Imperial College London

29 July 2020

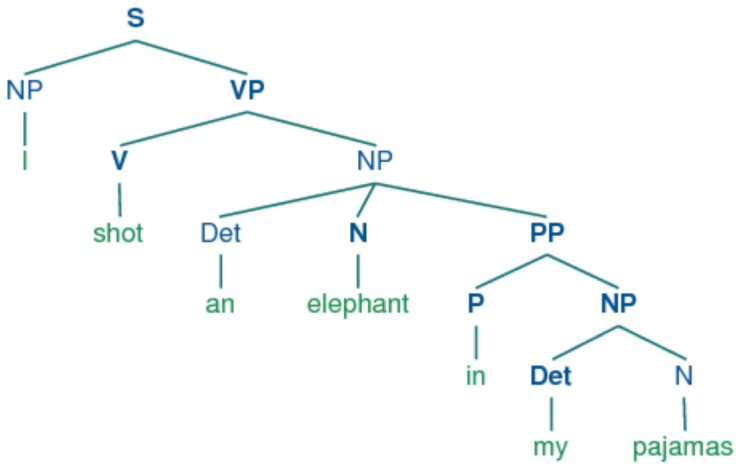
CYK Algorithm for Parsing

https://github.com/CRLala/CYK_Algorithm_for_Parsing

Chiraag Lala

Imperial College London

Constituency Parsing



Given a sequence of tokens (a sentence), generate its syntactic structure (a parse tree) which conforms to a Grammar.

Imperial College London

Applications of Parsing

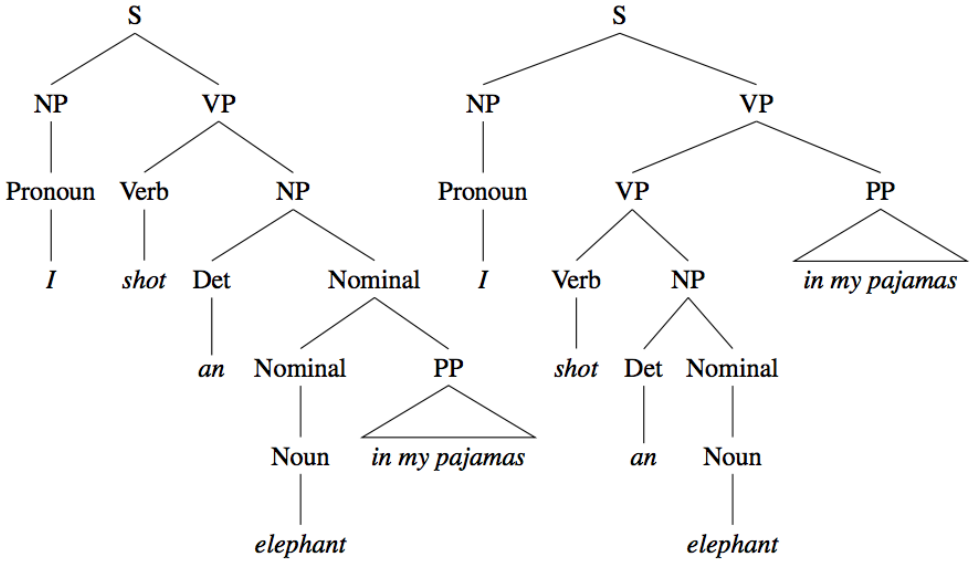
- Grammar checking
- As representation / features for downstream tasks:
  - Question Answering
  - Information Extraction
  - Named Entity Recognition
  - Machine Translation
  - …

Imperial College London

Challenges

1. It is hard. The simple brute-force method has exponential time complexity.

2. Structural Ambiguity



Imperial College London

Addressing the Challenges

1. Cocke-Younger-Kasami (CYK) Algorithm

2. Probabilistic Context-Free Grammar (PCFG)

Imperial College London

Context-Free Grammar (CFG) in Chomsky Normal Form (CNF)

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

Rules are of the form

$\mathbf{X} \rightarrow \mathbf{Y}\ \mathbf{Z}$ (Grammar)

or

$\mathbf{X} \rightarrow \mathbf{w}$ (Lexicon)

Imperial College London

Cocke-Younger-Kasami Algorithm

Dynamic programming algorithm with polynomial time complexity.

Input: sequence of words (sentence) and a CFG in CNF

Returns: n x n matrix of all possible parse trees

n is length of the sentence

Kasami, T. (1965). An efficient recognition and syntax analysis algorithm for context-free languages. Tech. rep. AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

Younger, D. H. (1967). Recognition and parsing of contextfree languages in time n 3 . Information and Control, 10, 189–208.

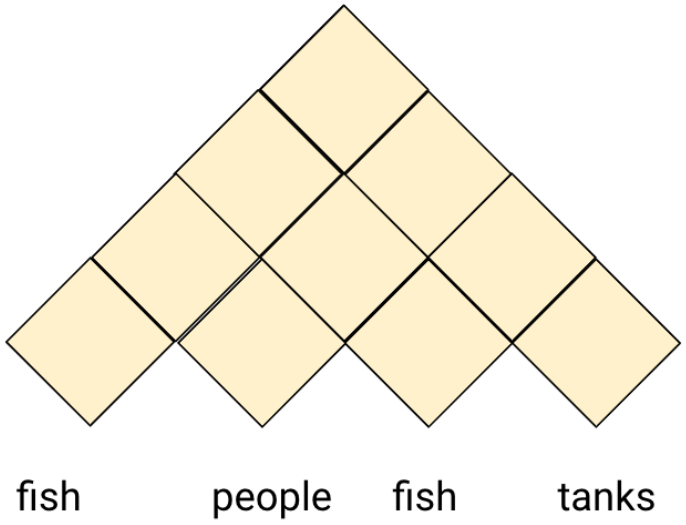
Imperial College London

CYK Algorithm: Pseudocode

```
function CKY(sentence, grammar) returns table
    for j from 1 to length(sentence) do
        for all {A | A → sentence[j] ∈ grammar}
            table[j, j] ← table[j, j] ∪ A
    for step from 1 to length(sentence)-1 do
        for row from 1 to length(sentence)-step do
            col ← row + step
            for k from 0 to step-1 do
                for all {A | A → BC ∈ grammar
                                and B ∈ table[row, row+k]
                                and C ∈ table[row+k+1, col]}
                    table[row][col] ← table[row, col] ∪ A
```

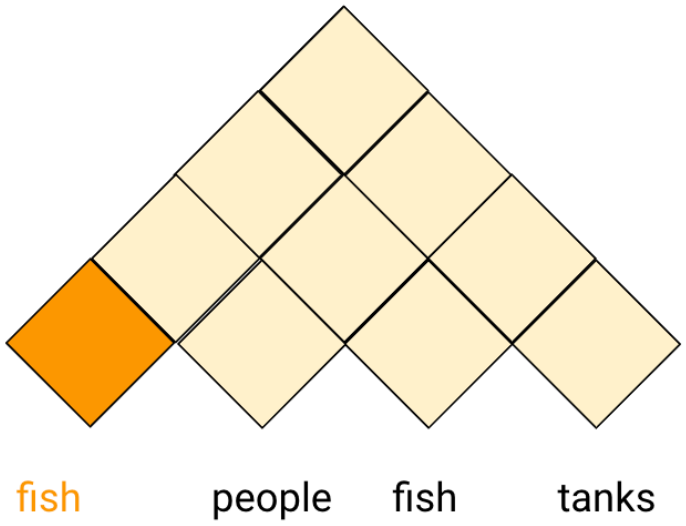
Imperial College London

CYK Algorithm: Intuition



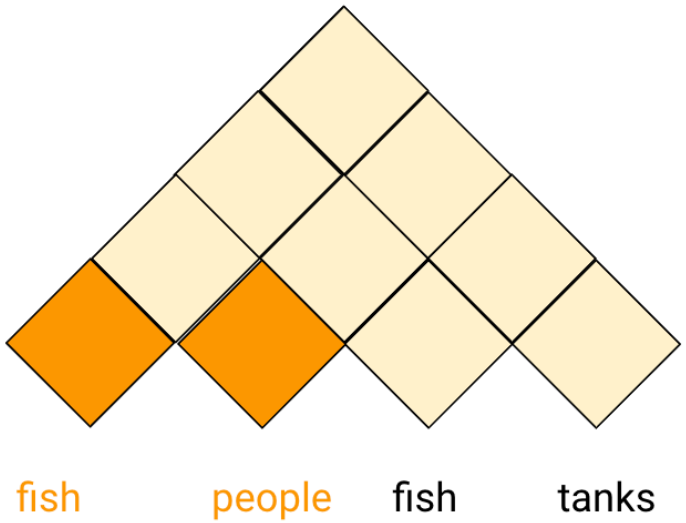
Imperial College London

CYK Algorithm: Intuition



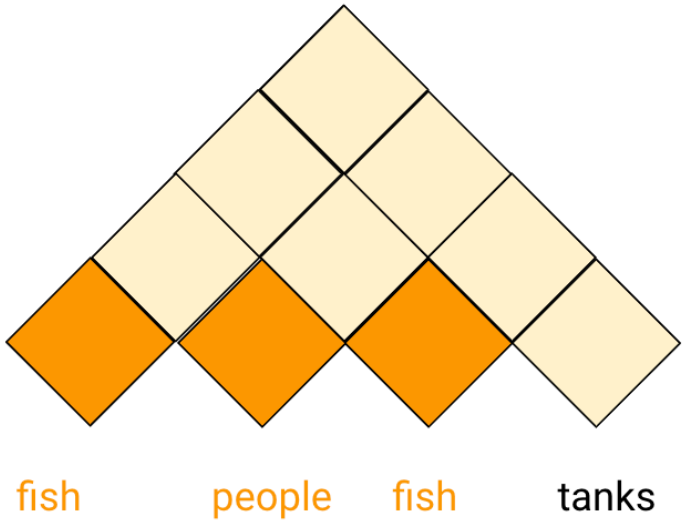
Imperial College London

CYK Algorithm: Intuition



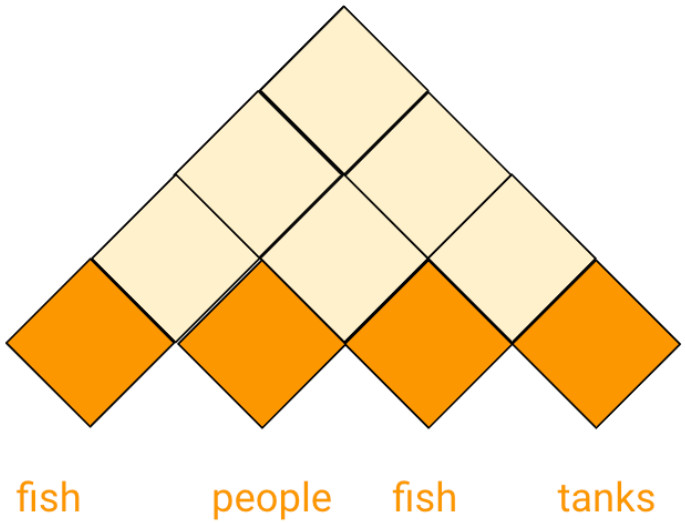
Imperial College London

CYK Algorithm: Intuition



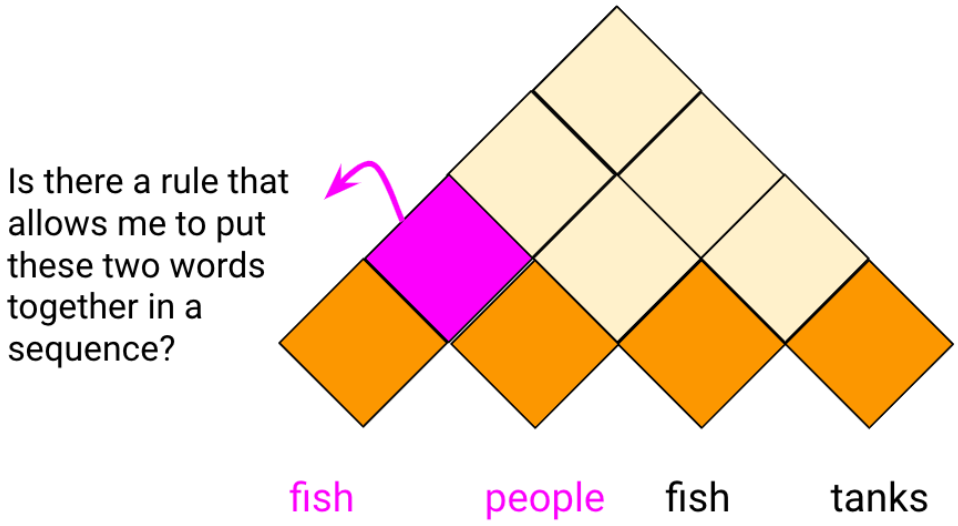
Imperial College London

CYK Algorithm: Intuition



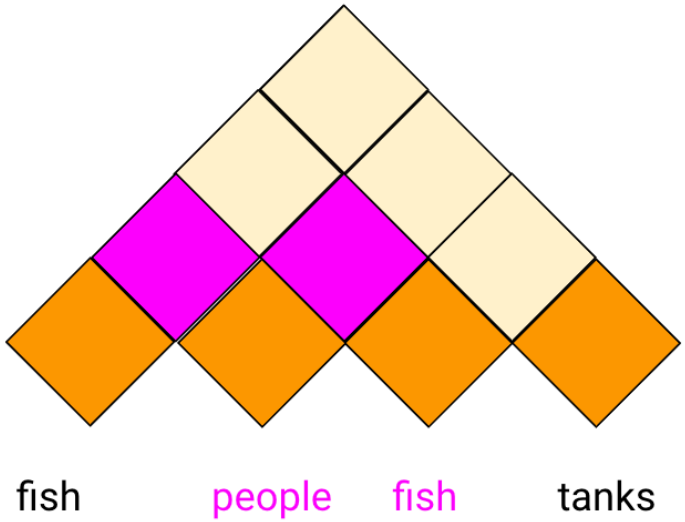
Imperial College London

CYK Algorithm: Intuition



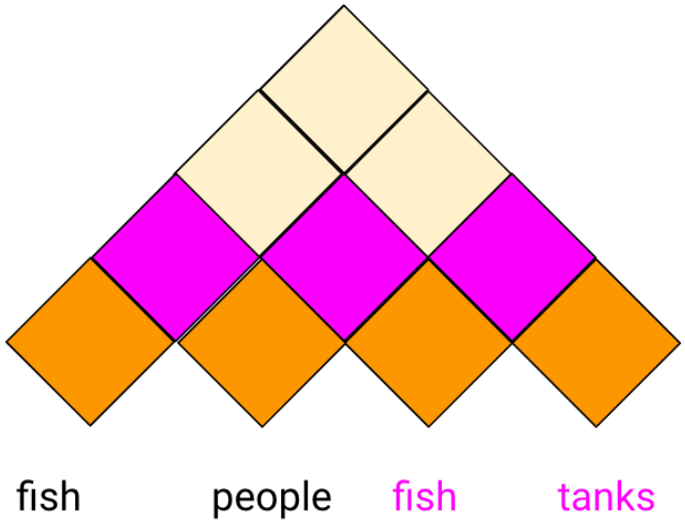
Imperial College London

CYK Algorithm: Intuition



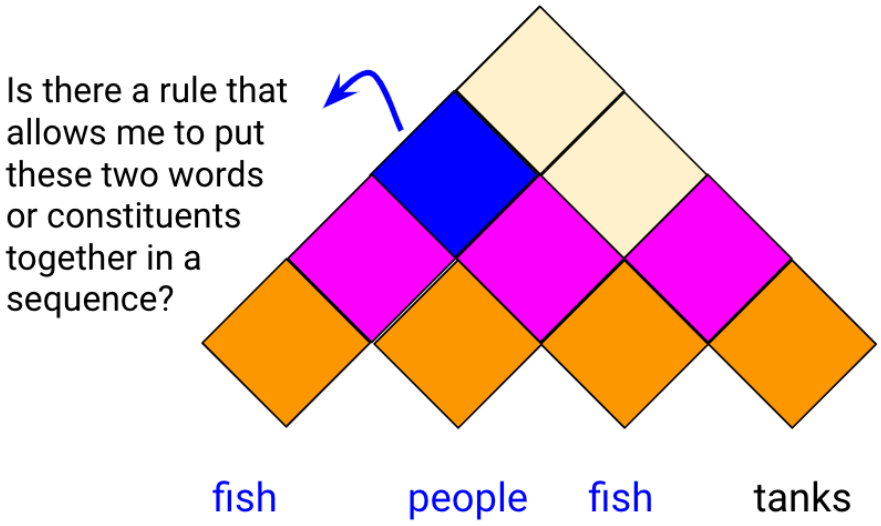
Imperial College London

CYK Algorithm: Intuition



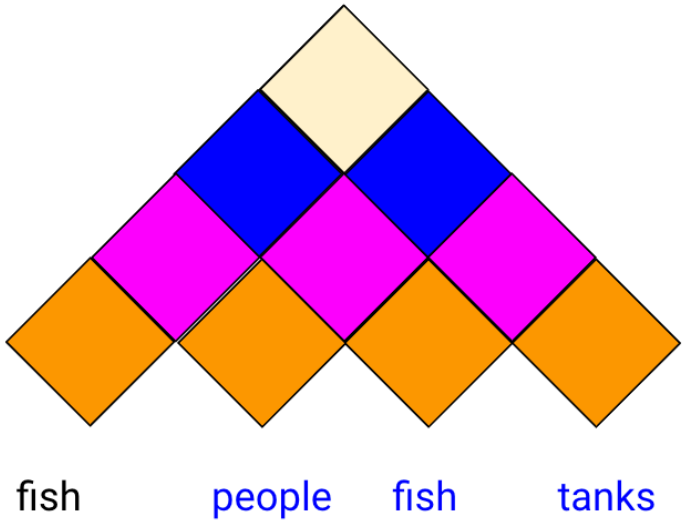
Imperial College London

CYK Algorithm: Intuition



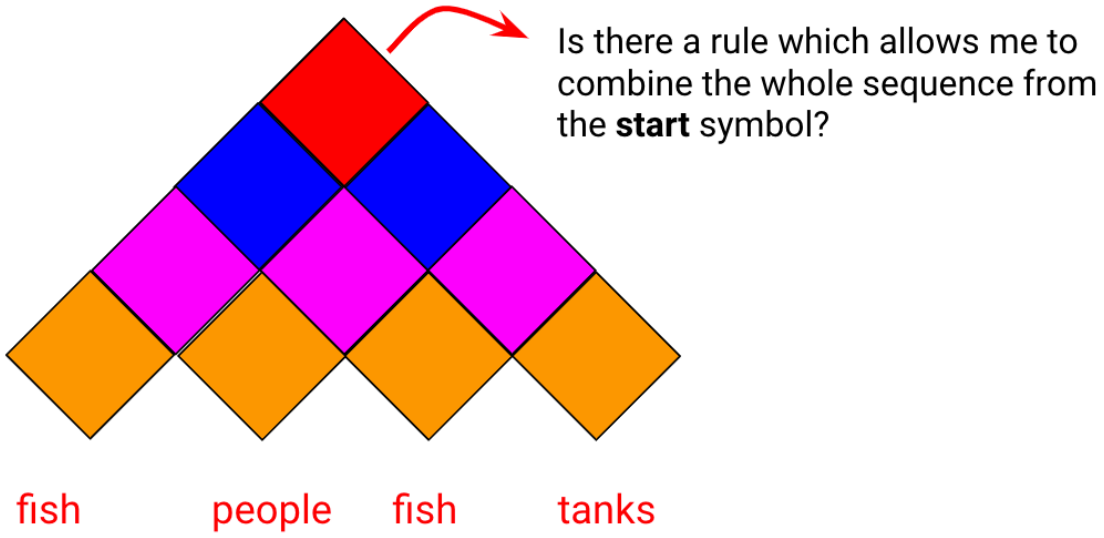
Imperial College London

CYK Algorithm: Intuition



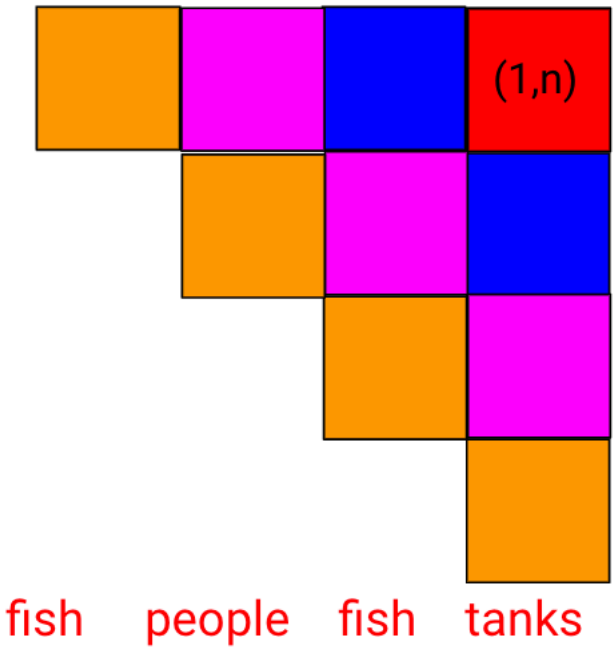
Imperial College London

CYK Algorithm: Intuition



Imperial College London

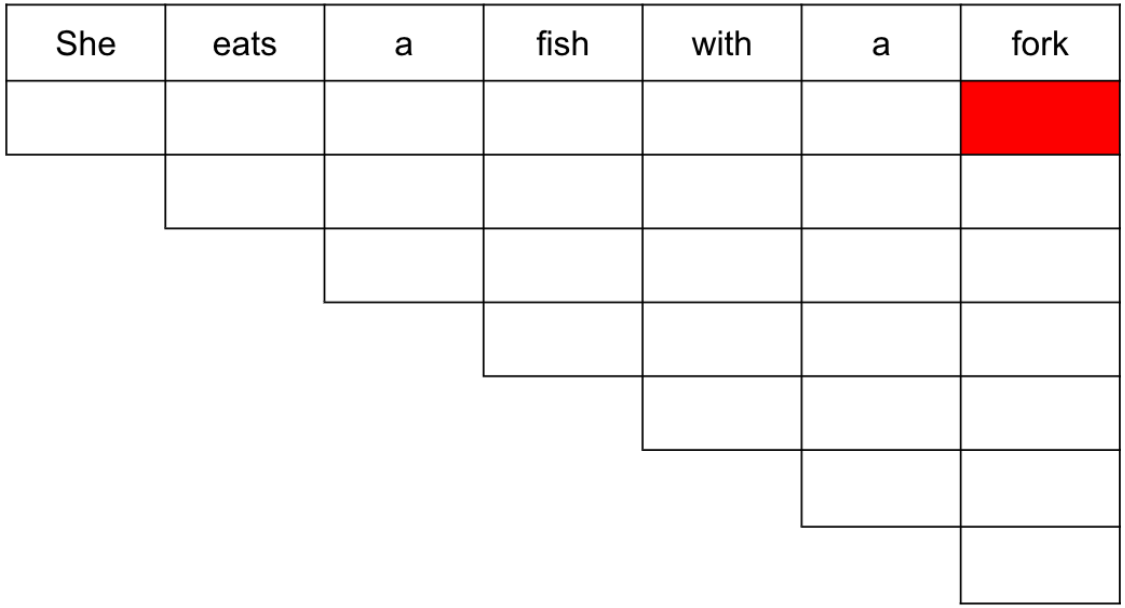
CYK Algorithm: Intuition



Imperial College London

CYK Algorithm: Example

S	→	NP VP
VP	→	VP PP
VP	→	V NP
VP	→	eats
PP	→	P NP
NP	→	Det N
NP	→	she
V	→	eats
P	→	with
N	→	fish
N	→	fork
Det	→	a



Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP						
	V, VP					
		Det				
			N			
				P		
					Det	
						N

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP						
	V, VP					
		Det				
			N			
				P		
					Det	
						N

Are there rules of the type
? → NP V
or
? → NP VP

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S					
	V, VP					
		Det				
			N			
				P		
					Det	
						N

Are there rules of the type
? → NP V
or
? → NP VP

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S					
	V, VP					
		Det				
			N			
				P		
					Det	
						N

Are there rules of the type
? → V Det
or
? → VP Det

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S					
	V, VP	-------				
		Det				
			N			
				P		
					Det	
						N

Are there rules of the type
? → V Det
or
? → VP Det

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S					
	V, VP	-------				
		Det				
			N			
				P		
					Det	
						N

Are there rules of the type
? → Det N

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S					
	V, VP	-------				
		Det	NP			
			N			
				P		
					Det	
						N

Are there rules of the type
? → Det N

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S					
	V, VP	-------				
		Det	NP			
			N	-------		
				P	-------	
					Det	NP
						N

Keep going for each 2 words

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S					
	V, VP	-------				
		Det	NP			
			N	-------		
				P	-------	
					Det	NP
						N

Sequences of 3 words, but our grammar is binary. So check for rules for combining words in 2 substrings
? → <she> <eats a>
? → <she eats> <a>

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

? → NP -------

She	eats	a	fish	with	a	fork
NP	S					
	V, VP	-------				
		Det	NP			
			N	-------		
				P	-------	
					Det	NP
						N

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

? → S Det

She	eats	a	fish	with	a	fork
NP	S					
	V, VP	-------				
		Det	NP			
			N	-------		
				P	-------	
					Det	NP
						N

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

? → S Det

She	eats	a	fish	with	a	fork
NP	S	-------				
	V, VP	-------				
		Det	NP			
			N	-------		
				P	-------	
					Det	NP
						N

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	------				
	V, VP	-------				
		Det	NP			
			N	-------		
				P	-------	
					Det	NP
						N

? → <eats> <a fish>
? → <eats a> <fish>

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	------				
	V, VP	-------				
		Det	NP			
			N	-------		
				P	-------	
					Det	NP
						N

? → V NP
? → VP NP

(both V and VP → eats)

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	-------				
	V, VP	-------	VP			
		Det	NP			
			N	-------		
				P	-------	
					Det	NP
						N

? → V NP
? → VP NP

(both V and VP → eats)

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

? → ------- N

She	eats	a	fish	with	a	fork
NP	S	-------				
	V, VP	-------	VP			
		Det	NP			
			N	-------		
				P	-------	
					Det	NP
						N

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	-------				
	V, VP	-------	VP			
		Det	NP	-------		
			N	-------	-------	
				P	-------	PP
					Det	NP
						N

Keep going for each 3 words.
Last PP:
? → <with> <a fork>
? → <with a> <fork>
i.e.:
? → P NP (yes, that's PP)
? → ------- N (no)

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	-------				
	V, VP	-------	VP			
		Det	NP	-------		
			N	-------	-------	
				P	-------	PP
					Det	NP
						N

Sequences of 4 words, but our grammar is binary. So check for rules for combining words in 2 substrings
? → <she> <eats a fish>
? → <she eats> < a fish>
? → <she eats a> <fish>

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	-------				
	V, VP	-------	VP			
		Det	NP	-------		
			N	-------	-------	
				P	-------	PP
					Det	NP
						N

? → <she> <eats a fish>

? → NP VP

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	-------	S			
	V, VP	-------	VP			
		Det	NP	-------		
			N	-------	-------	
				P	-------	PP
					Det	NP
						N

? → <she> <eats a fish>

? → NP VP

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	-------	S			
	V, VP	-------	VP	-------		
		Det	NP	-------	-------	
			N	-------	-------	-------
				P	-------	PP
					Det	NP
						N

Keep going for each 4 words

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	-------	S	-------		
	V, VP	-------	VP	-------	-------	
		Det	NP	-------	-------	-------
			N	-------	-------	-------
				P	-------	PP
					Det	NP
						N

Keep going for each 5 words

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	------	S	------	------	
	V, VP	------	VP	------	------	VP
		Det	NP	------	------	------
			N	------	------	------
				P	------	PP
					Det	NP
						N

Sequences of 6 words
? → <she> <eats a fish with a>
? → <she eats> < a fish with a>
? → <she eats a> <fish with a>
....
? → <eats a fish> <with a fork>
...
i.e. (last one above)
? → VP PP (yes, VP!)

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	-------	S	-------	-------	
	V, VP	-------	VP	-------	-------	VP
		Det	NP	-------	-------	-------
			N	-------	-------	-------
				P	-------	PP
					Det	NP
						N

Sequences of 7 words
? → <she> <eats a fish with a fork>
? → <she eats> < a fish with a fork>
? → <she eats a> <fish with a fork>
....
I.e. (for the first one above)
? → NP VP (yes, S!)

Imperial College London

CYK Algorithm: Example

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

She	eats	a	fish	with	a	fork
NP	S	------	S	------	-------	S
	V, VP	-------	VP	-------	-------	VP
		Det	NP	------	-------	-------
			N	-------	-------	-------
				P	-------	PP
					Det	NP
						N

Sequences of 7 words
? → <she> <eats a fish with a fork>
? → <she eats> < a fish with a fork>
? → <she eats a> <fish with a fork>
....
I.e. (for the first one above)
? → NP VP (yes, S!)

Imperial College London

CYK Algorithm: What is the time complexity?

Pause the video and calculate

Imperial College London

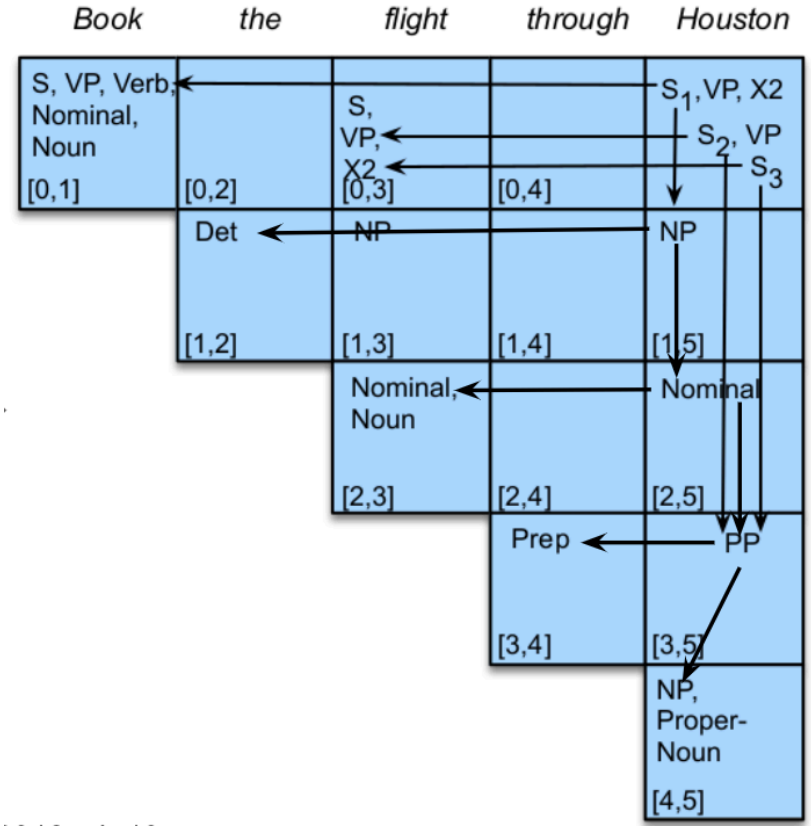
CYK Algorithm: How to retrieve parse trees?

Backpointer!

When filling up a cell with a particular non-terminal, also store the cells and the symbols used.

Imperial College London

CYK Algorithm: How to retrieve parse trees?



From Jurafsky, D and Martin, J, “Speech and Language Processing,” 2018, ch 13

Imperial College London

How to retrieve the most probable parse tree?

Probabilistic Context-Free Grammar (PCFG) in Chomsky Normal Form

Imperial College London

Probabilistic Context-Free Grammar

PCFG is a CFG where each rule (X → Y Z) comes with a probability score p.

$$p = p(X \rightarrow YZ \mid X) = \frac{count(X \rightarrow YZ)}{count(X \rightarrow **)}$$

Imperial College London

PCFG Example

S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	IN	NP	1.0

Vi	⇒	sleeps	1.0
Vt	⇒	saw	1.0
NN	⇒	man	0.7
NN	⇒	woman	0.2
NN	⇒	telescope	0.1
DT	⇒	the	1.0
IN	⇒	with	0.5
IN	⇒	in	0.5

Imperial College London

CYK Algorithm using PCFG

Like Viterbi Algorithm, you keep track of Viterbi score at each step.

For instance, let **X** → **Y Z** be a rule with probability $\boldsymbol{p}$.
Also, suppose a particular cell '**y**' contains **Y** with Viterbi score $\boldsymbol{Vy}$
And suppose a particular cell '**z**' contain **Z** with Viterbi score $\boldsymbol{Vz}$

If according to CYK algorithm we fill a cell '**x**' with **X** using the above rule, then we should also store a Viterbi score $\boldsymbol{Vx = Vy * Vz * p}$

If cell '**x**' already has **X** in it with a Viterbi score $\boldsymbol{Vx\_old}$, then we compare the newly computed $\boldsymbol{Vx}$ with it. If $\boldsymbol{Vx > Vx\_old}$ then replace backpointers and Viterbi score with the new ones.

Imperial College London

Exercises and Tutorial

https://github.com/CRLala/CYK_Algorithm_for_Parsing

Imperial College London

Questions and Answers

Piazza

Q&A sessions according to time-zones at specified times

Imperial College London

Summary

- Constituency parsing: generate a syntactic structure (parse tree) of a sequence of tokens.
- Constituency parsing is hard (in terms of time complexity) and may have structural ambiguities (multiple parse trees).
- CYK algorithm pseudocode, intuition, and example. Improves efficiency from exponential time complexity to polynomial time complexity.
- CYK algorithm using a PCFG in CNF is a Viterbi-style algorithm which can help retrieve the most probable parse tree.

Imperial College London

Next lecture

Dependency parsing

Imperial College London

Thank you!

Chiraag Lala

clala@imperial.ac.uk