	0, 3	Completer (S6, S16)
3	S18	S → VP •	0, 3	Completer (S17)

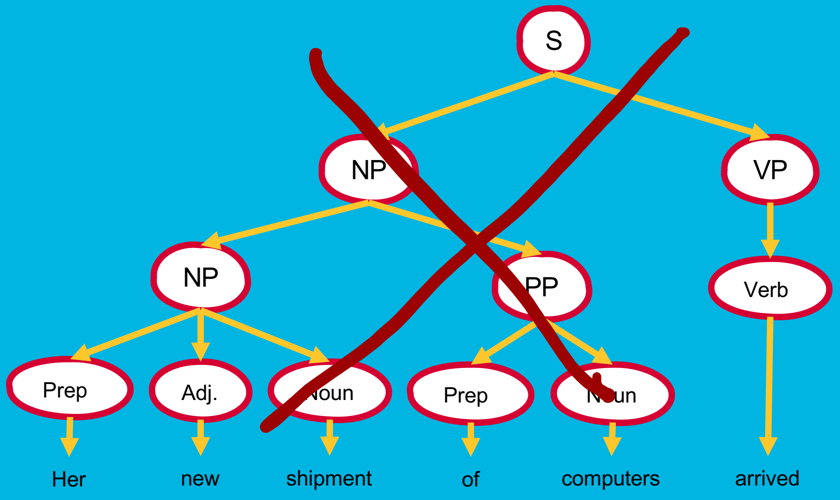
Successful Final Parse

Chart	State	Rule	Start, End	Added By (Backward Pointer)
0	S0	γ → • S	0, 0	Start State
0	S1	S → • NP VP	0, 0	Predictor
0	S2	S → • VP	0, 0	Predictor
0	S3	NP → • Det Nominal	0, 0	Predictor
0	S4	VP → • Verb	0, 0	Predictor
0	S5	VP → • Verb NP	0, 0	Predictor
1	S6	Verb → book •	0, 1	Scanner
1	S7	VP → Verb •	0, 1	Completer
1	S8	VP → Verb • NP	0, 1	Completer
1	S9	S → VP •	0, 1	Completer
1	S10	NP → • Det Nominal	1, 1	Predictor
2	S11	Det → that •	1, 2	Scanner
2	S12	NP → Det • Nominal	1, 2	Completer
2	S13	Nominal → • Noun	2, 2	Predictor
3	S14	Noun → flight •	2, 3	Scanner
3	S15	Nominal → Noun •	2, 3	Completer (S14)
3	S16	NP → Det Nominal •	1, 3	Completer (S11, S15)
3	S17	VP → Verb NP •	0, 3	Completer (S6, S16)
3	S18	S → VP •	0, 3	Completer (S17)



What if we don't need a full parse tree?

- Full parse trees can be complex and time-consuming to build
- Many NLP tasks don't require full hierarchical parses
- In those cases we can perform **partial parsing**, or **shallow parsing**, by **chunking** an input



$\boxed{\text{[Her new shipment]}_{NP}\ \text{[of]}_{PP}\ \text{[computers]}_{NP}\ \text{[arrived]}_{VP}}$

How is chunking performed?

Segmentation: Identify the non-overlapping, fundamental phrases

[Her new order] [of] [computers] [arrived]

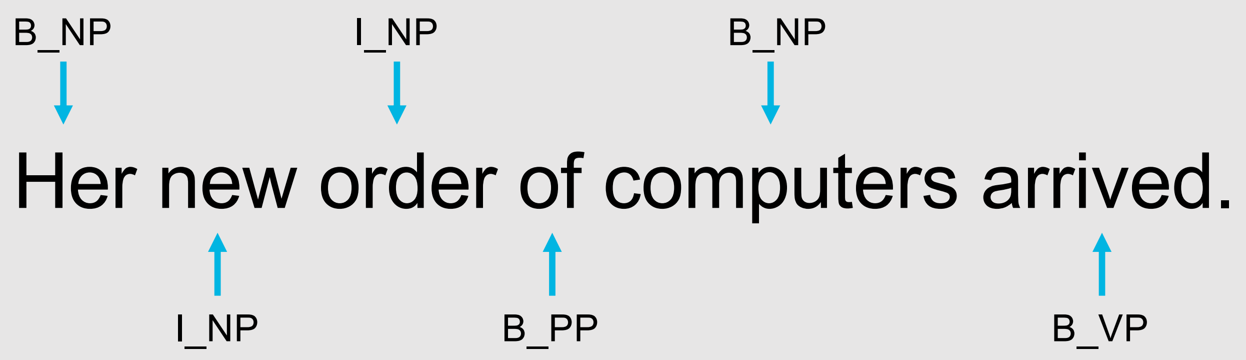
Labeling: Assign labels to those phrases

[Her new order]$_{NP}$ [of]$_{PP}$ [computers]$_{NP}$ [arrived]$_{VP}$

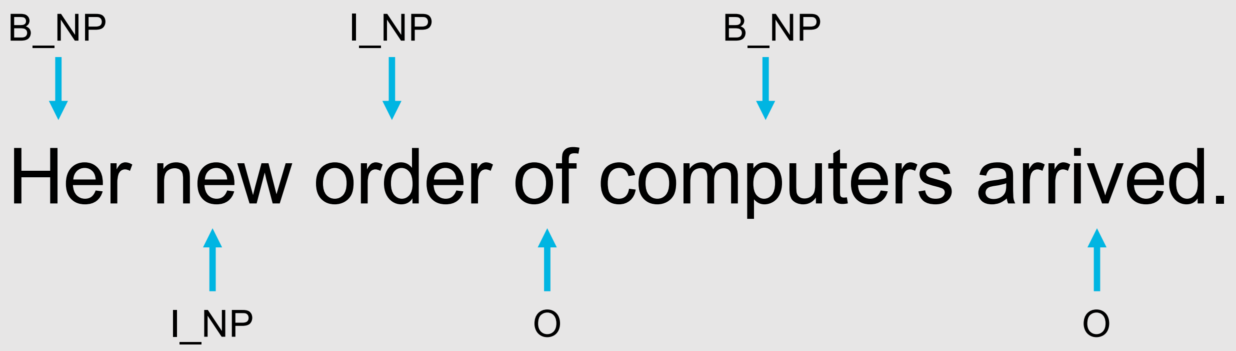
How do we segment text into spans?

- **IOB tagging**
  - **I:** Tokens **inside** a span
  - **O:** Tokens **outside** any span
  - **B:** Tokens **beginning** a span

Task: IOB Tagging (All Constituent Types)



Task: IOB Tagging (Noun Phrases)



We typically evaluate chunking systems using standard NLP performance metrics, including precision, recall, and F1.

This Week's Topics

Tuesday

Parts of Speech
POS Tagsets
POS Tagging
Context-Free Grammars

Thursday

Hierarchical Parsing
Dynamic Programming Parsing Algorithms
★ Probabilistic CKY
Lexicalized Grammars

How can we resolve some of the parsing ambiguities we've observed?

- **Probabilistic Context-Free Grammars:** Can help determine which parse out of multiple valid parses should be selected, based on how likely the parse tree is to occur in a large corpus
- Same core components as regular CFGs:
  - A set of non-terminals, N
  - A set of terminal symbols, Σ
  - A set of rules or productions, R
  - A designated start symbol, S
- However, R is augmented with a probability, [p], learned from a corpus
- The sum of all probabilities for a given non-terminal is 1.0
- For example, if the following three expansions for S were possible, they might have the probabilities:
  - S → NP VP [0.80]
  - S → Aux NP VP [0.15]
  - S → VP [0.05]

Probabilistic Context-Free Grammars

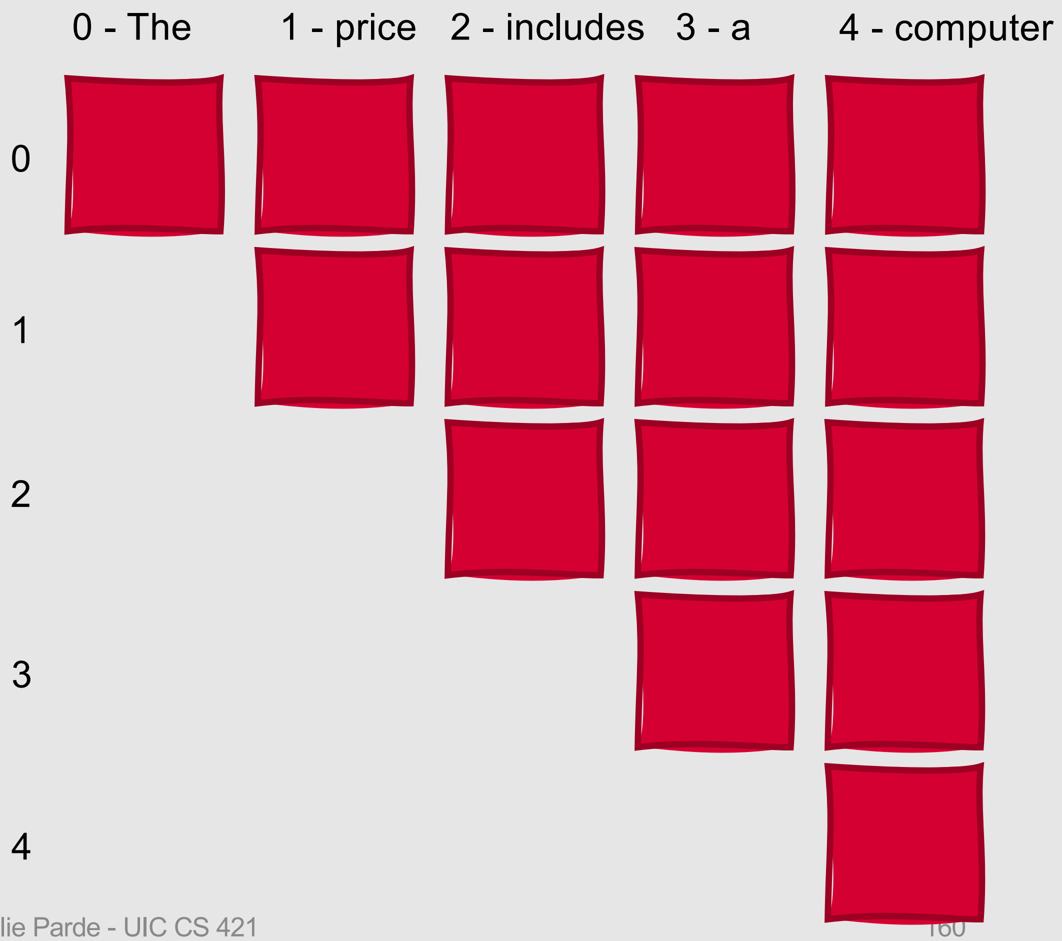
- The probability of sentence S having a parse tree T is the product of the individual probabilities associated with its constituent rules
  - $P(T,S) = \prod_{i=1}^{n} P(\beta_i | A_i)$
- To disambiguate between multiple valid parses, we find the parse tree T that results in the highest probability for the sentence S
  - $\hat{T}(S) = \underset{T\ s.t.\ S=\text{yield}(T)}{\text{argmax}} P(T)$
- We can compute the probabilities for our parse trees by extending the parsing algorithms we already have
- Probabilities are typically learned from a labeled corpus:
  - $P(\alpha \rightarrow \beta | \alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_{\gamma} Count(\alpha \rightarrow \gamma)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

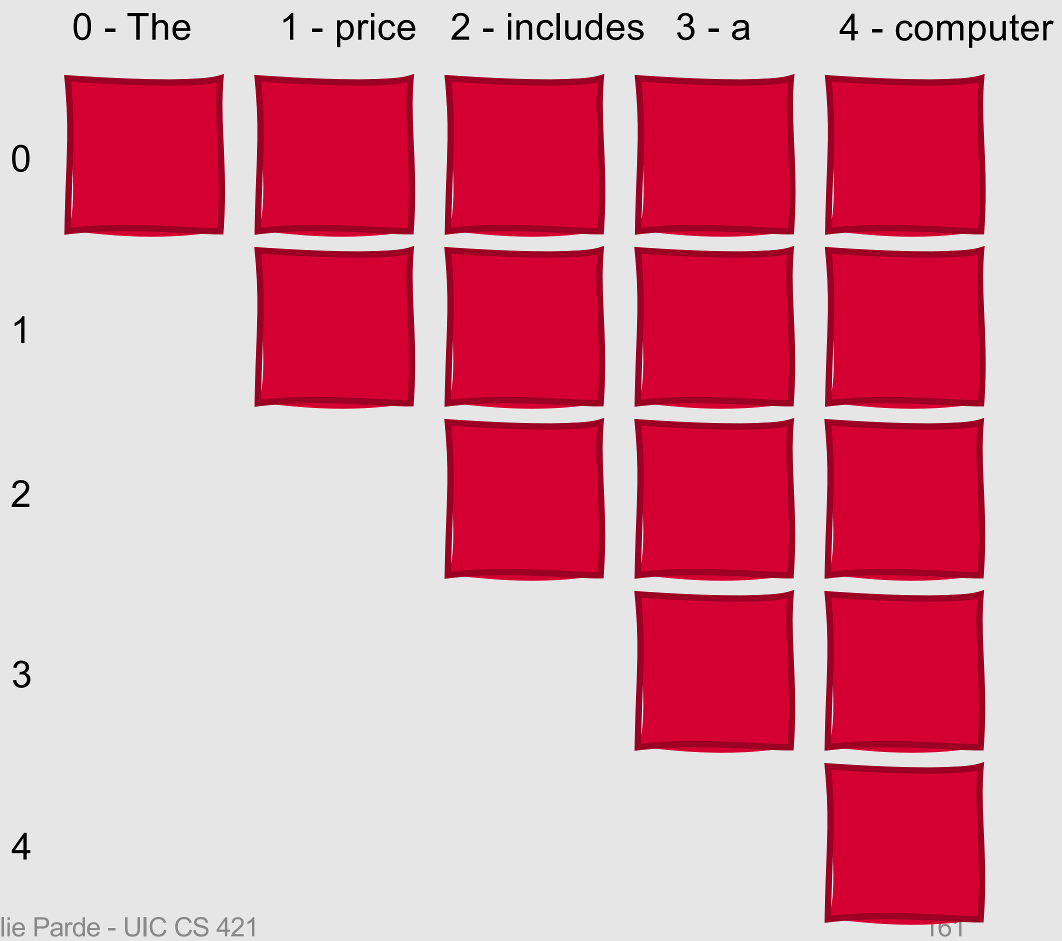
Still assume grammar is in Chomsky normal form!



Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02



Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)				
1					
2					
3					
4					

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)				
1		N (0.01)			
2					
3					
4					

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)				
1		N (0.01)			
2			V (0.05)		
3					
4					

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)				
1		N (0.01)			
2			V (0.05)		
3				Det (0.40)	
4					

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)				
1		N (0.01)			
2			V (0.05)		
3				Det (0.40)	
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)			
2			V (0.05)		
3				Det (0.40)	
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)	🙁		
2			V (0.05)		
3				Det (0.40)	
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)	☹		
2			V (0.05)	☹	
3				Det (0.40)	
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)	☹		
2			V (0.05)	☹	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)	☹️		
2			V (0.05)	☹️	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)	☹		
2			V (0.05)	☹	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹		
1		N (0.01)	☹		
2			V (0.05)	☹	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	🙁		
1		N (0.01)	🙁		
2			V (0.05)	🙁	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	🙁		
1		N (0.01)	🙁		
2			V (0.05)	🙁	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	🙁		
1		N (0.01)	🙁	🙁	
2			V (0.05)	🙁	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹		
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹		
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹		
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹		
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹		
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹		
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	🙁	🙁	
1		N (0.01)	🙁	🙁	
2			V (0.05)	🙁	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹️	☹️	
1		N (0.01)	☹️	☹️	☹️
2			V (0.05)	☹️	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	
1		N (0.01)	☹	☹	☹
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	S (0.8 * 0.0012 * 0.000024 = $2.304*10^{-8}$)
1		N (0.01)	☹	☹	☹
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	🙁	🙁	S (0.8 * 0.0012 * 0.000024 = $2.304*10^{-8}$)
1		N (0.01)	🙁	🙁	🙁
2			V (0.05)	🙁	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	S (0.8 * 0.0012 * 0.000024 = 2.304*10^{-8})
1		N (0.01)	☹	☹	☹
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

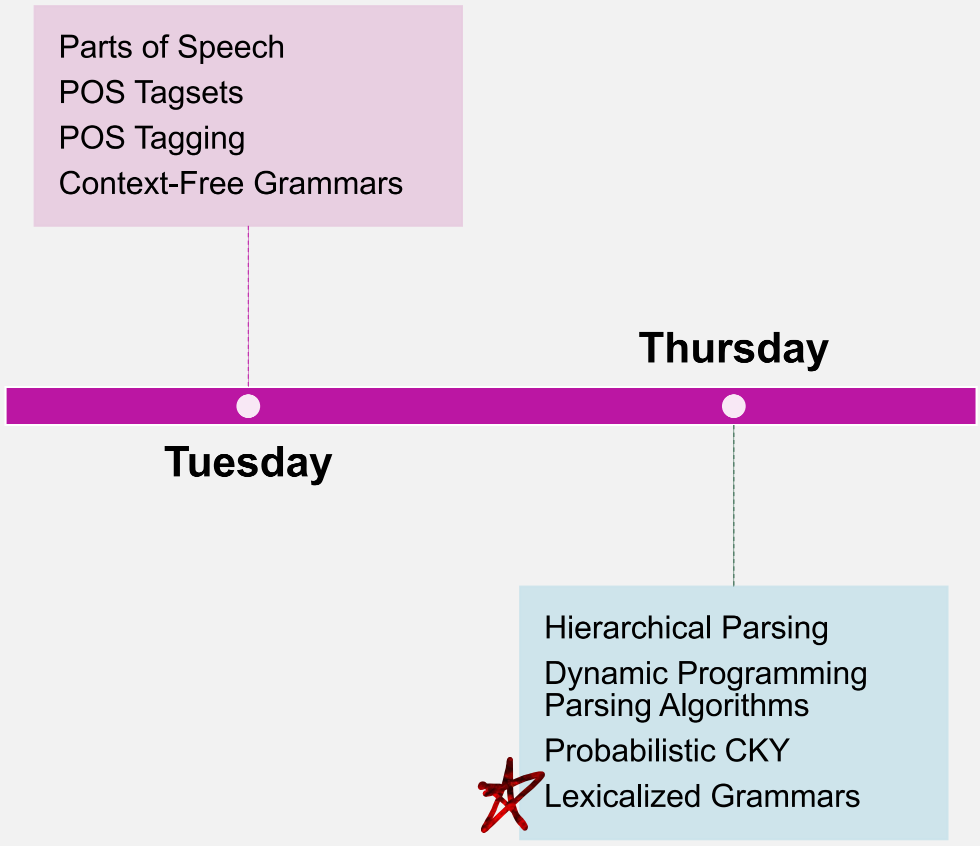
Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	S (0.8 * 0.0012 * 0.000024 = $2.304*10^{-8}$)
1		N (0.01)	☹	☹	☹
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Challenges Associated with PCFGs

- PCFGs solve many issues associated with resolving ambiguities, but they still have:
  - **Poor independence assumptions**, which may make it difficult to model important **structural dependencies** in the parse tree
  - **Lack of lexical conditioning**, which may allow **lexical dependency issues** (e.g., those dealing with preposition attachment or other syntactic properties) to arise
- How can we address these lingering limitations?
  - Adding extra constraints to rules by splitting them based on their parents or their syntactic positions
  - Using slightly different grammatical paradigms, such as **probabilistic lexicalized CFGs**

This Week's Topics



Parts of Speech
POS Tagsets
POS Tagging
Context-Free Grammars

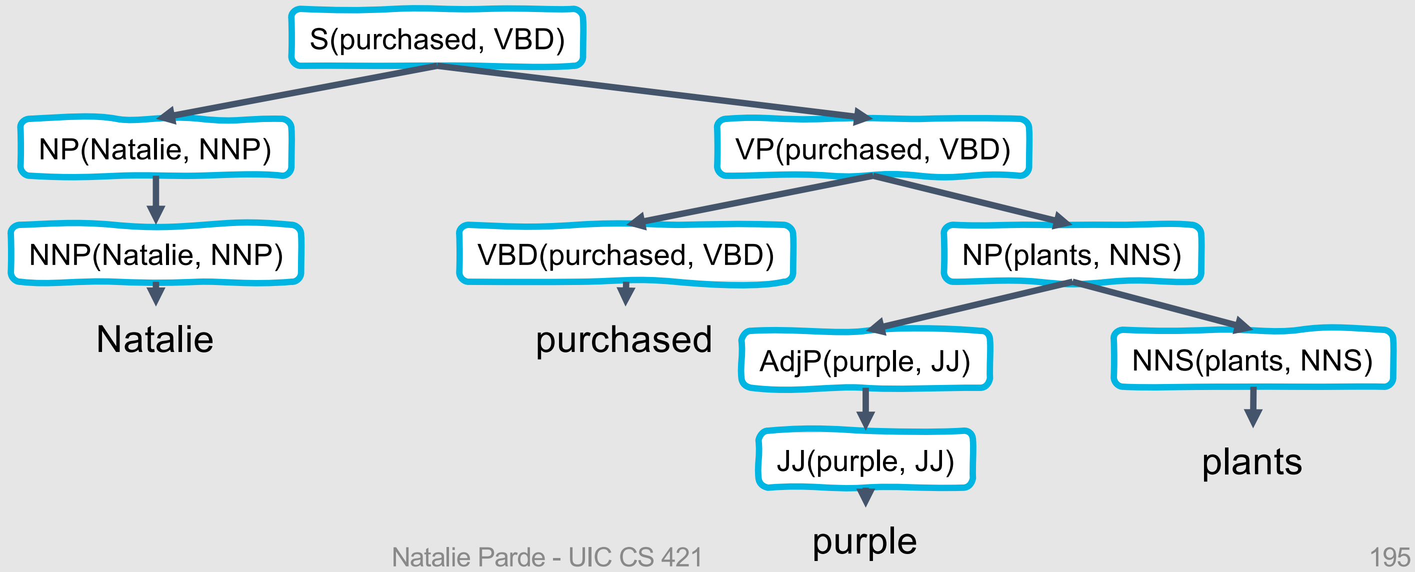
Thursday

Tuesday

Hierarchical Parsing
Dynamic Programming Parsing Algorithms
Probabilistic CKY
Lexicalized Grammars

Lexicalized Parsers

- Allow lexicalized rules
  - Non-terminals specify lexical heads and associated POS tags
  - NP(plants, NNS) → AdjP(purple, JJ) NNS(plants, NNS)



Lexicalized Production Rules

- Lexicalized rules thus are more complex:
  - $LHS \rightarrow L_n \; L_{n-1} \ldots L_1 \; H \; R_1 \ldots R_{n-1} \; R_n$

Non-terminal LHS
NP(plants, NNS)

Non-terminal left of head
AdjP(purple, JJ)

Head
NNS(plants, NNS)

Non-terminal right of head
PP(under, P)

P_L: Probability for generating dependents on the left

P_H: Probability for generating heads

P_R: Probability for generating dependents on the right

The Collins Parser

- Goal: Use P_H, P_L, and P_R to estimate the overall probability for the production rule
- Method:
  - Surround the righthand side of the rule with STOP non-terminals
    - NP(plants, NNS) → STOP AdjP(purple, JJ) NNS(plants, NNS) PP(under, IN) STOP
  - Compute the individual P_H, P_L, and P_R values for the head and the non-terminals to its left and right (including STOP non-terminals)
  - Multiply these together

Grab the **purple plants under the bookcase**.

The Collins Parser

- Consider the following generic production rule:
  - $LHS \rightarrow L_n \; L_{n-1} \ldots L_1 \; H \; R_1 \ldots R_{n-1} \; R_n$

Non-terminal LHS
NP(plants, NNS)

Non-terminal left of head
AdjP(purple, JJ)

Head
NNS(plants, NNS)

Non-terminal right of head
PP(under, IN)

***P_L*: Probability for generating dependents on the left**

***P_H*: Probability for generating heads**

***P_R*: Probability for generating dependents on the right**

Grab the **purple plants under the bookcase**.

NP(facemasks, NNS) → STOP AdjP(purple, JJ) NNS(plants, NNS) **PP(under, IN) STOP**

= P_H(H|LHS) * P_L(STOP|LHS H) * P_L(L_1|LHS H) * P_R(R_1|LHS H) * P_R(STOP|LHS H)

P_H(H|LHS) = P(NNS(plants, NNS) | NP(plants, NNS))

P_L(STOP|LHS H) = P(STOP | NP(plants, NNS) NNS(plants, NNS))

P_L(L_1|LHS H) = P(AdjP(purple, JJ) | NP(plants, NNS) NNS(plants, NNS))

P_R(R_1|LHS H) = P(PP(under, IN) | NP(plants, NNS) NNS(plants, NNS))

P_R(STOP|LHS H) = P(STOP | NP(plants, NNS) NNS(plants, NNS))

Estimate the individual probabilities using maximum likelihood estimates.

$P_R(R_1|\text{LHS H}) = P(\text{PP(under, IN)} \mid \text{NP(plants, NNS) NNS(plants, NNS)})$

$$\downarrow$$

$$\frac{\text{Count}(\text{NP}(\text{plants}, \text{NNS})\ \textit{with}\ \text{PP}(\text{under}, \text{IN})\ \textit{as a child to the right})}{\text{Count}(\text{NP}(\text{plants}, \text{NNS}))}$$

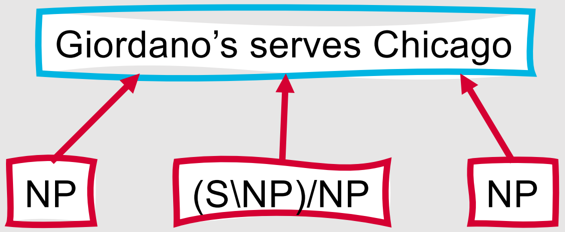
Combinatory Categorial Grammars (CCGs)

- *Heavily* lexicalized way to group words from a lexicon into categories and define rules indicating how those categories may be combined
- CCG categories include:
  - **Atomic elements**
    - $\mathcal{A} \subseteq \mathcal{C}$, where $\mathcal{A}$ is a set of atomic elements, and $\mathcal{C}$ is the set of categories for the grammar
    - Simple noun phrases
  - **Single-argument functions**
    - (X/Y), (X\Y) $\in \mathcal{C}$, if $\mathrm{X}, \mathrm{Y} \in \mathcal{C}$
      - (X/Y): Seeks a constituent of type Y to the right, and returns X
      - (X\Y): Seeks a constituent of type Y to the left, and returns X
    - Verb phrases, more complex noun phrases, etc.

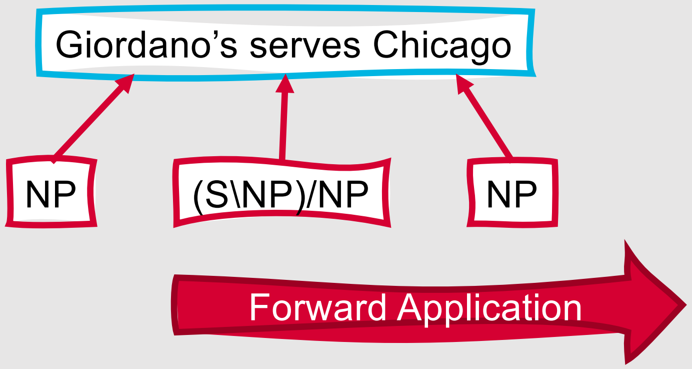
CCG Lexica and Rules

- CCG lexica assign CCG categories to words
  - Chicago: NP
    - **Atomic category**
  - cancel: (S\NP)/NP
    - **Functional category**
    - Seeks an NP to the right, returning (S\NP), which seeks an NP to the left, returning S
- CCG rules specify how functions and their arguments may be combined
  - **Forward function application:** Applies the function to its argument on the right, resulting in the specified category
    - X/Y Y ⇒ X
  - **Backward function application:** Applies the function to its argument on the left, resulting in the specified category
    - Y X\Y ⇒ X
  - A coordination rule can also be applied
    - X CONJ X ⇒ X

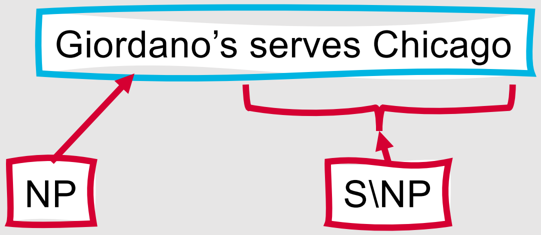
CCG Rules: Example



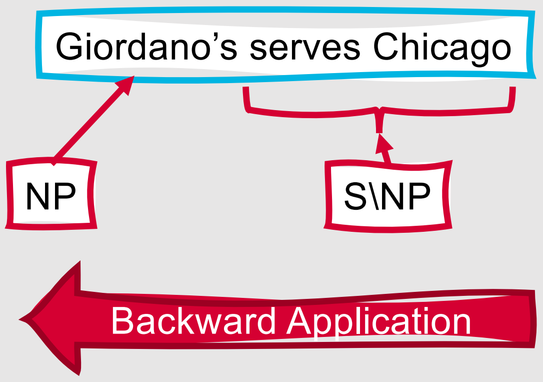
CCG Rules: Example



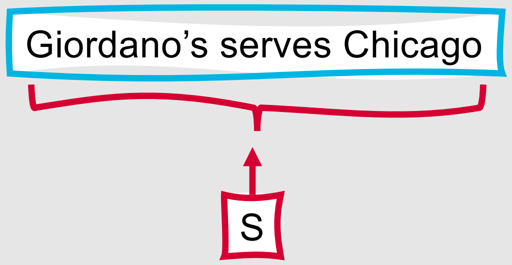
CCG Rules: Example



CCG Rules: Example



CCG Rules: Example



We now know how to build parsers (in many different ways)! How can we evaluate them?

- **PARSEVAL measures:** Seek to determine how close a predicted parse is to a gold standard parse for the same text, based on its individual constituents
  - Constituent is correct if it matches a constituent in the gold standard in terms of its:
    - Starting point
    - Ending point
    - Non-terminal symbol

Once constituent correctness is defined….

- We can apply the same metrics we use for other NLP problems!
  - $\text{Recall} = \frac{\#\text{ correct constituents in predicted parse}}{\#\text{ constituents in gold standard parse}}$
  - $\text{Precision} = \frac{\#\text{ correct constituents in predicted parse}}{\#\text{ constituents in predicted parse}}$

Summary: Constituency Parsing

Constituency parsing is a way to automatically describe the structure of an input sentence according to a constituency grammar

Constituency parsing can be performed using either a **top-down** or a **bottom-up approach**

The **CKY algorithm** and **Earley algorithm** are popular dynamic programming approaches to parsing that work in a bottom-up and top-down manner, respectively

We can select the best parse for a sentence using **probabilistic context-free grammars**

The **CKY algorithm** can be updated to incorporate these probabilities for use with PCFG parsing

An alternative parsing paradigm uses **lexicalized grammar frameworks**

We can evaluate parsers using standard NLP metrics applied based on the number of **correctly identified constituents** in a predicted parse