Parsing Natural Language

CKY-Algorithm

Parsing

- We have seen, that in order to get a parse tree, we need any context free grammar (CFG)
- And a parsing algorithm which can find the tree-structures according to this grammar (not unique in the natural language setting)

- Since we are only interested in a single output, we also want to find the **best of these trees**

Parsing

- We can now phrase the problem of parsing:

Find the ordering of rules from our Grammar G, which produces the best syntax (or parse-) tree for a given sentence!

➔ We are looking for all valid trees (according to our grammar), score them, and decide for the tree with the highest score

$$\widehat{tree} = \underset{t \in valid\ Trees}{\operatorname{argmax}} \; score(t, X)$$

Parsing

$$\widehat{tree} = \underset{t \in valid\ Trees}{\arg\max}\ score(t, X)$$

- We are facing two major challenges:

  1. How can we (efficiently!) get all valid trees?
  2. How can we score a tree?

Parsing -Scores

$$\widehat{tree} = \underset{t \in valid\ Trees}{\operatorname{argmax}} \; score(t, X)$$

- We are yet again looking for a repeated structure, that we can assign scores to

➔In this case, we repeatedly apply rules from our grammar, therefore one very intuitive approach would be to score a tree as:

$$score(t, X) = p(tree, X) = \prod_{i=1}^{n} score(rule_i)$$

Parsing -Scores

$$p(tree) = \prod_{i=1}^{n} score(rule_i)$$

- What would be a very easy way to get a score for a rule?

  → Read a probability q from a labelled corpus!

  → $q(\alpha \to \beta) = \frac{Count(\alpha\to\beta)}{Count(\alpha\ )}$

- Example:

$$q(VP \to Vt) = \frac{105}{1000}$$

Probabilistic context-free grammars (PCFG)

- This leads us to Probabilistic Context Free Grammars (PCFG), defined as follows:

1. A context free grammar $G = (N, \Sigma, R, S\ )$
2. A parameter $q(\alpha \to \beta)$ for each rule $\alpha \to \beta \in R$.

The parameter $q(\alpha \to \beta)$ can be interpreted as a probability.
Therefore it holds:

$$\sum_{\alpha \to \beta \in R: \alpha = X, X \in N} q(\alpha \to \beta) = 1$$

$$q(\alpha \to \beta) \geq 0$$

Probabilistic Context Free Grammar - Example

$N = \{\text{S, NP, VP, PP, DT, Vi, Vt, NN, IN}\}$
$S = \text{S}$
$\Sigma = \{\text{sleeps, saw, man, woman, dog, telescope, the, with, in}\}$

$R, q =$

S	$\rightarrow$	NP	VP	1.0
VP	$\rightarrow$	Vi		0.3
VP	$\rightarrow$	Vt	NP	0.5
VP	$\rightarrow$	VP	PP	0.2
NP	$\rightarrow$	DT	NN	0.8
NP	$\rightarrow$	NP	PP	0.2
PP	$\rightarrow$	IN	NP	1.0

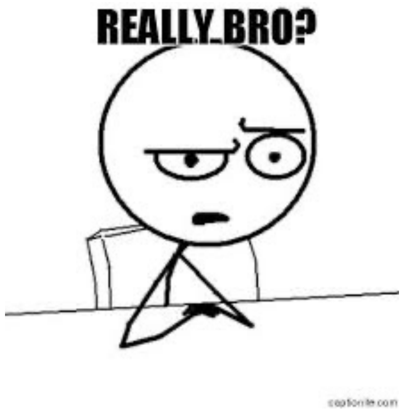
Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0
NN	$\rightarrow$	man	0.1
NN	$\rightarrow$	woman	0.1
NN	$\rightarrow$	telescope	0.3
NN	$\rightarrow$	dog	0.5
DT	$\rightarrow$	the	1.0
IN	$\rightarrow$	with	0.6
IN	$\rightarrow$	in	0.4

CYK Algorithmus

- We found a way to decide for the best tree, now we need an algorithm capable of producing all valid trees that we can score!
- The CYK (or CKY) algorithm can do this for us, and the scoring process can be integrated into the algorithm to be even more efficient!

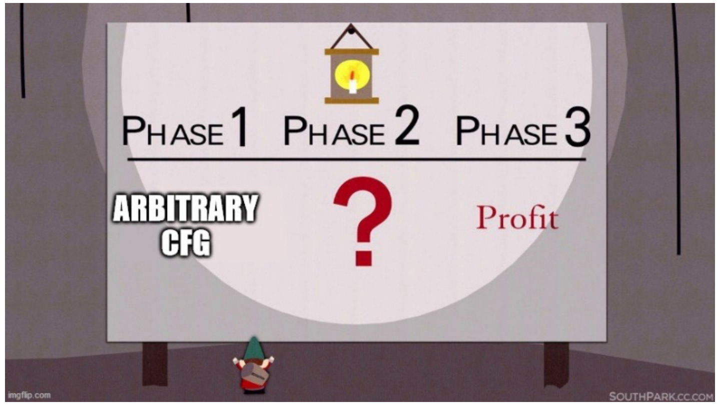
- Caveat: Our CFG has to take a special form!

→ That’s why we have to convert our CFG into „Chomsky normal form“



Parsing Natural Language

Converting a CFG into Chomsky Normal Form



Chomsky Normal Form (CNF)

- A CFG is said to be in Chomsky normal form, if every rule $\alpha \rightarrow \beta$ is in either of the two following forms:

$$X \rightarrow Y_1 Y_2 \text{ and } X, Y_1, Y_2 \in N$$
$$X \rightarrow Y \text{ and } X \in N, Y \in \Sigma$$

- Each rule is either:
  - A non-terminal producing two non-terminals
  - A non-terminal producing exactly one terminal

Chomsky Normal Form (CNF)

N = {S, NP, VP, PP, DT, Vi, Vt, NN, IN}
S = S
Σ = {sleeps, saw, man, woman, dog, telescope, the, with, in}

$R, q =$

S	$\rightarrow$	NP	VP	1.0
VP	$\rightarrow$	Vi		0.3
VP	$\rightarrow$	Vt	NP	0.5
VP	$\rightarrow$	VP	PP	0.2
NP	$\rightarrow$	DT	NN	0.8
NP	$\rightarrow$	NP	PP	0.2
PP	$\rightarrow$	IN	NP	1.0

Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0
NN	$\rightarrow$	man	0.1
NN	$\rightarrow$	woman	0.1
NN	$\rightarrow$	telescope	0.3
NN	$\rightarrow$	dog	0.5
DT	$\rightarrow$	the	1.0
IN	$\rightarrow$	with	0.6
IN	$\rightarrow$	in	0.4

➔ We have to transform a single rule!

Chomsky Normal Form (CNF)

- Before transforming a CFG into Chomsky Normal Form, these issues may occur in our grammar:

1. The start symbol S appears in the right hand side(RHS) (e.g. $S \rightarrow NP\ S$)
2. There are „Null“-rules (e.g. $NP \rightarrow \epsilon$) („we can delete a NP“)
3. Unit-production rules (e.g. $NP \rightarrow VP$)
4. Production rules having more than 2 elements on the RHS (e.g. $NP \rightarrow PP\ VP\ PP$)
5. „Mixed“ production rules (e.g. $PP \rightarrow Der\ NP$)

Chomsky Normal Form (CNF)

- Sketch of the algorithm for the conversion of a CFG into CNF:

1. If S appears on the RHS, introduce a new rule: $S_0 \rightarrow S$
2. Eliminate ϵ rules
3. Remove Unit-production rules as shown in the coming example
4. Clean up (optional)
5. Remove all rules with RHS > 2 as shown in the example

Chomsky Normal Form (CNF)

- Remove ϵ-rules by making them explicit

1. Find a non-terminal which produces ϵ “($A \rightarrow \epsilon$)”
2. Introduce a new symbol A' for A
3. Loop through your grammar, and whenever A occurs on the RHS of a rule
   ➔Replace that rule accordingly
4. Add rules starting with A' as LHS
5. Repeat until no more reachable ϵ productions in your grammar

Chomsky Normal Form (CNF)

- Example ϵ elimination

$$
\begin{aligned}
S &\rightarrow NP \text{ and } PP \\
NP &\rightarrow NP\ PP \\
NP &\rightarrow \epsilon \\
PP &\rightarrow PP\ PP \\
PP &\rightarrow \epsilon
\end{aligned}
$$

1. Find a non-terminal which produces ϵ “$(A \rightarrow \epsilon)$”

Chomsky Normal Form (CNF)

- Example ϵ elimination

2. Introduce a new symbol A' for A (we use PP')

$$
\begin{aligned}
S &\rightarrow NP \text{ and } PP \\
NP &\rightarrow NP\ PP \\
NP &\rightarrow \epsilon \\
PP &\rightarrow PP\ PP \\
PP &\rightarrow \epsilon
\end{aligned}
$$

Chomsky Normal Form (CNF)

- Example ϵ elimination

3. Loop through your grammar, and whenever A occurs on the RHS of a rule
➔Replace that rule accordingly

$$
\begin{aligned}
S &\rightarrow NP \text{ and } PP \\
NP &\rightarrow NP\ PP \\
NP &\rightarrow \epsilon \\
PP &\rightarrow PP\ PP \\
PP &\rightarrow \epsilon
\end{aligned}
$$

Add one rule of the form:
$LHS \rightarrow \alpha A'\beta$

Add one rule of the form:
$LHS \rightarrow \alpha\beta$

Chomsky Normal Form (CNF)

- Example ϵ elimination

3. Loop through your grammar, and whenever A occurs on the RHS of a rule
   ➔Replace that rule accordingly

Add one rule of the form:
$LHS \rightarrow \alpha A' \beta$

Add one rule of the form:
$LHS \rightarrow \alpha\beta$

$$
\begin{aligned}
S &\rightarrow NP \text{ and } PP \\
NP &\rightarrow NP\ PP \\
NP &\rightarrow \epsilon \\
PP &\rightarrow PP\ PP \\
PP &\rightarrow \epsilon \\
S &\rightarrow NP \text{ and } PP' \\
S &\rightarrow NP \text{ and} \\
NP &\rightarrow NP\ PP' \\
NP &\rightarrow NP
\end{aligned}
$$

Chomsky Normal Form (CNF)

- Example ϵ elimination

3. Loop through your grammar, and whenever A occurs on the RHS of a rule
   ➔Replace that rule accordingly

Add one rule of the form:
$LHS \rightarrow \alpha A' \beta$

Add one rule of the form:
$LHS \rightarrow \alpha\beta$

$\cancel{S \rightarrow NP \text{ and } PP}$
$\cancel{NP \rightarrow NP\ PP}$
$NP \rightarrow \epsilon$
$\cancel{PP \rightarrow PP\ PP}$
$PP \rightarrow \epsilon$
$S \rightarrow NP \text{ and } PP'$
$S \rightarrow NP \text{ and}$
$NP \rightarrow NP\ PP'$
$NP \rightarrow NP$
$PP \rightarrow PP'\ PP'$
$PP \rightarrow PP'$

Chomsky Normal Form (CNF)

- Example ϵ elimination

4. Add rules starting with A' as LHS (for every non ϵ rule of A)

$$NP \rightarrow \epsilon$$
$$PP \rightarrow \epsilon$$
$$S \rightarrow NP \text{ and } PP'$$
$$S \rightarrow NP \text{ and}$$
$$NP \rightarrow NP\ PP'$$
$$NP \rightarrow NP$$
$$PP \rightarrow PP'PP'$$
$$PP \rightarrow PP'$$
$$PP' \rightarrow PP'PP'$$
$$PP' \rightarrow PP'$$

Chomsky Normal Form (CNF)

- Example ϵ elimination

5. Repeat until no more reachable ϵ productions in your grammar

$$NP \rightarrow \epsilon$$
$$\cancel{PP \rightarrow \epsilon}$$
$$S \rightarrow NP \text{ and } PP'$$
$$S \rightarrow NP \text{ and}$$
$$NP \rightarrow NP\ PP'$$
$$NP \rightarrow NP$$
$$PP \rightarrow PP'PP'$$
$$PP \rightarrow PP'$$
$$PP' \rightarrow PP'PP'$$
$$PP' \rightarrow PP'$$

Chomsky Normal Form (CNF)

- Example ϵ elimination

- Can ϵ rules even occur in NLP?

➔ Yes! If you allow rules with Kleene-* notations

- E.g. :
  - $NP?$ Means an optional nominal phrase
  - $NP*$ Means arbitrarily many or 0 NPs

$$NP \rightarrow \epsilon$$
$$\cancel{PP \rightarrow \epsilon}$$
$$S \rightarrow NP \text{ and } PP'$$
$$S \rightarrow NP \text{ and}$$
$$NP \rightarrow NP\ PP'$$
$$NP \rightarrow NP$$
$$PP \rightarrow PP'PP'$$
$$PP \rightarrow PP'$$
$$PP' \rightarrow PP'PP'$$
$$PP' \rightarrow PP'$$

Chomsky Normal Form (CNF)

- Sketch of the algorithm for the conversion of a CFG into CNF:

  1. If S appears on the RHS, introduce a new rule: $S_0 \rightarrow S$
  2. Eliminate ϵ rules
  3. Remove Unit-production rules as shown in the coming example
  4. Clean up (optional)
  5. Remove all rules with RHS > 2 as shown in the example

Example conversion to CNF

3. Remove Unit-production rules“ ($S \rightarrow VP$) (We yet again make them explicit!)

$S \rightarrow NP\ VP$

$\boxed{S \rightarrow VP}$

$VP \rightarrow V\ NP$

$VP \rightarrow V$

$VP \rightarrow V\ NP\ PP$

$VP \rightarrow V\ PP$

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$VP \rightarrow V$

$VP \rightarrow V\ NP\ PP$

$VP \rightarrow V\ PP$

$S \rightarrow V\ NP$

$S \rightarrow V$

$S \rightarrow V\ NP\ PP$

$S \rightarrow V\ PP$

This will usually produce more unit productions rules, so repeat until none left!

Chomsky Normal Form (CNF)

- Sketch of the algorithm for the conversion of a CFG into CNF:

1. If S appears on the RHS, introduce a new rule: $S_0 \rightarrow S$
2. Eliminate ϵ rules
3. Remove Unit-production rules as shown in the coming example
4. Clean up (optional)
5. Remove all rules with RHS > 2 as shown in the example

Example conversion to CNF

4. Clean up (optional)

- This step has the job, to remove all useless rules (e.g. $NP \rightarrow NP$)
- And the remove all symbols that can never be reached
  - Usually leftovers from $\epsilon - \mathrm{elimination}$
- And remove symbols that can not produce any terminals („non-productive non terminals“

Chomsky Normal Form (CNF)

- Sketch of the algorithm for the conversion of a CFG into CNF:

1. If S appears on the RHS, introduce a new rule: $S_0 \rightarrow S$
2. Eliminate ϵ rules
3. Remove Unit-production rules as shown in the coming example
4. Clean up (optional)
5. Remove all rules with RHS > 2 as shown in the example

Example conversion to CNF

5. Eliminate all „RHS >2 production rules“ (by introducing placeholder non-terminals)

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$VP \rightarrow V$ “and” PP

$VP \rightarrow V\ PP$

$S \rightarrow V\ NP$

$S \rightarrow V\ NP\ PP$

$S \rightarrow V\ PP$

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$VP \rightarrow V\ NP\ PP$

$VP \rightarrow V\ PP$

$S \rightarrow V\ NP$

$\cancel{S \rightarrow V\ NP\ PP}$

$S \rightarrow V\ PP$

$S \rightarrow X0\ PP$

$X0 \rightarrow V\ NP$

Example conversion to CNF

5. Eliminate all „RHS >2 production rules“ (by introducing placeholder non-terminals)

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$VP \rightarrow V$ “and” PP

$VP \rightarrow V\ PP$

$S \rightarrow V\ NP$

$S \rightarrow V\ NP\ PP$

$S \rightarrow V\ PP$

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$VP \rightarrow V\ NP\ PP$

$VP \rightarrow V\ PP$

$S \rightarrow V\ NP$

$S \rightarrow V\ PP$

$S \rightarrow X0\ PP$

$X0 \rightarrow V\ NP$

$VP \rightarrow V\ X1\ PP$

$X1 \rightarrow$ “and”

Chomsky Normal Form (CNF)

- Sketch of the algorithm for the conversion of a CFG into CNF:

1. If S appears on the RHS, introduce a new rule: $S_0 \rightarrow S$
2. Eliminate ϵ rules
3. Remove Unit-production rules as shown in the coming example
4. Cleanup (optional)
5. Remove all rules with RHS > 2 as shown in the example

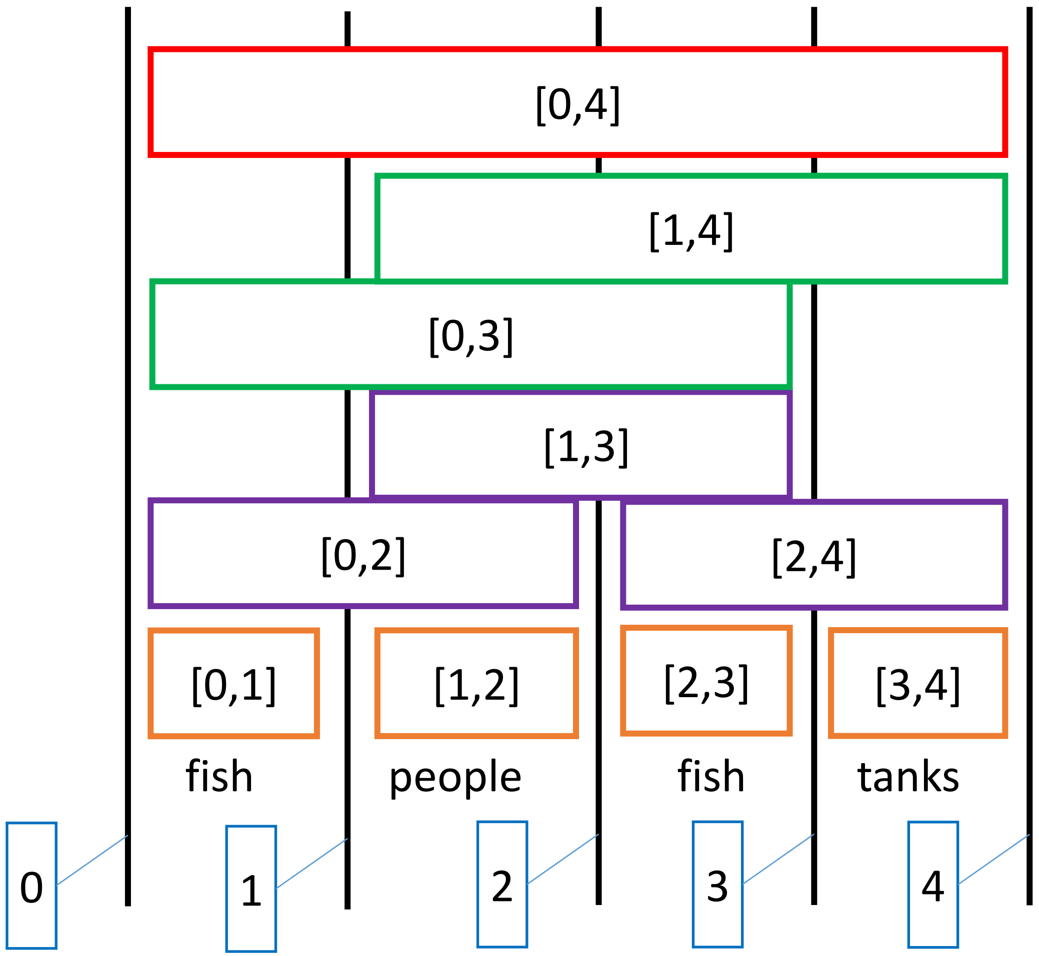
CKY Parsing

A worked example

CKY-Parsing

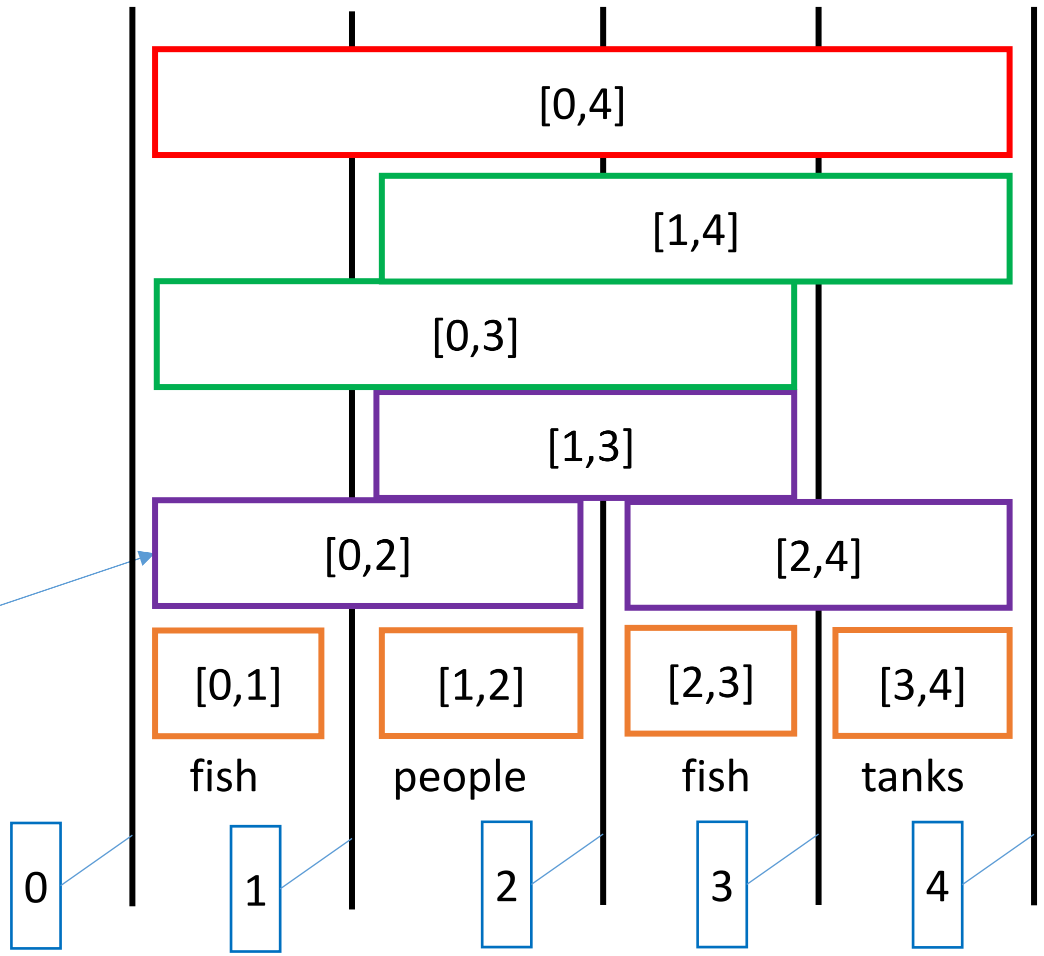
- We are given sentence, which is already separated into tokens
- And a grammar in Chomsky normal form
- The CKY algorithm produces the parse-tree with highest score in runtime $O(n^3)$ and $O(n^2)$ used memory
- We reduce the exponential nature of the search by storing the right things (dynamic programming)

CKY-Parsing-Intuition



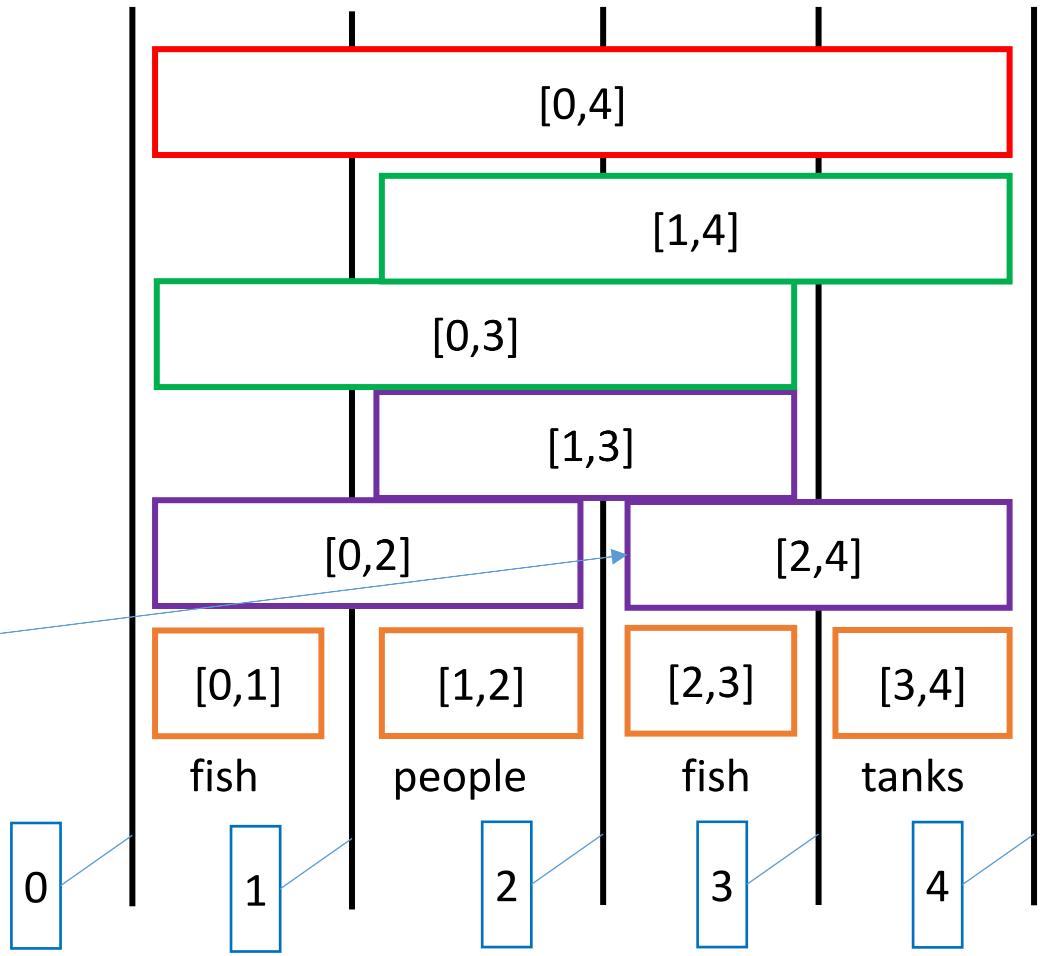
CKY-Parsing-Intuition

Can be built from [0,1] and [1,2]

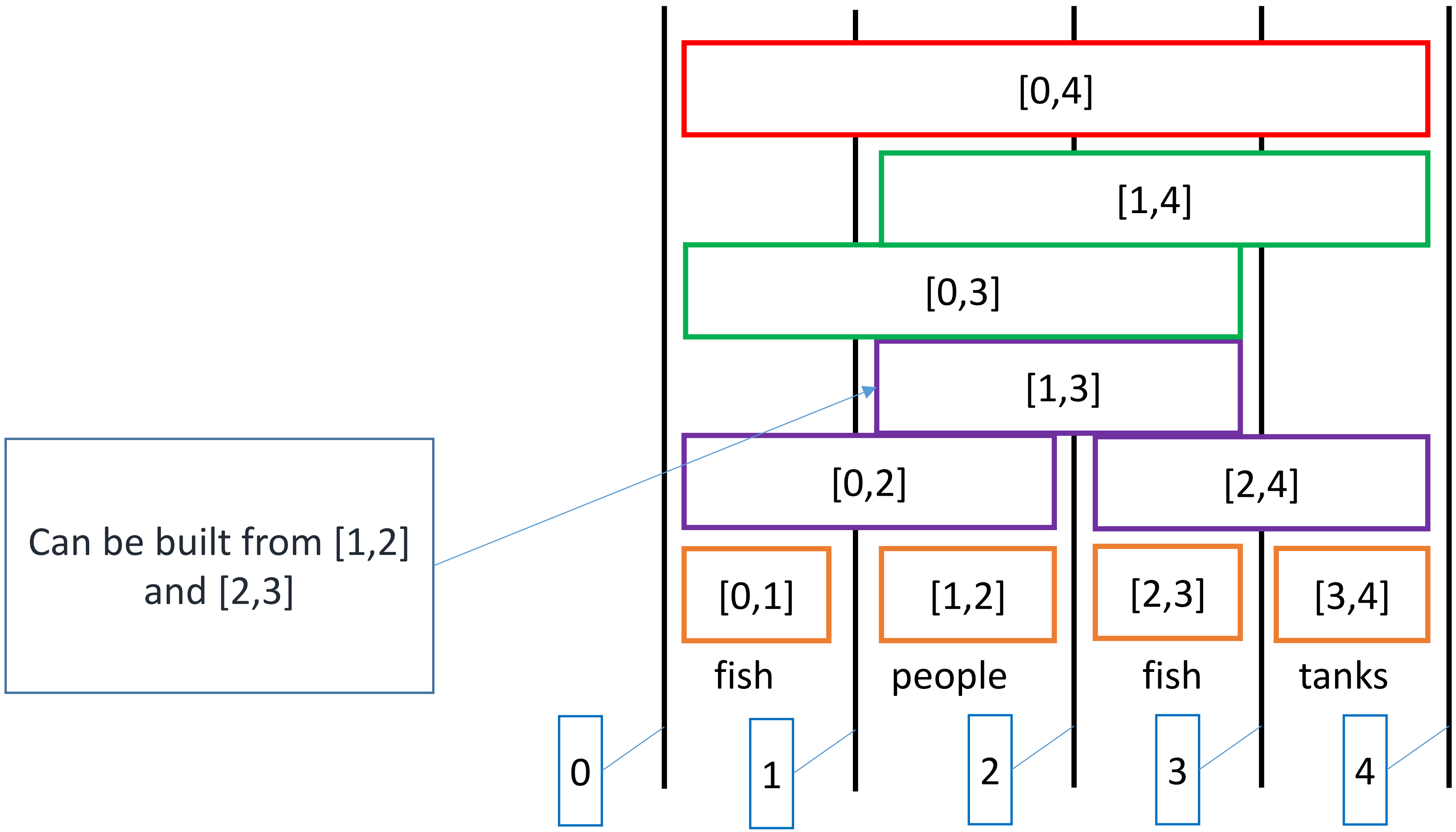


CKY-Parsing-Intuition

Can be built from [2,3] and [3,4]

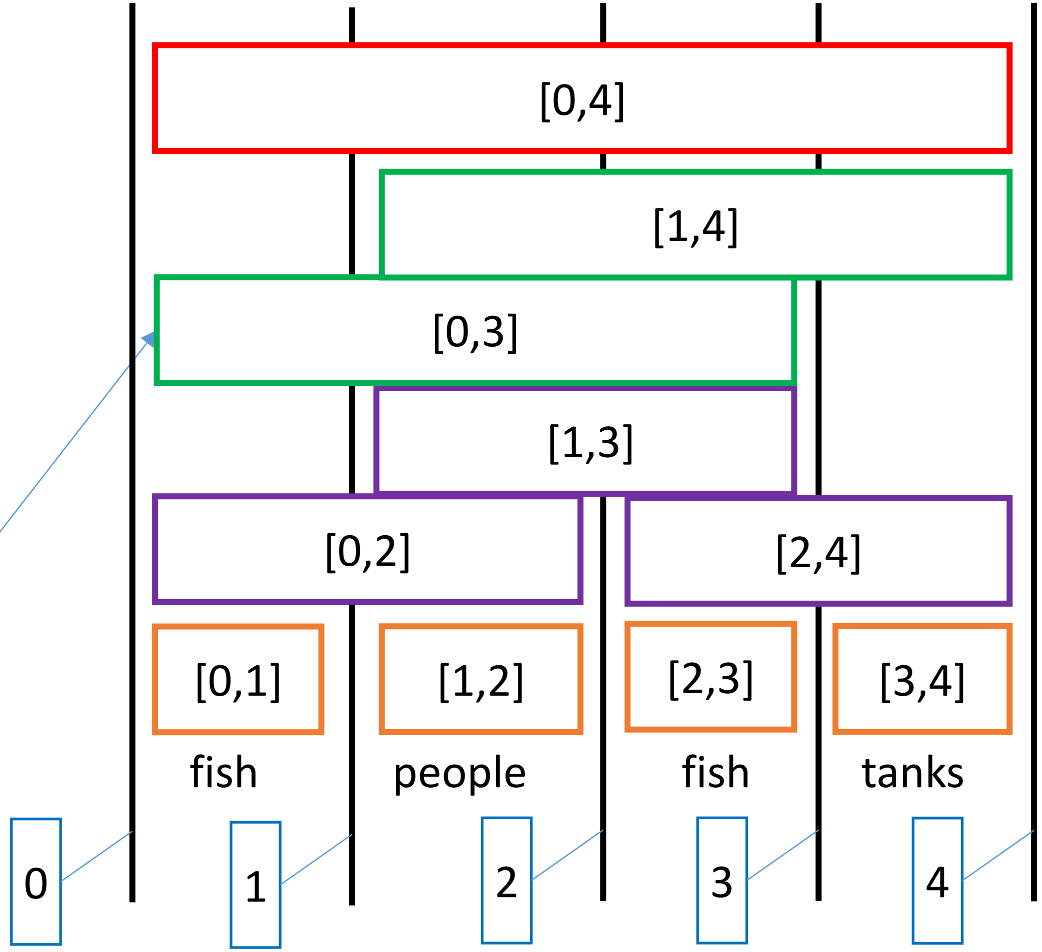


CKY-Parsing-Intuition

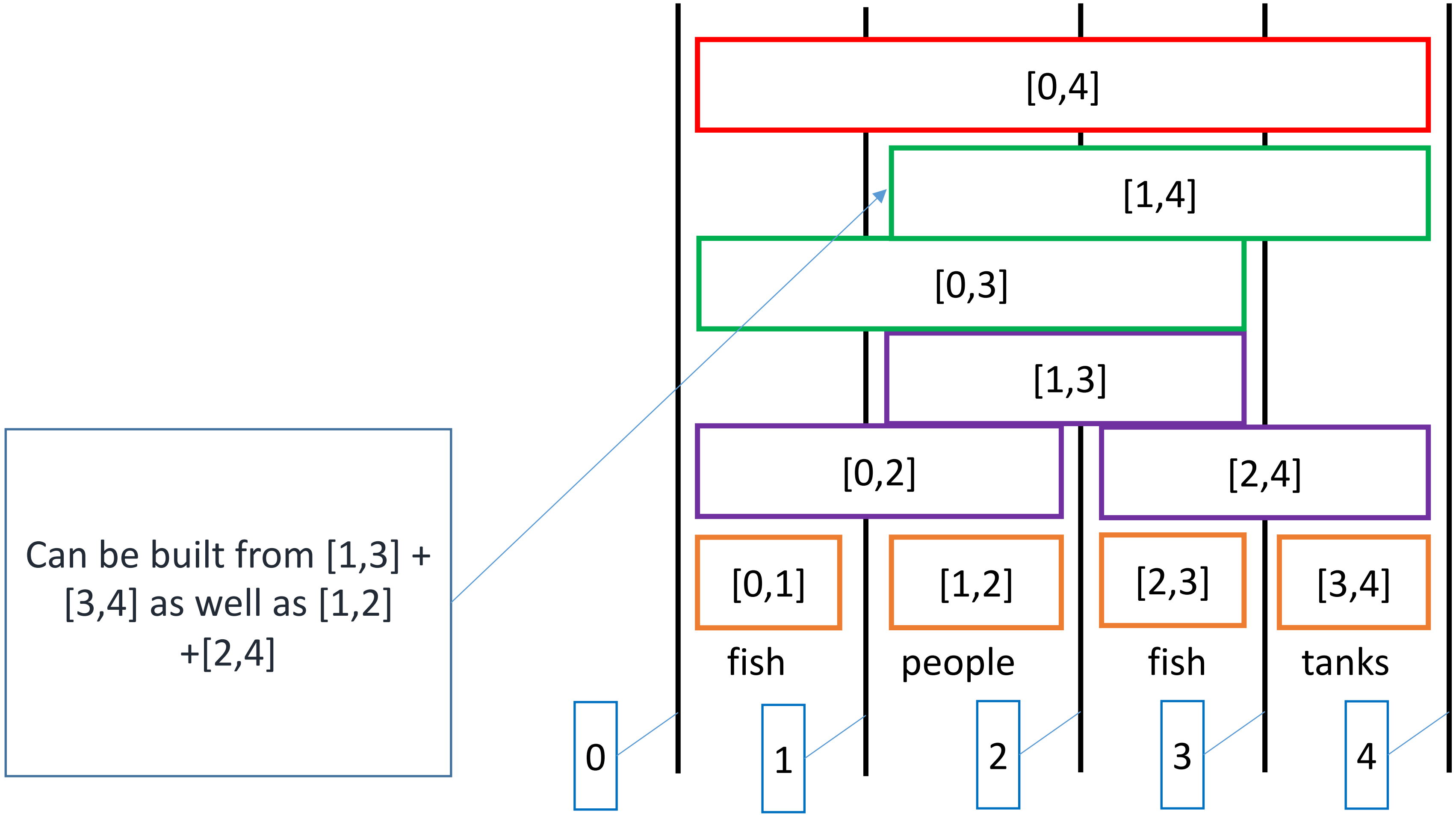


CKY-Parsing-Intuition

Can be built from [0,2] + [2,3] as well as [0,1] +[1,3]

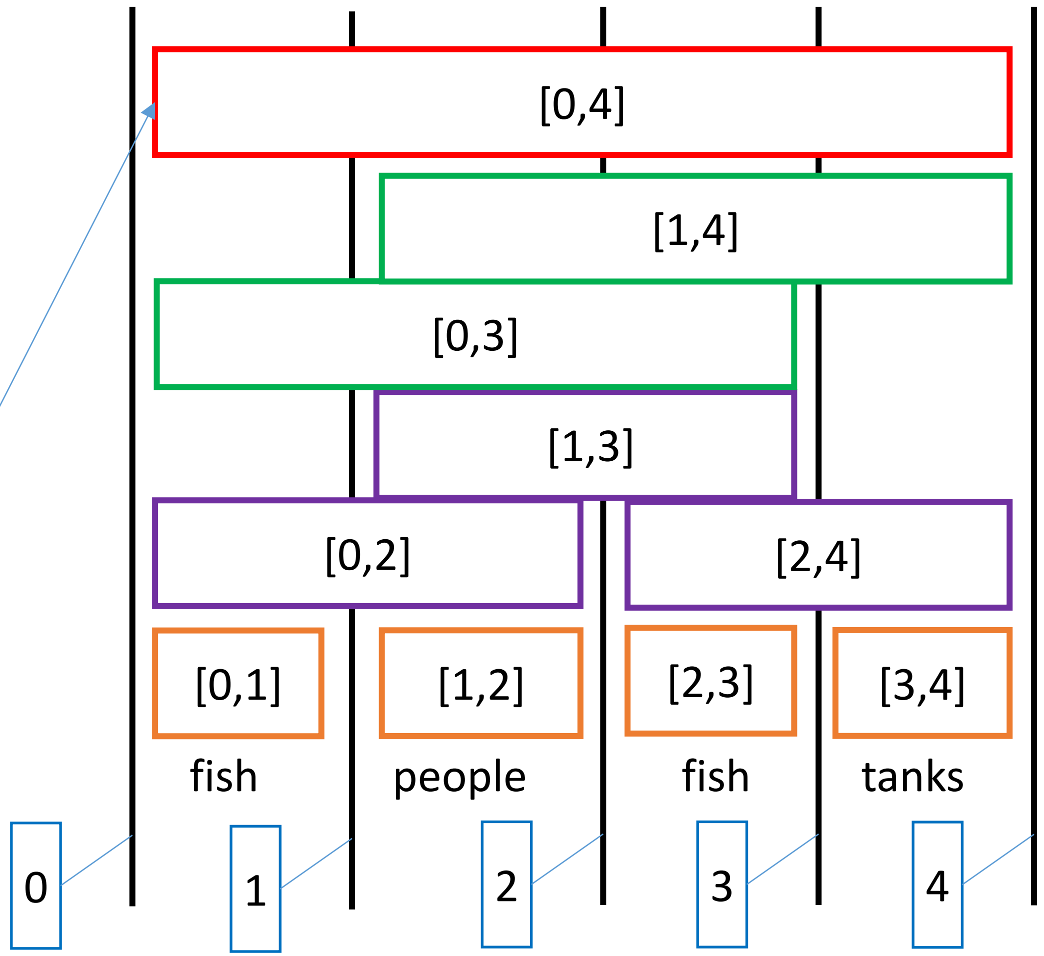


CKY-Parsing-Intuition



CKY-Parsing-Intuition

Can be built from [0,1] + [1,4] as well as [0,3] +[3,4] and [0,2]+[2,4]



Grammar:

S → NP VP 0.8
VP → V NP 0.5
S → V NP 0.2

NP → NP NP 0.3
V → *fish* 1
VP → *people* 0.1
VP → *fish* 0.4
NP → *people* 0.3
NP → *fish* 0.3
NP → *tanks* 0.3

		1	2	3	4
0	fish	score[0][1]	score[0][2]	score[0][3]	score[0][4]
1	people		score[1][2]	score[1][3]	score[1][4]
2	fish			score[2][3]	score[2][4]
3	tanks				score[3][4]
4					

Grammar:

S $\rightarrow$ NP VP 0.8
VP $\rightarrow$ V NP 0.5
S $\rightarrow$ V NP 0.2

NP $\rightarrow$ NP NP 0.3
V $\rightarrow$ *fish* 1
VP $\rightarrow$ *people* 0.1
VP $\rightarrow$ *fish* 0.4
NP $\rightarrow$ *people* 0.3
NP $\rightarrow$ *fish* 0.3
NP $\rightarrow$ *tanks* 0.3

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	score[0][2]	score[0][3]	score[0][4]
1		people	score[1][2]	score[1][3]	score[1][4]
2			fish	score[2][3]	score[2][4]
3				tanks	score[3][4]
4					

Grammar:

$S \rightarrow NP\ VP\ 0.8$
$VP \rightarrow V\ NP\ 0.5$
$S \rightarrow V\ NP\ 0.2$

$NP \rightarrow NP\ NP\ 0.3$
$V \rightarrow$ *fish* 1
$VP \rightarrow$ *people* 0.1
$VP \rightarrow$ *fish* 0.4
$NP \rightarrow$ *people* 0.3
$NP \rightarrow$ *fish* 0.3
$NP \rightarrow$ *tanks* 0.3

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	score[0][2]	score[0][3]	score[0][4]
1		people	VP: 0.1 NP:0.3	score[1][3]	score[1][4]
2			fish	score[2][3]	score[2][4]
3				tanks	score[3][4]
4					

Grammar:

S → NP VP 0.8
VP → V NP 0.5
S → V NP 0.2

NP → NP NP 0.3
V → *fish* 1
VP → *people* 0.1
VP → *fish* 0.4
NP → *people* 0.3
NP → *fish* 0.3
NP → *tanks* 0.3

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	score[0][2]	score[0][3]	score[0][4]
1	people		VP: 0.1 NP:0.3	score[1][3]	score[1][4]
2	fish			V: 1 NP: 0.3 VP:0.4	score[2][4]
3	tanks				score[3][4]
4					

Grammar:

S $\rightarrow$ NP VP 0.8
VP $\rightarrow$ V NP 0.5
S $\rightarrow$ V NP 0.2

NP $\rightarrow$ NP NP 0.3
V $\rightarrow$ *fish* 1
VP $\rightarrow$ *people* 0.1
VP $\rightarrow$ *fish* 0.4
NP $\rightarrow$ *people* 0.3
NP $\rightarrow$ *fish* 0.3
NP $\rightarrow$ *tanks* 0.3

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	score[0][2]	score[0][3]	score[0][4]
1	people		VP: 0.1 NP:0.3	score[1][3]	score[1][4]
2	fish			V: 1 NP: 0.3 VP:0.4	score[2][4]
3	tanks				NP: 0.3
4					

Grammar:

$S \rightarrow NP\ VP$ 0.8
$VP \rightarrow V\ NP$ 0.5
$S \rightarrow V\ NP$ 0.2

$NP \rightarrow NP\ NP$ 0.3
$V \rightarrow$ *fish* 1
$VP \rightarrow$ *people* 0.1
$VP \rightarrow$ *fish* 0.4
$NP \rightarrow$ *people* 0.3
$NP \rightarrow$ *fish* 0.3
$NP \rightarrow$ *tanks* 0.3

red + yellow = green

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	S:0.3 *0.1*0.8 VP: 1*0.3*0.5 S:1*0.3*0.2 NP:0.3*0.3*0.3	score[0][3]	score[0][4]
1		people	VP: 0.1 NP:0.3	score[1][3]	score[1][4]
2			fish	V: 1 NP: 0.3 VP:0.4	score[2][4]
3				tanks	NP: 0.3
4					

Grammar:

red + yellow = green

- S → NP VP 0.8
- VP → V NP 0.5
- S → V NP 0.2
- NP → NP NP 0.3
- V → *fish* 1
- VP → *people* 0.1
- VP → *fish* 0.4
- NP → *people* 0.3
- NP → *fish* 0.3
- NP → *tanks* 0.3

	1	2	3	4
0 fish	V: 1 NP: 0.3 VP:0.4	~~S:0.3 *0.1*0.8~~ VP: 1*0.3*0.5 S:1*0.3*0.2 NP:0.3*0.3*0.3	score[0][3]	score[0][4]
1 people		VP: 0.1 NP:0.3	score[1][3]	score[1][4]
2 fish			V: 1 NP: 0.3 VP:0.4	score[2][4]
3 tanks				NP: 0.3
4				

red + yellow = green

Grammar:

S $\rightarrow$ NP VP 0.8
VP $\rightarrow$ V NP 0.5
S $\rightarrow$ V NP 0.2

NP $\rightarrow$ NP NP 0.3
V $\rightarrow$ *fish* 1
VP $\rightarrow$ *people* 0.1
VP $\rightarrow$ *fish* 0.4
NP $\rightarrow$ *people* 0.3
NP $\rightarrow$ *fish* 0.3
NP $\rightarrow$ *tanks* 0.3

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	~~S:0.3 *0.1*0.8~~ VP: 1*0.3*0.5 S:1*0.3*0.2 NP:0.3*0.3*0.3	score[0][3]	score[0][4]
1	people		VP: 0.1 NP:0.3	S:0.3*0.4*0.8 NP:0.3*0.3*0.3	score[1][4]
2	fish			V: 1 NP: 0.3 VP:0.4	score[2][4]
3	tanks				NP: 0.3
4					

red + yellow = green

Grammar:

S → NP VP 0.8
VP → V NP 0.5
S → V NP 0.2

NP → NP NP 0.3
V → *fish* 1
VP → *people* 0.1
VP → *fish* 0.4
NP → *people* 0.3
NP → *fish* 0.3
NP → *tanks* 0.3

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	~~S:0.3 *0.1*0.8~~ VP: 1*0.3*0.5 S:1*0.3*0.2 NP:0.3*0.3*0.3	score[0][3]	score[0][4]
1		people	VP: 0.1 NP:0.3	S:0.3*0.4*0.8 NP:0.3*0.3*0.3	score[1][4]
2			fish	V: 1 NP: 0.3 VP:0.4	VP:1*0.3*0.5 S:1*0.2*0.3 NP:0.3*0.3*0.3
3				tanks	NP: 0.3
4					

Grammar:

S → NP VP 0.8
VP → V NP 0.5
S → V NP 0.2

NP → NP NP 0.3
V → *fish* 1
VP → *people* 0.1
VP → *fish* 0.4
NP → *people* 0.3
NP → *fish* 0.3
NP → *tanks* 0.3

red + yellow = green

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	VP: 0.15 S:0.06 NP:0.027	S: 0.027*0.4*0.8 NP:0.027*0.3*0.3	score[0][4]
1	people		VP: 0.1 NP:0.3	S:0.096 NP:0.027	score[1][4]
2	fish			V: 1 NP: 0.3 VP:0.4	VP:0.15 S:0.06 NP:0.027
3	tanks				NP: 0.3
4					

Grammar:

S → NP VP 0.8
VP → V NP 0.5
S → V NP 0.2

NP → NP NP 0.3
V → *fish* 1
VP → *people* 0.1
VP → *fish* 0.4
NP → *people* 0.3
NP → *fish* 0.3
NP → *tanks* 0.3

red + yellow = green

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	VP: 0.15 S:0.06 NP:0.027	S: 0.027*0.4*0.8 NP:0.027*0.3*0.3 VP:1*0.027*0.5 S:1*0.027*0.2	score[0][4]
1	people		VP: 0.1 NP:0.3	S:0.096 NP:0.027	score[1][4]
2	fish			V: 1 NP: 0.3 VP:0.4	VP:0.15 S:0.06 NP:0.027
3	tanks				NP: 0.3
4					

Grammar:

S → NP VP 0.8
VP → V NP 0.5
S → V NP 0.2

NP → NP NP 0.3
V → *fish* 1
VP → *people* 0.1
VP → *fish* 0.4
NP → *people* 0.3
NP → *fish* 0.3
NP → *tanks* 0.3

red + yellow = green

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	VP: 0.15 S:0.06 NP:0.027	S: 0.027*0.4*0.8 NP:0.027*0.3*0.3 VP:1*0.027*0.5 ~~S:1*0.027*0.2~~	score[0][4]
1	people		VP: 0.1 NP:0.3	S:0.096 NP:0.027	NP:0.027*0.3*0.3
2	fish			V: 1 NP: 0.3 VP:0.4	VP:0.15 S:0.06 NP:0.027
3	tanks				NP: 0.3
4					

red + yellow = green

Grammar:

S $\rightarrow$ NP VP 0.8
VP $\rightarrow$ V NP 0.5
S $\rightarrow$ V NP 0.2

NP $\rightarrow$ NP NP 0.3
V $\rightarrow$ *fish* 1
VP $\rightarrow$ *people* 0.1
VP $\rightarrow$ *fish* 0.4
NP $\rightarrow$ *people* 0.3
NP $\rightarrow$ *fish* 0.3
NP $\rightarrow$ *tanks* 0.3

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	VP: 0.15 S:0.06 NP:0.027	S: 0.027*0.4*0.8 NP:0.027*0.3*0.3 VP:1*0.027*0.5 ~~S:1*0.027*0.2~~	score[0][4]
1	people		VP: 0.1 NP:0.3	S:0.096 NP:0.027	NP:0.027*0.3*0.3 S:0.3*0.15*0.8 NP:0.3*0.027*0.3
2	fish			V: 1 NP: 0.3 VP:0.4	VP:0.15 S:0.06 NP:0.027
3	tanks				NP: 0.3
4					

red + yellow = green

Grammar:

S $\rightarrow$ NP VP 0.8
VP $\rightarrow$ V NP 0.5
S $\rightarrow$ V NP 0.2

NP $\rightarrow$ NP NP 0.3
V $\rightarrow$ *fish* 1
VP $\rightarrow$ *people* 0.1
VP $\rightarrow$ *fish* 0.4
NP $\rightarrow$ *people* 0.3
NP $\rightarrow$ *fish* 0.3
NP $\rightarrow$ *tanks* 0.3

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	VP: 0.15 S:0.06 NP:0.027	S: 0.00864 NP:0.00243 VP:0.0135	NP:0.00243*0.3*0.3
1	people		VP: 0.1 NP:0.3	S:0.096 NP:0.027	NP:0.00243 S:0.036
2	fish			V: 1 NP: 0.3 VP:0.4	VP:0.15 S:0.06 NP:0.027
3	tanks				NP: 0.3
4					

red + yellow = green

Grammar:

S $\rightarrow$ NP VP 0.8
VP $\rightarrow$ V NP 0.5
S $\rightarrow$ V NP 0.2

NP $\rightarrow$ NP NP 0.3
V $\rightarrow$ *fish* 1
VP $\rightarrow$ *people* 0.1
VP $\rightarrow$ *fish* 0.4
NP $\rightarrow$ *people* 0.3
NP $\rightarrow$ *fish* 0.3
NP $\rightarrow$ *tanks* 0.3

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	VP: 0.15 S:0.06 NP:0.027	S: 0.00864 NP:0.00243 VP:0.0135	NP:0.00243*0.3*0.3 S:0.027*0.15*0.8 NP:0.027*0.027*0.3
1	people		VP: 0.1 NP:0.3	S:0.096 NP:0.027	NP:0.00243 S:0.036
2	fish			V: 1 NP: 0.3 VP:0.4	VP:0.15 S:0.06 NP:0.027
3	tanks				NP: 0.3
4					

Grammar:

S → NP VP 0.8
VP → V NP 0.5
S → V NP 0.2

NP → NP NP 0.3
V → *fish* 1
VP → *people* 0.1
VP → *fish* 0.4
NP → *people* 0.3
NP → *fish* 0.3
NP → *tanks* 0.3

red + yellow = green

	1	2	3	4
0 fish	V: 1 NP: 0.3 VP:0.4	VP: 0.15 S:0.06 NP:0.027	S: 0.00864 NP:0.00243 VP:0.0135	NP:0.00243*0.3*0.3 S:0.027*0.15*0.8 NP:0.027*0.027*0.3 VP:1*0.00243*0.5 S:1*0.00243*0.2 NP:0.3*0.00243*0.3
1 people		VP: 0.1 NP:0.3	S:0.096 NP:0.027	NP:0.00243 S:0.036
2 fish			V: 1 NP: 0.3 VP:0.4	VP:0.15 S:0.06 NP:0.027
3 tanks				NP: 0.3
4				

Grammar:

red + yellow = green

S → NP VP 0.8
VP → V NP 0.5
S → V NP 0.2

NP → NP NP 0.3
V → *fish* 1
VP → *people* 0.1
VP → *fish* 0.4
NP → *people* 0.3
NP → *fish* 0.3
NP → *tanks* 0.3

		1	2	3	4
0	fish	V: 1 NP: 0.3 VP:0.4	VP: 0.15 S:0.06 NP:0.027	S: 0.00864 NP:0.00243 VP:0.0135	~~NP:0.00243*0.3*0.3~~ S:0.00324 ~~NP:0.027*0.027*0.3~~ ~~VP:1*0.00243*0.5~~ S:0.000486 ~~NP:0.3*0.00243*0.3~~
1	people		VP: 0.1 NP:0.3	S:0.096 NP:0.027	NP:0.00243 S:0.036
2	fish			V: 1 NP: 0.3 VP:0.4	VP:0.15 S:0.06 NP:0.027
3	tanks				NP: 0.3
4					

CKY-Parsing Result

- Consider the cell which contains the entire sentence:

~~NP:0.00243*0.3*0.3~~
S:0.00324
~~NP:0.027*0.027*0.3~~
~~VP:1*0.00243*0.5~~
S:0.000486
~~NP:0.3*0.00243*0.3~~

➔ If there is an S, the sentence is valid according to our grammar

➔ Parse is built by „following backpointers“

➔ Probability (or score) of the best tree: 0.00324

CKY-Parsing

function PROBABILISTIC-CKY(*words*,*grammar*) **returns** most probable parse
and its probability

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{ A \mid A \rightarrow words[j] \in grammar\}$
    $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{ A \mid A \rightarrow BC \in grammar,$
        and $table[i,k,B] > 0$ **and** $table[k,j,C] > 0 \}$
        if $(table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ **then**
          $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
          $back[i,j,A] \leftarrow \{k,B,C\}$
  return BUILD_TREE($back$[1, LENGTH(*words*), S]), $table$[1, LENGTH(*words*), S]

Evaluation

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1 = \frac{2Prec \cdot Rec}{Prec+Rec}$
- Is addressed in more detail in the chapter "Evaluation"