Algorithms for NLP

Parsing II

Yulia Tsvetkov – CMU

Slides: Ivan Titov – University of Edinburgh,
Chris Dyer – Deepmind
Announcements

- HW2 out
- Today: Sachin will give an overview of HW2
- Recitation on EM next week 10/12
- Recitation on HW2 the week after 10/19
- Yulia office hours
  - today: 3:30-4:00
  - next week Yulia is away, no office hours
The move followed a round of similar increases by other lenders, reflecting a continuing decline in that market.
# Context Free Grammar (CFG)

<table>
<thead>
<tr>
<th>Grammar (CFG)</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT $\rightarrow$ S</td>
<td>NP $\rightarrow$ NP PP</td>
</tr>
<tr>
<td>S $\rightarrow$ NP VP</td>
<td>VP $\rightarrow$ VBP NP</td>
</tr>
<tr>
<td>NP $\rightarrow$ DT NN</td>
<td>VP $\rightarrow$ VBP NP PP</td>
</tr>
<tr>
<td>NP $\rightarrow$ NN NNS</td>
<td>PP $\rightarrow$ IN NP</td>
</tr>
</tbody>
</table>

Other grammar formalisms: LFG, HPSG, TAG, CCG…
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
## Parsing with CKY

<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

- \( S \rightarrow NP \ VP \)
- \( VP \rightarrow M \ V \)
- \( VP \rightarrow V \)
- \( NP \rightarrow N \)
- \( NP \rightarrow N \ NP \)

### 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

<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

$S \rightarrow NP \ VP$

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

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

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

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

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

Chart (aka parsing triangle)

max = 1  max = 2  max = 3

min = 0

min = 1

min = 2

$S?$
Preterminal rules

$S \rightarrow NP \ VP$

Inner rules

$VP \rightarrow M \ V$

$NP \rightarrow N$

$NP \rightarrow N \ NP$

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$
Preterminal rules

\[ S \rightarrow NP \ VP \]

Inner rules

\[ VP \rightarrow M \ V \]
\[ NP \rightarrow N \]
\[ NP \rightarrow N \ NP \]

Preterminal rules

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

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

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

<table>
<thead>
<tr>
<th>min = 0</th>
<th>min = 1</th>
<th>min = 2</th>
</tr>
</thead>
</table>

max = 1     max = 2     max = 3

\[ S \rightarrow NP \ VP \]

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

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

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

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

\[ V \rightarrow poison \]
\[ V \rightarrow lead \]
\[ S \rightarrow NP \ VP \]

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

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

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

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

\[ V \rightarrow poison \]
\[ V \rightarrow lead \]
$S \rightarrow NP \ VP$

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

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

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

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

$V \rightarrow poison$
$V \rightarrow lead$
Preterminal rules

lead | can | poison
0    1    2    3

max = 1  max = 2  max = 3

min = 0  ?
2       ?
3       ?

Inner rules

\[ S \rightarrow NP \ VP \]
\[ VP \rightarrow M \ V \]
\[ VP \rightarrow V \]
\[ NP \rightarrow N \]
\[ NP \rightarrow N \ NP \]

Preterminal rules

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

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

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

\[ S \rightarrow NP \; VP \]

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

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

Inner rules

Preterminal 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>

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

**Preterminal rules**

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

**Inner 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>

<table>
<thead>
<tr>
<th>min = 0</th>
<th>min = 1</th>
<th>min = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N, V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NP, VP</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>N, M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>N, V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NP, VP</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>?</td>
<td></td>
</tr>
</tbody>
</table>

Preterminal rules:

- \( S \to NP \ VP \)

Inner rules:

- \( VP \to M \ V \)
- \( VP \to V \)
- \( NP \to N \)
- \( NP \to N \ NP \)

- \( N \to can \)
- \( N \to lead \)
- \( N \to poison \)
- \( M \to can \)
- \( M \to must \)
- \( V \to poison \)
- \( V \to lead \)
Preterminal rules

lead | can | poison
0    1    2    3

max = 1  max = 2  max = 3

1  N, V
   NP, VP

2  N, M
   NP

3  N, V
   NP, VP

4  ?

min = 0

min = 1

min = 2

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

Preterminal rules
$S \rightarrow NP \ VP$

$VP \rightarrow M \ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N \ NP$

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$
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

\( N, V \)
\( NP, VP \)

Inner rules

\[ S \rightarrow NP \ VP \]

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

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

Preterminal rules

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

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

\[ V \rightarrow poison \]
\[ V \rightarrow lead \]
<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>3</td>
<td></td>
<td>max = 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>max = 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>max = 1</td>
</tr>
<tr>
<td>min = 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>min = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>min = 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{array}{c}
\text{Preterminal rules} \\
S \rightarrow \text{NP } \text{VP} \\
\text{Inner rules} \\
VP \rightarrow \text{M } \text{V} \\
\text{NP} \rightarrow \text{N} \\
\text{NP} \rightarrow \text{N} \text{ NP} \\
\text{NP} \rightarrow \text{N} \\
\text{NP} \rightarrow \text{lead} \\
\text{NP} \rightarrow \text{can} \\
\text{NP} \rightarrow \text{poison} \\
\text{M} \rightarrow \text{can} \\
\text{M} \rightarrow \text{must} \\
\text{V} \rightarrow \text{poison} \\
\text{V} \rightarrow \text{lead} \\
\end{array} \]
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

1  N, V  4  NP
NP, VP

min = 0

2  N, M
NP

min = 1

3  N, V
NP, VP

min = 2

6  ?

Inner rules

\[
S \rightarrow NP \ VP
\]

\[
VP \rightarrow M \ V
\]

\[
NP \rightarrow N
\]

\[
NP \rightarrow N \ NP
\]

Preterminal rules

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

M \rightarrow can
M \rightarrow must

V \rightarrow poison
V \rightarrow lead
**Preterminal rules**

- $S \rightarrow NP \ VP$

**Inner rules**

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

**Terminal rules**

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

- $N \rightarrow can$
- $N \rightarrow lead$
- $N \rightarrow poison$
- $M \rightarrow can$
- $M \rightarrow must$
- $V \rightarrow poison$
- $V \rightarrow lead$

### Table

<table>
<thead>
<tr>
<th></th>
<th>min = 0</th>
<th>min = 1</th>
<th>min = 2</th>
<th>max = 1</th>
<th>max = 2</th>
<th>max = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$N, V$</td>
<td>$N, M$</td>
<td></td>
<td>$NP, VP$</td>
<td>$S, NP$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$NP$</td>
<td>$S, NP$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$NP$</td>
<td>$NP$</td>
<td></td>
<td>$NP$</td>
<td>$NP, VP$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$NP$</td>
<td>$NP$</td>
<td></td>
<td>$NP$</td>
<td>$NP, VP$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$NP$</td>
<td>$NP$</td>
<td></td>
<td>$NP$</td>
<td>$NP, VP$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$NP$</td>
<td>$NP$</td>
<td></td>
<td>$NP$</td>
<td>$NP, VP$</td>
<td></td>
</tr>
</tbody>
</table>

### Mid = 1

- $mid = 1$
<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

1. \(N, V\), \(NP, VP\)
2. \(N, M\), \(NP\)
3. \(N, V\), \(NP, VP\)
4. \(NP\)
5. \(S, VP, NP\)
6. \(S, NP, S(?)\)
```

**Preterminal rules**

- \(S \rightarrow NP \ VP\)
- \(VP \rightarrow M \ V\)
- \(VP \rightarrow V\)
- \(NP \rightarrow N\)
- \(NP \rightarrow N \ NP\)

**Inner rules**

- \(N \rightarrow can\)
- \(N \rightarrow lead\)
- \(N \rightarrow poison\)
- \(M \rightarrow can\)
- \(M \rightarrow must\)
- \(V \rightarrow poison\)
- \(V \rightarrow lead\)
Apparently the sentence is ambiguous for the grammar: (as the grammar overgenerates)
PCFGs

\[ p(T) = 1.0 \times 0.2 \times 1.0 \times 0.4 \times 0.5 \times 0.3 \times 0.5 \times 0.3 \times 0.2 \times 1.0 \times 0.6 \times 0.5 \times 0.3 \times 0.7 \]
\[ = 2.26 \times 10^{-5} \]
CKY with PCFGs

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

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

\[ C \rightarrow C_1 \quad C_2 \]

For every \( C \) choose \( C_1, C_2 \) and mid such that

\[ P(T_1) \times P(T_2) \times P(C \rightarrow C_1C_2) \]

is maximal, where \( T_1 \) and \( T_2 \) are left and right subtrees.
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 -> 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)
          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
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
Consider

\[ NP \rightarrow DT\ NNP\ VBG\ NN \]

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?

\[
\begin{align*}
NP & \rightarrow DT \ X \\
X & \rightarrow NNP \ Y \\
Y & \rightarrow VBG \ NN
\end{align*}
\]
Transformation to CNF form: binarization

- **Consider**
  
  \[ NP \rightarrow DT \ NNP \ VBG \ NN \]

  \[
  \begin{array}{c}
  NP \\
  \downarrow \\
  DT \quad NNP \quad VBG \quad NN \\
  \mid \\
  the \quad Dutch \quad publishing \quad group
  \end{array}
  \]

- **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**

  \[
  NP \rightarrow DT \ @NP\|DT \\
  @NP\|DT \rightarrow NNP \ @NP\|DT\_NNP \\
  @NP\|DT\_NNP \rightarrow VBG \ NN
  \]
Transformation to CNF form: binarization

- Consider

  \[ NP \rightarrow DT \ NNP \ VBG \ NN \ 0.2 \]

  \[
  \begin{align*}
  NP & : 0.2 \\
  DT & : \text{the} \\
  NNP & : \text{Dutch} \\
  VBG & : \text{publishing} \\
  NN & : \text{group}
  \end{align*}
  \]

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

  \[ NP \rightarrow DT \ X \ 1.0 \]
  \[ X \rightarrow NNP \ Y \ 1.0 \]
  \[ Y \rightarrow VBG \ NN \ 0.2 \]
Transformation to CNF form: binarization

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

  ![Binarized Tree Diagram]

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

  Can be easily reversed on postprocessing.
Unary Rules

- CNF includes only two types of rules:
  
  \[ C \rightarrow x \]
  \[ C \rightarrow C_1 C_2 \]

- What about unary rules:

  \[ C \rightarrow C_1 \]
Unary Rules

**CFG**

A → X
B → X
C → X
...
X → C₁C₂
...
X → run
X → play
X → sleep
X → love

**CNF**

A → run,
B → run,
C → run,
X → run,
A → play,
B → play,
C → play,
X → play,
A → sleep,
B → sleep,
C → sleep,
X → sleep,
A → love
B → love
C → love
X → love
...
...
...
...
A → C₁C₂
B → C₁C₂
C → C₁C₂
X → C₁C₂

- explode the grammar
- make it hard to reverse
Unary rules

- How to integrate unary rules $C \rightarrow C_1$?

  for each max from 1 to n
  for each min from max - 1 down to 0

  // First, try all binary rules as before.

  ...

  // Then, try all unary rules.

  for each syntactic category C

    for each unary rule C $\rightarrow C_1$

    chart[min][max][C] = maximum (chart[min][max][C],
                                chart[min][max][C_1])
Unary closure

- What if the grammar contained 2 rules:
  
  $A \rightarrow B$
  
  $B \rightarrow C$

- But C can be derived from A by a chain of rules:
  
  $A \rightarrow B \rightarrow C$

- One could support chains in the algorithm but it is easier to extend the grammar, to get the transitive closure

  $A \rightarrow B$
  
  $B \rightarrow C$  \[\Rightarrow\]
  
  $A \rightarrow B$
  
  $B \rightarrow C$
  
  $A \rightarrow C$
Why unary closure

A → B
B → C

// Then, try all unary rules.

for each syntactic category C
for each unary rule C → C,

if chart[min][max][C] then
chart[min][max][C] = true
Why unary closure

A → B
B → C

A → C

scenario 1

C

B → C

A → B

C, B

S?
Why unary closure

A → B
B → C

A → C

scenario 1

B → C

A → B

scenario 2

A → B

B → C

C, B

A

C, B

C, B, A
Unary closure

- What if the grammar contained 2 rules:
  \[ A \rightarrow B \]
  \[ B \rightarrow C \]

- But C can be derived from A by a chain of rules:
  \[ A \rightarrow B \rightarrow C \]

- One could support chains in the algorithm but it is easier to extend the grammar, to get the transitive closure

\[
\begin{align*}
A &\rightarrow B \\
B &\rightarrow C
\end{align*}
\Rightarrow
\begin{align*}
A &\rightarrow B \\
B &\rightarrow C \\
A &\rightarrow C
\end{align*}
\]

Convenient for programming reasons in the PCFG case
Unary (reflexive transitive) closure

\[
\begin{align*}
A &\rightarrow B & 0.1 &\Rightarrow & A &\rightarrow B & 0.1 &\Rightarrow & A &\rightarrow A & 1 \\
B &\rightarrow C & 0.2 & & B &\rightarrow C & 0.2 & & B &\rightarrow B & 1 \\
& & & & A &\rightarrow C & 0.2 \times 0.1 & & C &\rightarrow C & 1
\end{align*}
\]

Note that this is not a PCFG anymore as the rules do not sum to 1 for each parent.
Unary (reflexive transitive) closure

\[
\begin{align*}
A & \rightarrow B & 0.1 \\
B & \rightarrow C & 0.2 \\
A & \rightarrow C & 0.2 \times 0.1 \\
A & \rightarrow A & 1 \\
B & \rightarrow B & 1 \\
C & \rightarrow C & 1
\end{align*}
\]

The fact that the rule is composite needs to be stored to recover the true tree.

Note that this is not a PCFG anymore as the rules do not sum to 1 for each parent.
Unary (reflexive transitive) closure

Note that this is not a PCFG anymore as the rules do not sum to 1 for each parent.

The fact that the rule is composite needs to be stored to recover the true tree.

What about loops, like: \( A \rightarrow B \rightarrow A \rightarrow C \)?
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 do we store in backpointers?
  - rule
  - for binary rules, midpoint

- Be careful with unary rules
  - Basically you can assume that you always used an unary rule from the closure (but it could be the trivial one $C \rightarrow C$)
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
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.
Preview: 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)
Estimating PCFGs
Estimating PCFGs

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

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

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

\[
\begin{align*}
S & \rightarrow NP \ VF & 1.0 & (NP \ A \ girl \) \ (VP \ ate \ a \ sandwich) \\
VP & \rightarrow V & 0.2 & \\
VP & \rightarrow V \ NF & 0.4 & (VP \ ate) \ (NP \ a \ sandwich) \\
VP & \rightarrow VP \ PF & 0.4 & (VP \ saw \ a \ girl) \ (PP \ with \ …) \\
NP & \rightarrow NP \ PF & 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 \ NF & 1.0 & (P \ with) \ (NP \ with \ a \ sandwich) \\
\end{align*}
\]

\[
\begin{align*}
\bar{N} & \rightarrow \text{girl} & 0.2 \\
N & \rightarrow \text{telescope} & 0.7 \\
N & \rightarrow \text{sandwich} & 0.1 \\
PN & \rightarrow I & 1.0 \\
V & \rightarrow \text{saw} & 0.5 \\
V & \rightarrow ate & 0.5 \\
P & \rightarrow with & 0.6 \\
P & \rightarrow in & 0.4 \\
D & \rightarrow a & 0.3 \\
D & \rightarrow \text{the} & 0.7 \\
\end{align*}
\]
Estimating PCFGs: Intuition

- Probabilistic Regular Grammar

\[ N^i \rightarrow w^j N^k \]

\[ N^i \rightarrow w^j \]

Start state, \( N^1 \)
Estimating PCFGs: Intuition

- Probabilistic Regular Grammar

\[ N^i \rightarrow w^j N^k \]
\[ N^i \rightarrow w^j \]
Start state, \( N^1 \)

[Credit: Chris Manning]
Estimating PCFGs: Intuition

- Probabilistic Regular Grammar

\[ N^i \rightarrow w_j N^k \]
\[ N^i \rightarrow w^j \]

Start state, \( N^1 \)

\[ X: \quad N^1 \rightarrow N' \rightarrow N' \rightarrow N' \rightarrow \text{sink} \]

\[ O: \quad \text{the} \rightarrow \text{big} \rightarrow \text{brown} \rightarrow \text{box} \]
Estimating PCFGs: Intuition

\[ X: \quad \text{NP} \rightarrow \text{N'} \rightarrow \text{N'} \rightarrow \text{N'} \rightarrow \text{sink} \]

\[ O: \quad \text{the} \quad \text{big} \quad \text{brown} \quad \text{box} \]

\[ N^1 \]

\[ \text{the} \]
\[ N' \]
\[ \text{big} \]
\[ N' \]
\[ \text{brown} \]
\[ N^0 \]
\[ \text{box} \]
Estimating PCFGs: Intuition

\[
X: \quad \text{NP} \rightarrow \text{N'} \rightarrow \text{N'} \rightarrow \text{N'} \rightarrow \text{sink}
\]

\[
O: \quad \text{the} \quad \text{big} \quad \text{brown} \quad \text{box}
\]

\[
N^1
\]

\[
\alpha
\]

\[
\beta
\]
Unsupervised estimation of PCFGs

- Notation
- Calculating inside probabilities
- Calculating outside probabilities
- The inside-outside algorithm (EM) - preview
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 $\mathcal{N}^j$ spans $w_{pq}$ in the sentence

```
   VP
   / \                  VP_{13} =
  /   \                ate
 V   NP
/ \      DET
ate the
/ \     N
the orange
```

$w_1 \quad \text{p} \quad w_2 \quad \text{the} \quad w_3 \quad \text{orange}$
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:

\[ \beta_j(p, q) = \sum_{rs} \sum_{d=p}^{q-1} P(N_j \rightarrow N_r^j N_s^j) \beta_r(p, d) \beta_s(d + 1, q) \]

The grammar is binarized

let's draw...
Implementation: PCFG parsing

\[
\begin{align*}
\text{for each } & \text{ max from 2 to n} \\
\text{for each } & \text{ min from max - 2 down to 0} \\
\text{for each } & \text{ syntactic category } C \\
\text{double } & \text{ best = undefined} \\
\text{for each } & \text{ binary rule } C \rightarrow C_1 C_2 \\
\text{for each } & \text{ mid from min + 1 to max - 1} \\
\text{double } & t_1 = \text{ chart}[\text{min}][\text{mid}][C_1] \\
\text{double } & t_2 = \text{ chart}[\text{mid}][\text{max}][C_2] \\
\text{double } & \text{ candidate = } t_1 \times t_2 \times p(C \rightarrow C_1 C_2) \\
\text{if } & \text{ candidate > best then} \\
\text{best = candidate} \\
\text{chart}[\text{min}][\text{max}][C] = \text{ best}
\end{align*}
\]
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] = total
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] = total

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

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

\[ \beta_{DET}(1,1) \]
\[ \beta_N(2,2) \]
\[ \beta_{NP}(1,2) \]
\[ \beta_{NP}(1,2) \]
Inside probability: example

\[ \beta_{DET}(1,1) = P(\text{the} \mid DET_{11}, G) = P(DET \rightarrow \text{the} \mid G) = 0.4 \]

\[ \beta_N(2,2) \]

\[ \beta_{NP}(1,2) \]

\[ \beta_{NP}(1,2) \]
Inside probability: example

\[ \beta_{DET}(1,1) = P(\text{the} \mid DET_{11}, G) = P(DET \to \text{the} \mid G) = 0.4 \]

\[ \beta_{N}(2,2) = P(N \to \text{orange} \mid G) = 0.2 \]

\[ \beta_{NP}(1,2) \]

\[ \beta_{NP}(1,2) \]
Inside probability: example

\[
\beta_{DET}(1,1) = P(\text{the} \mid \text{DET}_{11}, G) = P(\text{DET} \rightarrow \text{the} \mid G) = 0.4
\]

\[
\beta_N(2,2) = P(N \rightarrow \text{orange} \mid G) = 0.2
\]

\[
\beta_{NP}(1,2) = P(\text{NP} \rightarrow \text{DET} \cdot N)\beta_{DET}(1,1)\beta_N(2,2)
\]

\[
= 0.8 \times 0.4 \times 0.2 = 0.064
\]

\[
\beta_{NP}(1,2) = 0.064
\]
Inside probability: example

\[ \beta_{DE}^{DET}, (1,1) = P(\text{the} \mid DET_{11}, G) = P(DET \rightarrow \text{the} \mid G) = 0.4 \]

\[ \beta_{N}^{N}, (2,2) = P(\text{orange} \mid G) = 0.2 \]

\[ \beta_{NP}^{NP}, (1,2) = P(NP \rightarrow DET \cdot N) \beta_{DE}^{DET}, (1,1) \times \beta_{N}^{N}, (2,2) \]

\[ = 0.8 \times 0.4 \times 0.2 \]

\[ \beta_{NP}^{NP}, (1,2) = 0.064 \]

\[ \beta_S(1, m) = P(S \rightarrow w_1, \ldots, w_m \mid G') \]
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
  \[ \alpha_1(1, m) = \alpha_S(1, m) = 1 \quad \alpha_{i \neq 1}(1, m) = 0 \]

- Induction?

- Intuition: \( N_{pq}^j \) must be either the L or R child of a parent node. We first consider the case when it is the L child.
The yellow area is the probability we would like to calculate

- How do we decompose it?
Calculating outside probability

Step 1: We assume that $N_{pe}^f$ is the parent of $N_{pt}^j$. 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 1: The red shaded area is the inside probability for $N_{(q+1)e}^g$, i.e. $\beta_q(q + 1, e)$
Step 3: The blue shaded area is just the production $N^f \rightarrow N^j N^g$, the corresponding probability $P(N^f \rightarrow N^j N^g | N^f, G)$.
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 $e$.

What if we do not want to assume this?
The joint probability corresponding to the yellow, red and blue areas, assuming $N^j_p$ 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:

\[ \sum_{f,g}^{p-1} \sum_{e=1}^{p-1} \alpha_f(e,q) \cdot \beta_g(e,p-1) \cdot P(N^f \rightarrow N^g N^j) \]
Calculating outside probability

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

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

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

\[
\alpha_j(p, q) = \sum_{f, g \neq j} \sum_{e=q+1}^{m} \alpha_f(p, e) \cdot \beta_g(q + 1, e) \cdot P(N^f \rightarrow N^j N^g) + \sum_{f, g} \sum_{e=1}^{p-1} \alpha_f(e, q) \cdot \beta_g(e, p - 1) \cdot P(N^f \rightarrow N^g N^j)
\]
Inside-outside algorithm

For PCFGs we need to compute:

$$\theta^t = P(N^j \rightarrow N^r N^s | N^j)$$
Given two events, $x$ and $y$, the maximum likelihood estimation (MLE) for their conditional probability is:

$$P(x \mid y) = \frac{\text{count}(x, y)}{\text{count}(x)}$$

If they are observable, it’s easy to see what to do: just count the events in a representative corpus and use the MLE.
What are the hidden variables that cannot be observed directly?

Use a model $\mu$ and iteratively improve the model based on a corpus of observable data ($O$) generated by the hidden variables:

$$P_{\mu}(x \mid y) = \frac{E_\mu[\text{count}(x, y) \mid O]}{E_\mu[\text{count}(x) \mid O]}$$

It is worth noting that if you know how to calculate the numerator, the denominator is trivially derivable.
By updating \(\mu\) and iterating, the model converges to at least a local maximum.

This can be proven, but I will not do it here.
The inside-outside algorithm

- Goal: estimate a model $\mu$ that is a PCFG (in Chomsky normal form) that characterizes a corpus of text.

- Required input:
  - Size of non-terminal vocabulary, $n$
  - At least one sentence to be modeled, $O$
The inside-outside algorithm

- Stated with the general schema described earlier, we seek to the MLE probabilities for productions in the grammar

\[ P(N^j \rightarrow N^r N^s \mid N^j) = \frac{\text{count}(N^j \rightarrow N^r N^s, N^j)}{\text{count}(N^j)} \]

- (Observe that this would be trivially easy to calculate this with a treebank, since the non-terminals are observable in a treebank)
The inside-outside algorithm

- Since the non-terminals are not visible, we can use EM to estimate the probabilities iteratively:

\[
P_{\hat{\mu}}(N^j \rightarrow N^r N^s | N^j) = \frac{E_\mu[\text{count}(N^j \rightarrow N^r N^s, N^j) | O]}{E_\mu[\text{count}(N^j) | O]}
\]
To be continued...

- Next: recitation on EM