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
The move followed a round of similar increases by other lenders, reflecting a continuing decline in that market.
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

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

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

\( S = \{S\} \)

\( R \)

\( S \rightarrow NP \ VP \)

(NP  A girl)  (VP ate a sandwich)

\( VP \rightarrow V \)

\( VP \rightarrow V \ NF \)

(V  ate)  (NP a sandwich)

\( VP \rightarrow VP \ PF \)

(VP  saw a girl)  (PP with a telescope)

\( NP \rightarrow NP \ PF \)

(NP  a girl)  (PP with a sandwich)

\( NP \rightarrow D \ N \)

(D a)  (N sandwich)

\( NP \rightarrow PN \)

\( PP \rightarrow P \ NF \)

(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 $P(Y_1 Y_2 \ldots Y_k | X)$
An example PCFGs

Associate probabilities with the rules:

\[ p(X \rightarrow \alpha) \]

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

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

\[ S \rightarrow NP \ VF \quad 1.0 \quad (NP \ A \ girl) \ (VP \ ate \ a \ sandwich) \]

\[ VP \rightarrow V \quad 0.2 \]

\[ VP \rightarrow V \ NF \quad 0.4 \quad (VP \ ate) \ (NP \ a \ sandwich) \]

\[ VP \rightarrow VP \ PF \quad 0.4 \quad (VP \ saw \ a \ girl) \ (PP \ with \ …) \]

\[ NP \rightarrow NP \ PF \quad 0.3 \quad (NP \ a \ girl) \ (PP \ with \ ….) \]

\[ NP \rightarrow D \ N \quad 0.5 \quad (D \ a) \ (N \ sandwich) \]

\[ NP \rightarrow PN \quad 0.2 \]

\[ PP \rightarrow P \ NF \quad 1.0 \quad (P \ with) \ (NP \ with \ a \ sandwich) \]

\[ N \rightarrow \text{girl} \quad 0.2 \]

\[ N \rightarrow \text{telescope} \quad 0.7 \]

\[ N \rightarrow \text{sandwich} \quad 0.1 \]

\[ PN \rightarrow I \quad 1.0 \]

\[ V \rightarrow \text{saw} \quad 0.5 \]

\[ V \rightarrow ate \quad 0.5 \]

\[ P \rightarrow \text{with} \quad 0.6 \]

\[ P \rightarrow \text{in} \quad 0.4 \]

\[ D \rightarrow \text{a} \quad 0.3 \]

\[ D \rightarrow \text{the} \quad 0.7 \]

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

$S \rightarrow NP \ VP \ 1.0$

$NP \rightarrow NP \ PP \ 0.3$

$NP \rightarrow D \ N \ 0.5$

$NP \rightarrow PN \ 0.2$

$PP \rightarrow P \ NP \ 1.0$

$N \rightarrow girl \ 0.2$

$N \rightarrow telescope \ 0.7$

$VP \rightarrow V \ 0.2$

$VP \rightarrow V \ NF \ 0.4$

$VP \rightarrow VP \ PP \ 0.4$

$PN \rightarrow I \ 1.0$

$V \rightarrow saw \ 0.5$

$V \rightarrow ate \ 0.5$

$P \rightarrow with \ 0.6$

$P \rightarrow in \ 0.4$

$D \rightarrow a \ 0.3$

$D \rightarrow the \ 0.7$

\[ P(T) = 1.0 \times 0.2 \times 1.0 \times 0.4 \times 0.5 \times 0.3 \times 0.5 \times 0.3 \times 0.2 \times 1.0 \times 0.6 \times 0.5 \times 0.3 \times 0.7 = 2.26e-5 \]
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):
    \[
    \arg\max_{T \in G(x)} P(T)
    \]

- 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
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):

\[
N \rightarrow \text{telescope} \quad D \rightarrow \text{the}\]

**Binary inner rules**:

\[
S \rightarrow NP VF \quad NP \rightarrow D \quad N
\]
Constraints on the grammar

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

\[ C \to x \]
\[ C \to C_1 C_2 \]

- 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: $C \rightarrow C_1$
  - Get rid of N-ary rules: $C \rightarrow C_1 C_2 \ldots C_n \ (n > 2)$

Crucial to process them, as required for efficient parsing
Transformation to CNF form: binarization

- Consider $NP \rightarrow DT\ NNP\ VBG\ NN$

  ![Diagram](image.png)

- How do we get a set of binary rules which are equivalent?
Transformation to CNF form: binarization

- Consider $NP \rightarrow DT \ NNP \ VBG \ NN$

- How do we get a set of binary rules which are equivalent?
  $NP \rightarrow DT \ X$
  $X \rightarrow NNP \ Y$
  $Y \rightarrow VBG \ NN$
Transformation to CNF form: binarization

- Consider $NP \rightarrow DT\ NNP\ VBG\ NN$

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

- A more systematic way to refer to new non-terminals
  $NP \rightarrow DT\ \underline{\underline{NP}}\_DT$
  $\underline{\underline{NP}}\_DT \rightarrow NNP\ \underline{\underline{NP}}\_DT\_NNP$
  $\underline{\underline{NP}}\_DT\_NNP \rightarrow VBG\ NN$
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 \(w = (w_1, w_2, \ldots, w_n)\)
- **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 \(w = (w_1, w_2, \ldots, w_n)\)
- **Goal** is to produce a parse tree for \(w\)
- We need an easy way to refer to substrings of \(w\)

\textit{span} \((i, j)\) refers to words between fenceposts \(i\) and \(j\)
Parsing one word

\[ C \rightarrow w_i \]
Parsing one word

\[ C \rightarrow w_i \]
Parsing one word

$C \rightarrow w_i$

covers all words between $i - 1$ and $i$
Parsing one word

$C \rightarrow C_1 \ C_2$

Check through all $C_1, C_2, \text{mid}$

covers all words btw $min$ and $mid$
covers all words btw $mid$ and $max$
Parsing one word

\[ C \rightarrow C_1 \ C_2 \]

Check through all C1, C2, \textit{mid}

covers all words btw \textit{min} and \textit{mid}
covers all words btw \textit{mid} and \textit{max}
Parsing one word

covers all words
between min and max
Preterminal rules

<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

inner rules

$S \rightarrow NP \ VP$

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

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

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

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

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

Chart (aka parsing triangle)
Preterminal rules

<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Inner rules

\[ S \rightarrow NP \ VP \]
\[ VP \rightarrow M \ V \]
\[ NP \rightarrow N \]
\[ NP \rightarrow N \ NP \]
\[ N \rightarrow can \]
\[ N \rightarrow lead \]
\[ N \rightarrow poison \]
\[ M \rightarrow can \]
\[ M \rightarrow must \]
\[ V \rightarrow poison \]
\[ V \rightarrow lead \]
Preterminal rules

<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Inner rules

\[
S \rightarrow NP \ VP
\]

\[
VP \rightarrow M \ V
\]

\[
NP \rightarrow N
\]

\[
NP \rightarrow N \ NP
\]

\[
N \rightarrow can
\]

\[
N \rightarrow lead
\]

\[
N \rightarrow poison
\]

\[
M \rightarrow can
\]

\[
M \rightarrow must
\]

\[
V \rightarrow poison
\]

\[
V \rightarrow lead
\]
Preterminal rules

$S \rightarrow NP \ VP$

$VP \rightarrow M \ V$

$NP \rightarrow N$

$NP \rightarrow N \ NP$

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$
<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

max = 1  max = 2  max = 3

min = 0

min = 1

min = 2

\[ S \rightarrow NP \ VP \]

\[ VP \rightarrow M \ V \]
\[ VP \rightarrow V \]

\[ NP \rightarrow N \]
\[ NP \rightarrow N \ NP \]

\[ N \rightarrow can \]
\[ N \rightarrow lead \]
\[ N \rightarrow poison \]

\[ M \rightarrow can \]
\[ M \rightarrow must \]

\[ V \rightarrow poison \]
\[ V \rightarrow lead \]
Preterminal rules

- $V \rightarrow \text{poison}$
- $M \rightarrow \text{must}$
- $N \rightarrow \text{lead}$

Inner rules

- $NP \rightarrow N\ NP$
- $VP \rightarrow M\ V$
- $S \rightarrow NP\ VP$
Preterminal rules

\[ S \rightarrow NP \ VP \]

\[ VP \rightarrow M \ V \]
\[ VP \rightarrow V \]

\[ NP \rightarrow N \]
\[ NP \rightarrow N \ NP \]

\[ N \rightarrow can \]
\[ N \rightarrow lead \]
\[ N \rightarrow poison \]

\[ M \rightarrow can \]
\[ M \rightarrow must \]

\[ V \rightarrow poison \]
\[ V \rightarrow lead \]
<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

```
max = 1  max = 2  max = 3
```

```
min = 0

1

?  

min = 1

2

?  

min = 2

3  

?  

min = 3
```

\[ S \rightarrow NP \ VP \]

\[ VP \rightarrow M \ V \]

\[ VP \rightarrow V \]

\[ NP \rightarrow N \]

\[ NP \rightarrow N \ NP \]

\[ N \rightarrow can \]

\[ N \rightarrow lead \]

\[ N \rightarrow poison \]

\[ M \rightarrow can \]

\[ M \rightarrow must \]

\[ V \rightarrow poison \]

\[ V \rightarrow lead \]
<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

\[
\begin{array}{ccc}
\text{max} = 1 & \text{max} = 2 & \text{max} = 3 \\
\text{min} = 0 & \text{?} & \\
\text{min} = 1 & & \\
\text{min} = 2 & & \\
\end{array}
\]

Preterminal rules:

\[
S \rightarrow NP \ VP
\]

\[
VP \rightarrow M \ V
\]

\[
VP \rightarrow V
\]

\[
NP \rightarrow N
\]

\[
NP \rightarrow N \ NP
\]

Inner rules:

\[
N \rightarrow \text{can}
\]

\[
N \rightarrow \text{lead}
\]

\[
N \rightarrow \text{poison}
\]

\[
M \rightarrow \text{can}
\]

\[
M \rightarrow \text{must}
\]

\[
V \rightarrow \text{poison}
\]

\[
V \rightarrow \text{lead}
\]
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>lead</td>
<td>can</td>
<td>poison</td>
<td></td>
</tr>
</tbody>
</table>

```
max = 1
max = 2
max = 3

min = 0
1 N, V

min = 1
2 N, M

min = 2
3 N, V
```

Preterminal rules:

- $S \rightarrow NP \ VP$
- $VP \rightarrow M \ V$
- $VP \rightarrow V$
- $NP \rightarrow N$
- $NP \rightarrow N \ NP$

Inner rules:

- $N \rightarrow can$
- $N \rightarrow lead$
- $N \rightarrow poison$
- $M \rightarrow can$
- $M \rightarrow must$
- $V \rightarrow poison$
- $V \rightarrow lead$
<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Preterminal rules**

```
S → NP VP
```

**Inner rules**

```
VP → M V
VP → V
NP → N
NP → N NP
```

```
N → can
N → lead
N → poison
```

```
M → can
M → must
```

```
V → poison
V → lead
```
Preterminal rules

<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

max = 1  max = 2  max = 3

\[
\begin{array}{c|c|c|c}
\text{min = 0} & N, V & NP, VP & ? \\
\hline
\text{min = 1} & N, M & NP & \\
\hline
\text{min = 2} & N, V & NP, VP & \\
\end{array}
\]

Inner rules

\[
S \rightarrow NP \ VP
\]

\[
VP \rightarrow M \ V \\
VP \rightarrow V
\]

\[
NP \rightarrow N \\
NP \rightarrow N \ NP
\]

\[
N \rightarrow \text{can} \\
N \rightarrow \text{lead} \\
N \rightarrow \text{poison}
\]

\[
M \rightarrow \text{can} \\
M \rightarrow \text{must}
\]

\[
V \rightarrow \text{poison} \\
V \rightarrow \text{lead}
\]
<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

```
max = 1  max = 2  max = 3

min = 0

1

N, V
NP, VP

min = 1

2

N, M
NP

3

N, V
NP, VP

min = 2

4

NP
```

Preterminal rules

- \( S \rightarrow NP \ VP \)
- \( NP \rightarrow N \)
- \( NP \rightarrow N \ NP \)
- \( VP \rightarrow M \ V \)
- \( VP \rightarrow V \)
- \( N \rightarrow can \)
- \( N \rightarrow lead \)
- \( N \rightarrow poison \)
- \( M \rightarrow can \)
- \( M \rightarrow must \)
- \( V \rightarrow poison \)
- \( V \rightarrow lead \)

Inner rules
Preterminal rules

Inner rules

\[ S \rightarrow NP \; VP \]

\[ VP \rightarrow M \; V \]

\[ VP \rightarrow V \]

\[ NP \rightarrow N \]

\[ NP \rightarrow N \; NP \]

\[ N \rightarrow can \]

\[ N \rightarrow lead \]

\[ N \rightarrow poison \]

\[ M \rightarrow can \]

\[ M \rightarrow must \]

\[ V \rightarrow poison \]

\[ V \rightarrow lead \]
Preterminal rules

<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

max = 1  
max = 2  
max = 3

min = 0

1  
N, V  
NP, VP

min = 1

2  
N, M  
NP

min = 2

3  
N, V  
NP, VP

4  
NP

5  
S, VP,  
NP

Inner rules

\[ S \rightarrow NP \ VP \]

\[ VP \rightarrow M \ V \]

\[ VP \rightarrow V \]

\[ NP \rightarrow N \]

\[ NP \rightarrow N \ NP \]

\[ N \rightarrow \text{can} \]

\[ N \rightarrow \text{lead} \]

\[ N \rightarrow \text{poison} \]

\[ M \rightarrow \text{can} \]

\[ M \rightarrow \text{must} \]

\[ V \rightarrow \text{poison} \]

\[ V \rightarrow \text{lead} \]
Preterminal rules

<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

max = 1    max = 2    max = 3

min = 0

1. $N, V$
   $NP, VP$

2. $N, M$
   $NP$

3. $N, V$
   $NP, VP$

4. $NP$

5. $S, VP,$
   $NP$

6. $?$

min = 1

min = 2

Inner rules

$S \rightarrow NP \ VP$

$VP \rightarrow M \ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N \ NP$

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$
<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

### Preterminal rules

- \( S \rightarrow NP \ VP \)

### Inner rules

- \( VP \rightarrow M \ V \)
- \( VP \rightarrow V \)
- \( NP \rightarrow N \)
- \( NP \rightarrow N \ NP \)
  
- \( N \rightarrow can \)
- \( N \rightarrow lead \)
- \( N \rightarrow poison \)
- \( M \rightarrow can \)
- \( M \rightarrow must \)
- \( V \rightarrow poison \)
- \( V \rightarrow lead \)
Preterminal rules

<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c|c|c|c|c|c}
  & 0 & 1 & 2 & 3 \\
\hline
\text{min = 0} & N, V & NP, VP & & & \\
\text{min = 1} & N, M & NP & & & \\
\text{min = 2} & N, V & NP, VP & & & \\
\hline
\text{max = 1} & & & S, NP & & \\
\text{max = 2} & & S, VP, NP & & & \\
\text{max = 3} & & & & & \\
\end{array}
\]

\[S \rightarrow NP \; VP\]

\[VP \rightarrow M \; V\]

\[VP \rightarrow V\]

\[NP \rightarrow N\]

\[NP \rightarrow N \; NP\]

\[N \rightarrow \text{can}\]

\[N \rightarrow \text{lead}\]

\[N \rightarrow \text{poison}\]

\[M \rightarrow \text{can}\]

\[M \rightarrow \text{must}\]

\[V \rightarrow \text{poison}\]

\[V \rightarrow \text{lead}\]
Preterminal rules

<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

max = 1  max = 2  max = 3

min = 0  min = 1  min = 2

mid=2

Inner rules

\[ S \rightarrow NP \ VP \]

\[ VP \rightarrow M \ V \]
\[ VP \rightarrow V \]

\[ NP \rightarrow N \]
\[ NP \rightarrow N \ NP \]

\[ N \rightarrow can \]
\[ N \rightarrow lead \]
\[ N \rightarrow poison \]

\[ M \rightarrow can \]
\[ M \rightarrow must \]

\[ V \rightarrow poison \]
\[ V \rightarrow lead \]
Preterminal rules

lead | can | poison
0    1    2    3

max = 1  max = 2  max = 3

min = 0

1  N, V
   NP, VP

2  N, M
   NP

3  N, V
   NP, VP

4  NP

5  S, VP,
   NP

6  S, NP
   S(?!)

min = 1

min = 2

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

Inner rules

S → NP VP

VP → M V
VP → V

NP → N
NP → N NP

N → can
N → lead
N → poison

M → can
M → must

V → poison
V → lead
No subject-verb agreement, and *poison* used as an intransitive verb
CKY implementation

Chart can be represented by a Boolean 3D array $\text{chart}[\min][\max][C]$

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

$\text{chart}[\min][\max][C] = \text{true}$ if the signature $(\min, \max, C)$ is already added to the chart;

$\text{false}$ otherwise.

$max = 1 \quad max = 2 \quad max = 3$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>min = 0</td>
<td>$N,V$</td>
<td>$NP$</td>
<td>$S,VP,$</td>
</tr>
<tr>
<td></td>
<td>$NP,VP$</td>
<td></td>
<td>$NP$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>5</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>min = 1</td>
<td>$N,M$</td>
<td>$S,VP,$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$NP$</td>
<td>$NP$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>min = 2</td>
<td>$N,V$</td>
</tr>
<tr>
<td></td>
<td>$NP,VP$</td>
</tr>
</tbody>
</table>

Here we assume that labels (C) are integer indices.
Implementation: preterminal rules

for each $w_i$ from left to right

   for each preterminal rule $C \rightarrow w_i$

       \[ \text{chart}[i-1][i][C] = \text{true} \]
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?

\[
\text{for each max from 2 to } n \\
\quad \text{for each min from max - 2 down to 0} \\
\quad \quad \text{for each syntactic category } C \\
\quad \quad \quad \text{for each binary rule } C \rightarrow C_1 C_2 \\
\quad \quad \quad \quad \text{for each mid from min + 1 to max - 1}
\]

\[O(n^3|R|)\] where \(|R|\) is the number of rules in the grammar
Practical time complexity

\[ \sim \eta^{3.6} \]
No subject-verb agreement, and *poison* used as an intransitive verb.
CKY Parsing with PCFGs
PCFGs

\[ P(T) = 1.0 \times 0.2 \times 1.0 \times 0.4 \times 0.5 \times 0.3 \times 0.5 \times 0.3 \times 0.2 \times 1.0 \times 0.6 \times 0.5 \times 0.3 \times 0.7 = 2.26e-5 \]
CKY with PCFGs

- Chart is represented by a 3d array of floats:
  \[ \text{chart}[\text{min}][\text{max}][\text{label}] \]
  - It stores probabilities for the most probable subtree with a given signature.

- \[ \text{chart}[0][n][S] \] will store the probability of the most probable full parse tree.
Intuition

For every $C$ choose $C_1$, $C_2$, and $\text{mid}$ such that

$$P(T_1) \times P(T_2) \times P(C \rightarrow C_1C_2)$$

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 $C \rightarrow w_i$

$\text{chart}[i-1][i][C] = p(C \rightarrow w_i)$
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
F1 bracket score

- Treebank PCFG
- Unlexicalized PCFG (Klein and Manning, 2003)
- Lexicalized PCFG (Collins, 1999)
- Automatically Induced PCFG (Petrov et al., 2006)
- The best results reported (as of 2012)
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

  \[
  S \rightarrow \text{NP} \ \text{VP} \\
  \text{NP} \rightarrow \text{DT} \ \text{JJ} \ \text{NN} \ \text{NN} \\
  \text{VP} \rightarrow \text{VBD}
  \]

- 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

```
S → NP VP
NP → DT JJ NN NN 1
VP → VBD 1
```

- Vanilla CKY only allows *binary* rules.

```
S
  | NP
  |   VP
  |    
  |     
DT  JJ  JJ  NN  VBD
  | The  fat orange cat sat
```


Option 1: Binarize the Grammar

S → NP VP
NP → DT JJ NN NN
VP → VBD

S → NP VP
NP → DT @NP[DT]
@NP[DT] → JJ @NP[DT JJ]
@NP[DT JJ] → NN NN

S → NP VBD
NP → DT @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:**

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

\[ \text{NP} \rightarrow \text{DT} \@NP[\text{DT}] \]

\[ \@NP[\text{DT}] \rightarrow \text{JJ} \@NP[\text{DT JJ}] \]

\[ \@NP[\text{DT JJ}] \rightarrow \text{NN} \@NP[\text{DT JJ NN}] \]

**Unary Rules:**

\[ \text{VP} \rightarrow \text{VBD} \]

\[ \@NP[\text{DT JJ NN}] \rightarrow \text{NN} \]
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 applying unary rules
- Fill by applying rules to elements of the binary chart
CKY with TreeBank PCFG

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

  ![Diagram](image)

  - **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

- 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?

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>72.0</td>
</tr>
</tbody>
</table>
F1 bracket score

- Treebank PCFG
- Unlexicalized PCFG (Klein and Manning, 2003)
- Lexicalized PCFG (Collins, 1999)
- Automatically Induced PCFG (Petrov et al., 2006)
- The best results reported (as of 2012)
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

```
S

NP

PRP  VBD  NP

She  heard  DT  NN

the  noise
```
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**
  - NP: 11%
  - PP: 9%
  - DT NN: 6%

- **NPs under S**
  - NP: 9%
  - PP: 9%
  - DT NN: 21%

- **NPs under VP**
  - NP: 23%
  - PP: 7%
  - DT NN: 4%

- **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)
Back to our binarized tree

- How much parent annotating are we doing?

The fat house cat sat
Back to our binarized tree

- Are we doing any other structured annotation?

The fat house cat sat
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

NP
  DT JJ NN NN
Binarization / Markovization

v=1, h=∞
Binarization / Markovization

v=1, h=∞

v=1, h=1
Binarization / Markovization

v=1, h=∞

DT (NP (JJ (NN) (NN)))

v=1, h=1

DT (NP (JJ) (NN))

v=1, h=0

DT (NP (JJ) (NN))
Binarization / Markovization

v=2, h=∞

NP
- DT
- JJ
- NN
- NN

v=2, h=1

NP^VP
- DT^NP
- @NP^VP[DT]

NP
- JJ^NP
- @NP^VP[DT, JJ]

NP
- NN^NP
- @NP^VP[DT, JJ, NN]

NN^NP
- NN^NP

v=2, h=0

NP^VP
- DT^NP
- @NP^VP[DT]

NP
- JJ^NP
- @NP^VP[..., JJ]

NP
- NN^NP
- @NP^VP[..., NN]

NN^NP
- NN^NP

NN^NP
- NN^NP

NN^NP
A Fully Annotated (Unlex) Tree

```
ROOT
  |   |
S\^ROOT-v
  |   |
S \ NP\^S-B
  |   |   |   |
''S \ DT-U\^NP \ VBZ\^BE\^VP \ NP\^VP-B
  |   |   |   |   |   |   |   |   |
 '' '' This is \ NN\^NP \ NN\^NP
   |   |   |   |   |   |   |   |   |
   panic buying
```

## Some Test Set Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>LP</th>
<th>LR</th>
<th>F1</th>
<th>CB</th>
<th>0 CB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magerman 95</td>
<td>84.9</td>
<td>84.6</td>
<td><strong>84.7</strong></td>
<td>1.26</td>
<td>56.6</td>
</tr>
<tr>
<td>Collins 96</td>
<td>86.3</td>
<td>85.8</td>
<td><strong>86.0</strong></td>
<td>1.14</td>
<td>59.9</td>
</tr>
<tr>
<td>Unlexicalized</td>
<td><strong>86.9</strong></td>
<td><strong>85.7</strong></td>
<td><strong>86.3</strong></td>
<td>1.10</td>
<td><strong>60.3</strong></td>
</tr>
<tr>
<td>Charniak 97</td>
<td>87.4</td>
<td>87.5</td>
<td><strong>87.4</strong></td>
<td>1.00</td>
<td>62.1</td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.7</td>
<td>88.6</td>
<td><strong>88.6</strong></td>
<td>0.90</td>
<td>67.1</td>
</tr>
</tbody>
</table>

- 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

NP
  └── DT
    └── JJ
        └── NN
  └── @NP
    └── @NP
        └── NN

Fine Grammar

NP^VP
  └── DT^NP
    └── @NP^VP[DT]
        └── JJ^NP
            └── @NP^VP[...JJ]
                └── NN^NP
                    └── @NP^VP[...NN]
                        └── NN^NP

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 $P(X \text{ at } [i,j] | \text{ sentence})$:

$$
\frac{P_{IN}(X, i, j) \cdot P_{OUT}(X, i, j)}{P_{IN}(\text{root}, 0, n)}
$$

coarse: \hspace{1cm} ... \hspace{1cm} \begin{array}{c} \text{QP} \\ \text{NP} \\ \text{VP} \end{array} \hspace{1cm} ...
Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability $P(X \text{ at } [i,j] \mid \text{sentence})$:

$$\frac{P_{IN}(X, i, j) \cdot P_{OUT}(X, i, j)}{P_{IN}(\text{root}, 0, n)} < \text{threshold}$$

coarse: ... [QP NP VP] ...
Coarse-to-Fine Pruning

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

$$\frac{P_{IN}(X, i, j) \cdot P_{OUT}(X, i, j)}{P_{IN}(\text{root}, 0, n)} < \text{threshold}$$
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

\[
\text{VP}_{13} = \text{ate} \quad \text{DET} \quad \text{orange} \\
\text{P} \quad w_1 \quad w_2 \quad w_3
\]
Inside probability

- Definition (compare with backward prob for HMMs):
  \[
  \beta_j(p, q) = P(w_p, \ldots, w_q | N_{pq}^j, G) = P(N_{pq}^j \rightarrow w_{pq} | G)
  \]

- Computed recursively
  - Base case:
    \[
    \beta_j(k, k) = P(w_k | N_{kk}^j, G) = P(N_j \rightarrow w_k | G)
    \]
  - Induction:
    \[
    \beta_j(p, q) = \sum_{r} \sum_{s \leq d = p} P(N_j \rightarrow N_r^r N_s^s) \beta_r(p, d) \beta_s(d + 1, q)
    \]

The grammar is binarized
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
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 -> C_1 C_2
                for each mid from min + 1 to max - 1
                    double t_1 = chart[min][mid][C_1]
                    double t_2 = chart[mid][max][C_2]
                    double candidate = t_1 * t_2 * p(C -> C_1 C_2)
                    total = total + candidate
            chart[min][max][C] = best
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

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

- **Definition (compare with forward prob for HMMs):**
  \[ \alpha_j(p, q) = P(w_{1(p-1)}, N_{pq}^j, w_{(q+1)m}|G') \]

- 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?
Step 1: We assume that $N_{pe}^f$ is the parent of $N_{pq}^j$. 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 $N_{(q+1)\epsilon}^g$ i.e. $\beta_q(q + 1, e)$
Calculating outside probability

Step 3: The blue shaded area is just the production $N^f ightarrow N^j N^g$, 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?
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?

\[
\frac{P_{\text{IN}}(X, i, j) \cdot P_{\text{OUT}}(X, i, j)}{P_{\text{IN}}(\text{root}, 0, n)} < \text{threshold}
\]

- 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.

<table>
<thead>
<tr>
<th>(v, h)</th>
<th>NTs</th>
<th>UPs</th>
<th>BPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v = 1, h = 0)</td>
<td>98</td>
<td>1,000</td>
<td>3,794</td>
</tr>
<tr>
<td>(v = 1, h = 1)</td>
<td>608</td>
<td>1,510</td>
<td>8,700</td>
</tr>
<tr>
<td>(v = 1, h = \infty)</td>
<td>8,371</td>
<td>9,273</td>
<td>23,034</td>
</tr>
<tr>
<td>(v = 2, h = 2)</td>
<td>6,042</td>
<td>12,258</td>
<td>26,175</td>
</tr>
<tr>
<td>(v = 2, h = 3)</td>
<td>9,740</td>
<td>15,956</td>
<td>31,207</td>
</tr>
<tr>
<td>(v = 3, h = 1)</td>
<td>7,332</td>
<td>15,946</td>
<td>32,125</td>
</tr>
<tr>
<td>(v = 3, h = 2)</td>
<td>13,153</td>
<td>21,767</td>
<td>43,754</td>
</tr>
<tr>
<td>(v = 3, h = 3)</td>
<td>17,770</td>
<td>26,384</td>
<td>49,030</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Decoding Time</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKY</td>
<td>6.1 mins</td>
<td>80.55</td>
</tr>
<tr>
<td>\textbf{Threshold = -5}</td>
<td>3.01 mins</td>
<td>\textbf{79.5}</td>
</tr>
<tr>
<td>\textbf{Threshold = -10}</td>
<td>4.66 mins</td>
<td>80.52</td>
</tr>
<tr>
<td>\textbf{Threshold = -25}</td>
<td>10.59 mins</td>
<td>80.56</td>
</tr>
</tbody>
</table>

Table 5: Performance of CKY parser with and without coarse-to-fine pruning for \(v = 2, h = 2\) when \(\text{maxTrainLength} = 20\) and \(\text{maxTestLength} = 20\)

<table>
<thead>
<tr>
<th></th>
<th>Decoding Time</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKY</td>
<td>13.8 secs</td>
<td>83.39</td>
</tr>
<tr>
<td>\textbf{Threshold = -5}</td>
<td>12.72 secs</td>
<td>\textbf{83.14}</td>
</tr>
<tr>
<td>\textbf{Threshold = -10}</td>
<td>16.95 secs</td>
<td>83.42</td>
</tr>
<tr>
<td>\textbf{Threshold = -25}</td>
<td>28.5 secs</td>
<td>83.39</td>
</tr>
</tbody>
</table>