Language Technologies Institute

Carnegie Mellon University

Algorithms for NLP

CS 11-711, Fall 2020

Lecture 12: PCFGs & CKY algorithm

Sanket Vaibhav Mehta (SVM)
Slides adapted from: Yulia Tsvetkov, Anjalie Field – CMU

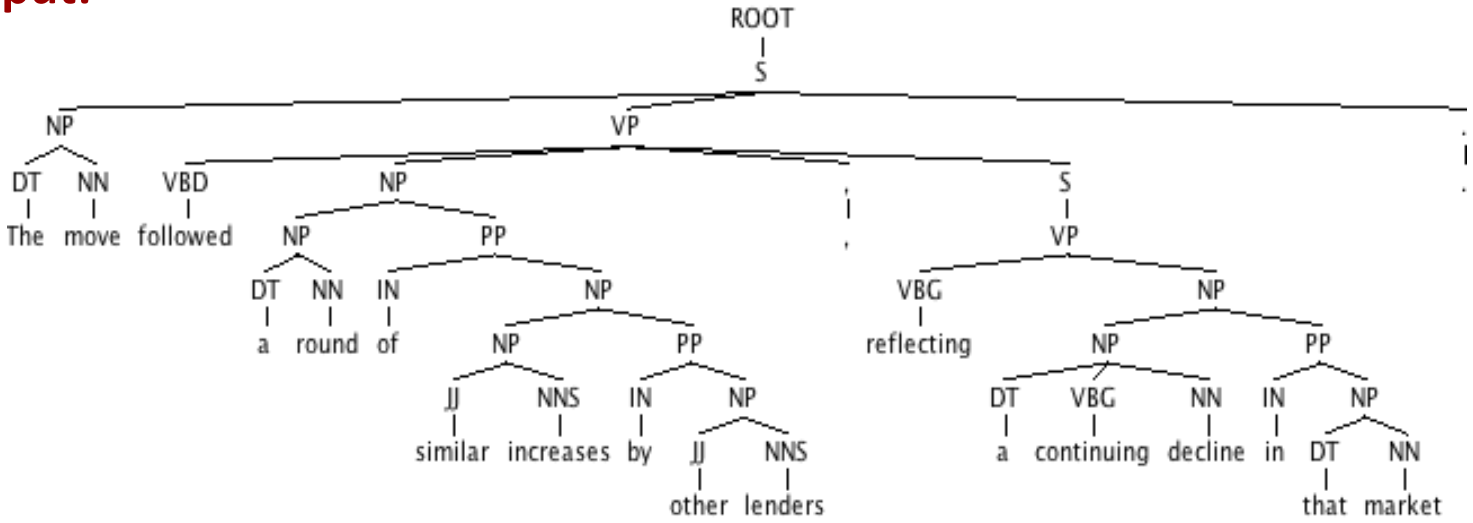
Agenda

- Recap: Syntactic Parsing, Constituent Trees, PCFGs
- CKY Algorithm
- CKY with PCFGs
- Parser Evaluation
- Improvements to CKY

Syntactic parsing

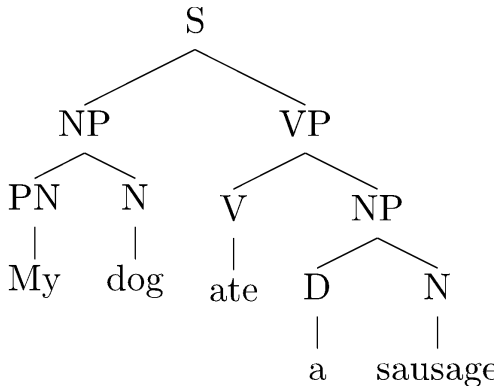
- **Input:**
  - The move followed a round of similar increases by other lenders, reflecting a continuing decline in that market

- **Output:**



Constituent trees

- **Internal nodes correspond to phrases**
  - S: a sentence
  - NP (Noun Phrase): My dog, a sandwich, lakes,..
  - VP (Verb Phrase): ate a sausage, barked, …
  - PP (Prepositional phrases): with a friend, in a car, …



- **Nodes immediately above words are PoS tags (aka preterminals)**
  - PN: pronoun
  - D: determiner
  - V: verb
  - N: noun
  - P: preposition

Context-free grammars (CFGs)

- A context-free grammar is a 4-tuple <*N, T, S, R*>
  - ***N* : the set of non-terminals**
    - Phrasal categories: S, NP, VP, ADJP, etc.
    - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
  - ***T* : the set of terminals (the words)**
  - ***S* : the start symbol**
    - Often written as ROOT or TOP
    - *Not* usually the sentence non-terminal S
  - ***R* : the set of rules**
    - Of the form $X \rightarrow Y_1 Y_2 \ldots Y_k$, with $X, Y_i \in N$
    - Examples: $S \rightarrow NP\ VP$, $VP \rightarrow VP\ CC\ VP$
    - Also called rewrites, productions, or local trees

An example grammar

$N = \{S, VP, NP, PP, N, V, PN, P\}$

$T = \{girl, telescope, sandwich, I, saw, ate, with, in, a, the\}$

$S = \{S\}$

R :

Called **Inner rules**

$S \rightarrow NP\ \ VP$ (NP A girl) (VP ate a sandwich)

$VP \rightarrow V$

$VP \rightarrow V\ \ NP$ (V ate) (NP a sandwich)

$VP \rightarrow VP\ \ PP$ (VP saw a girl) (PP with a telescope)

$NP \rightarrow NP\ \ PP$ (NP a girl) (PP with a sandwich)

$NP \rightarrow D\ \ N$ (D a) (N sandwich)

$NP \rightarrow PN$

$PP \rightarrow P\ \ NP$ (P with) (NP with a sandwich)

Preterminal rules

$N \rightarrow girl$

$N \rightarrow telescope$

$N \rightarrow sandwich$

$PN \rightarrow I$

$V \rightarrow saw$

$V \rightarrow ate$

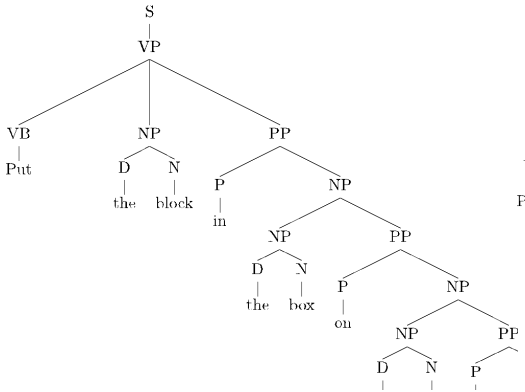
$P \rightarrow with$

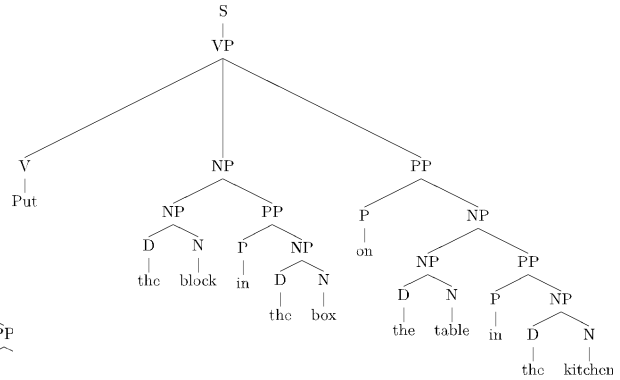
$P \rightarrow in$

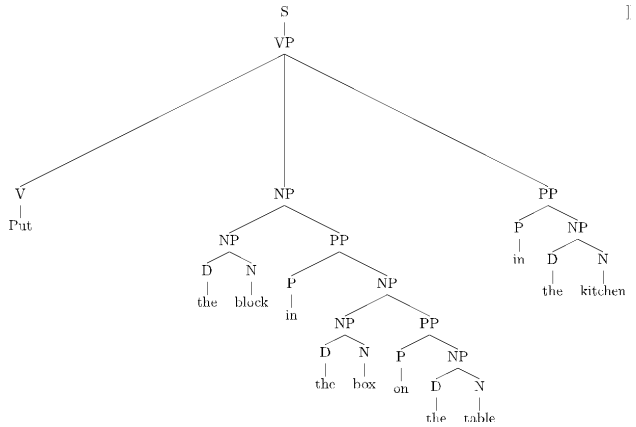
$D \rightarrow a$

$D \rightarrow the$

How to Deal with Ambiguity?







Put the block in the box on the table in the kitchen

- We want to score all the derivations to encode how plausible they are.

Probabilistic context-free grammars (PCFGs)

- A context-free grammar is a 4-tuple <*N, T, S, R*>
  - ***N* : the set of non-terminals**
    - Phrasal categories: S, NP, VP, ADJP, etc.
    - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
  - ***T* : the set of terminals (the words)**
  - ***S* : the start symbol**
    - Often written as ROOT or TOP
    - *Not* usually the sentence non-terminal S
  - ***R* : the set of rules**
    - Of the form $X \rightarrow Y_1\ Y_2 \ldots Y_k$, with $X, Y_i \in N$
    - Examples: $S \rightarrow NP\ VP$, $VP \rightarrow VP\ CC\ VP$
    - Also called rewrites, productions, or local trees
- A PCFG adds:
  - A top-down production probability per rule $P(Y_1\ Y_2 \ldots Y_k \mid X)$

An example PCFGs

Associate probabilities with the rules : $p(X \rightarrow \alpha)$

$$\forall\ X \rightarrow \alpha \in R: \quad 0 \leq p(X \rightarrow \alpha) \leq 1$$

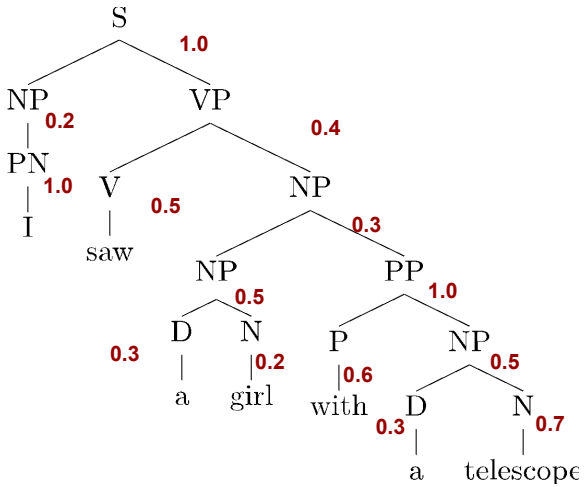
$$\forall X \in N: \quad \sum_{\alpha: X \rightarrow \alpha \in R} p(X \rightarrow \alpha) = 1$$

Now we can score a tree as a product of probabilities corresponding to the used rules

$S \rightarrow NP\ VP$	1.0	(NP A girl) (VP ate a sandwich)
$VP \rightarrow V$	0.2	
$VP \rightarrow V\ NP$	0.4	(VP ate) (NP a sandwich)
$VP \rightarrow VP\ PP$	0.4	(VP saw a girl) (PP with …)
$NP \rightarrow NP\ PP$	0.3	(NP a girl) (PP with ….)
$NP \rightarrow D\ N$	0.5	(D a) (N sandwich)
$NP \rightarrow PN$	0.2	
$PP \rightarrow P\ NP$	1.0	(P with) (NP with a sandwich)

$N \rightarrow girl$	0.2
$N \rightarrow telescope$	0.7
$N \rightarrow sandwich$	0.1
$PN \rightarrow I$	1.0
$V \rightarrow saw$	0.5
$V \rightarrow ate$	0.5
$P \rightarrow with$	0.6
$P \rightarrow in$	0.4
$D \rightarrow a$	0.3
$D \rightarrow the$	0.7

PCFGs



$S \rightarrow NP\ VP$ 1.0

$VP \rightarrow V$ 0.2
$VP \rightarrow V\ NP$ 0.4
$VP \rightarrow VP\ PP$ 0.4

$NP \rightarrow NP\ PP$ 0.3
$NP \rightarrow D\ N$ 0.5
$NP \rightarrow PN$ 0.2

$PP \rightarrow P\ NP$ 1.0

$N \rightarrow girl$ 0.2
$N \rightarrow telescope$ 0.7
$N \rightarrow sandwich$ 0.1
$PN \rightarrow I$ 1.0
$V \rightarrow saw$ 0.5
$V \rightarrow ate$ 0.5
$P \rightarrow with$ 0.6
$P \rightarrow in$ 0.4
$D \rightarrow a$ 0.3
$D \rightarrow the$ 0.7

$P(T) = 1.0 * 0.2 * 1.0 * 0.4 * 0.5 * 0.3 * 0.5 * 0.3 * 0.2 * 1.0 * 0.6 * 0.5 * 0.3 * 0.7 = 2.26e\text{-}5$

CKY Parsing

Parsing

- Parsing is search through the space of all possible parses
  - e.g., we may want either any parse, all parses or the highest scoring parse (if PCFG):

$$\underset{T \in G(x)}{\text{argmax}}\ P(T)$$

- Bottom-up
  - One starts from words and attempt to construct the full tree

- Top-down
  - Start from the start symbol and attempt to expand to get the sentence

CKY algorithm (aka CYK)

- Cocke-Kasami-Younger algorithm
  - Independently discovered in late 60s / early 70s

- An efficient bottom up parsing algorithm for (P)CFGs
  - can be used both for the recognition and parsing problems
  - Very important in NLP

- We will start with the non-probabilistic version

Constraints on the grammar

- The basic CKY algorithm supports only rules in the Chomsky Normal Form (CNF):

$$C \rightarrow x$$

$$C \rightarrow C_1 C_2$$

Unary preterminal rules (generation of words given PoS tags)

$N \rightarrow telescope$ $D \rightarrow the$

Binary inner rules $S \rightarrow NP\ VP$ $NP \rightarrow D\ N$

Constraints on the grammar

- The basic CKY algorithm supports only rules in the Chomsky Normal Form (CNF):

$$C \rightarrow x$$

$$C \rightarrow C_1 C_2$$

- Any CFG can be converted to an equivalent CNF
  - Equivalent means that they define the same language
  - However (syntactic) trees will look differently
  - It is possible to address it by defining such transformations that allows for easy reverse transformation

Transformation to CNF form

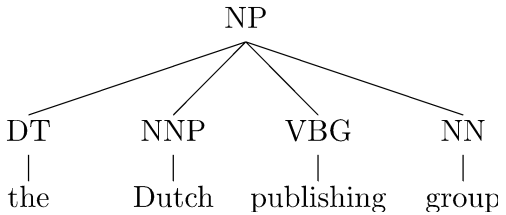
- What one need to do to convert to CNF form

  - Get rid of rules that mix terminals and non-terminals
  - Get rid of unary rules: $C \to C_1$
  - Get rid of N-ary rules: $C \to C_1\ C_2 \ldots C_n\ \ (n > 2)$

Crucial to process them, as required for efficient parsing

Transformation to CNF form: binarization

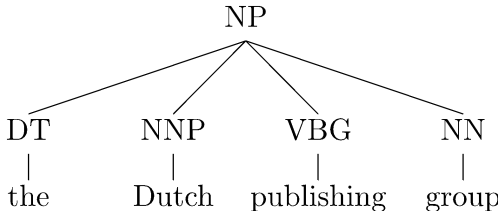
- Consider *NP→DT NNP VBG NN*



- How do we get a set of binary rules which are equivalent?

Transformation to CNF form: binarization

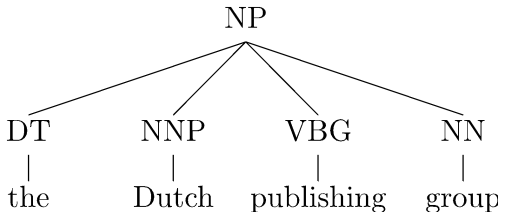
- Consider *NP→DT NNP VBG NN*



- How do we get a set of binary rules which are equivalent?
  NP→DT X
  X→NNP Y
  Y→VBG NN

Transformation to CNF form: binarization

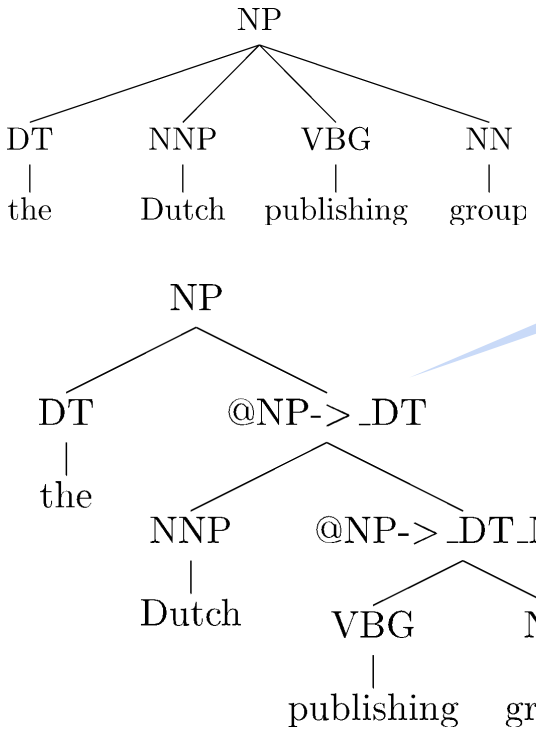
- Consider *NP→DT NNP VBG NN*



- How do we get a set of binary rules which are equivalent?
  NP→DT X
  X→NNP Y
  Y→VBG NN
- A more systematic way to refer to new non-terminals
  NP→DT @NP|_DT
  @NP|_DT → NNP @NP|_DT_NNP
  @NP|_DT_NNP → VBG NN

Transformation to CNF form: binarization

- Instead of binarizing tuples we can binarize trees on preprocessing:



Also known as **lossless Markovization** in the context of PCFGs

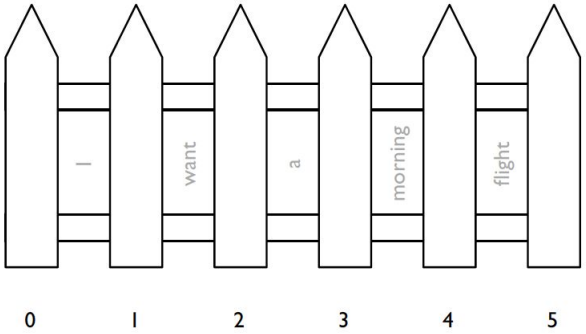
Can be easily reversed on post-processing

CKY: Parsing task

- We are given
  - a grammar <N, T, S, R>
  - a sequence of words $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$
- **Goal** is to produce a parse tree for w

CKY: Parsing task

- We are given
  - a grammar <N, T, S, R>
  - a sequence of words $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$
- **Goal** is to produce a parse tree for w
- We need an easy way to refer to substrings of w



indices refer to fenceposts

span (i, j) refers to words between fenceposts *i* and *j*

Parsing one word

$$C \rightarrow w_i$$

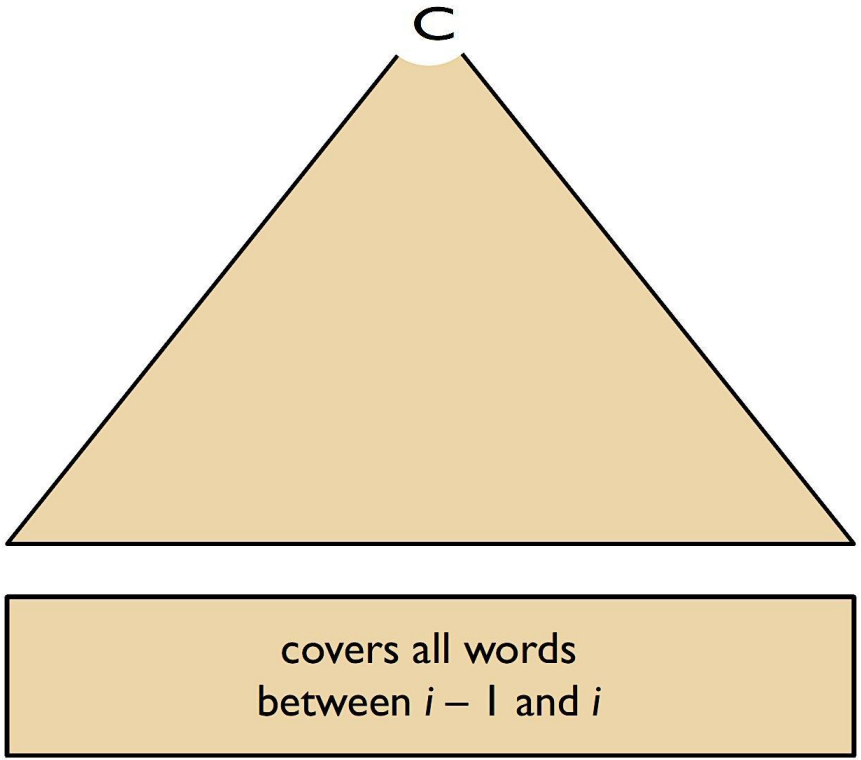
w_i

Parsing one word

$C \rightarrow w_i$

Parsing one word

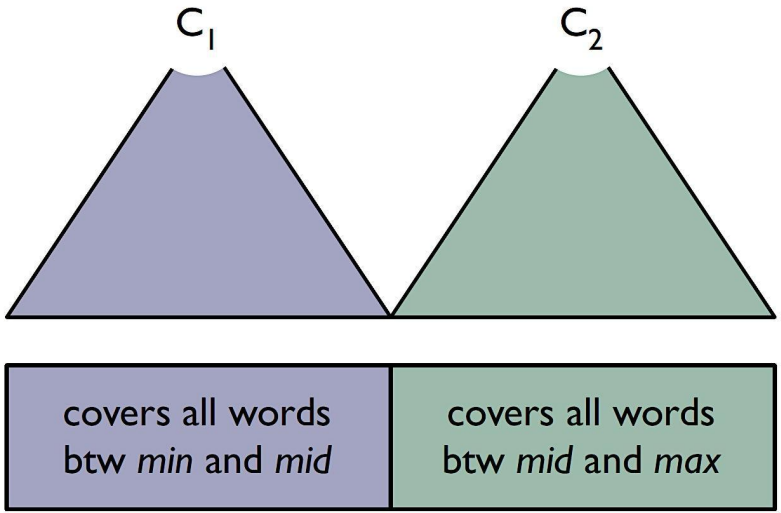
$C \rightarrow w_i$



Parsing one word

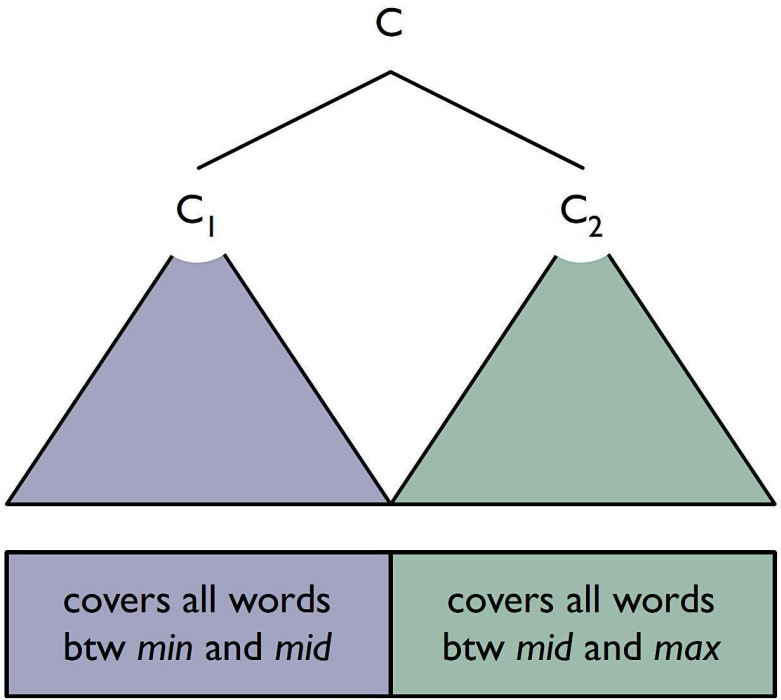
$$C \rightarrow C_1 \;\; C_2$$

Check through all C1, C2, **mid**



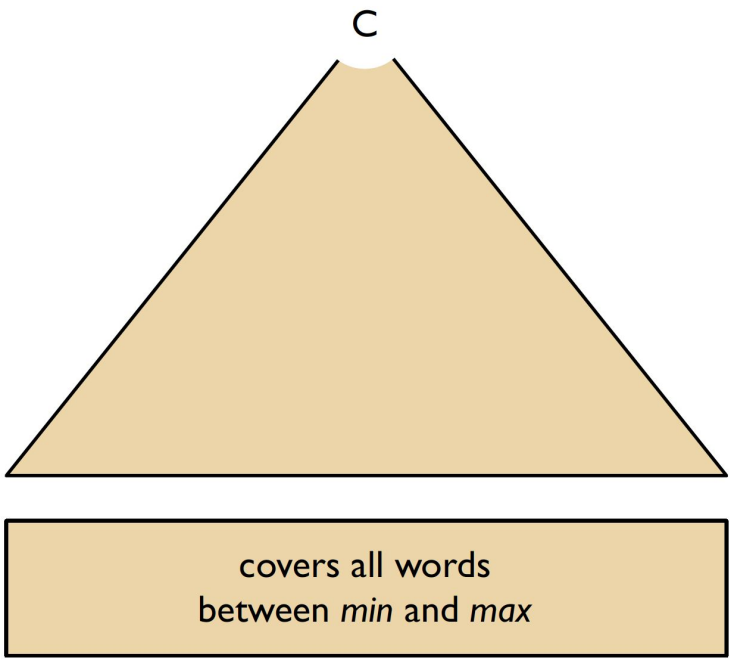
Parsing one word

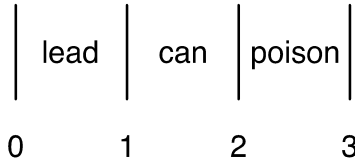
$$C \rightarrow C_1 \;\; C_2$$



Check through all
C1, C2, **mid**

Parsing one word





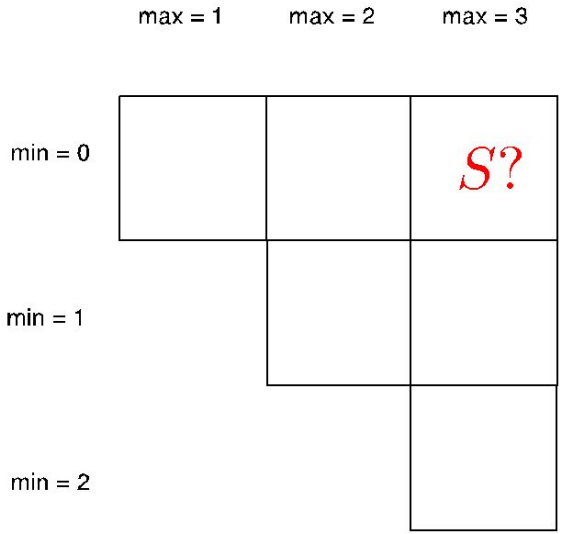


Chart (aka parsing triangle)

$S \rightarrow NP\ VP$

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N\ NP$

Inner rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

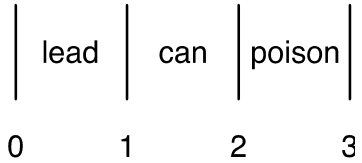
$M \rightarrow can$

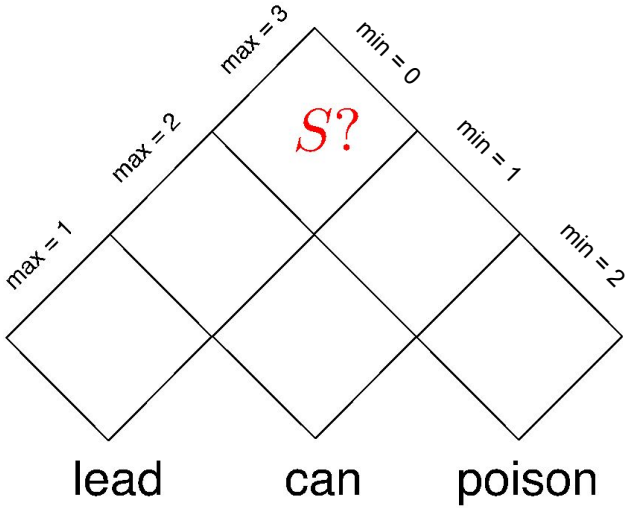
$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Preterminal rules





$$S \rightarrow NP \ VP$$

$$VP \rightarrow M \ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N \ NP$$

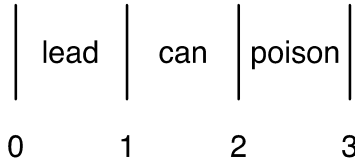
Inner rules

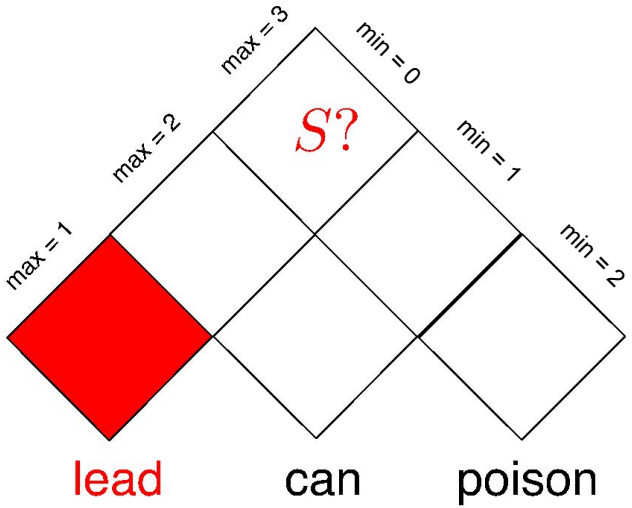
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \rightarrow NP \;\; VP$$

$$VP \rightarrow M \;\; V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N \;\; NP$$

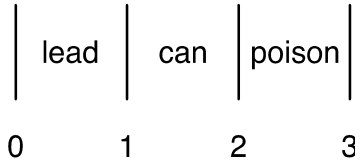
Inner rules

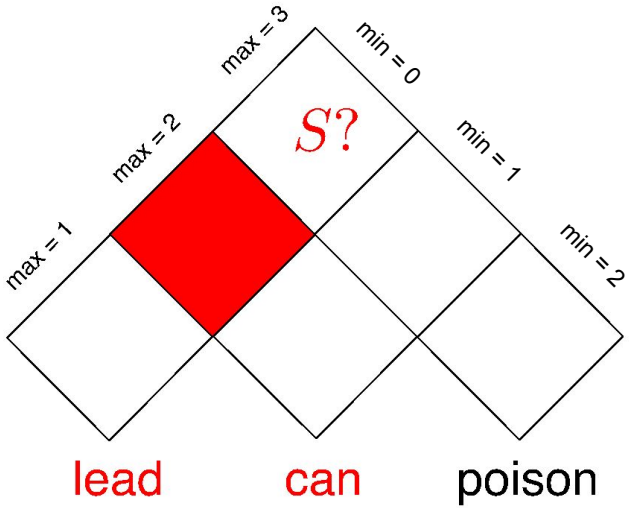
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \rightarrow NP \ VP$$

$$VP \rightarrow M \ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N \ NP$$

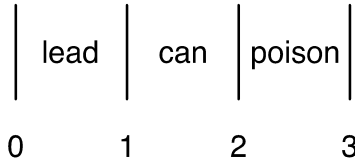
Inner rules

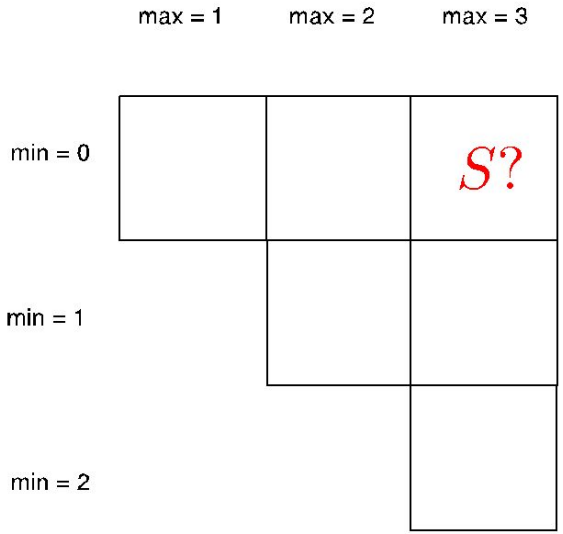
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

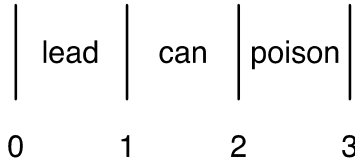
Inner rules

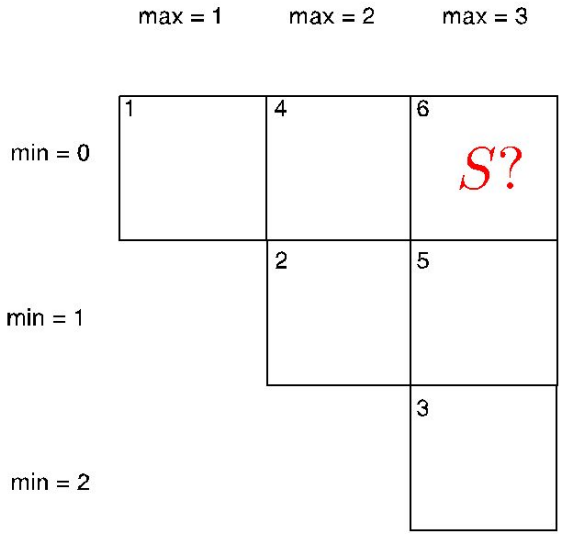
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

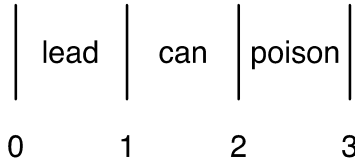
Inner rules

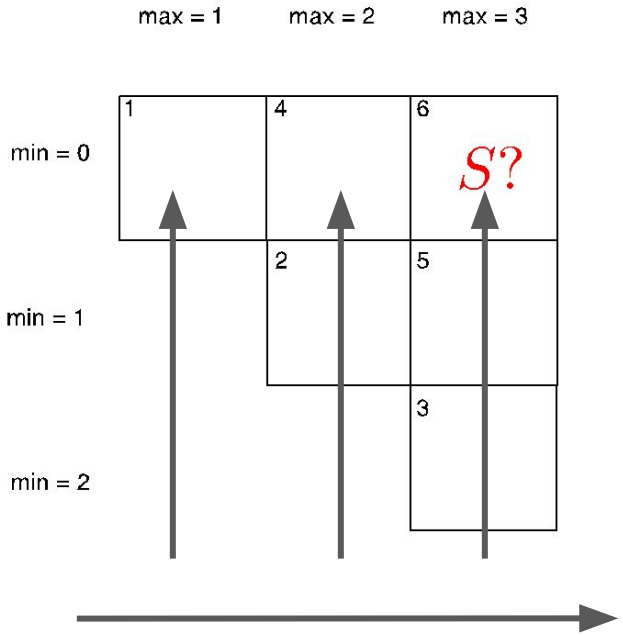
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

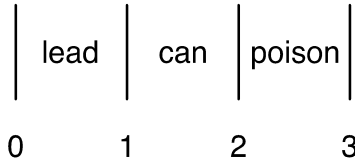
Inner rules

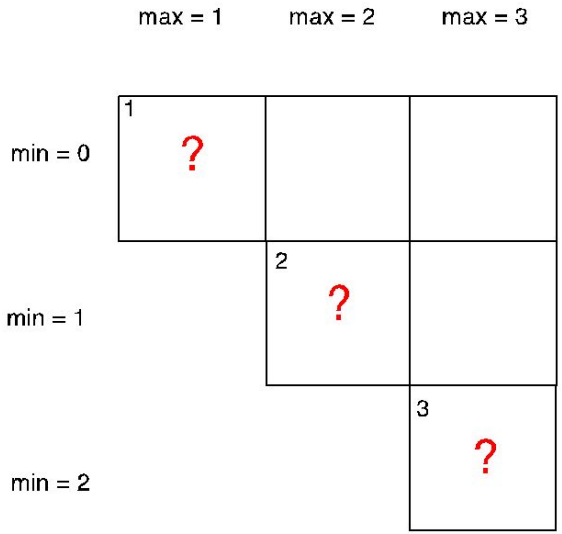
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

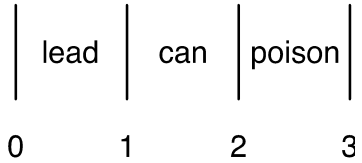
$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

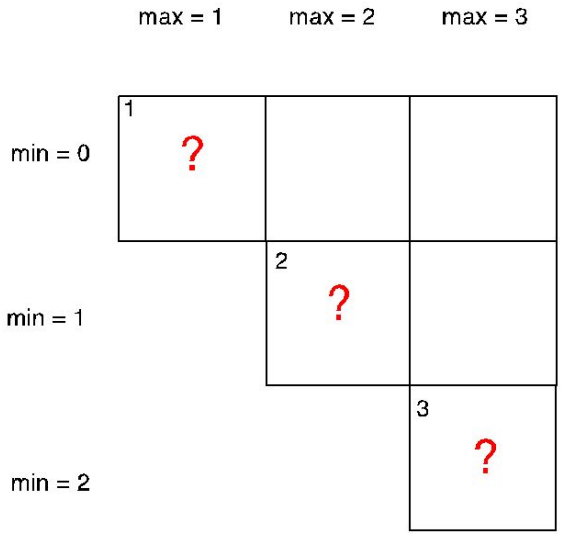
Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$





$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

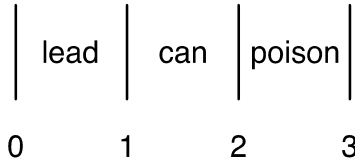
Inner rules

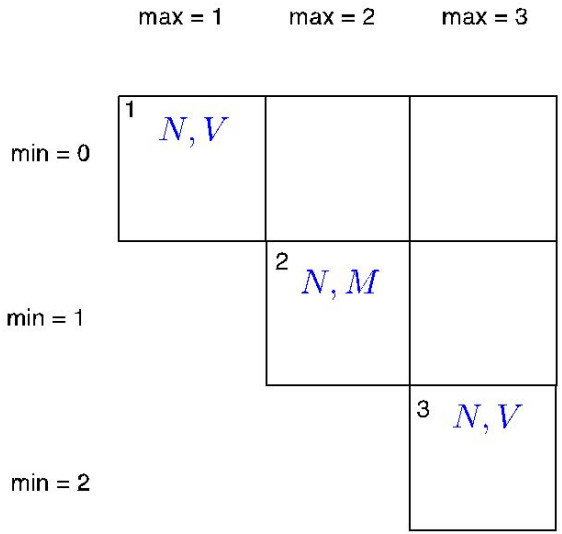
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$S \rightarrow NP\ VP$

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N\ NP$

Inner rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

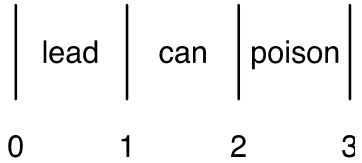
$M \rightarrow can$

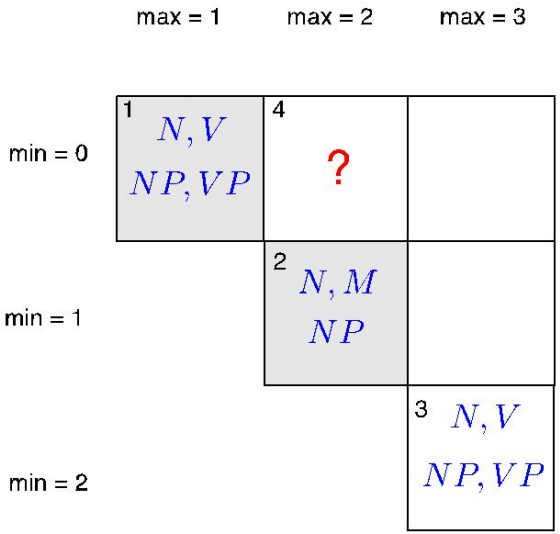
$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Preterminal rules





$S \rightarrow NP\ VP$

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N\ NP$

Inner rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

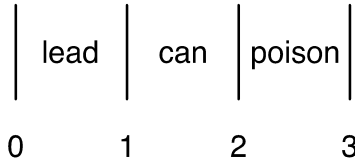
$M \rightarrow can$

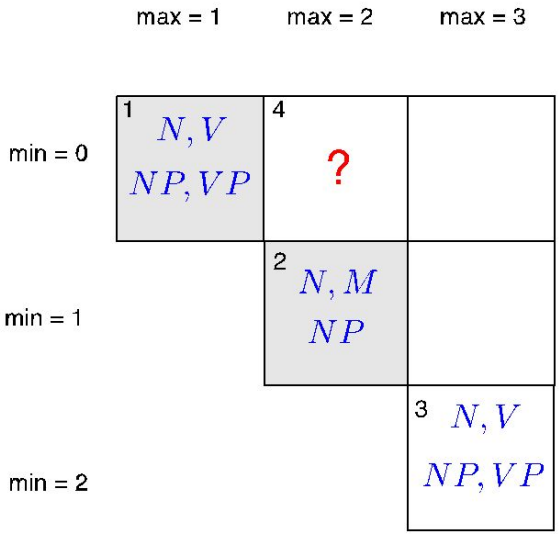
$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Preterminal rules





$S \rightarrow NP\ VP$

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N\ NP$

Inner rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

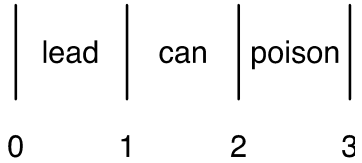
$M \rightarrow can$

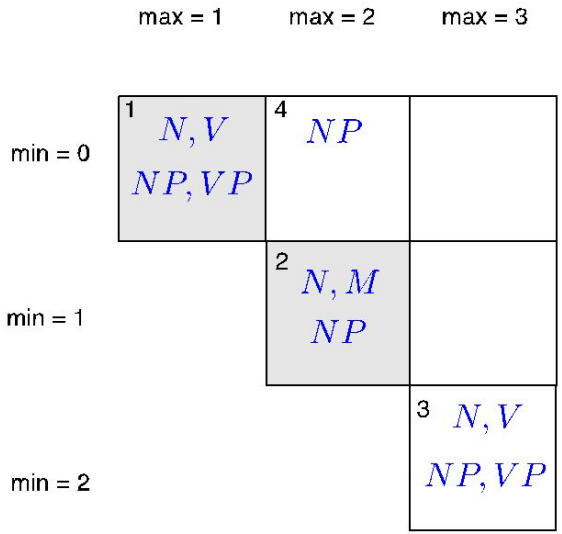
$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Preterminal rules





$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

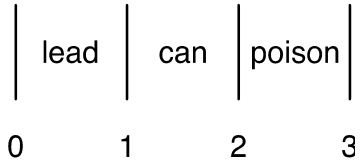
Inner rules

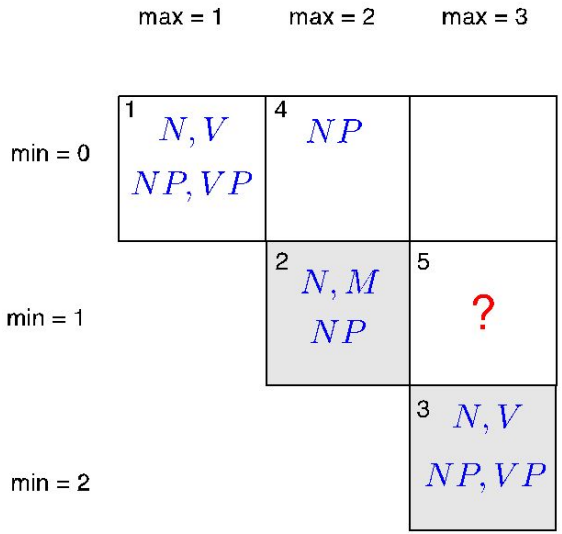
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$S \rightarrow NP\ VP$

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N\ NP$

Inner rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

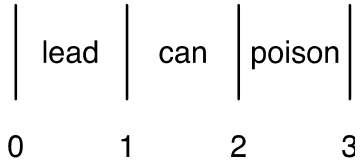
$M \rightarrow can$

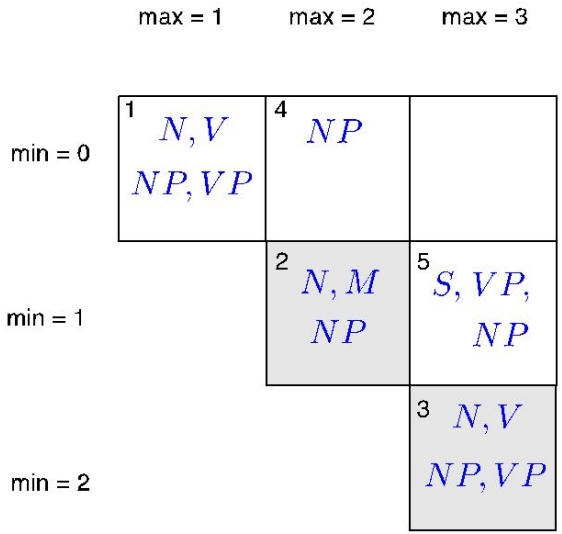
$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Preterminal rules





$S \rightarrow NP\ VP$

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N\ NP$

Inner rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

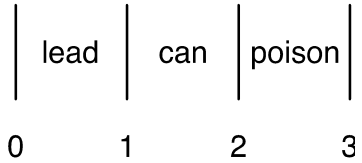
$M \rightarrow can$

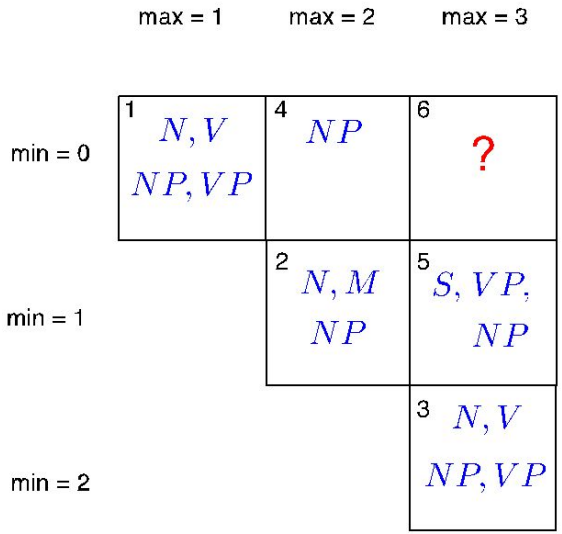
$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Preterminal rules





$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

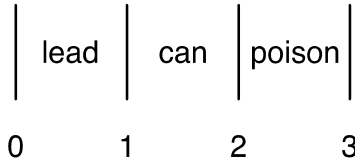
$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

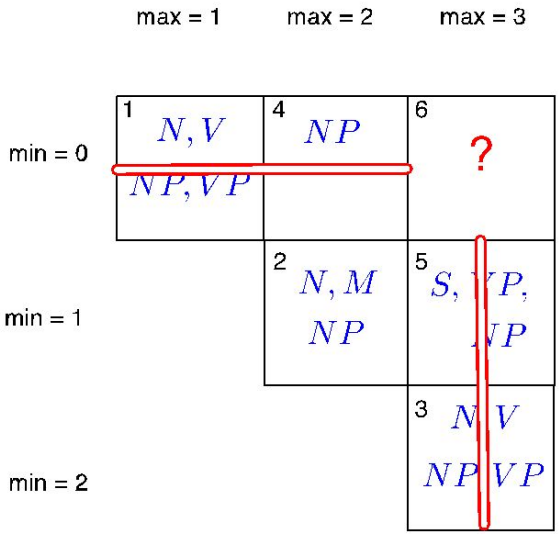
Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$





$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

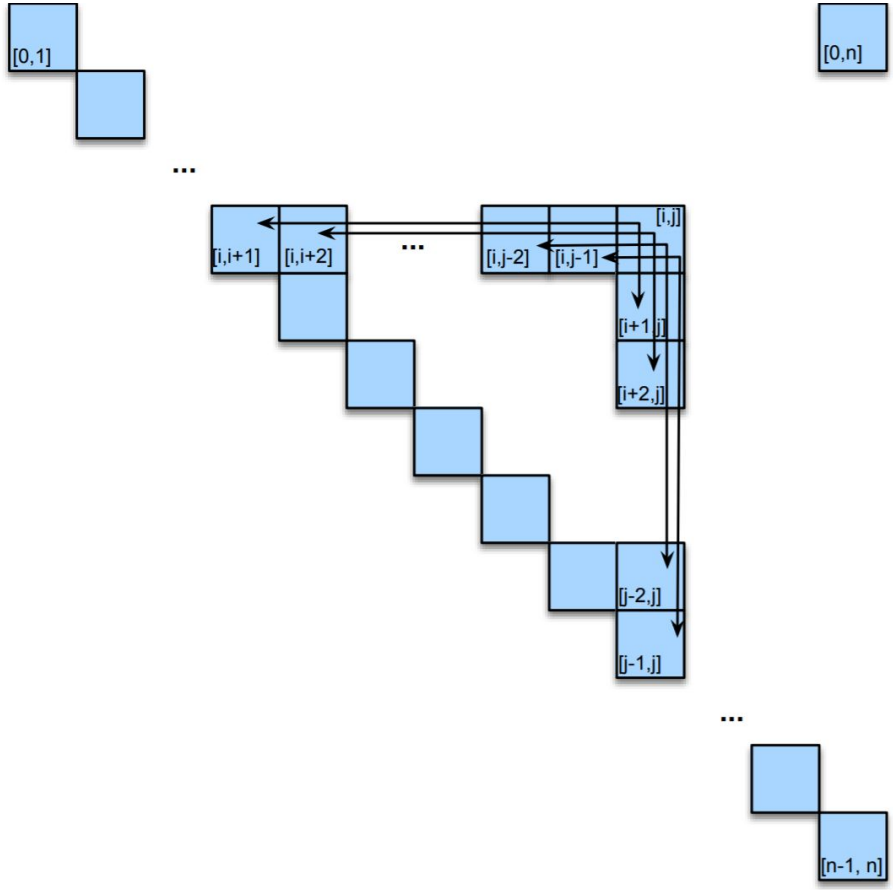
$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

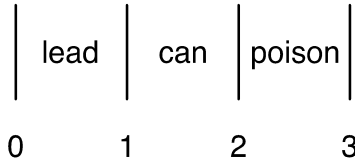
Preterminal rules

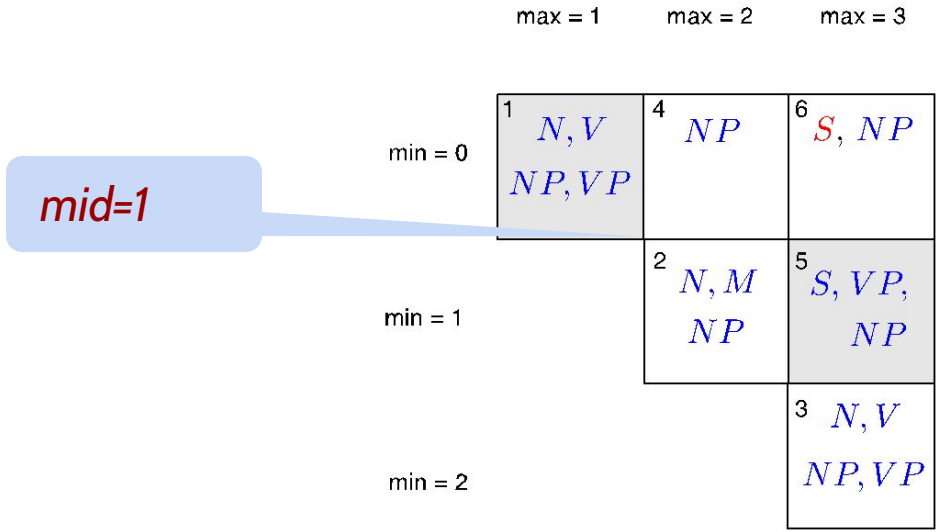
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$


[0,1]
[0,n]
...
[i,i+1]
[i,i+2]
...
[i,j-2]
[i,j-1]
[i,j]
[i+1,j]
[i+2,j]
[j-2,j]
[j-1,j]
...
[n-1, n]





$S \rightarrow NP\ VP$

$VP \rightarrow M\ V$

$VP \rightarrow V$

Inner rules

$NP \rightarrow N$

$NP \rightarrow N\ NP$

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

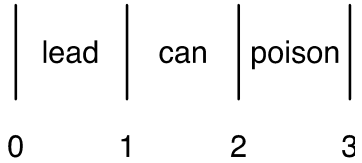
$M \rightarrow can$

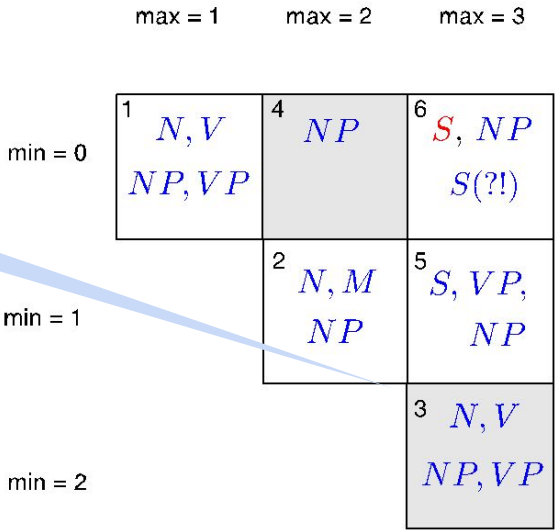
$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Preterminal rules





mid=2

$S \rightarrow NP\ VP$

Inner rules

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N\ NP$

Preterminal rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

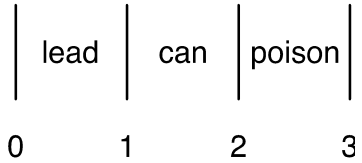
$M \rightarrow can$

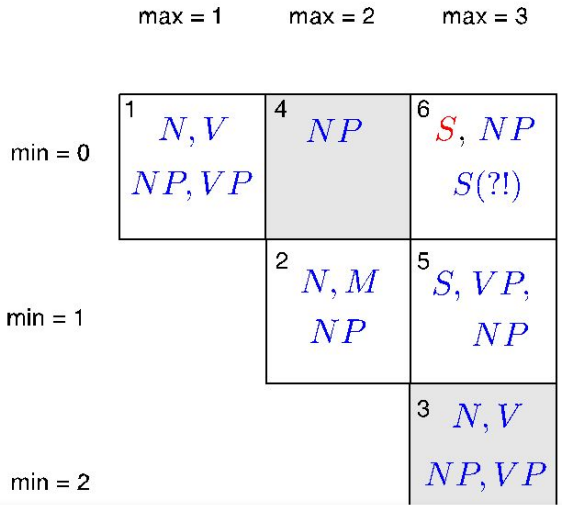
$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

$S \rightarrow NP\ VP$





Apparently the sentence is ambiguous for the grammar: (as the grammar overgenerates)

Inner rules

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N\ NP$

Preterminal rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

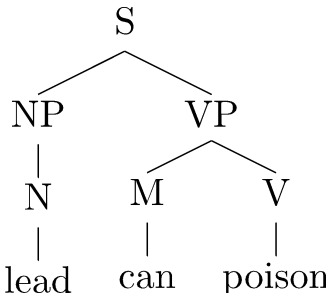
$M \rightarrow can$

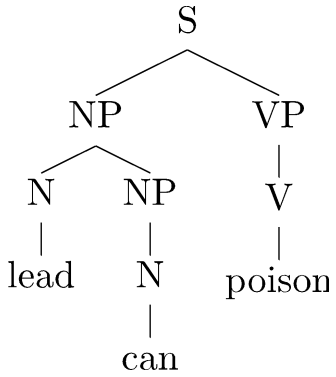
$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Ambiguity

- 



No subject-verb agreement, and *poison* used as an intransitive verb

CKY implementation

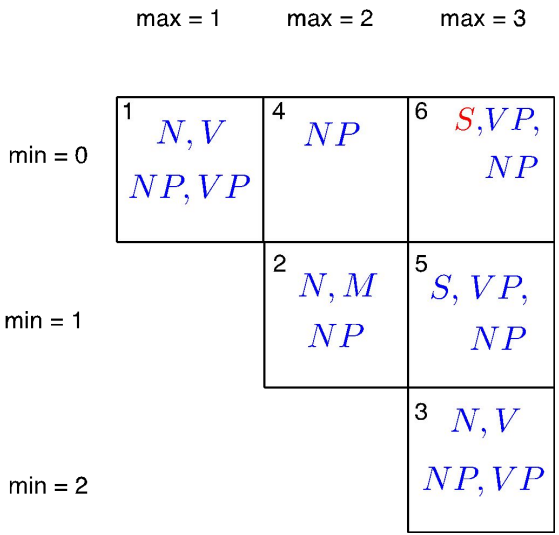
Here we assume that labels (C) are integer indices

Chart can be represented by a Boolean 3D array `chart[min][max][C]`

Relevant entries have $0 < min < max \leq n$

`chart[min][max][C] = true` if the signature (min, max, C) is already added to the chart;

`false` otherwise.

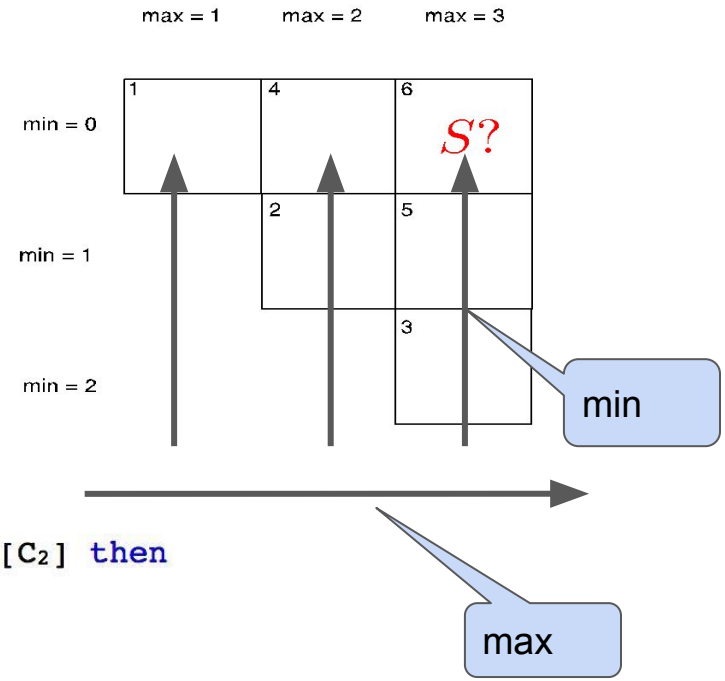


Implementation: preterminal rules

```
for each wi from left to right
  for each preterminal rule C -> wi
    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
```



Algorithm analysis

Time complexity?

```
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
```

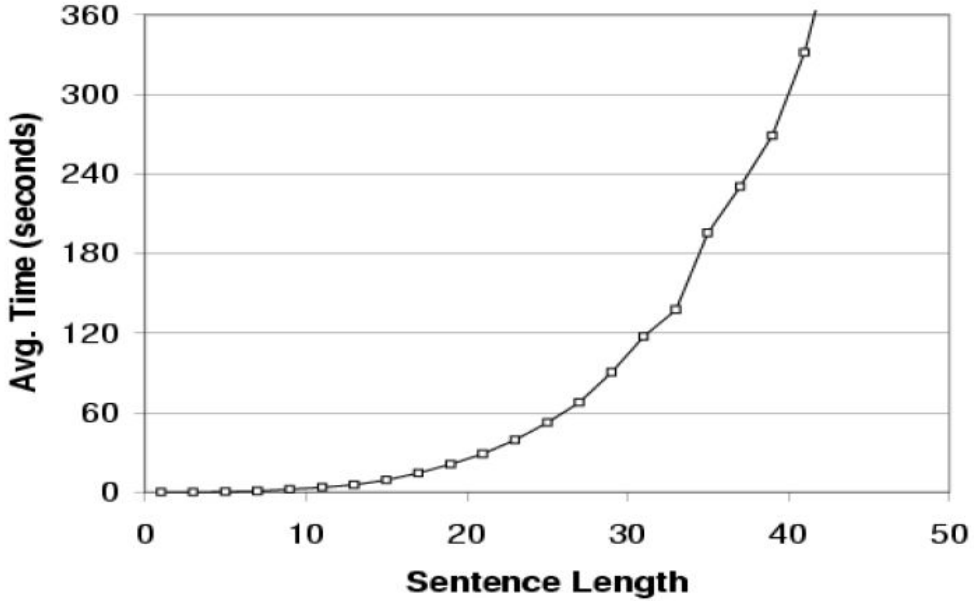
Algorithm analysis

Time complexity?

```
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
```

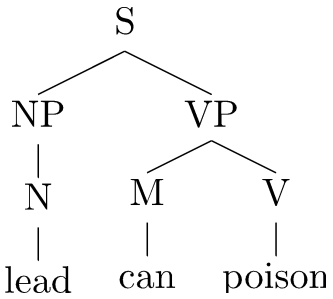
$O(n^3|R|)$ where **$|R|$** is is the number of rules in the grammar

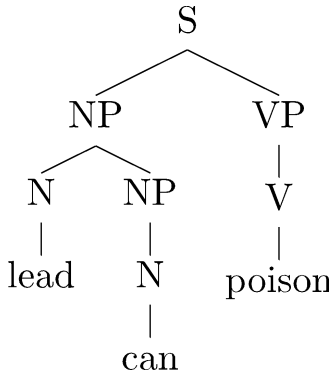
Practical time complexity



$\sim n^{3.6}$

Ambiguity

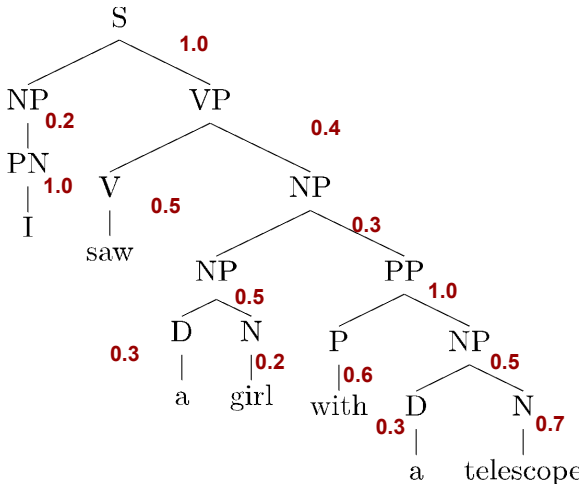
- 



No subject-verb agreement, and *poison* used as an intransitive verb

CKY Parsing with PCFGs

PCFGs



$S \rightarrow NP\ VP$ 1.0

$VP \rightarrow V$ 0.2
$VP \rightarrow V\ NP$ 0.4
$VP \rightarrow VP\ PP$ 0.4

$NP \rightarrow NP\ PP$ 0.3
$NP \rightarrow D\ N$ 0.5
$NP \rightarrow PN$ 0.2

$PP \rightarrow P\ NP$ 1.0

$N \rightarrow girl$ 0.2
$N \rightarrow telescope$ 0.7
$N \rightarrow sandwich$ 0.1
$PN \rightarrow I$ 1.0
$V \rightarrow saw$ 0.5
$V \rightarrow ate$ 0.5
$P \rightarrow with$ 0.6
$P \rightarrow in$ 0.4
$D \rightarrow a$ 0.3
$D \rightarrow the$ 0.7

$P(T) = 1.0 * 0.2 * 1.0 * 0.4 * 0.5 * 0.3 * 0.5 * 0.3 * 0.2 * 1.0 * 0.6 * 0.5 * 0.3 * 0.7 = 2.26e\text{-}5$

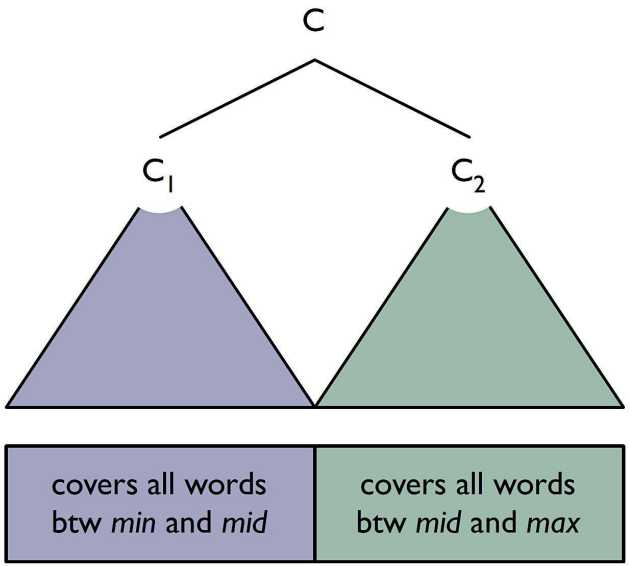
CKY with PCFGs

- Chart is represented by a 3d array of **floats**
  `chart[min][max][label]`
  - It stores probabilities for **the most probable subtree with a given signature**

- `chart[0][n][S]` will store the probability of **the most probable full parse tree**

Intuition

$$C \rightarrow C_1 \ C_2$$



For every C choose C_1, C_2 and mid such that

$$P(T_1) \times P(T_2) \times P(C \rightarrow C_1 C_2)$$

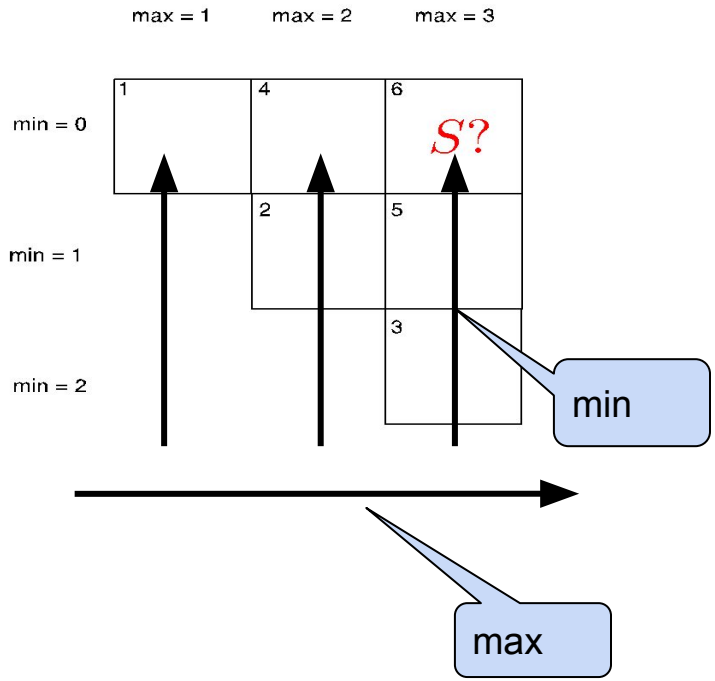
is maximal, where T_1 and T_2 are left and right subtrees.

Implementation: preterminal rules

```
for each w_i from left to right
  for each preterminal rule C -> w_i
    chart[i - 1][i][C] = p(C -> w_i)
```

Implementation: 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
```



Recovery of the tree

- For each signature we store backpointers to the elements from which it was built
  - start recovering from [0, n, S]

- What backpointers do we store?

Recovery of the tree

- For each signature we store backpointers to the elements from which it was built
  - start recovering from [0, n, S]

- What backpointers do we store?
  - rule
  - for binary rules, midpoint

Parsing evaluation

- **Intrinsic evaluation:**
  - Automatic: evaluate against annotation provided by human experts (gold standard) according to some predefined measure
  - Manual: … according to human judgment

- **Extrinsic evaluation:** score syntactic representation by comparing how well a system using this representation performs on some task
  - E.g., use syntactic representation as input for a semantic analyzer and compare results of the analyzer using syntax predicted by different parsers.

Standard evaluation setting in parsing

- **Automatic intrinsic evaluation** is used: parsers are evaluated against gold standard by provided by linguists
  - There is a standard split into the parts:
    - training set: used for estimation of model parameters
    - development set: used for tuning the model (initial experiments)
    - test set: final experiments to compare against previous work

Automatic evaluation of constituent parsers

- **Exact match:** percentage of trees predicted correctly

- **Bracket score:** scores how well individual phrases (and their boundaries) are identified

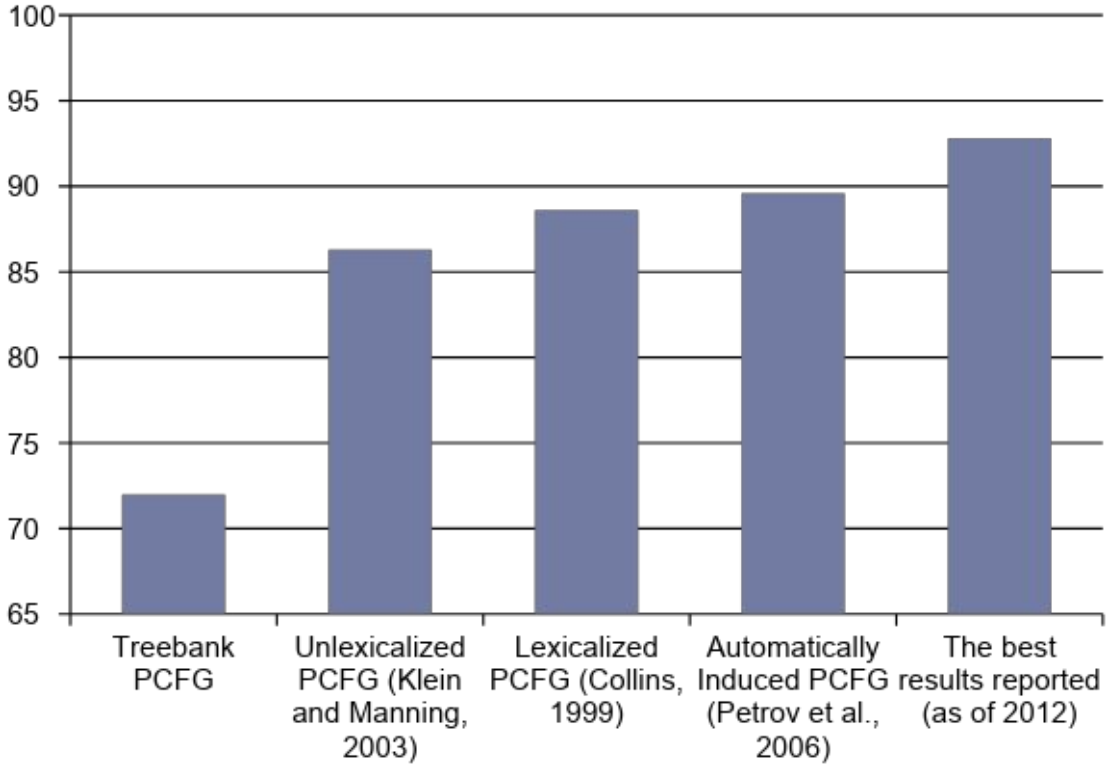
The most standard measure; we will focus on it

Brackets scores

- The most standard score is bracket score
- It regards a tree as a collection of brackets: $[min, max, C]$
- The set of brackets predicted by a parser is compared against the set of brackets in the tree annotated by a linguist
- Precision, recall and F1 are used as scores

Subtree signatures for CKY

F1 bracket score



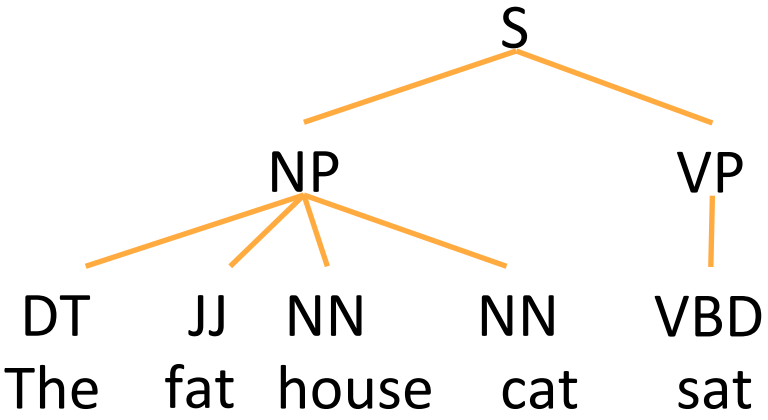
Improvements to CKY

- Tree binarization
- Relaxing independence assumptions
- Speeding up
- Additional grammar extensions

Binarization

Treebank PCFGs

- We can take a grammar straight off a tree, using counts to estimate probabilities



$S \rightarrow NP\ VP$ 1

$NP \rightarrow DT\ JJ\ NN\ NN$ 1

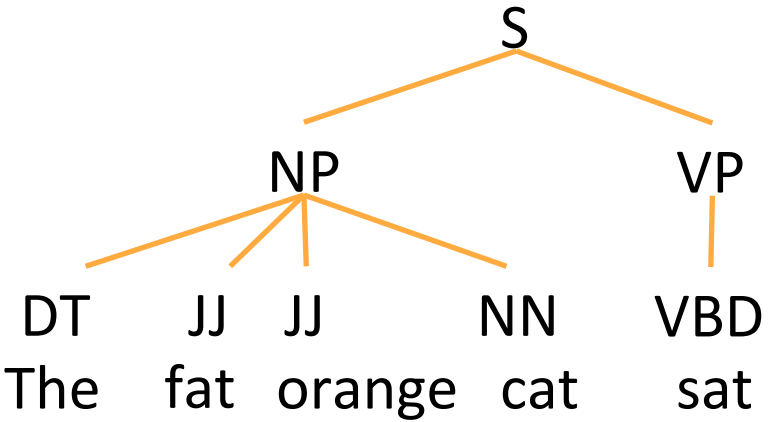
$VP \rightarrow VBD$ 1

…..

- Can we use CKY to parse sentences according to this grammar?

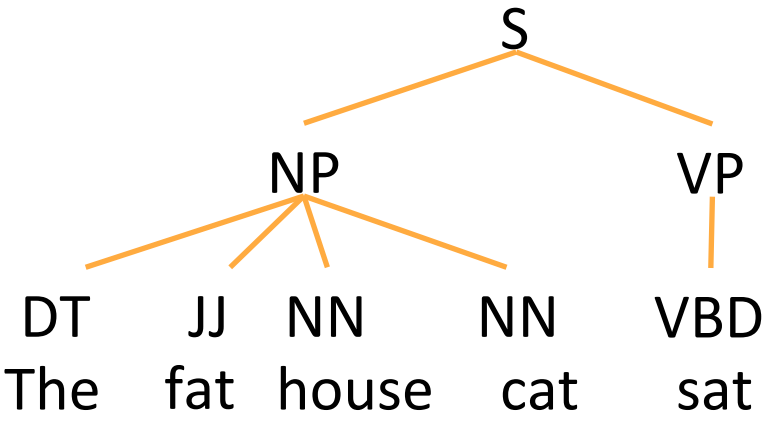
Treebank PCFGs

- We can take a grammar straight off a tree, using counts to estimate probabilities



- Vanilla CKY only allows *binary* rules

Option 1: Binarize the Grammar



S→ NP VP

NP→ DT JJ NN NN

VP→ VBD

S→ NP VP

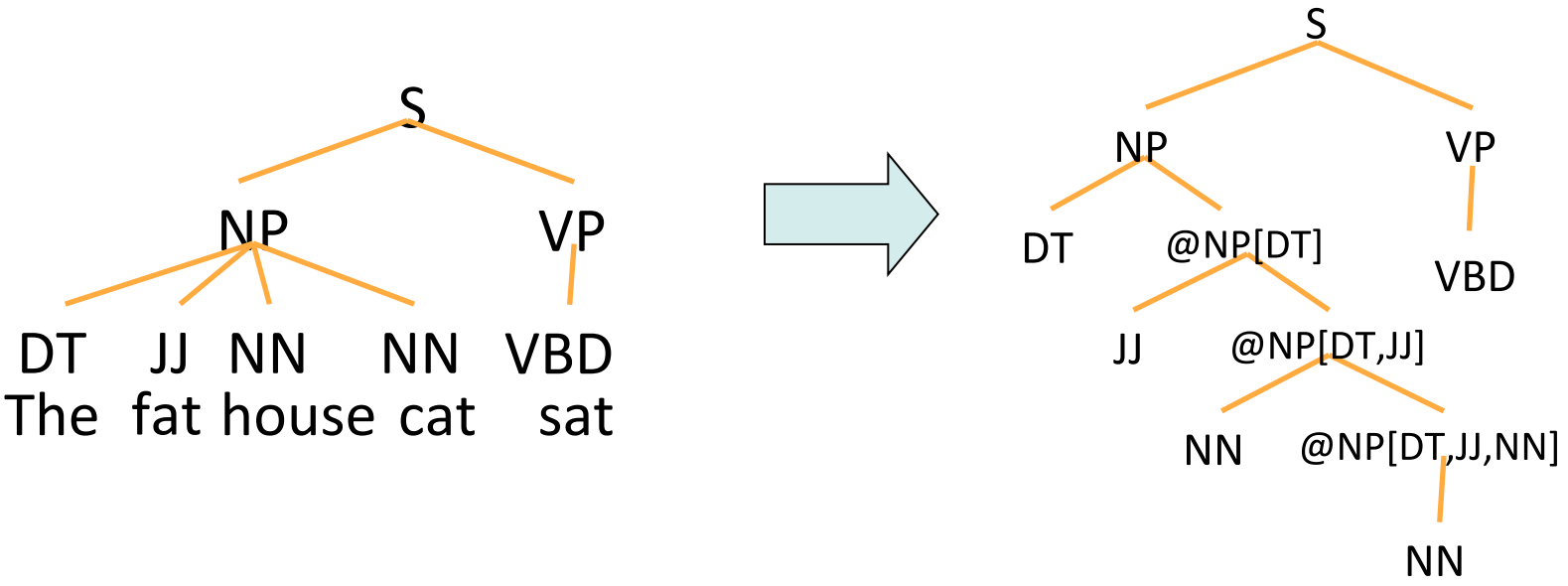
S→ NP VBD

NP→ DT @NP[DT]

@NP[DT]→ JJ @NP[DT JJ]

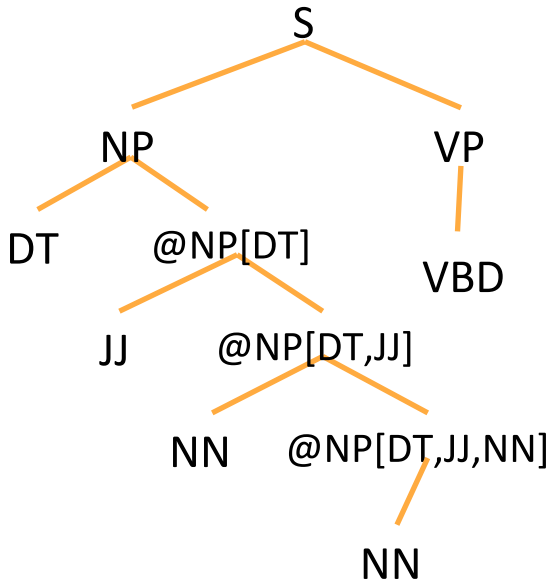
@NP[DT JJ]→ NN NN

Option 2: Binarize the Tree



- Can we use CKY to parse sentences according to the grammar pulled from this tree?

CKY: Modifications for Unary Rules



Binary Rules:

S→ NP VP

NP→ DT @NP[DT]

@NP[DT]→ JJ @NP[DT JJ]

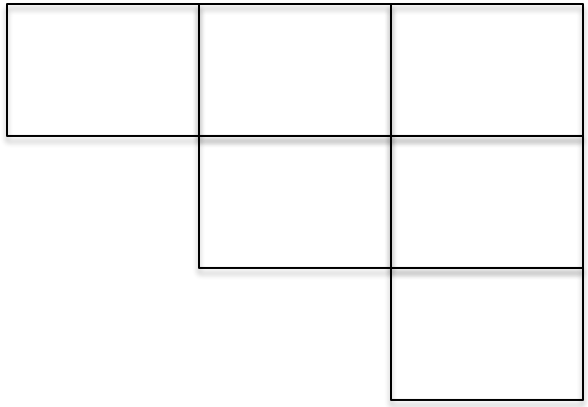
@NP[DT JJ]→ NN @NP[DT,JJ,NN]

Unary Rules:

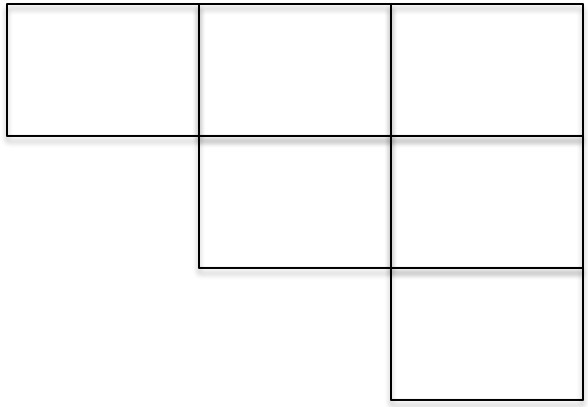
VP→ VBD

@NP[DT,JJ,NN] → NN

CKY: Incorporate Unary Rules



- **Binary chart:** Store the scores of non-terminals after applying binary rules
- Fill by applying rules to elements of the unary chart



- **Unary chart:** Store the scores of non-terminals after apply unary rules
- Fill by applying rules to elements of the binary chart

CKY with TreeBank PCFG

[Charniak 96]

- With these modifications, given a treebank we can:
  - Binarize the trees
  - Learn a PCFG from the binarized trees
  - Use the unary-binary chart variant of CKY to obtain parse trees for new sentences
- Does this work?

Typical Experimental Setup

- Corpus: Penn Treebank, WSJ

Training:	sections	02-21
Development:	section	22 (here, first 20 files)
Test:	section	23

- F1 (Bracket scores): harmonic mean of per-node **labeled** precision and recall.
- Here: also size – number of symbols in grammar.

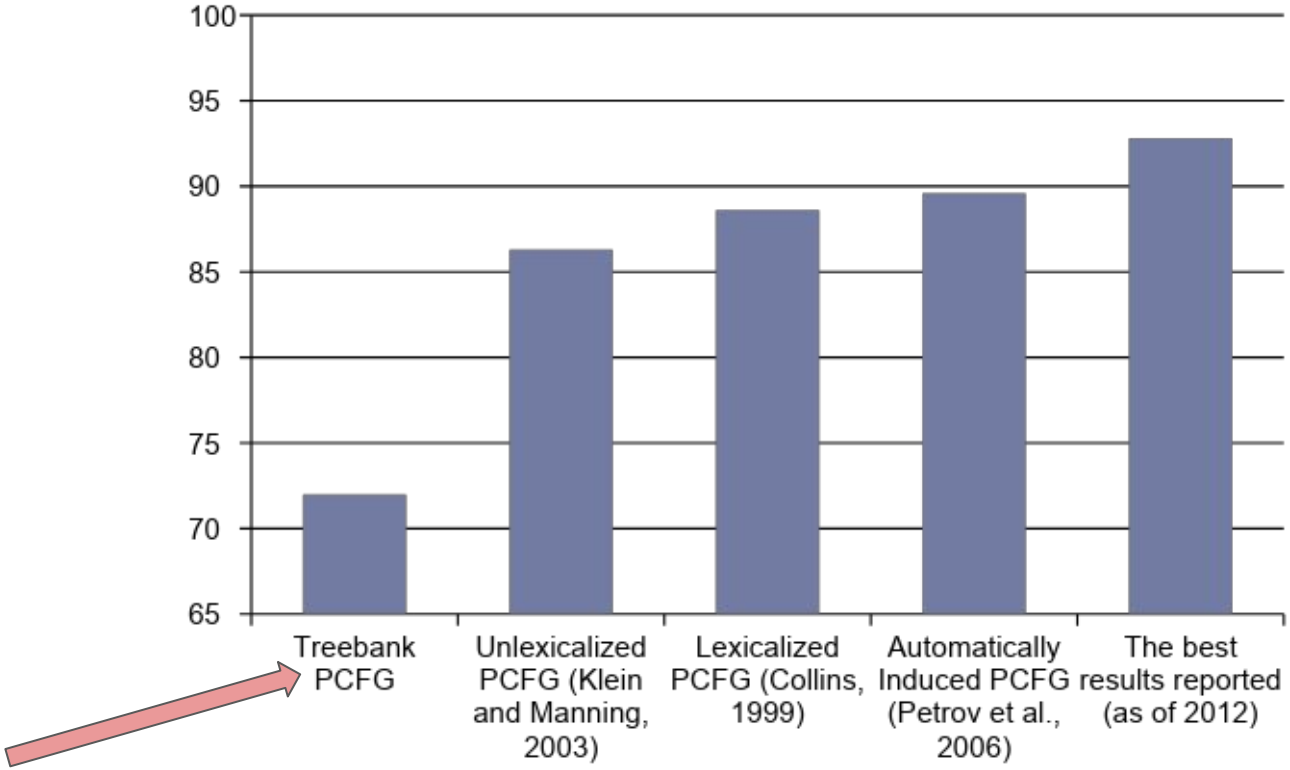
CKY with TreeBank PCFG

[Charniak 96]

- With these modifications, given a treebank we can:
  - Binarize the trees
  - Learn a PCFG from the binarized trees
  - Use the unary-binary chart variant of CKY to obtain parse trees for new sentences
- Does this work?

Model	*F1*
Baseline	72.0

F1 bracket score



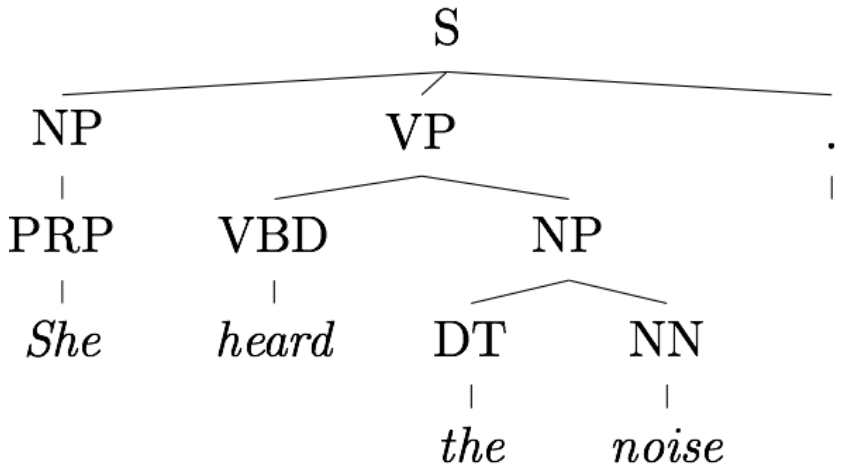
Model Assumptions

- Place Invariance
  - The probability of a subtree does not depend on where in the string the words it dominates are
- Context-free
  - The probability of a subtree does not depend on words not dominated by the subtree
- Ancestor-free
  - The probability of a subtree does not depend on nodes in the derivation outside the tree

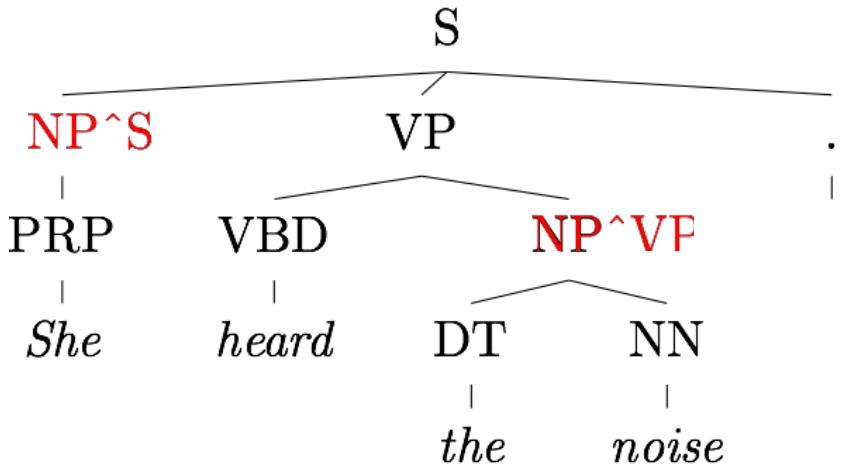
Model Assumptions

- We can relax some of these assumptions by *enriching* our grammar
  - We're already doing this in binarization
- Structured Annotation [Johnson '98, Klein&Manning '03]
  - Enrich with features about surrounding nodes
- Lexicalization [Collins '99, Charniak '00]
  - Enrich with word features
- Latent Variable Grammars [Matsuzaki et al. '05, Petrov et al. '06]

Grammar Refinement

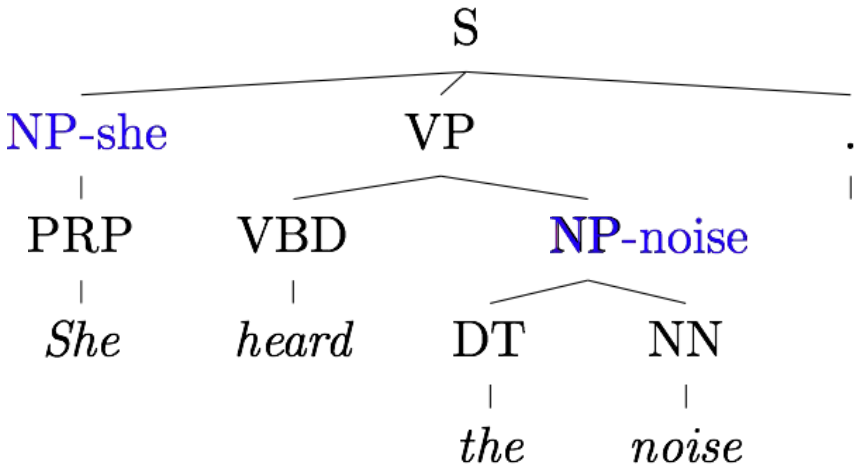


Grammar Refinement



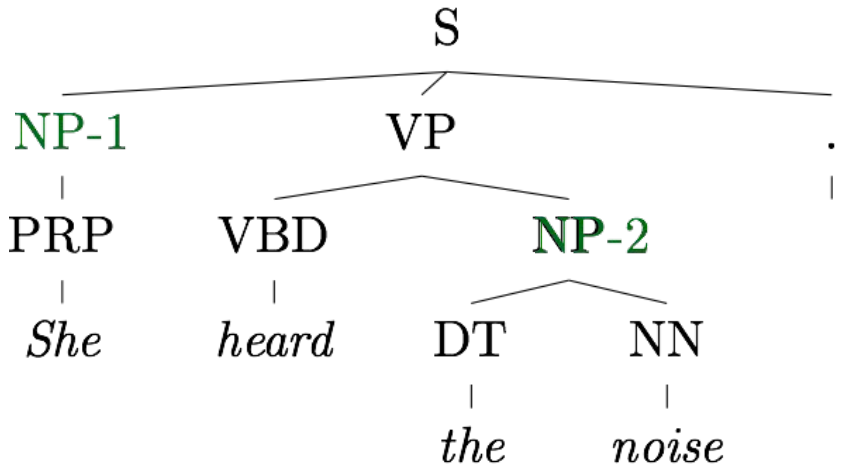
- Structural Annotation [Johnson ’98, Klein&Manning ’03]

Grammar Refinement



- Structural Annotation [Johnson ’98, Klein&Manning ’03]
- Lexicalization [Collins ’99, Charniak ’00]

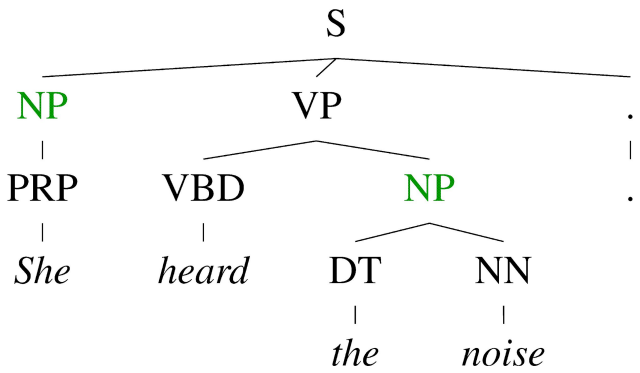
Grammar Refinement



- Structural Annotation [Johnson ’98, Klein&Manning ’03]
- Lexicalization [Collins ’99, Charniak ’00]
- Latent Variables [Matsuzaki et al. ’05, Petrov et al. ’06]

Structural Annotation

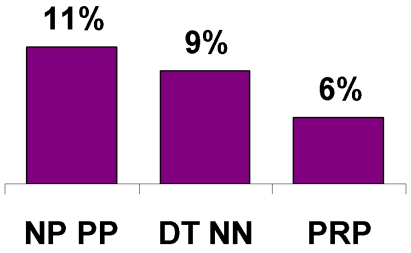
Ancestor-free assumption



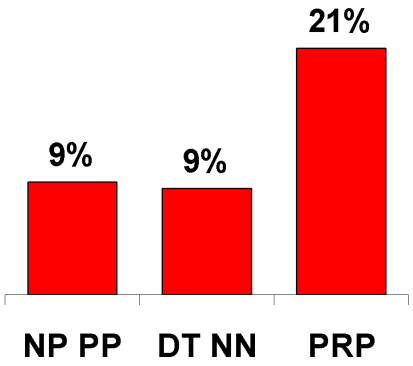
- Not every NP expansion can fill every NP slot

Ancestor-free assumption

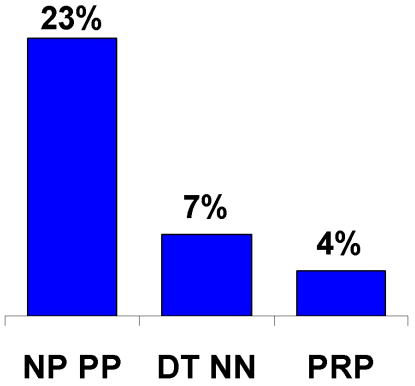
All NPs



NPs under S

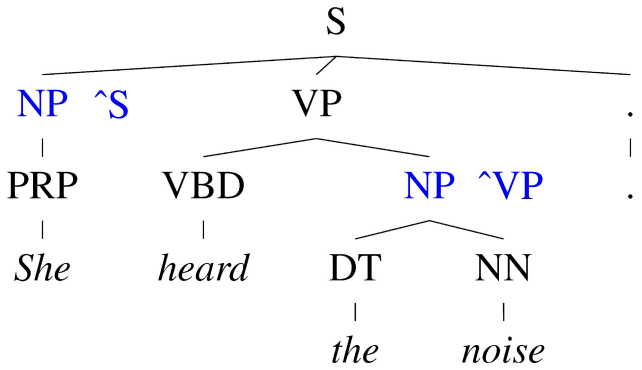


NPs under VP



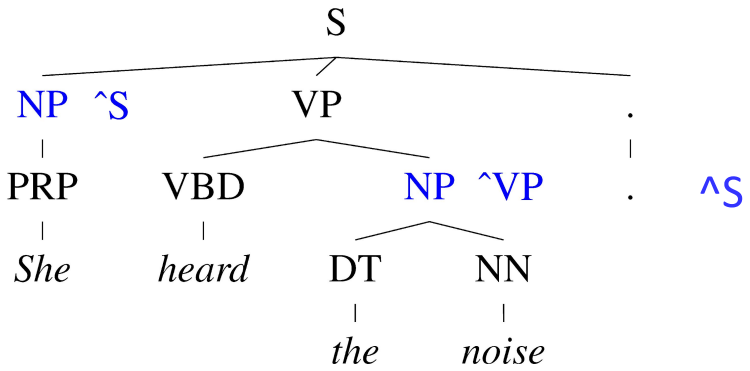
- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).
- Also: the subject and object expansions are correlated!

Parent Annotation



- Annotation refines base treebank symbols to improve statistical fit of the grammar

Parent Annotation

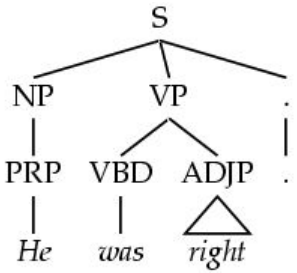


- Why stop at 1 parent?

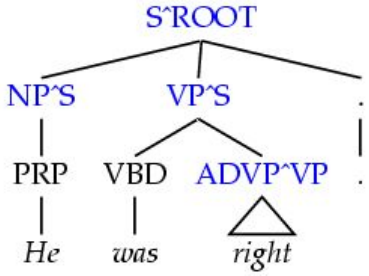
Vertical Markovization

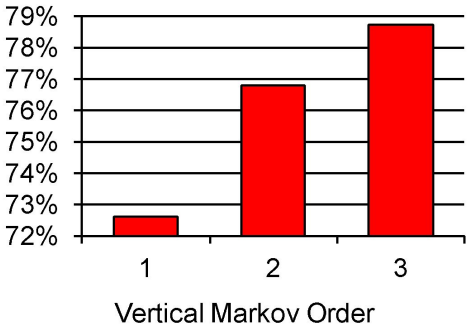
- **Vertical Markov order:** rewrites depend on past k ancestor nodes.
  (cf. parent annotation)

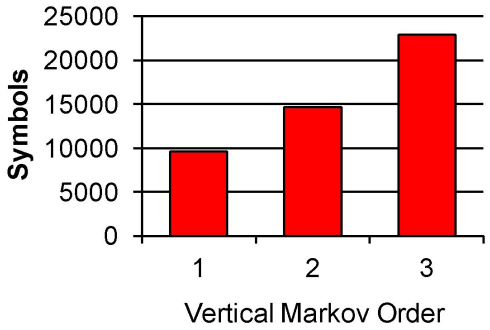
Order 1



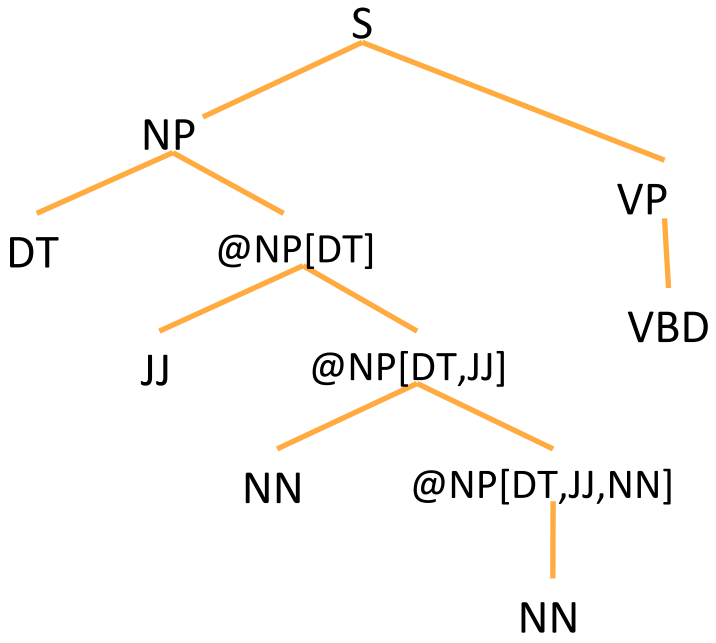
Order 2







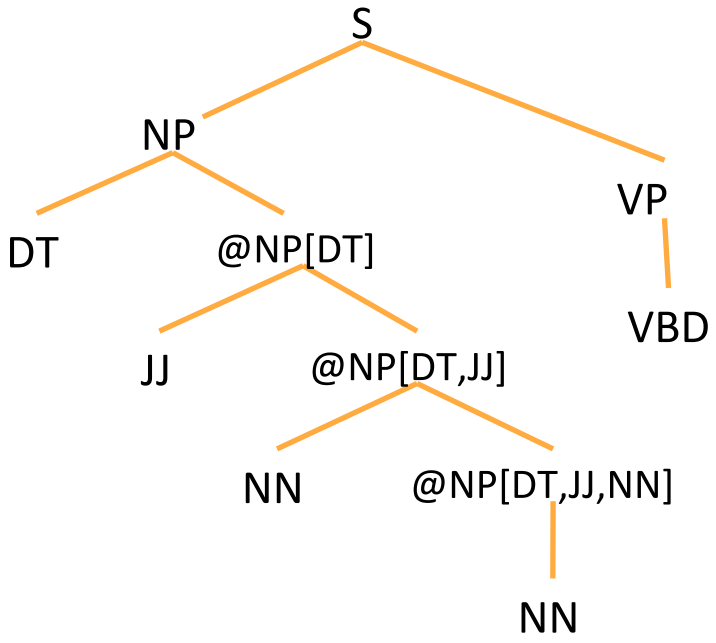
Back to our binarized tree



The fat house cat sat

- How much parent annotating are we doing?

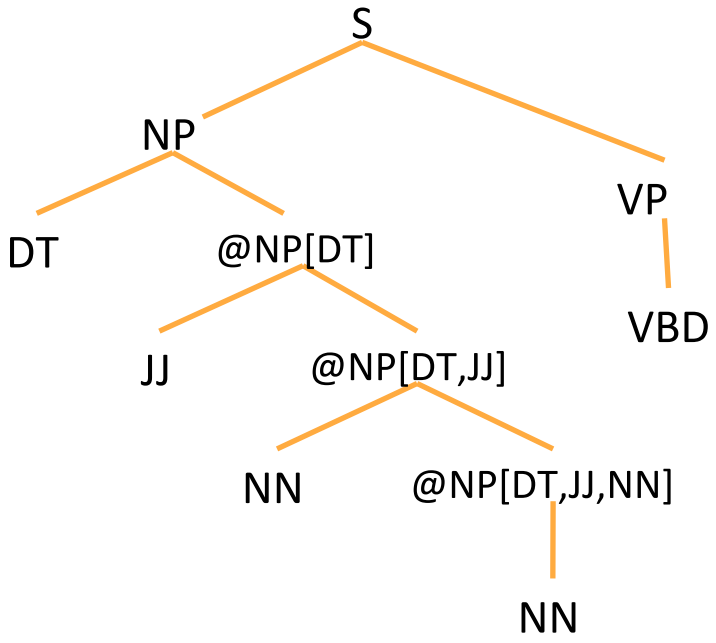
Back to our binarized tree



The fat house cat sat

- Are we doing any other structured annotation?

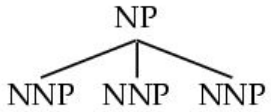
Back to our binarized tree



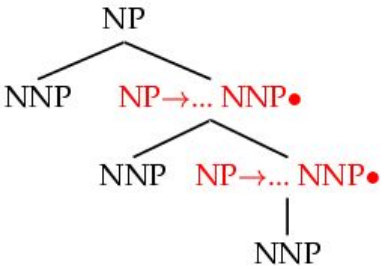
The fat house cat sat

- Remembering nodes to the left
- If parent annotation is termed “vertical” than this is “horizontal”

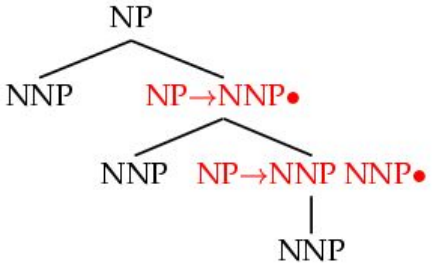
Horizontal Markovization

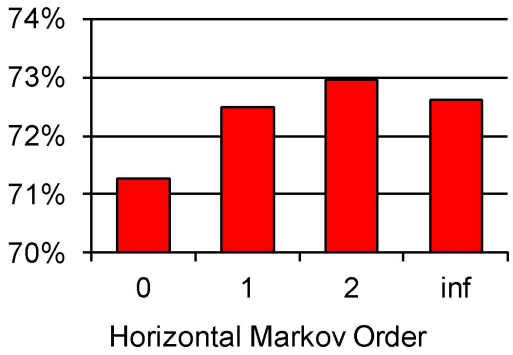


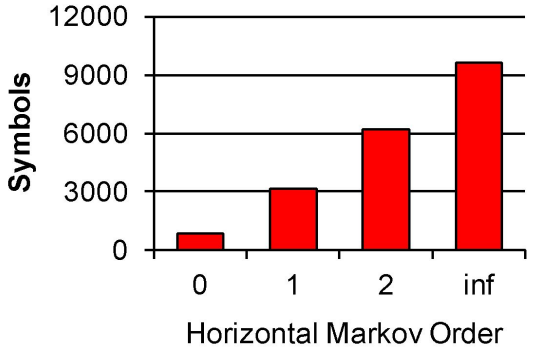
Order 1



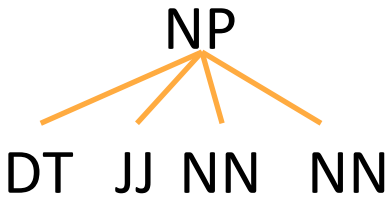
Order ∞



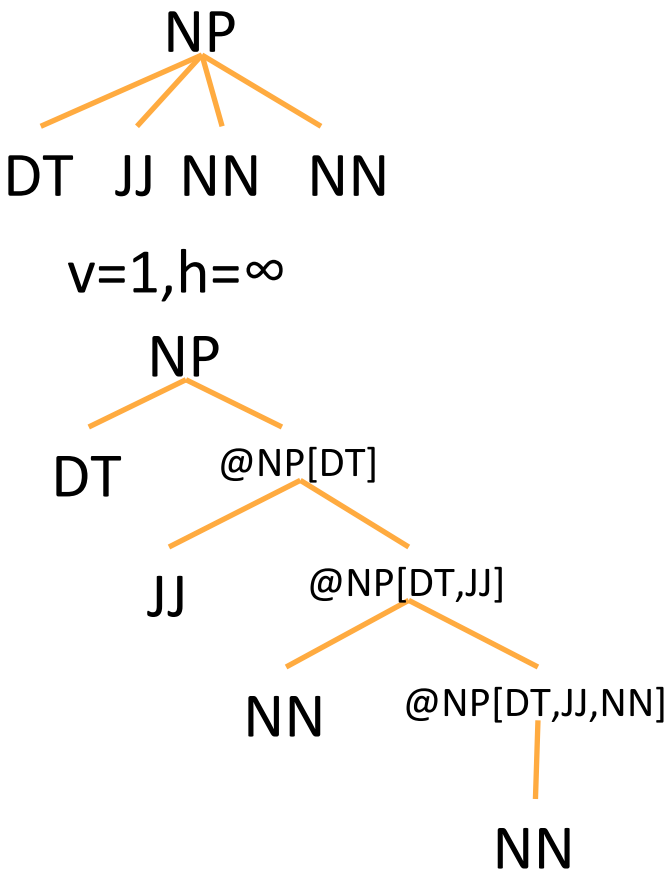




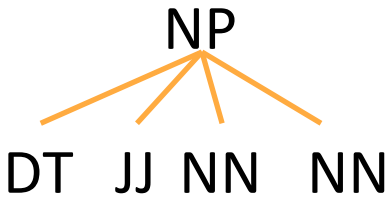
Binarization / Markovization



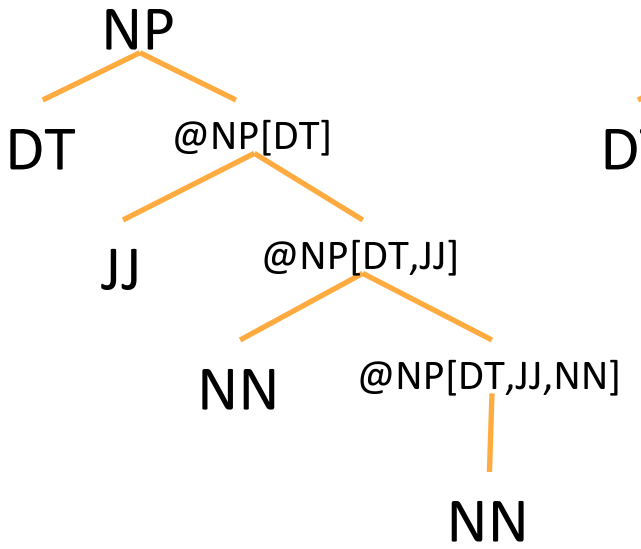
Binarization / Markovization



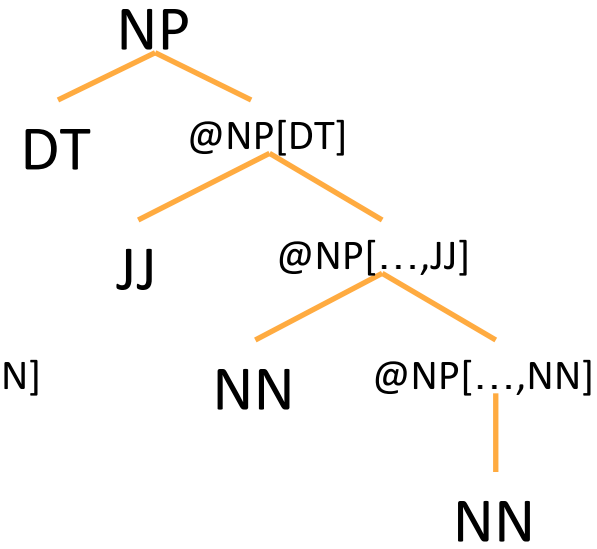
Binarization / Markovization



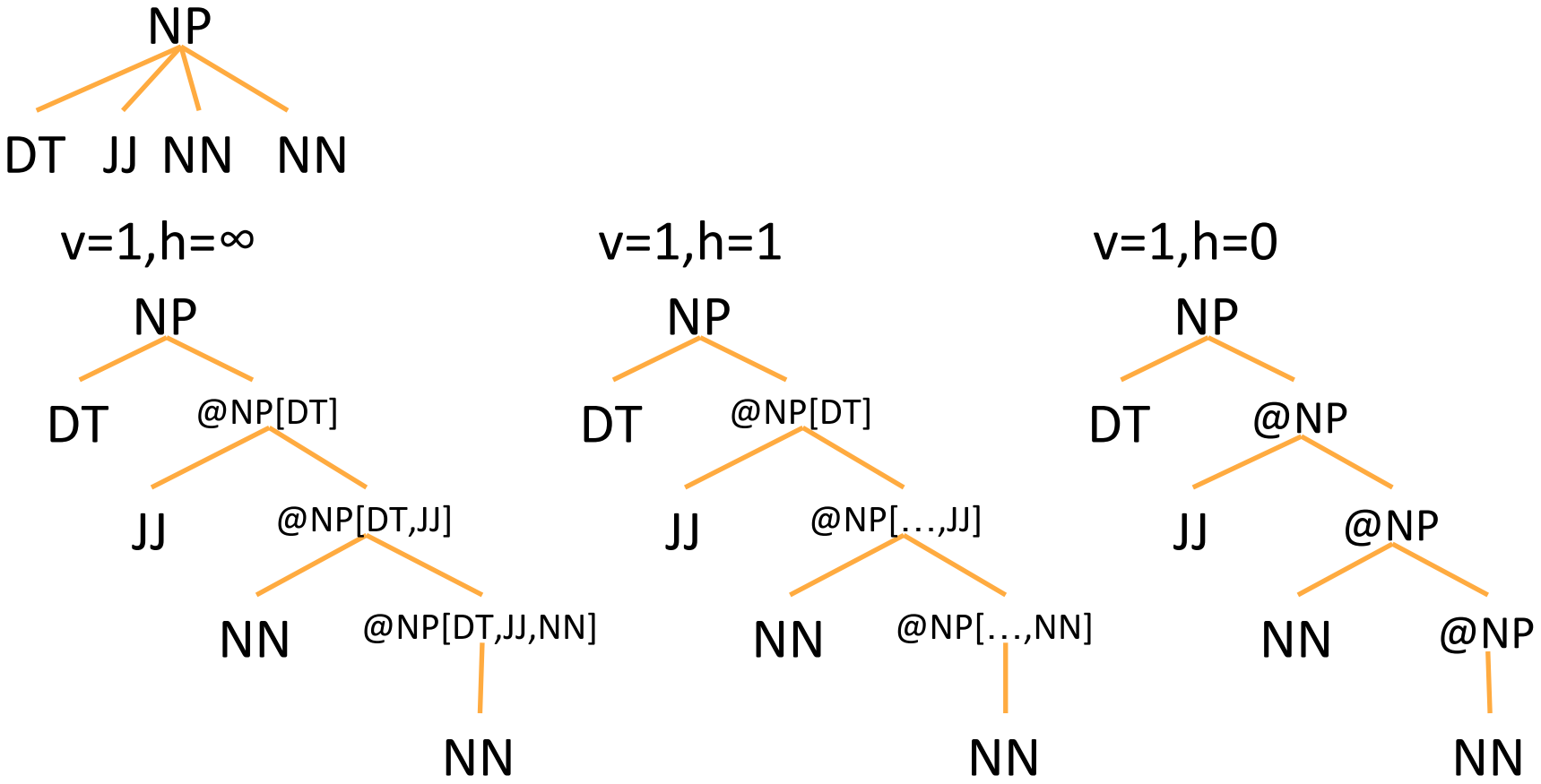
v=1,h=∞



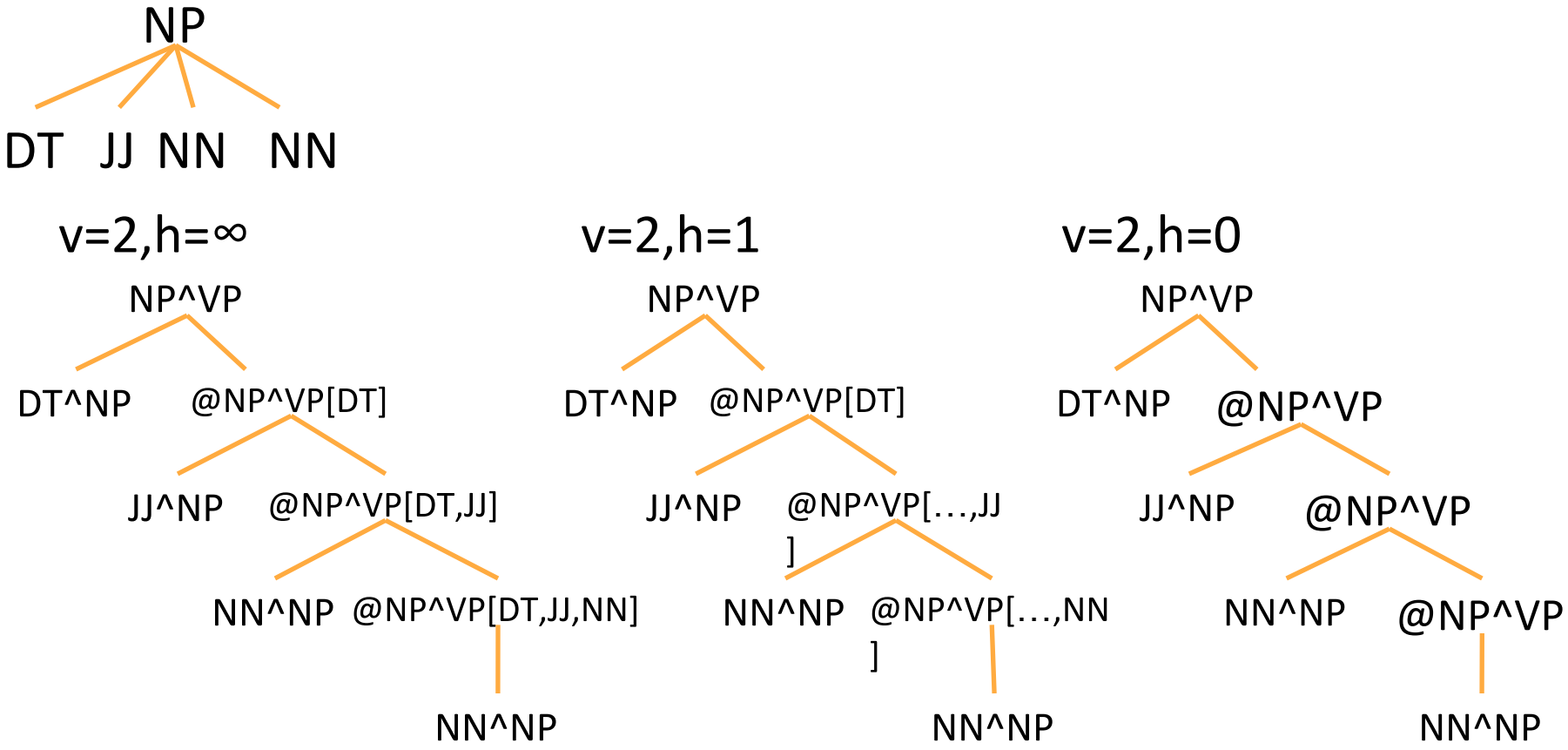
v=1,h=1



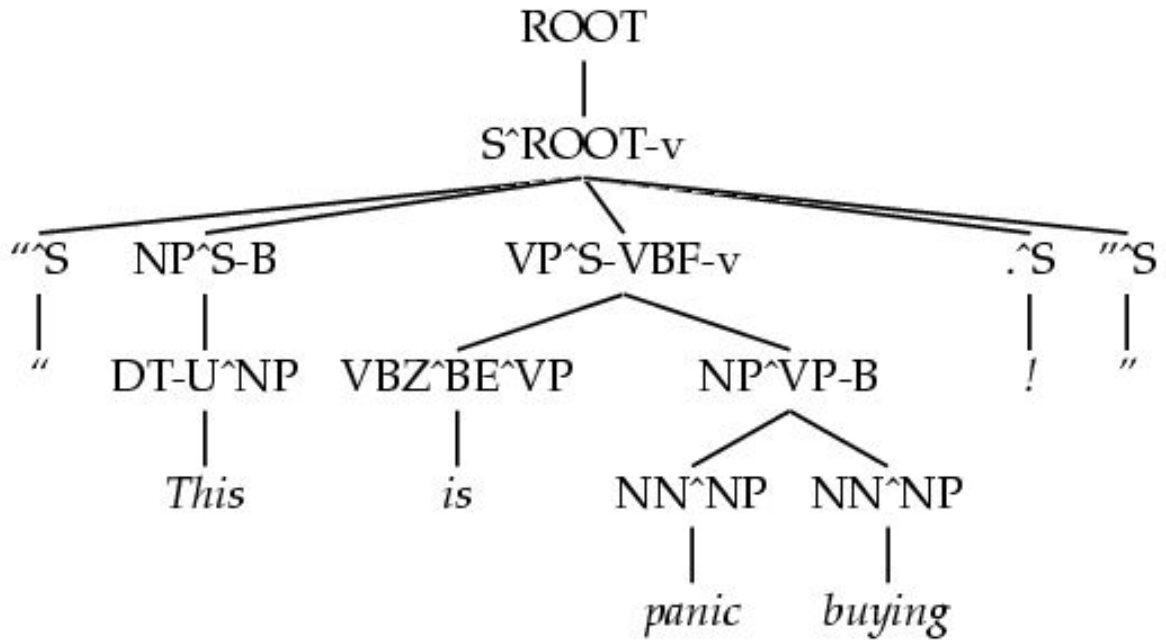
Binarization / Markovization



Binarization / Markovization



A Fully Annotated (Unlex) Tree



Some Test Set Results

Parser	LP	LR	**F1**	CB	0 CB
Magerman 95	84.9	84.6	**84.7**	1.26	56.6
Collins 96	86.3	85.8	**86.0**	1.14	59.9
Unlexicalized	86.9	85.7	**86.3**	1.10	60.3
Charniak 97	87.4	87.5	**87.4**	1.00	62.1
Collins 99	88.7	88.6	**88.6**	0.90	67.1

- Beats “first generation” lexicalized parsers.
- Lots of room to improve – more complex models next.

Efficient Parsing for Structural Annotation

Speeding up the algorithm

- **Basic pruning (roughly)**
  - For every span (i,j) store only labels which have the probability at most N times smaller than the probability of the most probable label for this span
  - Check not all rules but only rules yielding subtree labels having non-zero probability

- **Coarse-to-fine pruning**
  - Parse with a **smaller (simpler) grammar**, and **precompute (posterior) probabilities** for each spans, and use only the ones with non-negligible probability from the previous grammar

Overview: Coarse-to-Fine

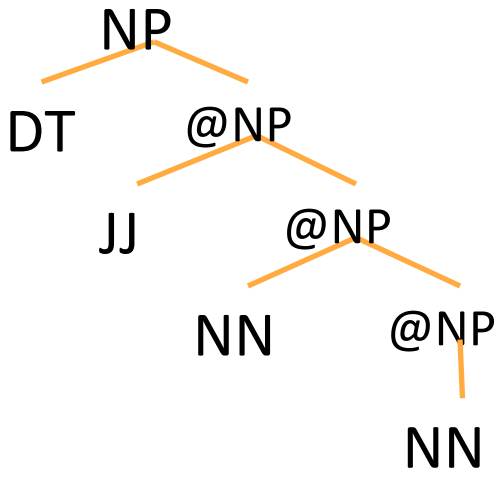
- We've introduce a lot of new symbols in our grammar: do we always need to consider all these symbols?
- **Motivation:**
  - If any NP is unlikely to span these words, than NP^S[DT], NP^VB[DT], NP^S[JJ], etc. are all unlikely
- High level:
  - First pass: compute probability that a coarse symbol spans these words
  - Second pass: parse as usual, but skip fine symbols that correspond with unprobable coarse symbols

Defining Coarse/Fine Grammars

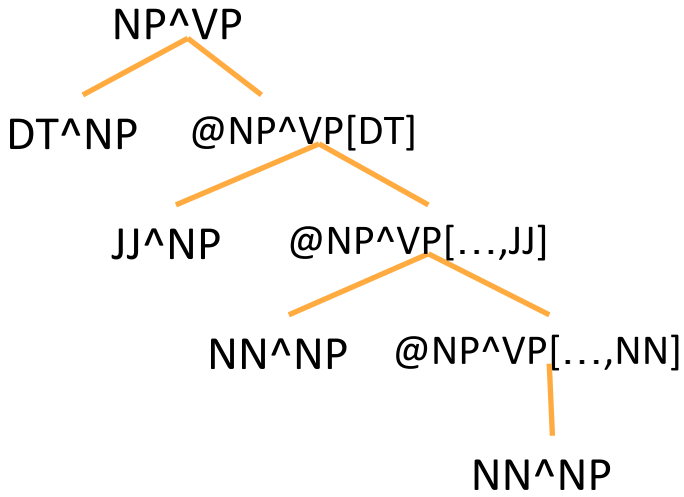
- [Charniak et al. 2006]
  - level 0: ROOT vs. not-ROOT
  - level 1: argument vs. modifier (i.e. two nontrivial nonterminals)
  - level 2: four major phrasal categories (verbal, nominal, adjectival and prepositional phrases)
  - level 3: all standard Penn treebank categories
- **Our version: stop at 2 passes**

Grammar Projections

Coarse Grammar



Fine Grammar



NP → DT @NP

NP^VP → DT^NP @NP^VP[DT]

Note: X-Bar Grammars are projections with rules like XP → Y @X or XP → @X Y or @X → X

Grammar Projections

Coarse Symbols

NP

@NP

DT

Fine Symbols

NP^VP

NP^S

@NP^VP[DT]

@NP^S[DT]

@NP^VP[…,JJ]

@NP^S[…,JJ]

DT^NP

Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability **P(*X at [i,j]* | *sentence)***:

$$\frac{\mathrm{P}_{\mathrm{IN}}(X,i,j)\cdot\mathrm{P}_{\mathrm{OUT}}(X,i,j)}{\mathrm{P}_{\mathrm{IN}}(root,0,n)}$$

coarse: ... | QP | NP | VP | ...

Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability **P(*X at [i,j]* | *sentence)***:

$$\frac{\mathrm{P}_{\mathrm{IN}}(X,i,j)\cdot\mathrm{P}_{\mathrm{OUT}}(X,i,j)}{\mathrm{P}_{\mathrm{IN}}(root,0,n)}$$

< threshold

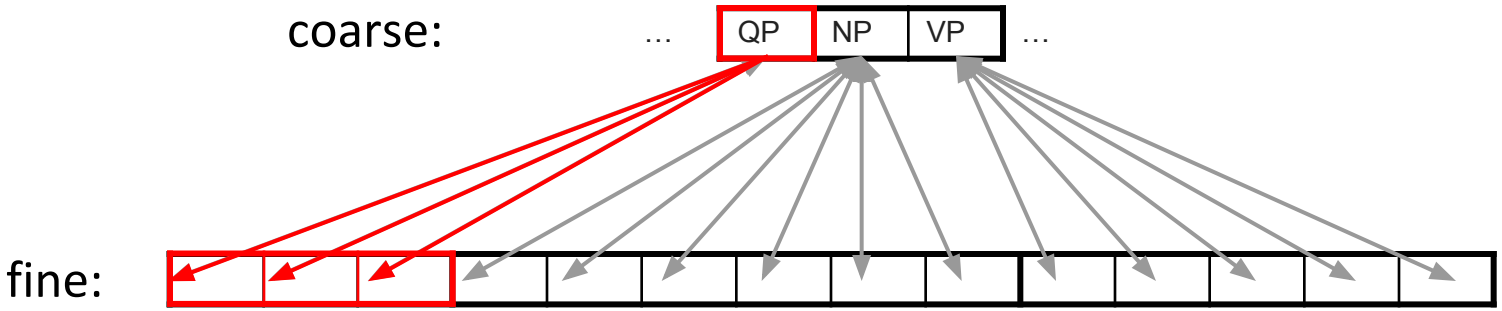
coarse:

Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability **P(*X at [i,j]* | *sentence)***:

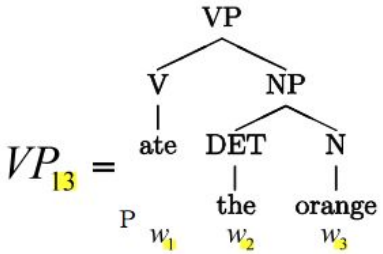
$$\frac{\mathrm{P}_{\mathrm{IN}}(X,i,j)\cdot\mathrm{P}_{\mathrm{OUT}}(X,i,j)}{\mathrm{P}_{\mathrm{IN}}(root,0,n)}$$

< threshold



Notation

- Non-terminal symbols (latent variables): $\{N^1, \ldots, N^n\}$
- Sentence (observed data): $\{w_1, \ldots, w_m\} = w_{1m}$
- N^j_{pq} denotes that N^j spans w_{pq} in the sentence



VP_{13} =

Inside probability

- Definition (compare with backward prob for HMMs):

$$\beta_j(p,q) = P(w_p, \ldots, w_q | N^j_{pq}, G) = P(N^j_{pq} \rightarrow w_{pq} | G)$$

- Computed recursively
  - **Base case:**

$$\beta_j(k,k) = P(w_k | N^j_{kk}, G) = P(N_j \rightarrow w_k | G)$$

The grammar is binarized

  - **Induction:**

$$\beta_j(p,q) = \sum_{rs} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)$$

Implementation: PCFG parsing

```
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
```

Implementation: inside

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double total = 0.0
      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)
          total = total + candidate
      chart[min][max][C] = best
```

Implementation: inside

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double total = 0.0
      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)
          total = total + candidate
      chart[min][max][C] = best
```

$$\beta_j(p,q) = \sum_{rs} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)$$

Implementation: inside

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double total = 0.0
      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)
          total = total + candidate
      chart[min][max][C] = best
```

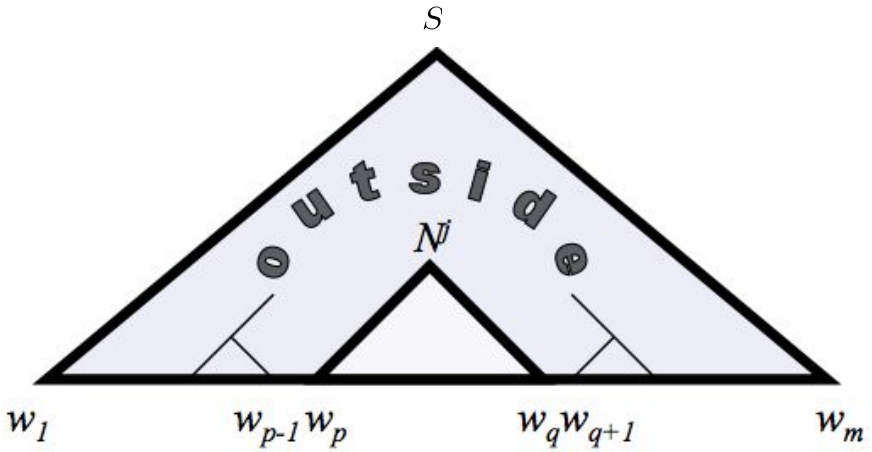
$$\beta_j(p,q) = \sum_{rs} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)$$

Outside probability

- Definition (compare with forward prob for HMMs):

$$\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G)$$

- The **joint probability** of starting with *S*, generating words $w_1, \ldots, w_{p-1}$, the non terminal N^j and words $w_{q+1}, \ldots, w_m$.



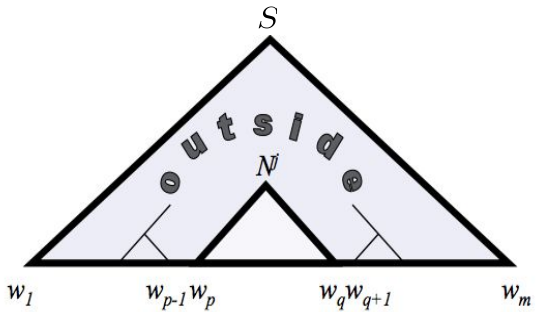
Calculating outside probability

- Computed recursively, base case

$$\alpha_1(1,m) = \alpha_S(1,m) = 1 \quad \alpha_{j\neq 1}(1,m) = 0$$

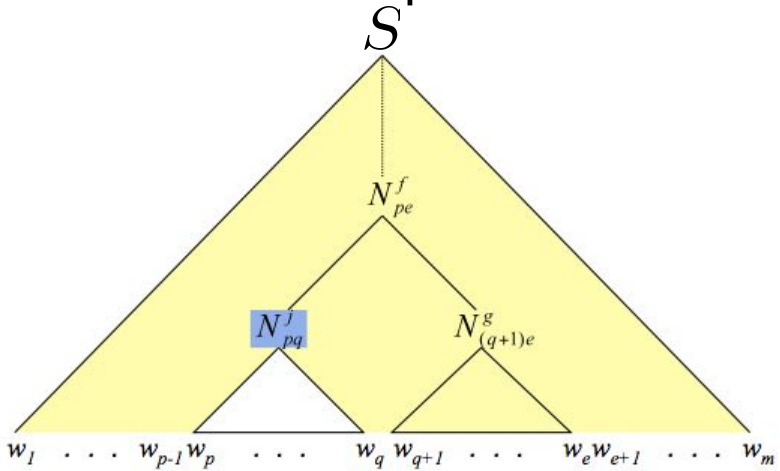
Induction?

Intuition: N^j_{pq}must be either the L or R child of a parent node. We first consider the case when it is the L child.



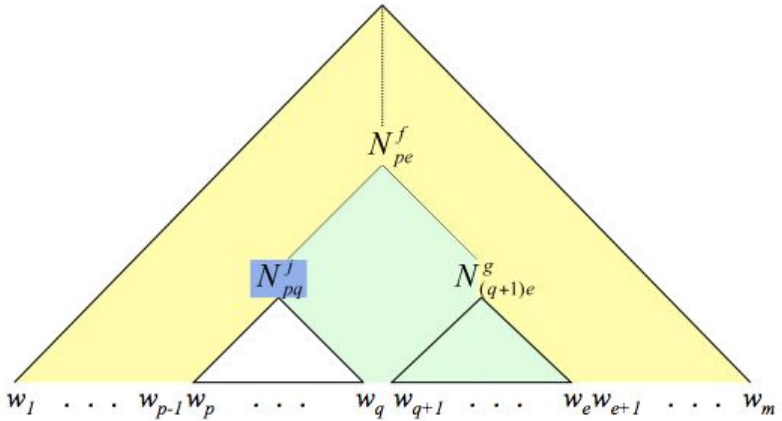
Calculating outside probability

- The yellow area is the probability we would like to calculate
  How do we decompose it?



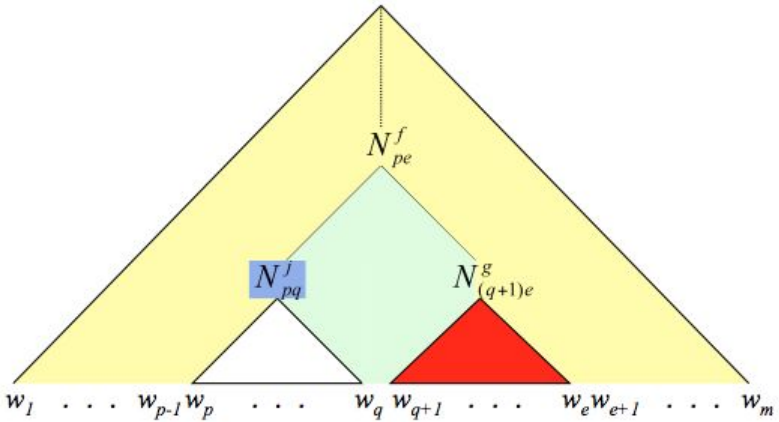
Calculating outside probability

Step 1: We assume that N^f_{pe} is the parent of N^j_{pq}. Its outside probability, $\alpha_f(p,e)$, (represented by the yellow shading) is available recursively. But how do we compute the green part?



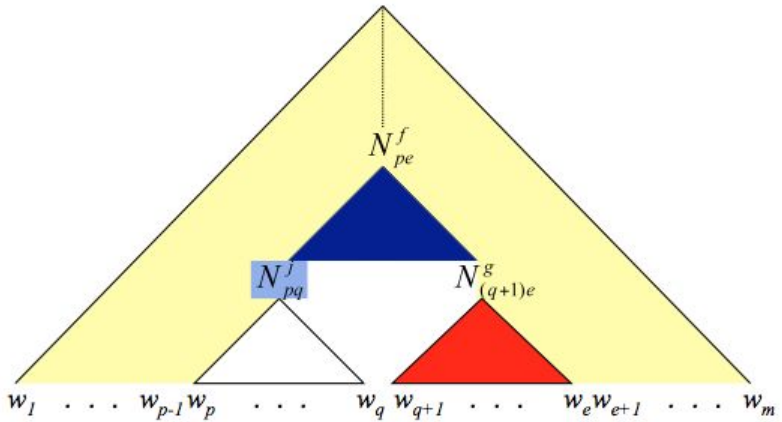
Calculating outside probability

Step 2: The red shaded area is the inside probability for $N^g_{(q+1)e}$ i.e. $\beta_g(q+1, e)$



Calculating outside probability

Step 3: The blue shaded area is just the production $N^f \rightarrow N^j N^g$, the corresponding probability $P(N^f \rightarrow N^j N^g | N^f, G)$

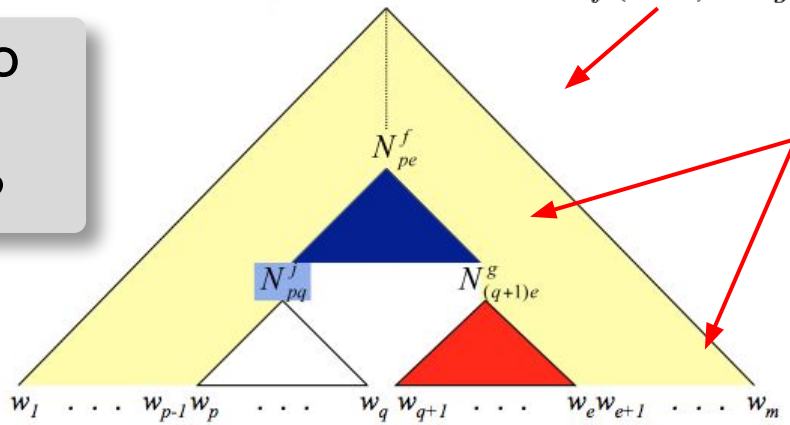


Calculating outside probability

If we multiply the terms together, we have the joint probability corresponding to the yellow, red and blue areas, assuming N^j was the L child of N^f, and give fixed non-terminals *f* and *g*, as well as a fixed partition *e*

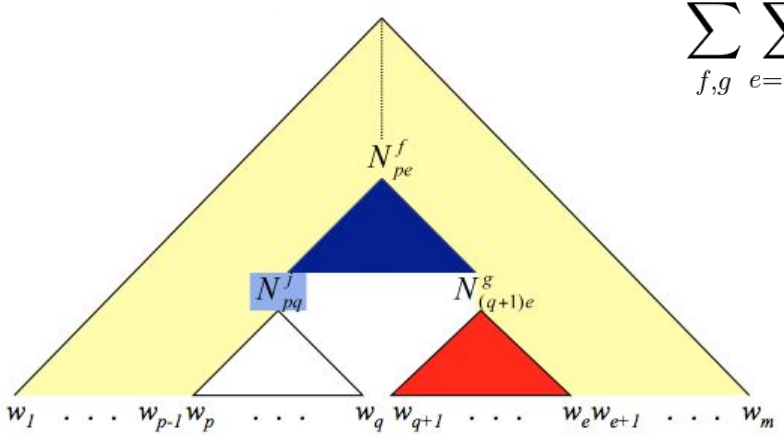
$$\alpha_f(p,e) \cdot \beta_g(q+1,e) \cdot P(N^f \rightarrow N^j N^g)$$

What if we do not want to assume this?



Calculating outside probability

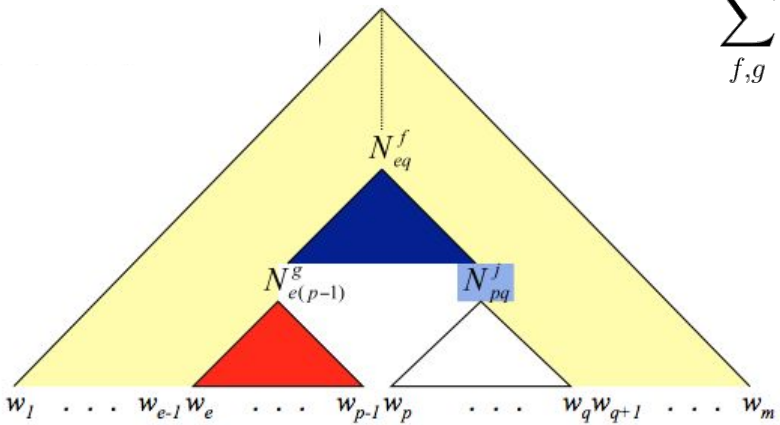
The joint probability corresponding to the yellow, red and blue areas, assuming N^j was the L child of some non-terminal:



$$\sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p,e) \cdot \beta_g(q+1,e) \cdot P(N^f \rightarrow N^j N^g)$$

Calculating outside probability

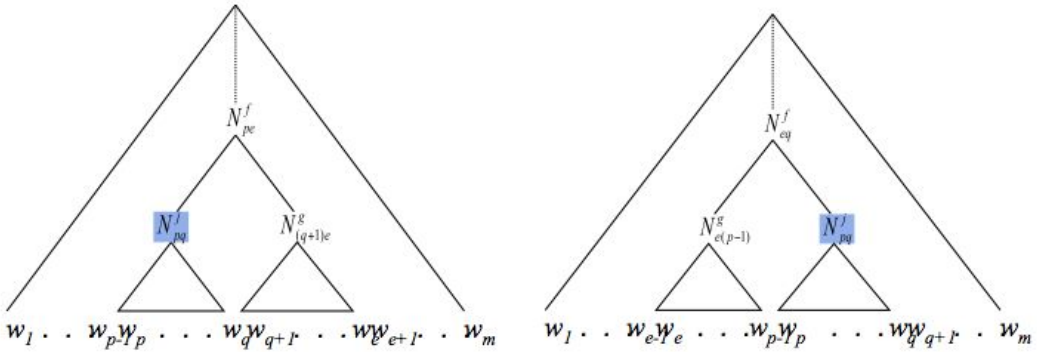
The joint probability corresponding to the yellow, red and blue areas, assuming N^j was the R child of some non-terminal:



$$\sum_{f,g}\sum_{e=1}^{p-1}\alpha_f(e,q)\cdot\beta_g(e,p-1)\cdot P(N^f \rightarrow N^g N^j)$$

Calculating outside probability

The joint final joint probability (the sum over the L and R cases):



$$\alpha_j(p,q) = \sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p,e) \cdot \beta_g(q+1,e) \cdot P(N^f \rightarrow N^j N^g) + \sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e,q) \cdot \beta_g(e,p-1) \cdot P(N^f \rightarrow N^g N^j)$$

Is C2F an Improvement?

$$\frac{\mathrm{P}_{\mathrm{IN}}(X,i,j)\cdot\mathrm{P}_{\mathrm{OUT}}(X,i,j)}{\mathrm{P}_{\mathrm{IN}}(root,0,n)} \quad < \quad \textit{threshold}$$

- Does coarse-to-fine pruning improve accuracy?
  - If your threshold is too high, it might throw away correct parses
- Does coarse-to-fine pruning improve speed?
  - Maybe, if your threshold is too low pruning might not be very useful

Improvements

Table 1: Impact of varying order(v, h) on grammar size, NT - Non-terminals, UP - Unary closure productions, and BP - Binary productions.

v, h	NTs	UPs	BPs
v = 1, h = 0	98	1,000	3,794
v = 1, h = 1	608	1,510	8,700
v = 1, h = ∞	8,371	9,273	23,034
v = 2, h = 2	**6,042**	**12,258**	**26,175**
v = 2, h = 3	9,740	15,956	31,207
v = 3, h = 1	7,332	15,946	32,125
v = 3, h = 2	**13,153**	**21,767**	**43,754**
v = 3, h = 3	17,770	26,384	49,030

Table 4: Performance of CKY parser with and without coarse-to-fine pruning for $v = 2, h = 2$.

	Decoding Time	$F1$
CKY	6.1 mins	80.55
Threshold = −5	**3.01 mins**	**79.5**
Threshold = −10	4.66 mins	80.52
Threshold = −25	10.59 mins	80.56

Table 5: Performance of CKY parser with and without coarse-to-fine pruning for $v = 2$, $h = 2$ when $maxTrainLength = 20$ and $maxTestLength = 20$

	Decoding Time	$F1$
CKY	13.8 secs	83.39
Threshold = −5	**12.72 secs**	**83.14**
Threshold = −10	16.95 secs	83.42
Threshold = −25	28.5 secs	83.39