



UPPSALA
UNIVERSITET

The CKY algorithm part 2: Probabilistic parsing

Syntactic analysis/parsing

2017-11-14

Sara Stymne

Department of Linguistics and Philology

Based on slides from Marco Kuhlmann





UPPSALA
UNIVERSITET

Recap: The CKY algorithm



The CKY algorithm

The CKY algorithm is an efficient bottom–up parsing algorithm for context-free grammars.

We use it to solve the following tasks:

- **Recognition:**
Is there any parse tree at all?
- **Probabilistic parsing:**
What is the most probable parse tree?



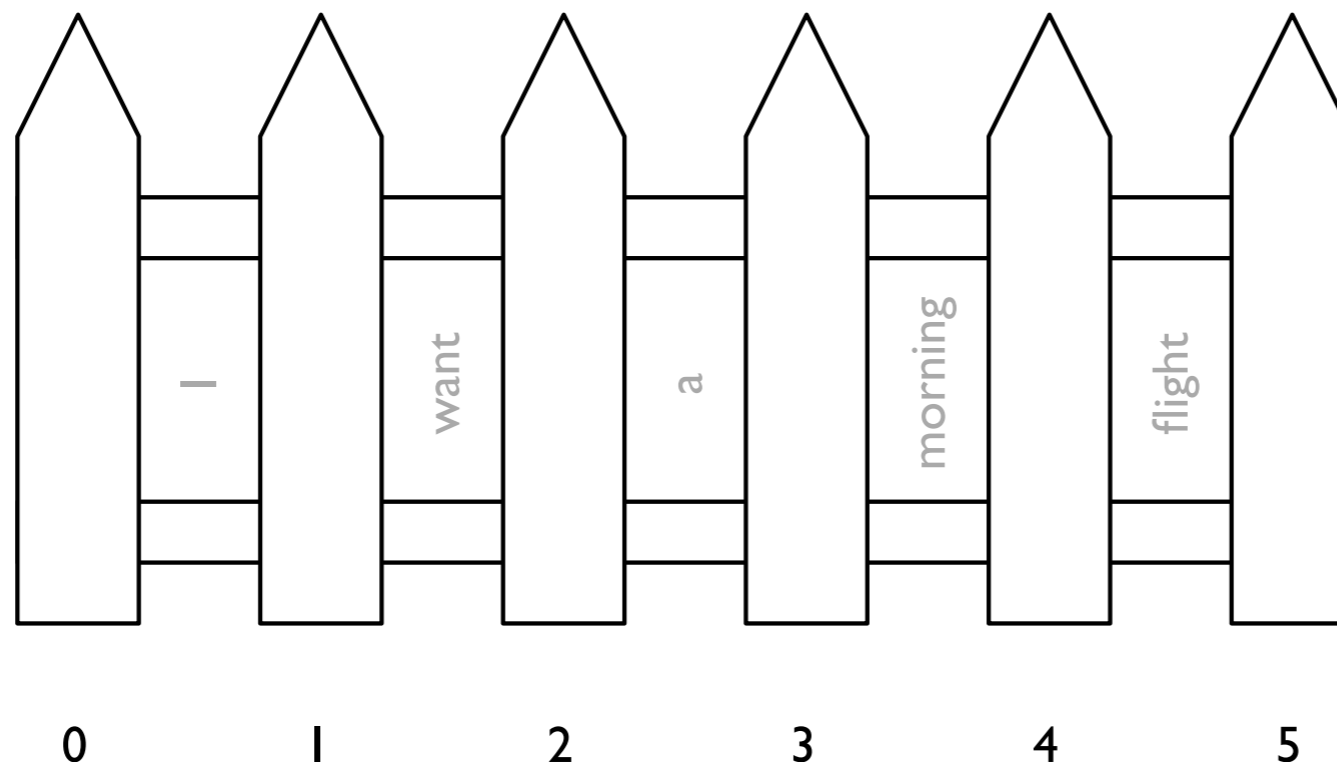
Restrictions

- The CKY algorithm as we present it here can only handle rules that are at most binary:
 $C \rightarrow w_i, C \rightarrow C_1 C_2, (C \rightarrow C_1)$
- This restriction is not a problem theoretically, but requires preprocessing (binarization) and postprocessing (debinarization).
- A parsing algorithm that does away with this restriction is Earley's algorithm (J&M 13.4.2).



Fencepost positions

We view the sequence w as a fence with n holes,
one hole for each token w_i ,
and we number the fenceposts from 0 till n .





Implementation

- The implementation uses a three-dimensional array *chart*.
- Whenever we have recognized a parse tree that spans all words between *min* and *max* and whose root node is labeled with C , we set the entry $chart[min][max][C]$ to *true*.



Implementation: preterminal rules

```
for each  $w_i$  from left to right
```

```
  for each preterminal rule  $C \rightarrow w_i$ 
```

```
    chart[i - 1][i][C] = true
```



Implementation: Binary rules

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      for each binary rule C -> C1 C2
        for each mid from min + 1 to max - 1
          if chart[min][mid][C1] and chart[mid][max][C2] then
            chart[min][max][C] = true
```




Question

How do we need to extend the code in order to handle unary rules $C \rightarrow C_1$?



Unary rules

```
for each max from 1 to n ← new bounds!  
  for each min from max - 1 down to 0 ← new bounds!  
  
    // First, try all binary rules as before.  
  
    ...  
  
    // Then, try all unary rules.  
  
    for each syntactic category C  
      for each unary rule C -> C1  
        if chart[min][max][C1] then  
          chart[min][max][C] = true
```



UPPSALA
UNIVERSITET

Implementation

Question

This is not quite right.

Why, and how could we fix the problem?



Structure

- Is there any parse tree at all?
- **What is the most probable parse tree?**



UPPSALA
UNIVERSITET

Probabilistic parsing



What is the most probable parse tree?

- The number of possible parse trees grows rapidly with the length of the input.
- But not all parse trees are equally useful.

Example: I booked a flight from Los Angeles.

- In many applications, we want the ‘best’ parse tree, or the first few best trees.
- Special case: ‘best’ = ‘most probable’



Probabilistic context-free grammars

A **probabilistic context-free grammar (PCFG)**

is a context-free grammar where

- each rule r has been assigned a probability $p(r)$ between 0 and 1
- the probabilities of rules with the same left-hand side sum up to 1



Example

Made up probabilities!

Rule	Probability
$S \rightarrow NPVP$	1
$NP \rightarrow \text{Pronoun}$	1/3
$NP \rightarrow \text{Proper-Noun}$	1/3
$NP \rightarrow \text{Det Nominal}$	1/3
$\text{Nominal} \rightarrow \text{Nominal PP}$	1/3
$\text{Nominal} \rightarrow \text{Noun}$	2/3
$VP \rightarrow \text{Verb NP}$	8/9
$VP \rightarrow \text{Verb NP PP}$	1/9
$PP \rightarrow \text{Preposition NP}$	1

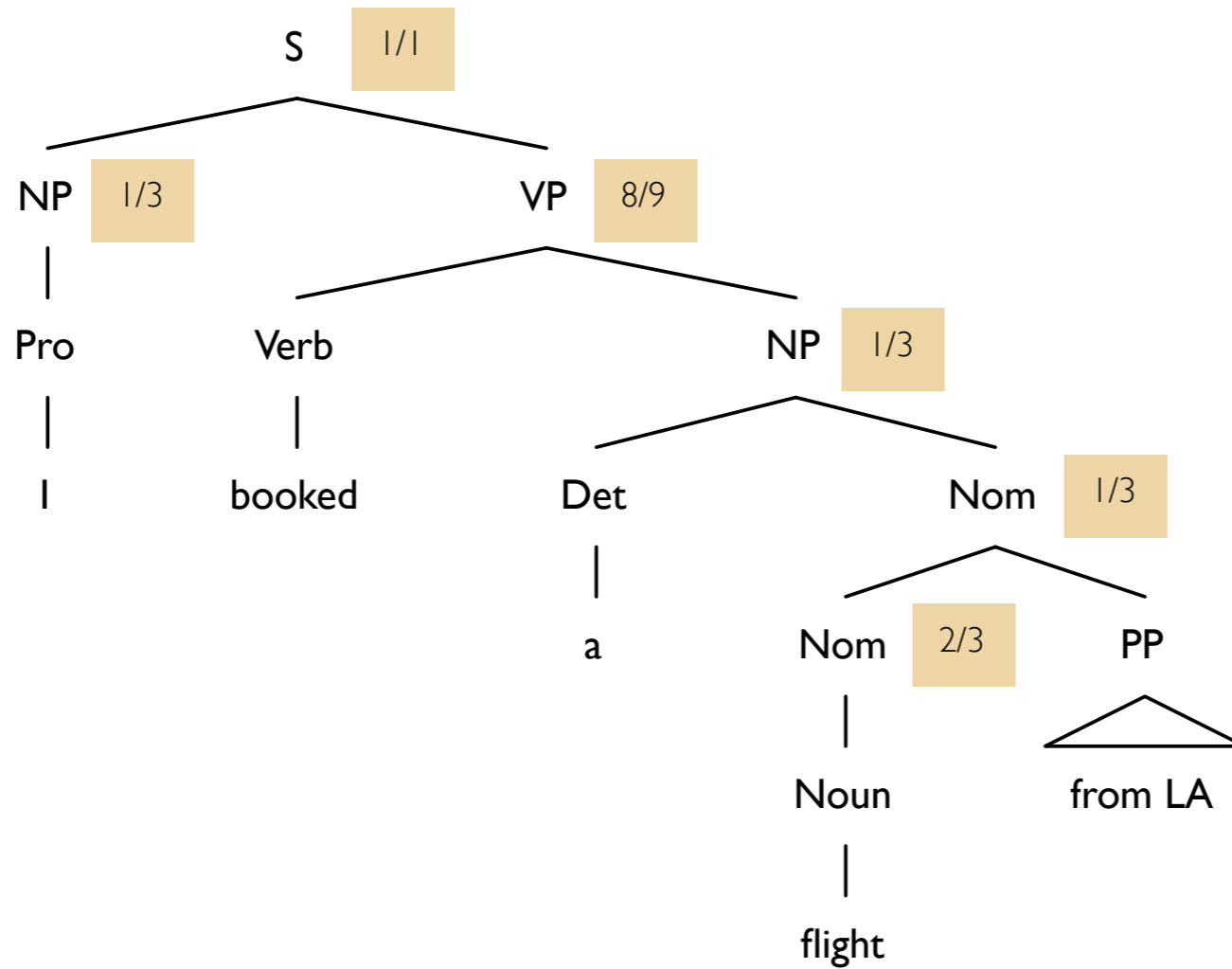


The probability of a parse tree

The probability of a parse tree is defined as the product of the probabilities of the rules that have been used to build the parse tree.



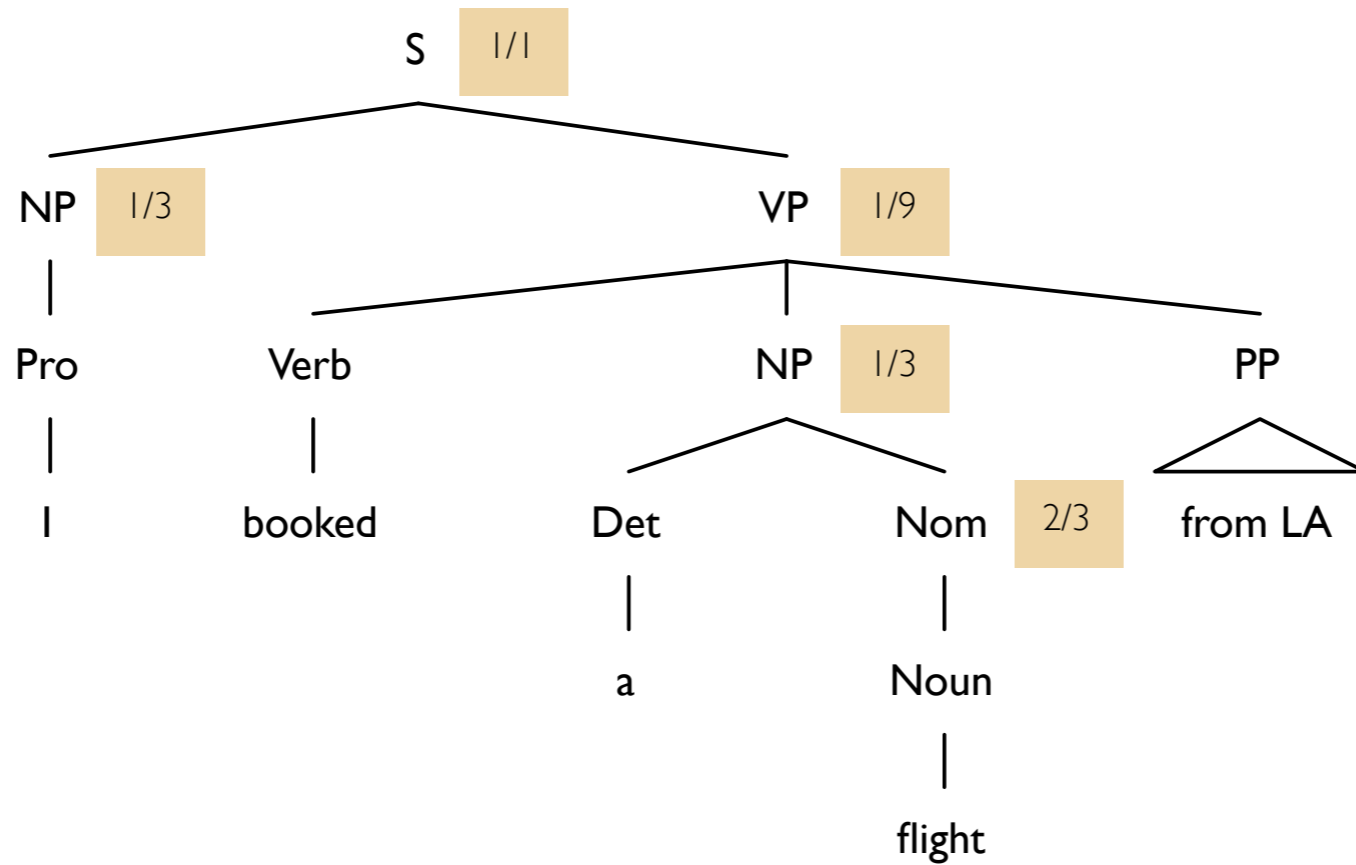
Example



Probability: 16/729



Example



Probability: 6/729



UPPSALA
UNIVERSITET

Probabilistic parsing

Small trees





UPPSALA
UNIVERSITET

Probabilistic parsing

Small trees

$$C \rightarrow w_i$$

Choose the most probable rule!

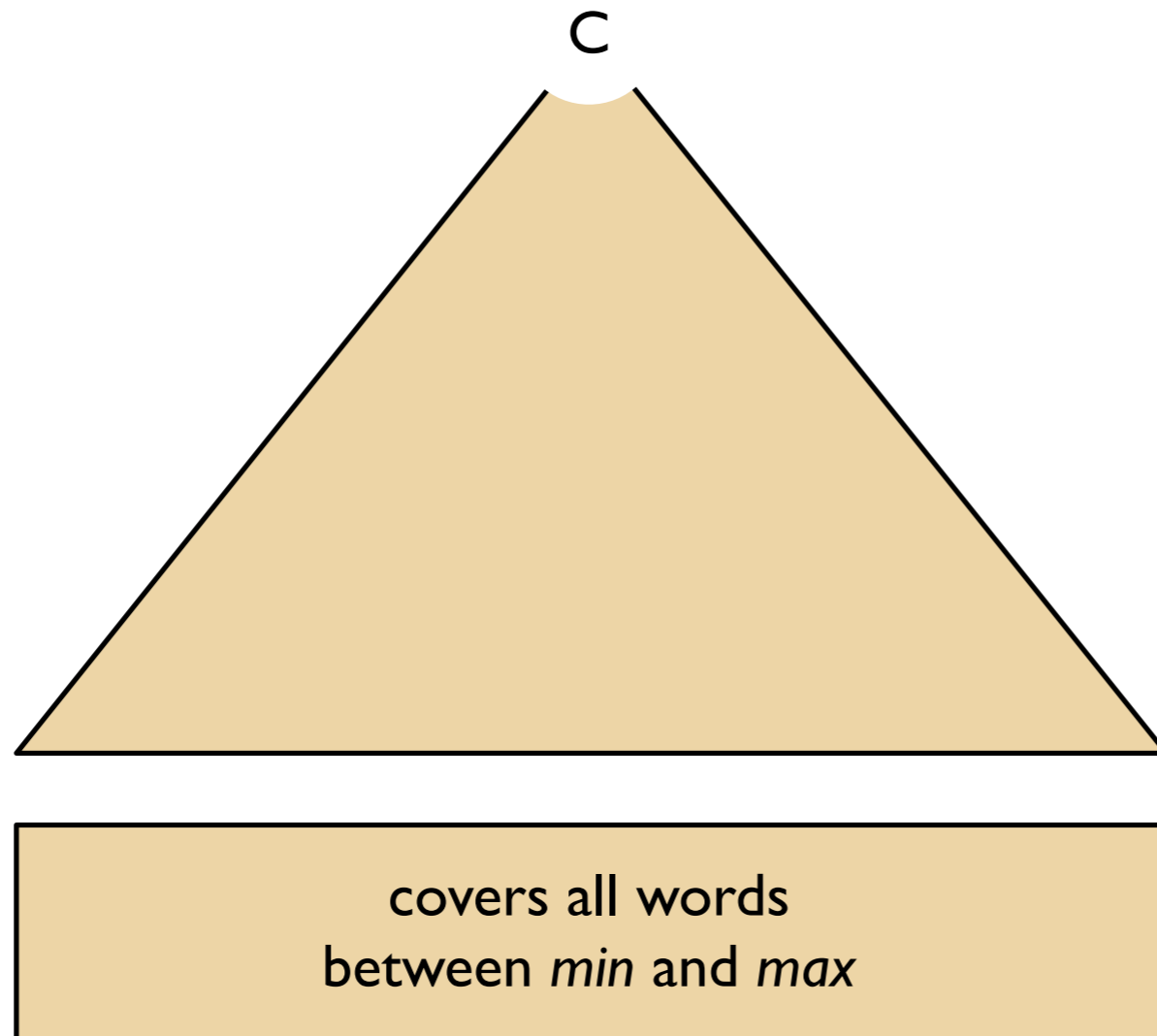
w_i



UPPSALA
UNIVERSITET

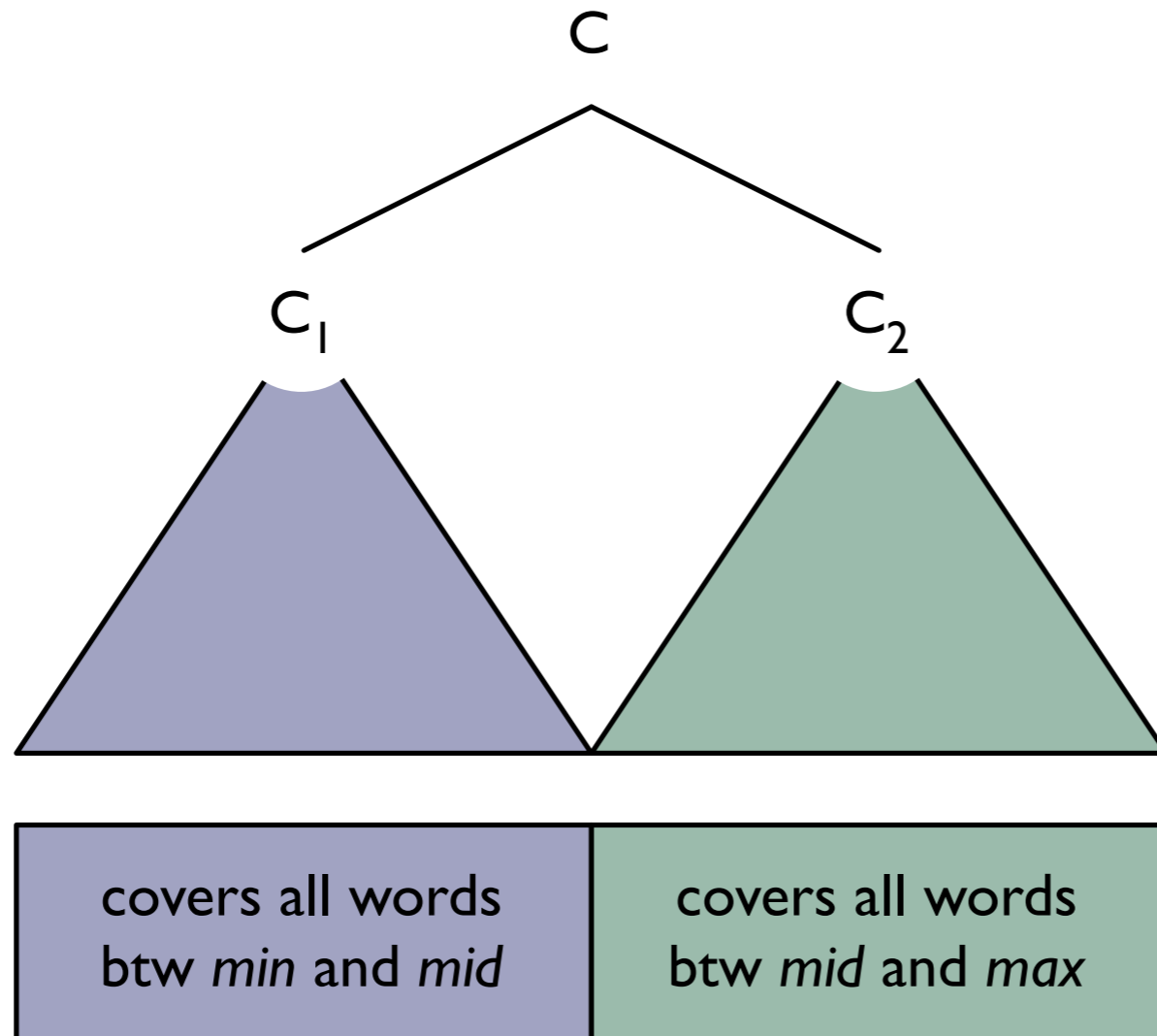
Probabilistic parsing

Small trees





Big trees

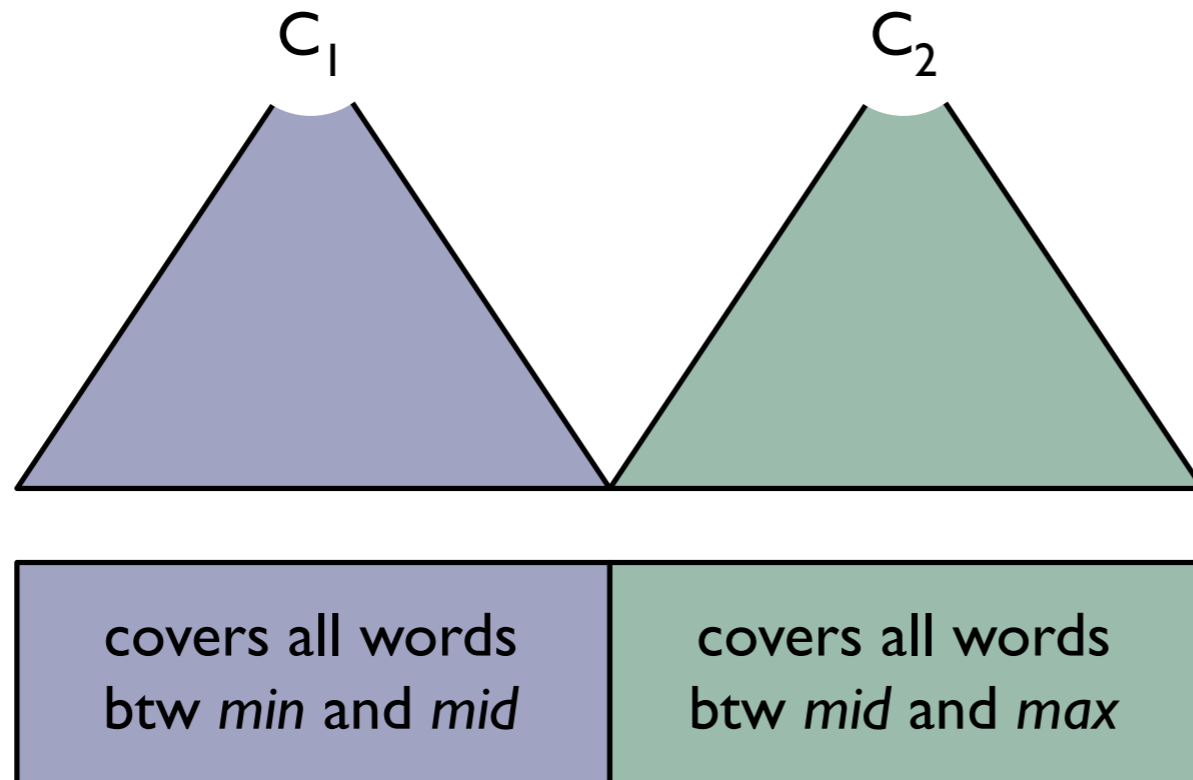




Big trees

$$C \rightarrow C_1 C_2$$

Choose the most probable rule!





Idea

- **For trees built using preterminal rules:**
Find a most probable rule. (apply all rules!)
- **For trees built using binary rules:**
Find a binary rule r and a split point mid such that $p(r) \times p(t_1) \times p(t_2)$ is maximal, where t_1 is a most probable left subtree and t_2 is a most probable right subtree.



Implementation

- Instead of an array with Boolean values, we now have an array with probabilities, i.e., *floats*.
- When all is done, we want to have $chart[min][max][C] = p$ if and only if a most probable parse tree with signature $[min, max, C]$ has probability p .



UPPSALA
UNIVERSITET

Probabilistic parsing

Preterminal rules

for each w_i from left to right

for each preterminal rule $C \rightarrow w_i$

$\text{chart}[i - 1][i][C] = p(C \rightarrow w_i)$



Binary rules

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double best = undefined
      for each binary rule C -> C1 C2
        for each mid from min + 1 to max - 1
          double t1 = chart[min][mid][C1]
          double t2 = chart[mid][max][C2]
          double candidate = t1 * t2 * p(C -> C1 C2)
          if candidate > best then
            best = candidate
      chart[min][max][C] = best
```



UPPSALA
UNIVERSITET

Probabilistic parsing

Question

How should we treat unary rules?



One more question

The only thing that we have done so far is to compute the *probability* of the most probable parse tree. But how does that parse tree look like?



Backpointers

- When we find a new best parse tree, we want to remember how we built it.
- For each element $t = \text{chart}[\text{min}][\text{max}][C]$, we also store **backpointers** to those elements from which t was built.
- Besides the ordinary chart of floats, we also have a backpointer chart



Backpointers

```
double best = undefined

Backpointer backpointer = undefined

...

if candidate > best then

    best = candidate

    // We found a better tree; update the backpointer!

    backpointer = (C -> C1 C2, min, mid, max)

...

chart[min][max][C] = best

backpointerChart[min][max][C] = backpointer
```


Implementation

Implementation ideas

```
# defaultdict is a suitable datastructure for charts!
```

```
    pi = defaultdict(float)
```

```
    bp = defaultdict(tuple)
```

```
# Recognize all parse trees built with with preterminal rules.
```

```
# Recognize all parse trees built with binary rules.
```

```
# "S" is not always the top category, the below is a simplification
```

```
return backtrace(bp[0, n, "S"], bp);
```



Assignment 1/2: CKY parsing

- Tips:
 - Use print statements to make sure your code does what you think it should
 - Use a small test set, and possibly a small grammar during development. The parser is slow
 - Start on the assignment now! Do not leave it until the last week!



Advanced models

- The CKY model is used in many competitive parsers
- To improve performance the grammar is often modified, e.g. by
 - Parent annotation
 - Lexicalized rules



Summary

- The CKY algorithm is an efficient parsing algorithm for context-free grammars.
- Today, we have used it for probabilistic parsing:
The task of computing the most probable parse tree for a given sentence.



Coming up

- Thursday:
 - Lecture on treebanks and PCFGs
- Next week:
 - Tuesday: Earley's algorithm
 - Thursday:
 - Advanced PCFG parsing (~ 45 minutes)
 - Time for supervision (~ 45 minutes)



Own work

- Reading:
 - CKY:J&M 14.1, 14.2
 - Treebanks: J&M 12.4, 14.3, 14.7
- Work on assignments
 - Start working on assignment 1/2: CKY
 - Contact me if you need help
 - You can also ask questions in the lectures



Deadlines

- Ordinary deadlines
 - Assignment 1+2 (b) 1 (m): 17-12-04
 - Assignment 2 (m): 17-12-18
 - Assignment 3+4: 18-01-12
- Resubmission deadline
 - All assignments: 18-02-09
- If you fail to meet these deadlines you will have to wait for the next time the course is given (Jan-Mar 2018)
- Assignments will only be graded in connection with each deadline
- In case of special circumstances, contact me **BEFORE** the deadline it concerns!