



NUS
National University
of Singapore

| **Computing**

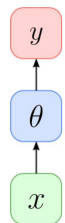
CS4248: Natural Language Processing

Lecture 9 — Trees

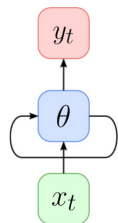
Recap of Week 08

Recurrent Neural Network — Basic Idea

Feedforward NN



Recurrent NN



x is now a sequence of vectors
(e.g., word embeddings)

Core concept of RNNs: **Hidden State**

- Additional vector incorporated into the network
- Commonly holds the last output of the hidden layer
→ size of hidden state = size of hidden layer
- Randomly initialized, and to be tuned through training (→ backpropagation)
- Basic recurrent formula:

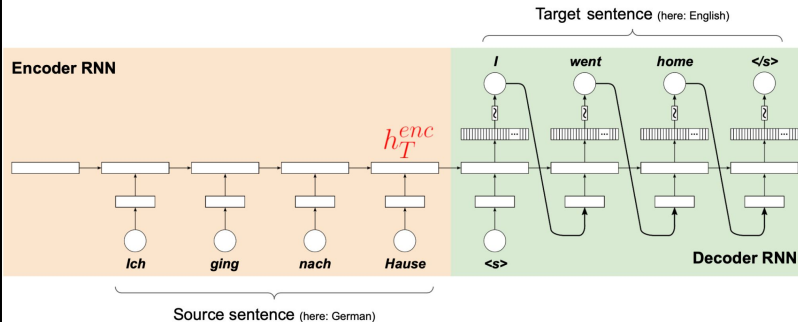
$$h_t = f_{\theta}(h_{t-1}, x_t)$$

hidden state of
time step $t-1$

input vector at
time step t

20

Encoder-Decoder (Sutskever et al.; 2014)



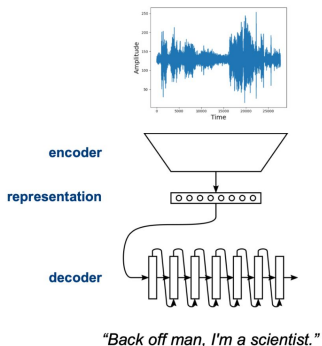
55

Encoder-Decoder Architecture

- Basic 2-component setup

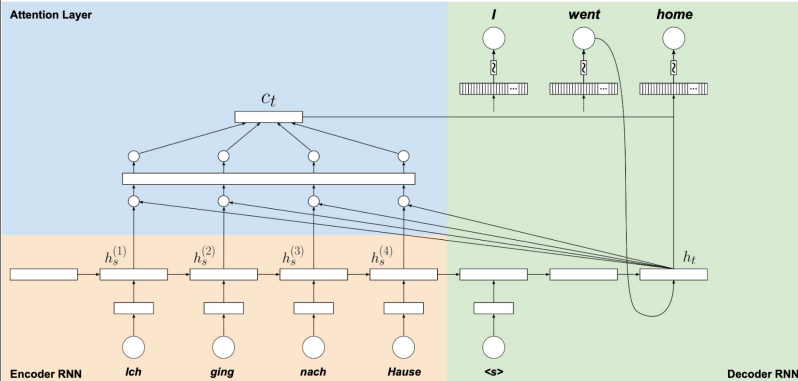
(1) Encoder

- Learns function that maps context into a fixed-size vector representation c
- Encoder architecture depending on context
(e.g., CNN for images, RNN for text)



50

Attention — Walkthrough



67

Announcements

Assignments

- Assignment 1: Graders are examining your requests.
- Assignment 1: We are also processing our requests on possible academic dishonesty.
- [Assignment 3](#) out: Theory and Practice, 3 problems.

Projects

- Intermediate Updates to be disseminated soon. 5% of your total grade, uniformly distributed.
- First [TEAMMATES](#) intra-peer evaluation has been completed. Please check your own results.
- Min and Chris will be announcing a sign-up sheet for teams to optionally consult us on their project.
- Teams can be nominated or self-nominate for STePS (Week 13 Wed) for poster styled presentation instead.

Outline

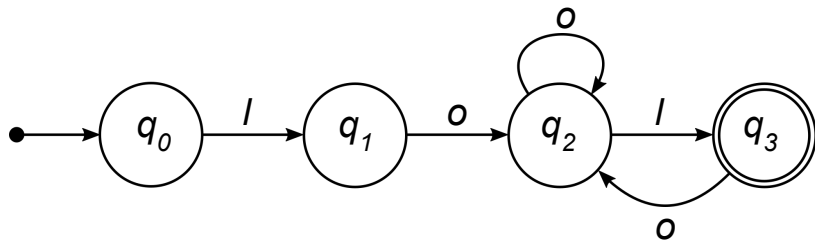
- **Syntactic Parsing**
 - **Quick recap: RegEx**
 - Context-free grammars
 - Structural Ambiguity
 - Chomsky Normal Form
- **CYK Parsing Algorithm**
 - Base membership algorithm
 - Find all parse trees with backtracking
 - Probabilistic parsing
 - Evaluation of parser

Throwback — Regular Expression

- Equivalence

- Regular Expressions describe **Regular Languages**
(most restricted types of languages w.r.t Chomsky Hierarchy)
- Regular Language = language accepted by a FSA

Example: FSA that accepts the Regular Language described by the Regular Expression $I(o+I)^+$



Regular Expression

$I(o+I)^+$

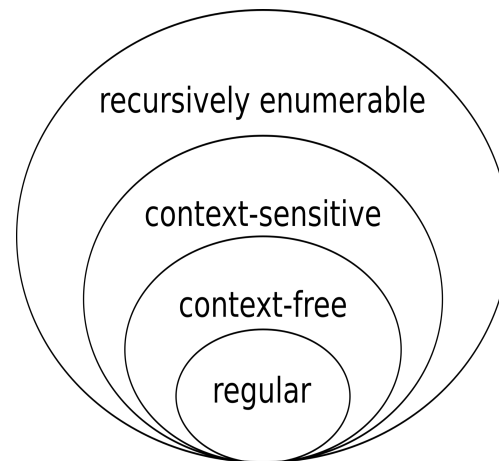


Regular Language

$\{I, Io, IIo, IIoI, IIoIoI, \dots\}$

Chomsky Hierarchy

(Source: [Wikipedia](https://en.wikipedia.org/wiki/Chomsky_hierarchy))





Quick Quiz

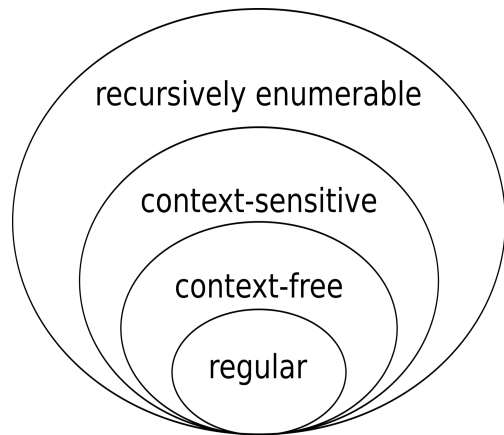


Regular Expressions — Limitations

- Not all languages can be described using RegEx

- Example:

$$\{0^n 1^n \mid n \geq 0\} = \{\epsilon, 01, 0011, 000111, 00001111, \dots\}$$



→ Natural Language is not a Regular Language

- Natural language allows for nested structures (**center embeddings**)

The food was delicious

*The food **Alice cooked** was delicious*

*The food **Alice the sister of Bob cooked** was delicious*

Syntax & Constituency

- Important questions

- How are words combined to form phrases?
- How are phrases combined to form longer phrases?
- How are phrases are combined to form sentences?

} How meaning is mapped onto what language structures?

- Important concept: constituency = phrase structure

- Constituent = group of words that behave as a single unit

Constituents

- **Constituent — Definition**

- Group of words that behaves as a single unit or phrase
(by default: individual words are constituents, but there are exceptions)
- Sentences can be described as a hierarchical structure of constituents (in a bit: parse trees)

- **Question: How do we know a group of words forms a constituent?**

- Handwavy answer: Group of words that “make sense” on its own

“She heard a loud shot from the bank during the time of the robbery.”

a loud shot

a loud shot from the

a loud shot from the bank

- Formal answer: **Constituency Tests**

Constituency Tests (some examples)

- Topicalization

- Only a constituent can be moved to different locations in a sentence

*“They met **at 8 pm** for dinner.” — “They met for dinner **at 8 pm**.” — “**At 8 pm**, they met for dinner.”*

- Proform substitution

- Only a constituent can be substituted with a proform like *it*, *that*, *them*, *then*, *there*, etc.

*“Chris went **back to Germany**.” → “Chris went there.”*

- Fragment Answers

- Only a constituent can answer a question while retaining the meaning of the original sentence.

*“Alice was hit by **the green car**.” — Q: “What hit Alice?” → “The green car.”*



Quick Quiz



Outline

- **Syntactic Parsing**

- Quick recap: RegEx
- **Context-free grammars**
- Structural Ambiguity
- Chomsky Normal Form

- **CYK Parsing Algorithm**

- Base membership algorithm
- Find all parse trees with backtracking
- Probabilistic parsing
- Evaluation of parser

Context-Free Grammars (CFGs)

- Context-Free Grammars

- Most common way to capture constituency and ordering → good fit for natural language!
(in fact, context-free grammars were first used to study human languages to describe the structure of sentences)
- Define what meaningful constituents are and how a constituent is formed out of other constituents
- More powerful than RegExs as they can express recursive structure
(in contrast, context free grammars can describe regular languages)

special start symbol → $S \rightarrow NP VP$

$NP \rightarrow Det Noun$

$VP \rightarrow Verb NP$

$Det \rightarrow a \mid the$

$Noun \rightarrow man \mid meal \mid flight$

$Verb \rightarrow saw \mid booked$

set of **rules** or **productions**

- Example

Non-terminal symbols

- Symbols that can be replaced according to rules
- For natural language grammars: phrase names, part of speech

Terminal symbols

- May be the output of a rule; cannot be changed/replaced further
- For natural language grammars: words/tokens

Context-Free Grammars (CFGs)

$S \rightarrow NP VP$
 $NP \rightarrow Det Noun$
 $VP \rightarrow Verb NP$
 $Det \rightarrow a \mid the$
 $Noun \rightarrow man \mid meal \mid flight$
 $Verb \rightarrow saw \mid booked$

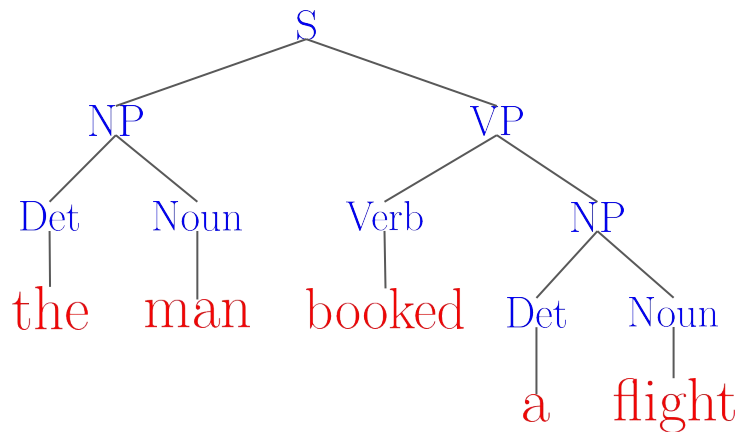
- Application of rules — example

$S \rightarrow NP VP$
 $\rightarrow Det Noun VP$
 $\rightarrow the Noun VP$
 $\rightarrow the man VP$
 $\rightarrow the man Verb NP$
 $\rightarrow the man booked NP$
 $\rightarrow the man booked Det Noun$
 $\rightarrow the man booked a Noun$
 $\rightarrow the man booked a flight$



Derivation: sequence of rules/productions used to generate a string of words

Visualization as Parse Tree



CFG — Formal Definition

- A CFG is a 4-tuple $\langle N, \Sigma, R, S \rangle$

- N — set of non-terminal symbols
- Σ — set of terminal symbols
- R — set of rules

Allowed format for all rules

$A \rightarrow \alpha$ with $A \in N, \alpha \in (N \cup \Sigma)$

- S — start symbol

Example

$N = \{\text{NP, VP, Det, Noun, Verb}\}$

$\Sigma = \{a, \text{the}, \text{man}, \text{meal}, \text{flight}, \text{saw}, \text{booked}\}$

$S \rightarrow \text{NP VP}$

$\text{NP} \rightarrow \text{Det Noun}$

$\text{VP} \rightarrow \text{Verb NP}$

$\text{Det} \rightarrow a \mid \text{the}$

...



Quick Quiz



CFG — Running Example

S \rightarrow NP VP
S \rightarrow Aux NP VP
S \rightarrow VP
NP \rightarrow Pronoun
NP \rightarrow ProperNoun
NP \rightarrow Det Nominal
Nominal \rightarrow Noun
Nominal \rightarrow Nominal Noun
Nominal \rightarrow Nominal PP
VP \rightarrow Verb
VP \rightarrow Verb NP
VP \rightarrow Verb NP PP
VP \rightarrow Verb PP
VP \rightarrow VP PP
PP \rightarrow Prep NP

Det \rightarrow *the* | *a* | *that* | *this*
Noun \rightarrow *book* | *flight* | *meal* | *money*
Verb \rightarrow *book* | *include* | *prefer*
Pronoun \rightarrow *I* | *she* | *he* | *me*
ProperNoun \rightarrow *Singapore* | *Frankfurt* | *SIA*
Aux \rightarrow *do* | *does* | *did*
Prep \rightarrow *from* | *to* | *on* | *near* | *through*

Important requirements to make it a CFG

- Only single non terminals on the right-hand side
- Application of a rule does not depend on context



Quick Quiz



Outline

- **Syntactic Parsing**
 - Quick recap: RegEx
 - Context-free grammars
 - **Structural Ambiguity**
 - Chomsky Normal Form
- **CYK Parsing Algorithm**
 - Base membership algorithm
 - Find all parse trees with backtracking
 - Probabilistic parsing
 - Evaluation of parser

Ambiguity

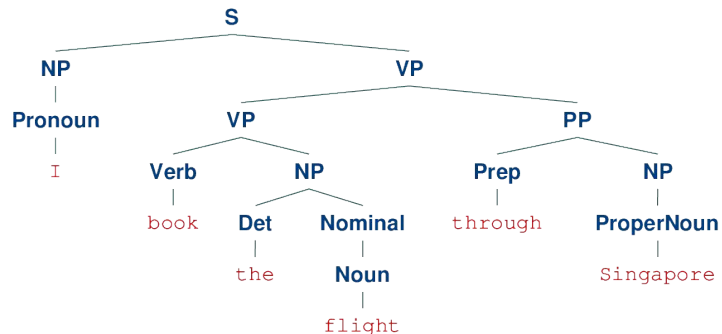
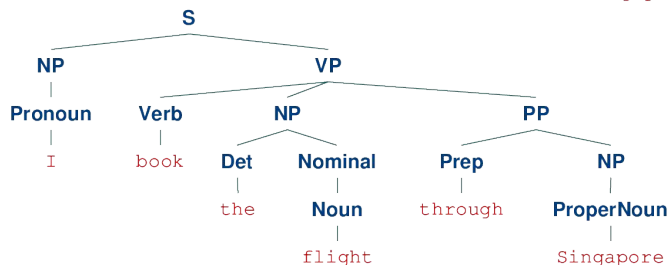
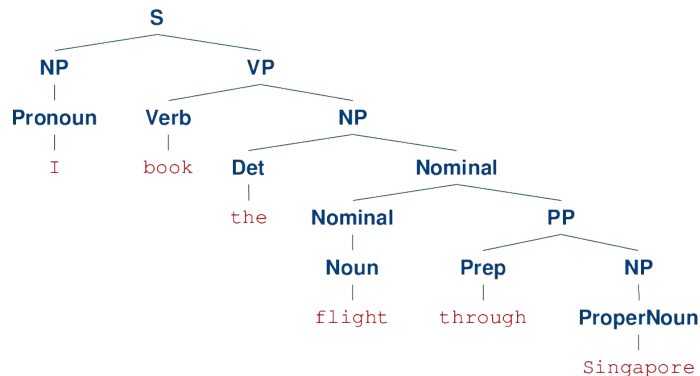
- Ambiguity of Natural Language

- Common: multiple ways to interpret a sentence
- Different interpretation → different meaning

→ Structural Ambiguity

- A grammar can assign more than one parse to a sentence
- Example (using our toy grammar):

“I book the flight through Singapore”





Structural Ambiguity

- Two common types of Structural Ambiguity

(1) Attachment Ambiguity

- A particular constituent can be attached to the parse tree at more than one place

(2) Coordination Ambiguity

- Phrases can be conjoined by conjunction like “*and*”, “*or*”, “*but*”, “*because*”, “*if*”, etc.
- Different types of conjunctions
(coordinating conjunctions, correlative conjunctions, correlative conjunctions)



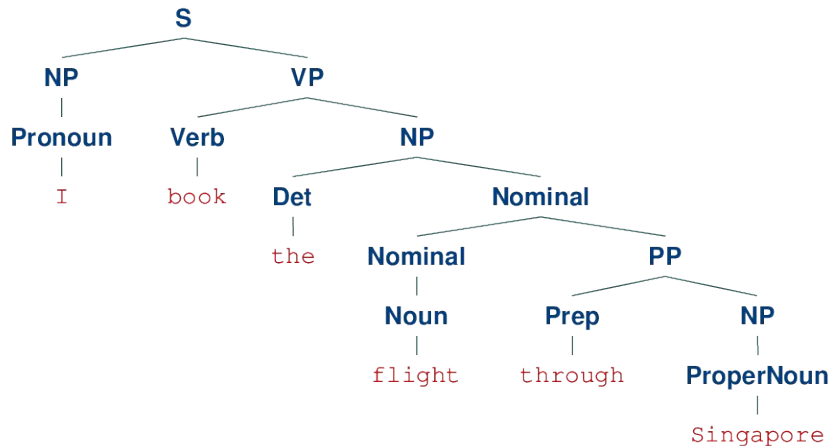
Which parse is correct?



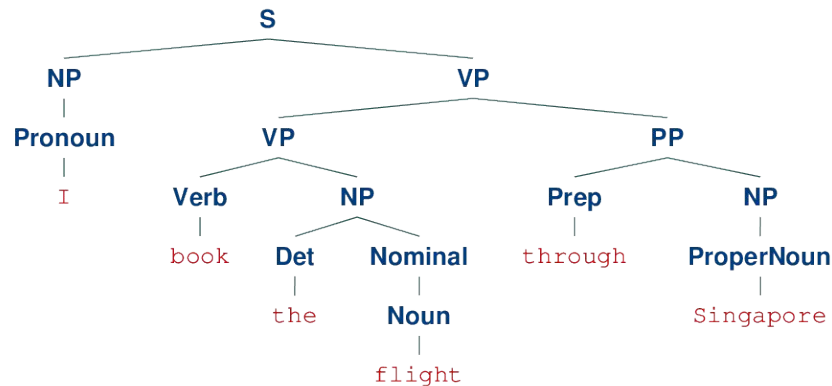


Attachment Ambiguity

“I book the flight through Singapore”



“through Singapore” attached to
noun phrase related to *“flight”*



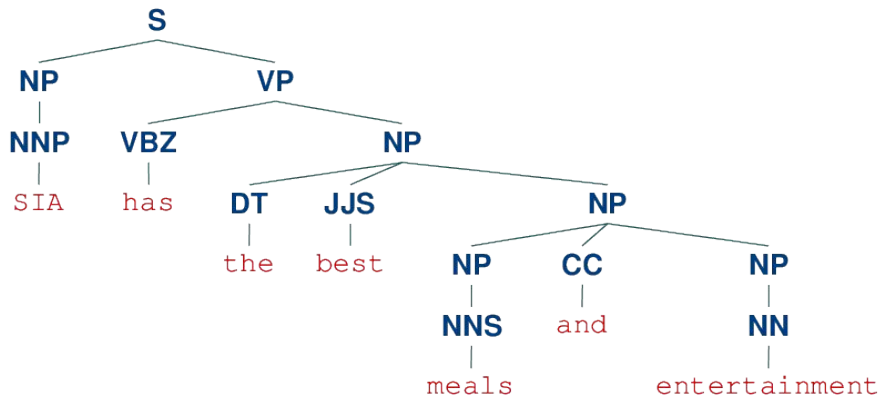
“through Singapore” attached to
verb phrase related to *“book”*

(like booking a flight through an agent)

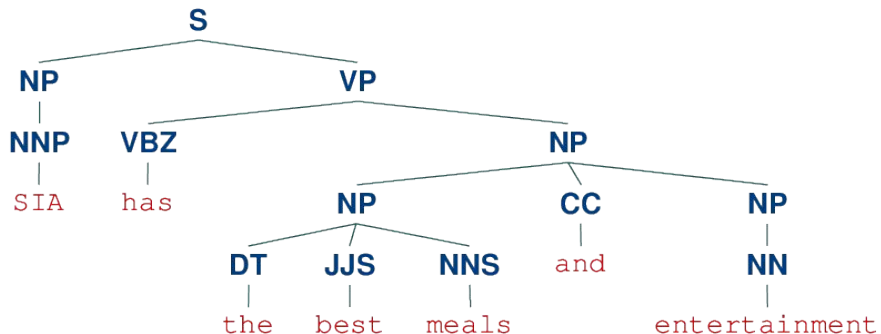


Coordination Ambiguity

“SIA has the best meals and entertainment”



“best” refers to both the meals
and the entertainment



“best” refers to only the meals
but not the entertainment

Note: This example used a different grammar since our toy grammar does not support conjunctions.

Pre-Lecture Activity for Last Week

- Assigned Task

- Watch the 9-minute YouTube video linked below
- Take an ambiguous [news headline](#) and explain one strategy mentioned in the video
- Post a 1-2 sentence answer to the following questions in your Tutorial Group's discussion

The Ling Space:

["How Do We Interpret Sentences? Parsing Strategies"](#)



Side notes:

- This task is meant as a warm-up to provide some context for the next lecture
- No worries if you get lost; we will talk about this in the next lecture
- You can just copy-&-paste others' answers but this won't help you learn better





Summary So Far...

- Parsing as a 2-part task

(1) Syntactic Parsing

- Extract all possible parses for a sentence
- Typically requires a grammar transformation step
(“binarization” of grammar to ensure efficient parsing)

(2) Syntactic Disambiguation

- Score all parses and return the best parse
- Scores commonly expressed as probability

Outline

- **Syntactic Parsing**
 - Quick recap: RegEx
 - Context-free grammars
 - Structural Ambiguity
 - **Chomsky Normal Form**
- **CYK Parsing Algorithm**
 - Base membership algorithm
 - Find all parse trees with backtracking
 - Probabilistic parsing
 - Evaluation of parser

Grammar Transformation (for CFGs)

- Important requirement: binarized rules
 - No more than 2 non-terminals on the right-hand side of rules
 - Crucial for a cubic time parsing of CFGs
- Common transformation: **Chomsky Normal Form**
 - Restrictions on rules compared to general CFG

Allowed format for all rules

$A \rightarrow \alpha$ with $A \in N$, $\alpha \in (N \cup \Sigma)$

α is either 1 terminal OR 2 non-terminals



A Great Way to Fly

Allowed format for all rules

$A \rightarrow \alpha$ with $A \in N$, $\alpha \in (N \cup \Sigma)$

α is either 1 terminal OR 2 non-terminals

$S \rightarrow NP VP$

$S \rightarrow Aux NP VP$

$S \rightarrow VP$

$NP \rightarrow Pronoun$

$NP \rightarrow ProperNoun$

$NP \rightarrow Det Nominal$

$Nominal \rightarrow Noun$

$Nominal \rightarrow Nominal Noun$

$Nominal \rightarrow Nominal PP$

$VP \rightarrow Verb$

$VP \rightarrow Verb NP$

$VP \rightarrow Verb NP PP$

$VP \rightarrow Verb PP$

$VP \rightarrow VP PP$

$PP \rightarrow Prep NP$

$Det \rightarrow the \mid a \mid that \mid this$

$Noun \rightarrow book \mid flight \mid meal \mid money$

$Verb \rightarrow book \mid include \mid prefer$

$Pronoun \rightarrow I \mid she \mid he \mid me$

$ProperNoun \rightarrow Singapore \mid Frankfurt \mid SIA$

$Aux \rightarrow do \mid does \mid did$

$Prep \rightarrow from \mid to \mid on \mid near \mid through$

Quick Quiz:

- Which rules do not conform to the Chomsky Normal Form?
- How can we transform the grammar to fix this?

Chomsky Normal Form (CNF)

- Two basic transformation steps

(1) Recursive removal of unary rules (and empty rules)

Nominal \rightarrow Noun
Noun $\rightarrow book \mid flight \mid meal \mid money$ \rightarrow Nominal $\rightarrow book \mid flight \mid meal \mid money$
Noun $\rightarrow book \mid flight \mid meal \mid money$

(2) Dividing n-ary rules by introducing new non-terminals

(n-ary rule = rule with $n > 2$ non-terminal on the right-hand side)

$S \rightarrow Aux \ NP \ VP$ \rightarrow $S \rightarrow X \ VP$
 $X \rightarrow Aux \ NP$

Toy Grammar in Chomsky Normal Form CNF

$S \rightarrow NP VP$
 $S \rightarrow X1 VP$
 $X1 \rightarrow Aux NP$
 $S \rightarrow Verb NP$
 $S \rightarrow X2 PP$
 $S \rightarrow Verb PP$
 $S \rightarrow VP PP$
 $NP \rightarrow Det Nominal$
 $Nominal \rightarrow Nominal Noun$
 $Nominal \rightarrow Nominal PP$
 $VP \rightarrow Verb NP$
 $VP \rightarrow X2 PP$
 $X2 \rightarrow Verb NP$
 $VP \rightarrow Verb PP$
 $VP \rightarrow VP PP$
 $PP \rightarrow Prep NP$

$Det \rightarrow the \mid a \mid that \mid this$
 $Noun \rightarrow book \mid flight \mid meal \mid money$
 $Verb \rightarrow book \mid include \mid prefer$
 $Pronoun \rightarrow I \mid she \mid he \mid me$
 $PropNoun \rightarrow Singapore \mid Frankfurt \mid SIA$
 $Aux \rightarrow do \mid does \mid did$
 $Prep \rightarrow from \mid to \mid on \mid near \mid through$
 $S \rightarrow book \mid include \mid prefer$
 $VP \rightarrow book \mid include \mid prefer$
 $NP \rightarrow I \mid she \mid he \mid me$
 $NP \rightarrow Singapore \mid Frankfurt \mid SIA$
 $Nominal \rightarrow book \mid flight \mid meal \mid money$

Allowed format for all rules

$A \rightarrow \alpha$ with $A \in N$, $\alpha \in (N \cup \Sigma)$

α is either 1 terminal OR 2 non-terminals



CFG to CNF — Summary

- Transformation of a CFG to a CNF
 - Every CFG can be transformed into a **weakly-equivalent** CNF

→ Weak equivalence

- Two grammars generate the same set of sentences (identical expressiveness)
- The derivations generating the same sentences may differ
(recall that the CNF may introduce additional non-terminals)

(Strong equivalence: identical expressiveness + identical derivations)

Break

Outline

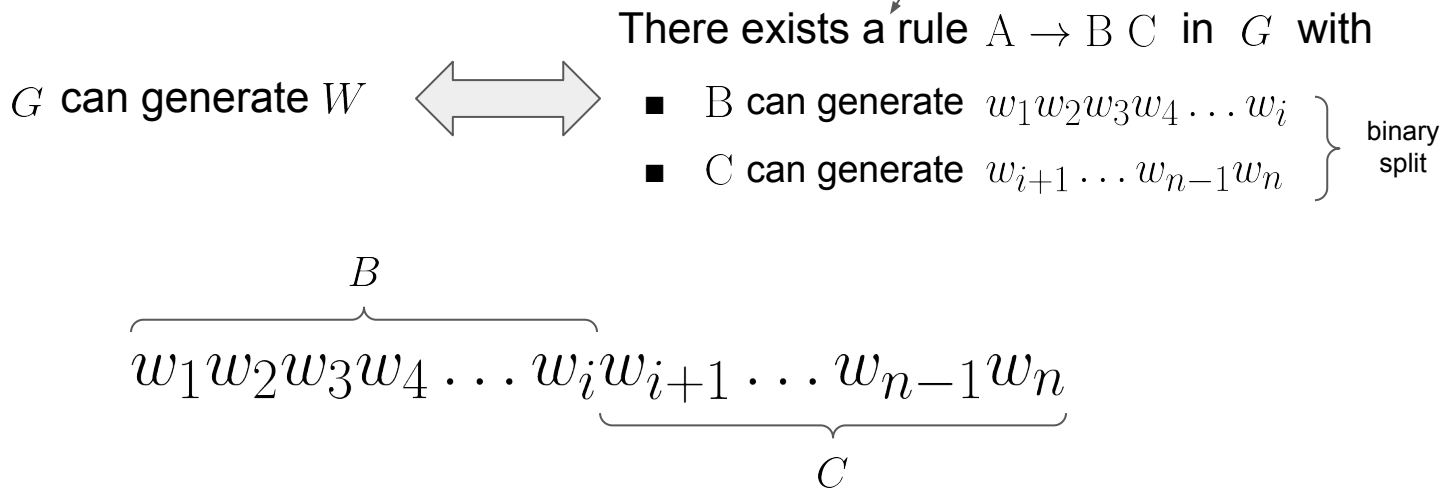
- Syntactic Parsing
 - Quick recap: RegEx
 - Context-free grammars
 - Structural Ambiguity
 - Chomsky Normal Form
- CYK Parsing Algorithm
 - **Base membership algorithm**
 - Find all parse trees with backtracking
 - Probabilistic parsing
 - Evaluation of parser

CYK Parsing Algorithm

- CYK Parsing Algorithm — basic intuition

- Given is a context-free grammar G in CNF
- Assume we have a sentence W comprising n words

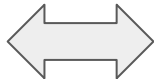
There can be multiple rules for different i ,
but at least one rule for at least one i .



Note: Appreciate how the “binarization” of rules helps here!

CYK Parsing Algorithm

$$\overbrace{w_1 w_2 w_3 w_4 \dots w_i}^B \underbrace{w_{i+1} \dots w_{n-1} w_n}_C$$

→ Recursive nature: G can generate B  There exists a rule $X \rightarrow Y Z$ so that X and Y can generate a binary split of $w_1 w_2 w_3 \dots w_i$

\vdots

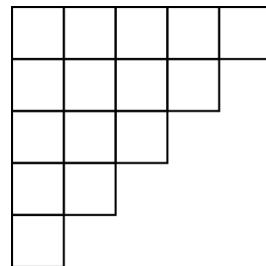
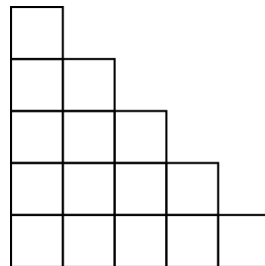
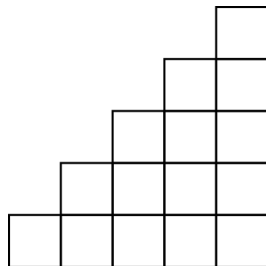
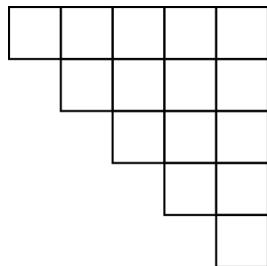
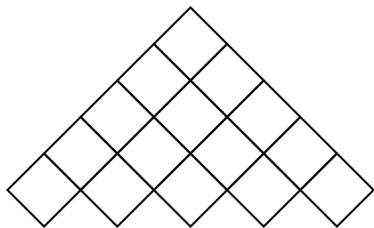
(until we reach individual words; then check the lexicon rules)

→ CYK Parsing Algorithm: Solve problem using Dynamic Programming

- Find all possible parses for all sequences of size k for k from 1 to n

CYK Parsing Algorithm

- Dynamic Programming approach
 - Completing the parse table in a bottom-up manner
(very similar idea as we have seen for calculating the Minimum Edit Distance)
 - Can to handle redundancy when computing the parse trees
- Different ways to visualize parse table
 - Completely identical, just that the indexing of table cells differs



CYK — Parse Table

[1,4] = all possible parses for span "book the flight"

I	book	the	flight	through	Singapore
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]	[0,6]
	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]
		[2,3]	[2,4]	[2,5]	[2,6]
			[3,4]	[3,5]	[3,6]
				[4,5]	[4,6]
					[5,6]

CYK parse table

- $N \times N$ table
(N = # words in sentence)
- Each cell represents all the possible parses for span $[i, j]$
- Algorithm: fill table starting with cells for spans of length $L = 1$ to cells for spans of increasing lengths

L	Cells
1	[0,1], [1,2], [2,3], [3,4], [4,5], [5,6]
2	[0,2], [1,3], [2,4], [3,5], [4,6]
3	[0,3], [1,4], [2,5], [3,6]
4	[0,4], [1,5], [2,6]
5	[0,5], [1,6]
6	[0,6]



Fill in the Lexicon



book



CYK — Walkthrough

Quick quiz: What does it mean that cell [1,2] contains start symbol S?

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2]	[0,3]	[0,4]	[0,5]	[0,6]
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4]	[1,5]	[1,6]
		[2,3] Det	[2,4]	[2,5]	[2,6]
			[3,4] Nominal, Noun	[3,5]	[3,6]
				[4,5] Prep	[4,6]
					[5,6] PropNoun, NP

Cells for spans of length $L = 1$

→ only need to check lexicon

Fill each cell with the non-terminals that can generate the corresponding word

Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | he | me
PropNoun → Singapore | Frankfurt | SIA
Aux → do | does | did
Prep → from | to | on | near | through
S → book | include | prefer
VP → book | include | prefer
NP → I | she | he | me
NP → Singapore | Frankfurt | SIA
Nominal → book | flight | meal | money

CYK — Walkthrough

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4]	[0,5]	[0,6]
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4]	[1,5]	[1,6]
		[2,3] Det	[2,4] NP	[2,5]	[2,6]
			[3,4] Nominal, Noun	[3,5]	[3,6]
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

Cells for spans of length $L > 1$

→ Check for each binary split if there is a production rule that can generate split

Example: Cell **[0,2]**

→ only 1 binary split: [0,1] / [1,2]

Check each possible pair of non-terminals of binary split is the RHS of an existing production rule → Yes, add LHS to cell

LHS	RHS
—	Pronoun S
—	Pronoun VP
—	Pronoun Nominal
—	Pronoun Noun
—	Pronoun Verb
—	NP S
S	NP VP
—	NP Nominal
—	NP Noun
—	NP Verb

Only this rule exists in our grammar

CYK — Walkthrough

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4]	[0,5]	[0,6]
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4] S, VP, X2	[1,5]	[1,6]
		[2,3] Det	[2,4] NP	[2,5]	[2,6]
			[3,4] Nominal, Noun	[3,5]	[3,6] Nominal
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

Quick quiz: Can you already guess how the parse table indicates that a sentence is valid?

Example: Cell [1,4]

- binary split: [1,2] / [2,4]
- binary split: [1,3] / [3,4]

Binary split: [1,2] / [2,4]

LHS	RHS
—	S NP
—	VP NP
—	Nominal NP
—	Noun NP
S, VP, X2	Verb NP

3 existing rules with the same RHS

Binary split: [1,3] / [3,4]

LHS	RHS
-----	-----

Empty because [1,3] is empty

CYK — Walkthrough

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4] S	[0,5]	[0,6]
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4] S, VP, X2	[1,5]	[1,6]
		[2,3] Det	[2,4] NP	[2,5]	[2,6] NP
			[3,4] Nominal, Noun	[3,5]	[3,6] Nominal
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

Example: Cell [2,6]

- binary split: [2,3] / [3,6]
- binary split: [2,4] / [4,6]
- binary split: [2,5] / [5,6]

Binary split: [2,3] / [3,6]

LHS	RHS
NP	Det Nominal

Binary split: [2,4] / [4,6]

LHS	RHS
—	NP PP

Binary split: [2,5] / [5,6]

LHS	RHS
-----	-----



CYK — TraceTogether

Example: Cell **[1,6]**

- binary split: [1,2] / [2,6]
- binary split: [1,3] / [3,6]
- binary split: [1,4] / [4,6]
- binary split: [1,5] / [5,6]

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4] S	[0,5]	[0,6]
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4] S, VP, X2	[1,5]	[1,6] S, VP, X2
		[2,3] Det	[2,4] NP	[2,5]	[2,6] NP
			[3,4] Nominal, Noun	[3,5]	[3,6] Nominal
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

CYK — Walkthrough (Done)

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4] S	[0,5]	[0,6]
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4] S, VP, X2	[1,5]	[1,6] S, VP, X2
		[2,3] Det	[2,4] NP	[2,5]	[2,6] NP
			[3,4] Nominal, Noun	[3,5]	[3,6] Nominal
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

Example: Cell [1,6]

- binary split: [1,2] / [2,6]
- binary split: [1,3] / [3,6] (empty!)
- binary split: [1,4] / [4,6]
- binary split: [1,5] / [5,6] (empty!)

Binary split: [1,2] / [2,6]

LHS	RHS
—	S NP
—	VP NP
—	Nominal NP
—	Noun NP
S, VP, X2	Verb NP

Binary split: [1,4] / [4,6]

LHS	RHS
—	S PP
S, VP	VP PP
S, VP	X2 PP

CYK — Walkthrough

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4] S	[0,5]	[0,6] S
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4] S, VP, X2	[1,5]	[1,6] S, VP, X2
		[2,3] Det	[2,4] NP	[2,5]	[2,6] NP
			[3,4] Nominal, Noun	[3,5]	[3,6] Nominal
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

Example: Cell [0,6]

- binary split: [0,1] / [1,6]
- binary split: [0,2] / [2,6]
- binary split: [0,3] / [3,6] (empty!)
- binary split: [0,4] / [4,6]
- binary split: [0,5] / [5,6] (empty!)

Binary split: [0,1] / [1,6]

LHS	RHS
—	Pronoun S
—	Pronoun VP
—	Pronoun X2
—	NP S
S	NP VP
—	NP X2

Binary split: [0,2] / [2,6]

LHS	RHS
—	S NP

Binary split: [0,4] / [4,6]

LHS	RHS
—	S PP

CYK — Walkthrough

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4] S	[0,5]	[0,6] S
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4] S, VP, X2	[1,5]	[1,6] S, VP, X2
		[2,3] Det	[2,4] NP	[2,5]	[2,6] NP
			[3,4] Nominal, Noun	[3,5]	[3,6] Nominal
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

Our grammar can generate this sentence since the start symbol S is in [0,6]

CYK Parsing Algorithm — Pseudo Code

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

```
for  $j \leftarrow$  from 1 to LENGTH(words) do  
    for all {  $A \mid A \rightarrow \text{words}[j] \in \text{grammar}$  }  
         $\text{table}[j-1, j] \leftarrow \text{table}[j-1, j] \cup A$ 
```

} Base case: For each terminal (i.e., word), find all terminals that can generate this terminal

```
for  $j \leftarrow$  from  $j-2$  down to 0 do  
    for  $k \leftarrow i+1$  to  $j-1$  do
```

} Loop over all possible binary splits of spans of size 2, and increase until sentence length

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

}
If there is a rule (or more) that can generate the current binary split, add the rule's LHS to the cell of the current span

CYK Parsing Algorithm — Basic Python Implementation

```
1 def cyk_parse(tokens, rules):
2     n = len(tokens)
3
4     # Initialize dynamic programming table
5     CYK = defaultdict(lambda: defaultdict(lambda: defaultdict(lambda: 0)))
6
7     # Initialize parse: span of length 1
8     for s in range(n):
9         # Find all non-terminals that can generate the terminal
10        for A, rhs in rules:
11            if rhs == (tokens[s],):
12                CYK[s][s+1][A] = 1
13
14        # Handle spans of length 2+ using dynamic programming
15        for length in range(2, n+1):
16            for start in range(0, n-length+1):          # Loop over all
17                end = start + length                    # the possible
18                for split in range(start+1, end):        # binary splits
19                    # Check each production rule (ignore lexicon rules)
20                    for A, (B, C) in [ r for r in rules if len(r[1]) == 2]:
21                        # is_valid = 1 if B and C can generate left and right part
22                        is_valid = CYK[start][split][B] * CYK[split][end][C]
23                        # The same LHS needs to be able to generate the RHS only once
24                        CYK[start][end][A] = np.max([ is_valid, CYK[start][end][A] ])
25
26    return CYK
```

CYK — Example: Invalid Parse

I	book	flight	the	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4]	[0,5]	[0,6]
	[1,2] S, VP, Nominal, Noun, Verb	[1,3] Nominal	[1,4]	[1,5]	[1,6]
		[2,3] Nominal, Noun	[2,4]	[2,5]	[2,6]
			[3,4] Det	[3,5]	[3,6]
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

CYK — Syntax vs. Semantic

- Syntactic parsing does not consider semantics
 - Any constituent can be replaced with another constituent of the same type
 - Example below: A noun can be replaced with any other noun

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4] S	[0,5]	[0,6] S
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4] S, VP, X2	[1,5]	[1,6] S, VP, X2
		[2,3] Det	[2,4] NP	[2,5]	[2,6] NP
			[3,4] Nominal, Noun	[3,5]	[3,6] Nominal
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

vs.

I	book	the	meal	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4] S	[0,5]	[0,6] S
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4] VP, S, X2	[1,5]	[1,6] VP, S, X2
		[2,3] Det	[2,4] NP	[2,5]	[2,6] NP
			[3,4] Nominal, Noun	[3,5]	[3,6] Nominal
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

CYK Parsing Algorithm — Limitation

- Basic CYK algorithm only solves the membership problem
 - Algorithm only checks if a sentence is a “member” of the language described by the grammar
 - What we also want
 - Finding all actual parse trees
(in case a sentence is valid; otherwise the result is empty)
 - Identifying the best parse tree(s)
(which requires a definition for what we mean by “best”)
- Good news: Only rather minor extension to base algorithm required

Outline

- Syntactic Parsing
 - Quick recap: RegEx
 - Context-free grammars
 - Structural Ambiguity
 - Chomsky Normal Form
- CYK Parsing Algorithm
 - Base membership algorithm
 - **Find all parse trees with backtracking**
 - Probabilistic parsing
 - Evaluation of parser

CYK — Get all Parse Trees (Derivations)

- Basic Idea: Keep track of backtrace
 - Remember which 2 cells matched an existing production rule

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4]	[0,5]	[0,6] S
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4] S, VP, X2	[1,5]	[1,6] S, VP X2
		[2,3] Det	[2,4] NP	[2,5]	[2,6] NP
			[3,4] Nominal, Noun	[3,5]	[3,6] Nominal
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

Binary split: [0,1] / [1,6]

LHS	RHS
—	Pronoun S
—	Pronoun VP
—	Pronoun X2
—	NP S
<u>S</u>	NP VP
—	NP X2

CYK — Get all Parse Trees (Derivations)

- Recall: Structural Ambiguity
 - In general, different production rules might match

I	book	the	flight	through	Singapore
[0,1] Pronoun, NP	[0,2] S	[0,3]	[0,4] S	[0,5]	[0,6] S
	[1,2] S, VP, Nominal, Noun, Verb	[1,3]	[1,4] S, VP, X2	[1,5]	[1,6] S, VP, X2
		[2,3] Det	[2,4] NP	[2,5]	[2,6] NP
			[3,4] Nominal, Noun	[3,5]	[3,6] Nominal
				[4,5] Prep	[4,6] PP
					[5,6] PropNoun, NP

Binary split: [1,2] / [2,6]

LHS	RHS
—	S NP
—	VP NP
—	Nominal NP
—	Noun NP
<u>S</u> , VP, X2	Verb NP

Binary split: [1,4] / [4,6]

LHS	RHS
—	S PP
<u>S</u> , VP	VP PP
<u>S</u> , VP	X2 PP

CYK — Get all Parse Trees (Pseudo Code)

function CKY-Parse(*words*, *grammar*) **returns** *table*, *pointer*

```
for  $j \leftarrow$  from 1 to LENGTH(words) do
    for all {  $A \mid A \rightarrow \text{words}[j] \in \text{grammar}$  }
         $\text{table}[j-1, j] \leftarrow \text{table}[j-1, j] \cup A$ 
         $\text{pointer}[j-1, j, A] \leftarrow \text{pointer}[j-1, j, A] \cup \text{words}[j]$ 
    }
for  $j \leftarrow$  from  $j-2$  down to 0 do
    for  $k \leftarrow i+1$  to  $j-1$  do
        for all {  $A \mid A \rightarrow BC \in \text{grammar}$  and  $B \in \text{table}[i, k]$  and  $C \in \text{table}[k, j]$  }
             $\text{table}[i, j] \leftarrow \text{table}[i, j] \cup A$ 
             $\text{pointer}[i, j, A] \leftarrow \text{pointer}[i, j, A] \cup ((i, k, B), (k, j, C))$ 
```

CYK — Get all Parse Trees (Python)

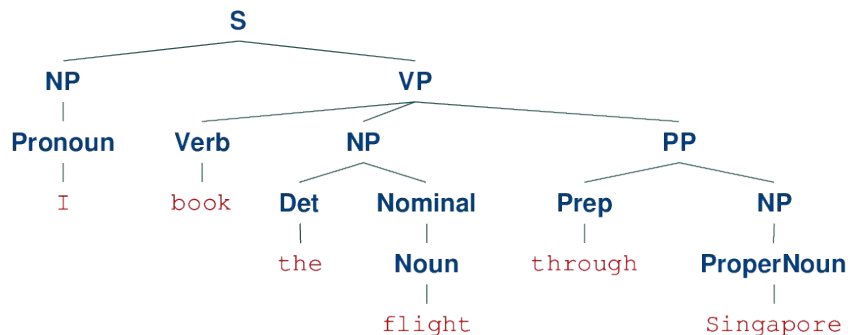
```
1 def cyk_parse_basic_ptr(tokens, rules):
2     n = len(tokens)
3
4     # Initialize dynamic programming table + backtrace pointers
5     CYK = defaultdict(lambda: defaultdict(lambda: defaultdict(lambda: 0)))
6     PTR = defaultdict(lambda: defaultdict(lambda: defaultdict(list)))
7
8     # Initialize parse: span of length 1
9     for s in range(n):
10         # Find all non-terminals that can generate the terminal
11         for A, rhs in rules:
12             if rhs == (tokens[s],):
13                 CYK[s][s+1][A] = 1
14                 PTR[s][s+1][A].append(tokens[s])
15
16     # Handle spans of length 2+ using dynamic programming
17     for length in range(2, n+1):
18         for start in range(0, n-length+1):      # Loop over all
19             end = start + length                # the possible
20             for split in range(start+1, end):    # binary splits
21                 # Check each production rule (ignore lexicon rules)
22                 for A, (B, C) in [ r for r in rules if len(r[1]) == 2]:
23                     # is_valid = 1 if B and C can generate left and right part
24                     is_valid = CYK[start][split][B] * CYK[split][end][C]
25                     # The same LHS needs to be able to generate the RHS only once
26                     CYK[start][end][A] = np.max([ is_valid, CYK[start][end][A] ])
27                     # If this is a valid split, remember from which cells we came
28                     if is_valid > 0:
29                         PTR[start][end][A].append(((start, split, B), (split, end, C)))
30
31     return CYK, PTR
```

The only additions to the base algorithm
(base algorithm = CYK for membership problem)

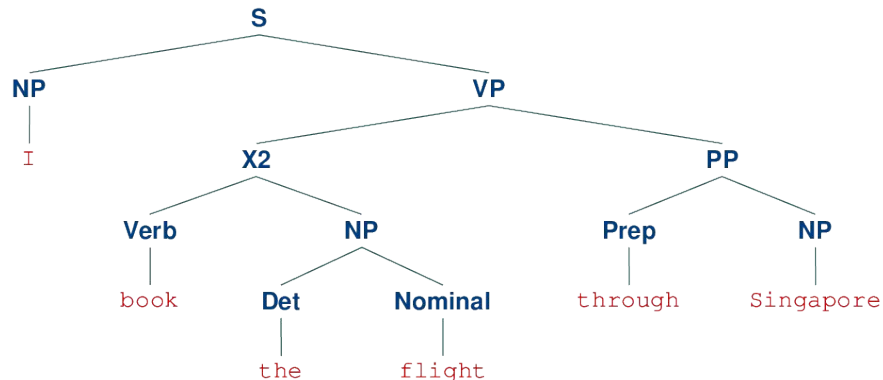
Parse Trees: CFG vs. CNF

- Converting a CFG into a CNF affects resulting parse trees
 - CFG parse trees can be recovered from CNF parse trees
(easy for newly split n-ary rules; a bit more tricky for unary rules)
 - Straightforward extension of CYK algorithm to support unary rules directly
(doesn't affect runtime complexity, but roughly doubles the required lines code)

Parse tree using original CFG



Parse tree using CNF (converted from CFG)



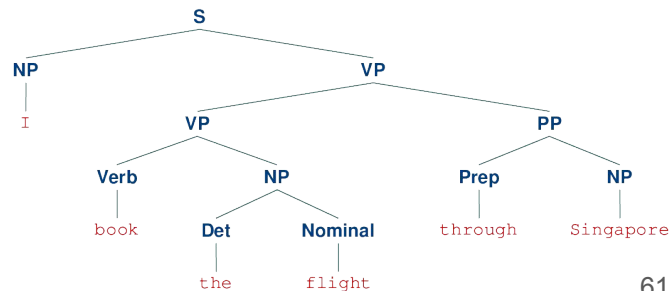
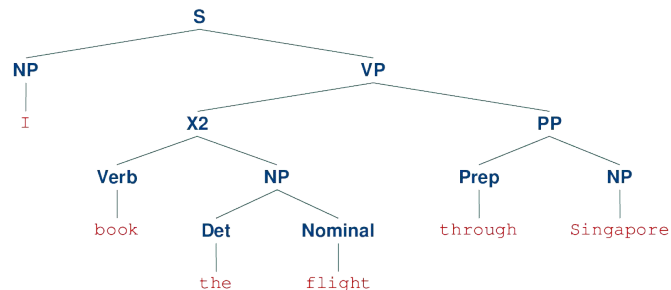
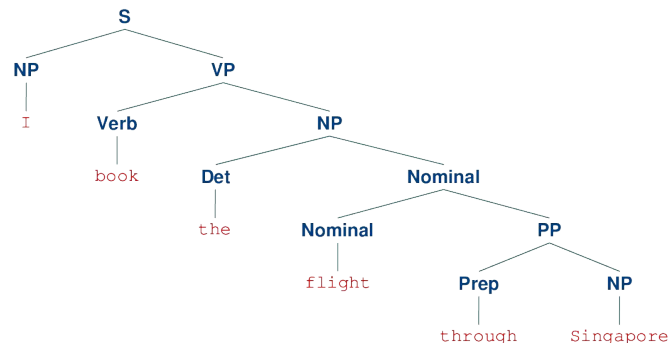
CYK — Parse Trees

- Parse tree for example

“I book the flight through Singapore”

- Observation

- Multiple valid parses
- Which is the best one?





Looking for ~~a few good~~ the best parses



Outline

- Syntactic Parsing
 - Quick recap: RegEx
 - Context-free grammars
 - Structural Ambiguity
 - Chomsky Normal Form
- CYK Parsing Algorithm
 - Base membership algorithm
 - Find all parse trees with backtracking
 - **Probabilistic parsing**
 - Evaluation of parser

Statistical Parsing

- Resolve structural ambiguity by choosing the most probable parse
 - Best parse = parse with the highest probability
 - Question: Where to get such probabilities from?

→ Probabilistic Context-Free Grammar (PCFG)

- Same as CFG, but each rule is associated with a probability
- Probabilities are learned from an annotated dataset

Given a parse tree T for a sentence S comprised of rules:

$$P(T, S) = \prod_i^n P(A \rightarrow \alpha) = \prod_i^n P(\alpha|A)$$

CFG — Formal Definition

- A CFG is a 4 tuple $\langle N, \Sigma, R, S \rangle$

- N — set of non-terminal symbols
- Σ — set of terminal symbols
- R — set of rules

Allowed format for all rules

$A \rightarrow \alpha$ [p] with $A \in N$, $\alpha \in (N \cup \Sigma)$

$$p = P(\alpha|A)$$

- S — start symbol

Example

$N = \{\text{NP, VP, Det, Noun, Verb}\}$

$\Sigma = \{a, \text{the, man, meal, flight, saw, booked}\}$

$S \rightarrow \text{NP VP}$ [0.4]

$\text{NP} \rightarrow \text{Det Noun}$ [0.5]

$\text{VP} \rightarrow \text{Verb NP}$ [0.2]

$\text{Det} \rightarrow a$ [0.3] | the [0.4]

...

Example CFG → Example PCFG

$$\begin{aligned}
 \sum = 1 & \left\{ \begin{array}{l} S \rightarrow NP VP [0.8] \\ S \rightarrow Aux NP VP [0.1] \\ S \rightarrow VP [0.1] \end{array} \right. \\
 \sum = 1 & \left\{ \begin{array}{l} NP \rightarrow Pronoun [0.2] \\ NP \rightarrow ProperNoun [0.2] \\ NP \rightarrow Det Nominal [0.6] \end{array} \right. \\
 \sum = 1 & \left\{ \begin{array}{l} Nominal \rightarrow Noun [0.3] \\ Nominal \rightarrow Nominal Noun [0.2] \\ Nominal \rightarrow Nominal PP [0.5] \end{array} \right. \\
 \sum = 1 & \left\{ \begin{array}{l} VP \rightarrow Verb [0.2] \\ VP \rightarrow Verb NP [0.4] \\ VP \rightarrow Verb NP PP [0.1] \\ VP \rightarrow Verb PP [0.1] \\ VP \rightarrow VP PP [0.2] \\ PP \rightarrow Prep NP [1.0] \end{array} \right.
 \end{aligned}$$

$$\begin{aligned}
 & Det \rightarrow the [0.4] \mid a [0.3] \mid that [0.2] \mid this [0.1] \\
 & Noun \rightarrow book [0.2] \mid flight [0.2] \mid meal [0.3] \mid money [0.3] \\
 & Verb \rightarrow book [0.4] \mid include [0.3] \mid prefer [0.3] \\
 & Pronoun \rightarrow I [0.4] \mid she [0.2] \mid he [0.2] \mid me [0.2] \\
 & ProperNoun \rightarrow Singapore [0.4] \mid Frankfurt [0.4] \mid SIA [0.2] \\
 & Aux \rightarrow do [0.5] \mid does [0.2] \mid did [0.3] \\
 & Prep \rightarrow from [0.2] \mid to [0.4] \mid on [0.2] \mid near [0.1] \mid through [0.1]
 \end{aligned}$$

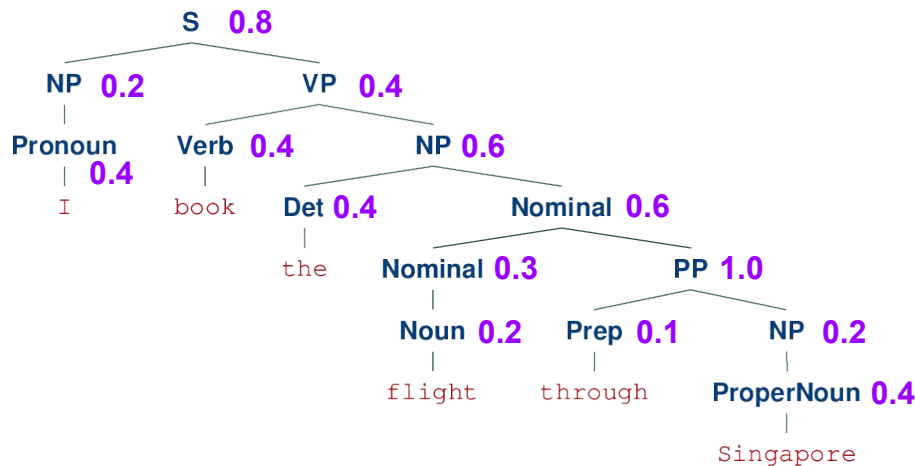
$\underbrace{\hspace{15em}}_{\sum = 1 \text{ for all right-hand sides}}$

Requirement for valid probabilities:

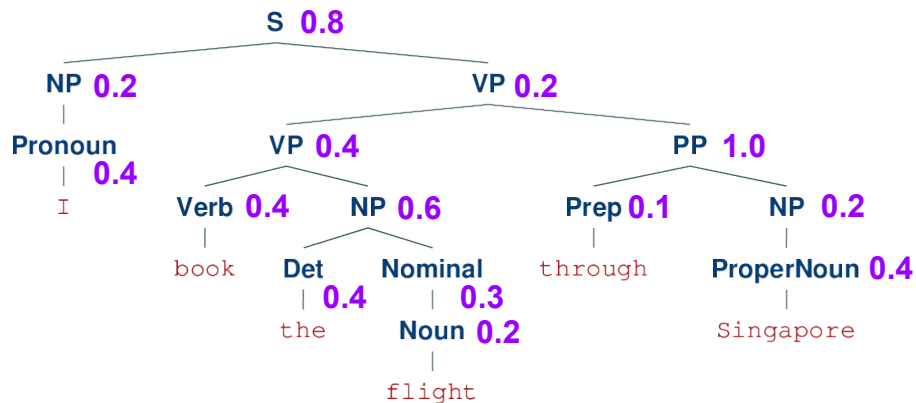
$$\sum_{\alpha} P(A \rightarrow \alpha) = \sum_{\alpha} P(\alpha|A) = 1$$

PCFG — Probability of a Parse Tree

- Probability of parse tree = product of probabilities of all rules
 - In practice, sum up log probabilities to avoid arithmetic underflow



$$P(T, S) = \prod_i^n P(A \rightarrow \alpha) = 0.00000071$$



$$P(T, S) = \prod_i^n P(A \rightarrow \alpha) = 0.00000024$$

PCFG — Calculating the Probability of a Rule

- Calculating $P(A \rightarrow \alpha)$ using Maximum Likelihood Estimation
 - Requires annotated dataset of parse trees


$$P(A \rightarrow \alpha) = P(\alpha|A) = \frac{\text{Count}(A \rightarrow \alpha)}{\text{Count}(A)}$$

Number of occurrences of
rule $P(A \rightarrow \alpha)$ in the dataset

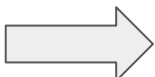
Number of occurrences of rules
in the dataset with A as the LHS

PCFG — Converting to CNF

(1) Dividing n-ary rules by introducing new non-terminals

$S \rightarrow NP VP [0.8]$		$S \rightarrow NP VP [0.8]$
$S \rightarrow Aux NP VP [0.1]$		$S \rightarrow X1 VP [0.1]$
...		$X1 \rightarrow Aux NP [1.0]$
		...

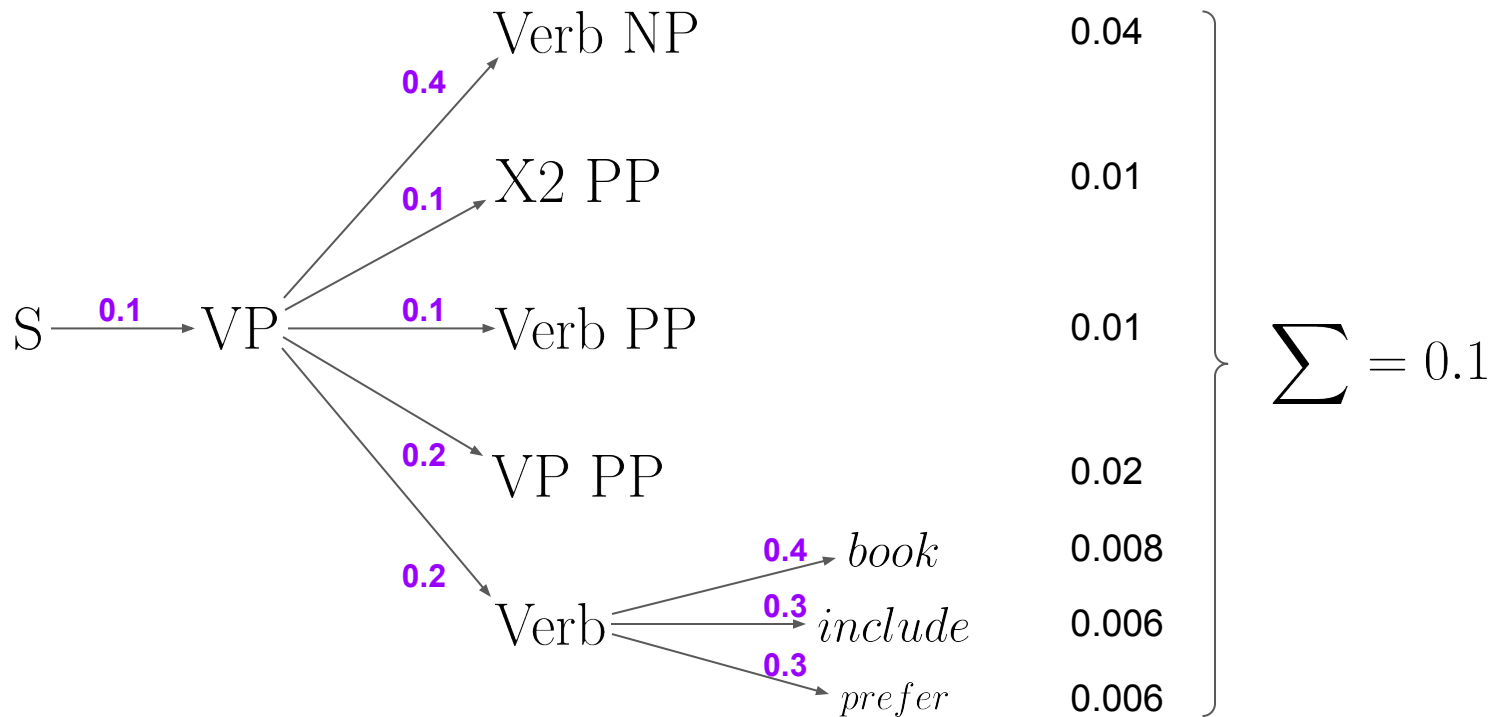
(2) Recursive removal of unary rules

$S \rightarrow NP VP [0.8]$		$S \rightarrow NP VP [0.8]$
$S \rightarrow Aux NP VP [0.1]$		$S \rightarrow X1 VP [0.1]$
$S \rightarrow VP [0.1]$		$X1 \rightarrow Aux NP [1.0]$
...		$S \rightarrow Verb NP [0.04]$
		$S \rightarrow X2 PP [0.01]$
		$S \rightarrow Verb PP [0.01]$
		$S \rightarrow VP PP [0.02]$
		$S \rightarrow book [0.008] \mid include [0.006] \mid prefer [0.006]$

How to compute these probabilities?

PCFG — Converting to CNF

- Multiply probabilities along the paths



PCFG — Converting to CNF

S → NP VP [0.8]
S → X1 VP [0.1]
X1 → Aux NP [1.0]
S → Verb NP [0.04]
S → X2 PP [0.01]
S → Verb PP [0.01]
S → VP PP [0.02]
NP → Det Nominal [0.6]
Nominal → Nominal Noun [0.2]
Nominal → Nominal PP [0.5]
VP → Verb NP [0.2]
VP → X2 PP [0.1]
X2 → Verb NP [1.0]
VP → Verb PP [0.1]
VP → VP PP [0.2]
PP → Prep NP [1.0]

Det → *the* [0.4] | *a* [0.3] | *that* [0.2] | *this* [0.1]
Noun → *book* [0.2] | *flight* [0.2] | *meal* [0.3] | *money* [0.3]
Verb → *book* [0.4] | *include* [0.3] | *prefer* [0.3]
Pronoun → *I* [0.4] | *she* [0.2] | *he* [0.2] | *me* [0.2]
PropNoun → *Singapore* [0.4] | *Frankfurt* [0.4] | *SIA* [0.2]
Aux → *do* [0.5] | *does* [0.2] | *did* [0.3]
Prep → *from* [0.2] | *to* [0.4] | *on* [0.2] | *near* [0.1] | *through* [0.1]
S → *book* [0.008] | *include* [0.006] | *prefer* [0.006]
VP → *book* [0.08] | *include* [0.06] | *prefer* [0.06]
NP → *I* [0.08] | *she* [0.04] | *he* [0.04] | *me* [0.04]
NP → *Singapore* [0.08] | *Frankfurt* [0.08] | *SIA* [0.04]
Nominal → *book* [0.06] | *flight* [0.06] | *meal* [0.09] | *money* [0.09]

pen & paper calculations...I hope the numbers add up :)

CYK — Get Best Parse Tree (Pseudo Code)

function CKY-Parse(*words*, *grammar*, *probs*) **returns** *table*, *pointer*

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

for all { $A \mid A \rightarrow \text{words}[j] \in \text{grammar}$ }

$\text{table}[j-1, j, A] \leftarrow \text{probs}[A \rightarrow \text{words}[j]]$

$\text{pointer}[j-1, j, A] \leftarrow \text{pointer}[j-1, j, A] \cup \text{words}[j]$

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

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

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

$p \leftarrow \text{table}[i, k, B] * \text{table}[k, j, C] * \text{probs}[A \rightarrow BC]$

if $p > \text{table}[i, j, A]$ **do**

$\text{table}[i, j, A] \leftarrow p$

$\text{pointer}[i, j, A] \leftarrow \text{pointer}[i, j, A] \cup ((i, k, B), (k, j, C))$

CYK — Get Best Parse Tree (Python)

```
1 def cyk_parse_probabilistic_ptr(tokens, rules, probs):
2     n = len(tokens)
3
4     # Initialize dynamic programming table
5     CYK = defaultdict(lambda: defaultdict(lambda: defaultdict(lambda: 0)))
6     PTR = defaultdict(lambda: defaultdict(lambda: defaultdict(list)))
7
8     # Initialize parse: span of length 1
9     for s in range(n):
10         # Find all non-terminals that can generate the terminal
11         for A, rhs in rules:
12             if rhs == (tokens[s],):
13                 CYK[s][s+1][A] = probs[A][token[s]]
14                 PTR[s][s+1][A].append(tokens[s])
15
16     # Handle spans of length 2+ using dynamic programming
17     for l in range(2, n+1):
18         for start in range(0, n-l+1):
19             end = start + l
20             for split in range(start+1, end):
21                 # Check each production rule (ignore lexicon rules)
22                 for A, (B, C) in [ r for r in rules if len(r[1]) == 2]:
23                     # Calculate probability of reaching the cell with the current rule
24                     p = CYK[start][split][B] * CYK[split][end][C] * probs[A][(B,C)]
25                     # If the probability is larger then the current one => update!
26                     if p > CYK[start][end][A]:
27                         CYK[start][end][A] = p
28                         PTR[start][end][A].append(((start, split, B), (split, end, C)))
29
30     return CYK, PTR
```

The only changes to the algorithm

Outline

- Syntactic Parsing
 - Quick recap: RegEx
 - Context-free grammars
 - Structural Ambiguity
 - Chomsky Normal Form
- CYK Parsing Algorithm
 - Base membership algorithm
 - Find all parse trees with backtracking
 - Probabilistic parsing
 - **Evaluation of parser**

Evaluation of Parse Trees

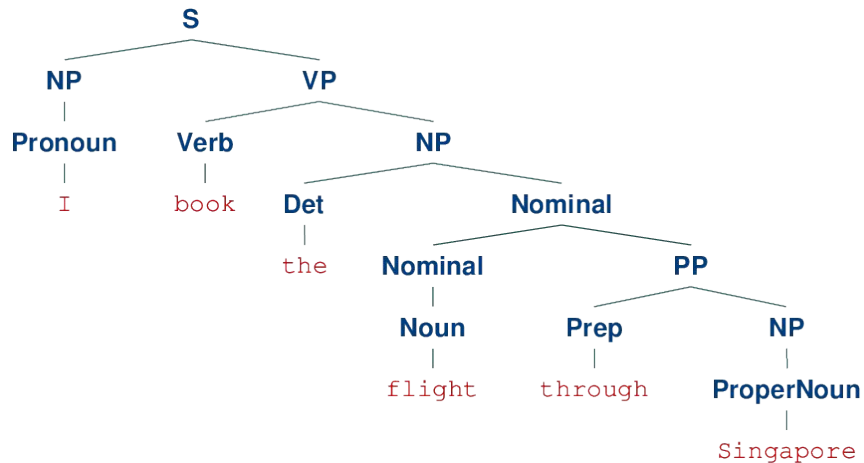
Note: This is not the only way to do it.

- Important: best parse \nRightarrow correct parse
 - Best parse = parse with the highest probability
 - Correct parse = parse that matches the gold-standard solution
- How evaluate parse trees?
 - Represent each parse tree as a set of tuples $\{\langle l_1, i_1, j_1 \rangle, \langle l_2, i_2, j_2 \rangle, \dots, \langle l_n, i_n, j_n \rangle\}$
 - l_k is the non-terminal labeling the k^{th} phrase
 - i_k is the index of the first word in the k^{th} word in the phrase
 - j_k is the index of the last word in the k^{th} word in the phrase

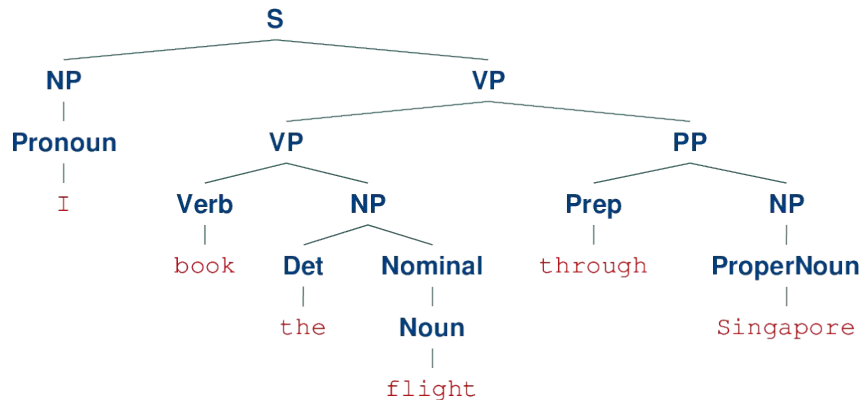
→ Use representations of computed parse and gold standard parse to estimate precision, recall and F1

Evaluation of Parse Trees — Example

Gold-standard (correct) parse tree



Computed “best” parse tree



Tuples only present in correct tree

$\langle \text{NP}, 3, 6 \rangle$
 $\langle \text{Nominal}, 4, 6 \rangle$

Tuples present in both trees

$\langle \text{NP}, 1, 1 \rangle$ $\langle \text{Pronoun}, 1, 1 \rangle$ $\langle \text{VP}, 2, 2 \rangle$ $\langle \text{Verb}, 2, 2 \rangle$
 $\langle \text{Det}, 3, 3 \rangle$ $\langle \text{Nominal}, 4, 4 \rangle$ $\langle \text{Noun}, 4, 4 \rangle$ $\langle \text{Prep}, 5, 5 \rangle$
 $\langle \text{ProperNoun}, 6, 6 \rangle$ $\langle \text{PP}, 5, 6 \rangle$ $\langle \text{NP}, 6, 6 \rangle$

Tuples only present in computed tree

$\langle \text{VP}, 2, 4 \rangle$
 $\langle \text{NP}, 3, 4 \rangle$

Evaluation of Parse Trees — Example

Tuples only present in correct tree

$\langle \text{NP}, 3, 6 \rangle$
 $\langle \text{Nominal}, 4, 6 \rangle$

Tuples present in both trees

$\langle \text{NP}, 1, 1 \rangle$ $\langle \text{Pronoun}, 1, 1 \rangle$ $\langle \text{VP}, 2, 2 \rangle$ $\langle \text{Verb}, 2, 2 \rangle$
 $\langle \text{Det}, 3, 3 \rangle$ $\langle \text{Nominal}, 4, 4 \rangle$ $\langle \text{Noun}, 4, 4 \rangle$ $\langle \text{Prep}, 5, 5 \rangle$
 $\langle \text{ProperNoun}, 6, 6 \rangle$ $\langle \text{PP}, 5, 6 \rangle$ $\langle \text{NP}, 6, 6 \rangle$

Tuples only present in computed tree

$\langle \text{VP}, 2, 4 \rangle$
 $\langle \text{NP}, 3, 4 \rangle$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{11}{11 + 2} = 0.85$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{11}{11 + 2} = 0.85$$

$$f1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 0.85$$

TP = # tuples in both trees

FP = # tuples only in the computed tree

FN = # tuples only in the correct tree

Summary

- Recursive nature of natural language
 - Natural language allows for nested structure
 - Basic building block: **constituents**
 - Most common way to capture constituency → **context-free grammars (CFGs)**
 - Syntactic parsing
 - Membership: check if a sentence can be generated by a grammar
 - Identification of all possible parse trees for a sentence
 - Identification of best parse tree for a sentence → **Probabilistic CFGs**
- } **CYK Parsing Algorithm**

U N I T E O R F A L L

Outlook for Next Week: Transformers



Photo credit: [Hasbro and Paramount Pictures](#)

Pre-Lecture Activity for Next Week

Read [8 Google Employees Invented Modern AI. Here's the Inside Story](#)
(Wired Article)

Apply your own (self-)attention to the article.
Quote a sentence of the article you think most
or least strikes your attention. Tell us why.

Side notes:

- This task is meant as a warm-up to provide some context for the next lecture
- No worries if you get lost; we will talk about this in the next lecture
- You can just copy-&-paste others' answers but this won't help you learn better

