Natural Language Models and Interfaces
Part B, lecture 2

Ivan Titov

Institute for Logic, Language and Computation
Today

- Parsing algorithms for CFGs
  - Recap, Chomsky Normal Form (CNF)
  - A dynamic programming algorithm for parsing (CKY)
  - Extension of CKY to support unary inner rules

- Parsing for PCFGs
  - Extension of CKY for parsing with PCFGs

- Parser evaluation (if we have time)

After this lecture you should be able to start working on the assignment step 2.1
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):
  \[
  \arg \max_{T \in G(x)} P(T)
  \]
  The probability by the PCFG model

- 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 (and beyond)

- 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):
  
  \[
  C \rightarrow x \\
  C \rightarrow C_1 C_2
  \]

- 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 but defining such transformations that allows for easy reverse transformation

Unary preterminal rules (generation of words given PoS tags \( N \rightarrow \text{telescope} \), \( D \rightarrow \text{the} \), ...)

Binary inner rules (e.g., \( S \rightarrow NP \ VP \), \( NP \rightarrow D \ N \))

Makes linguists unhappy
Transformation to CNF form

- What one need to do to convert to CNF form
  - Get rid of empty (aka epsilon) productions: \( C \rightarrow \epsilon \)
  - Get rid of unary rules: \( C \rightarrow C_1 \)
  - N-ary rules: \( C \rightarrow C_1 C_2 \ldots C_n \) \((n > 2)\)

Generally not a problem as there are not empty production in the standard (postprocessed) treebanks.

Not a problem, as our CKY algorithm will support unary rules.

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

» Consider \( NP \rightarrow DT \ NNP \ VBG \ NN \)

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

» 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

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

```
NP
  |      |      |      |
DT    NNP    VBG    NN
  the   Dutch  publishing  group
```

```
NP
  |      |
DT    @NP->_DT
  the

NP
  |      |      |
NNP    @NP->_DT_NNP
  |      |
Dutch    VBG    NN
  publishing  group
```

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

Can be easily reversed on postprocessing
Transformation to CNF form: binarization

Instead of binarizing rules we can binarize trees on preprocessing:

```
NP
  / \  /
 DT  NNP  VBG  NN
  |    |    |    |
  the Dutch publishing group

NP
  /        /
 DT  @NP->_DT
  |    |
  the

NP
  /          /
 NNP  @NP->_DT_NNP
  |            |
  Dutch

NP
  /          /
 VBG  @NP->_DT_NNP_VBG
  |            |
  publishing

NN
  |
  group
```

This is exactly the transformation used in the code of the assignment.
We are given

- a grammar \( G = (V, \Sigma, R, S) \)
- a sequence of words \( w = (w_1, w_2, \ldots, w_n) \)

Our goal is to produce a parse tree for \( w \)

We need an easy way to refer to substrings of \( w \)

\( \text{span} (i, j) \) refers to words between fenceposts \( i \) and \( j \)
Key problems

- **Recognition problem:** does the sentence belong to the language defined by CFG?
  - Is there a derivation which yields the sentence?

- **Parsing problem:** what is a derivation (tree) corresponding the sentence?
  - Probabilistic parsing: what is the most probable tree for the sentence?
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 longer spans

$C \rightarrow C_1 \ C_2$

Check through all $C_1, C_2, mid$

covers all words btw $min$ and $mid$
covers all words btw $mid$ and $max$
Parsing longer spans

$C \rightarrow C_1 \ C_2$

Check through all $C_1, C_2, mid$

covers all words btw $min$ and $mid$
covers all words btw $mid$ and $max$
Parsing longer spans

covers all words
between \textit{min} and \textit{max}
Applications of rules is independent of inner structure of a parse tree.

We only need to know the corresponding span and the root label of the tree.

- Its signature $[min, max, C]$ is also known as an edge.
Compute for every span a set of admissible labels (may be empty for some spans)

- Start from small trees (single words) and proceed to larger ones

When done, check if $S$ is among admissible labels for the whole sentence, if yes – the sentence belong to the language

- That is if a tree with signature $[0, n, S]$ exists

Unary rules?
# CKY in action

<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**

- $N \rightarrow can$
- $N \rightarrow lead$
- $N \rightarrow poison$
- $M \rightarrow can$
- $M \rightarrow must$
- $V \rightarrow poison$
- $V \rightarrow lead$
CKY in action

<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>

\[
\begin{array}{|c|c|c|}
\hline
\text{min} = 0 & \text{max} = 1 & S? \\
\hline
\text{min} = 1 & \text{max} = 2 \\
\hline
\text{min} = 2 \\
\hline
\end{array}
\]

Preterminal rules

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

Chart (aka parsing triangle)
CKY in action

<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

S?
CKY in action

<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

\[ S \rightarrow NP \ VP \]

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

Inner rules

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

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

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

\[ V \rightarrow poison \]
\[ V \rightarrow lead \]
### CKY in action

<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>

```
lead       can     poison
max = 1    max = 2    max = 3
min = 1    min = 2    min = 3
```

**Preterminal rules**

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

**Inner rules**

- \( NP \rightarrow N \)
- \( NP \rightarrow N \ NP \)
- \( N \rightarrow can \)
- \( N \rightarrow lead \)
- \( N \rightarrow poison \)
- \( M \rightarrow can \)
- \( M \rightarrow must \)
- \( V \rightarrow poison \)
- \( V \rightarrow lead \)
CKY in action

<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 4 6 

min = 1  

2 5 

min = 2  

3 

$S \rightarrow NP \ VP$

$VP \rightarrow M \ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N \ NP$

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$
CKY in action

<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

\[
\begin{array}{ccc}
1 & ? & \\
2 & ? & \\
3 & ? & \\
\end{array}
\]

Preterminal rules

\[
S \rightarrow NP \ VP
\]

\[
VP \rightarrow M \ V
\]

\[
VP \rightarrow V
\]

\[
NP \rightarrow N
\]

\[
NP \rightarrow N \ NP
\]

Inner rules

\[
N \rightarrow can
\]

\[
N \rightarrow lead
\]

\[
N \rightarrow poison
\]

\[
M \rightarrow can
\]

\[
M \rightarrow must
\]

\[
V \rightarrow poison
\]

\[
V \rightarrow lead
\]
CKY in action

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

max = 1  max = 2  max = 3

Preterminal rules

Inner rules

\[
\begin{align*}
S & \rightarrow NP \ VP \\
VP & \rightarrow M \ V \\
VP & \rightarrow V \\
NP & \rightarrow N \\
NP & \rightarrow N \ NP \\
N & \rightarrow can \\
N & \rightarrow lead \\
N & \rightarrow poison \\
M & \rightarrow can \\
M & \rightarrow must \\
V & \rightarrow poison \\
V & \rightarrow lead
\end{align*}
\]
CKY in action

<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>1</th>
<th>N, V</th>
</tr>
</thead>
<tbody>
<tr>
<td>min = 1</td>
<td>2</td>
<td>N, M</td>
</tr>
<tr>
<td>min = 2</td>
<td>3</td>
<td>N, V</td>
</tr>
</tbody>
</table>

max = 1 max = 2 max = 3

Preterminal rules

Inner rules

\[
S \rightarrow NP \ VP
\]

\[
VP \rightarrow M \ V
\]

\[
VP \rightarrow V
\]

\[
NP \rightarrow N
\]

\[
NP \rightarrow N \ NP
\]

\[
N \rightarrow can
\]

\[
N \rightarrow lead
\]

\[
N \rightarrow poison
\]

\[
M \rightarrow can
\]

\[
M \rightarrow must
\]

\[
V \rightarrow poison
\]

\[
V \rightarrow lead
\]
CKY in action

<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>$N, V$</td>
<td>$N, M$</td>
<td>$N, V$</td>
</tr>
<tr>
<td>$NP, VP$</td>
<td>$NP$</td>
<td>$NP, VP$</td>
</tr>
</tbody>
</table>

max = 1, max = 2, max = 3

Preterminal rules

$S \rightarrow NP \ VP$

Inner rules

$VP \rightarrow M \ V$

$NP \rightarrow N$

$NP \rightarrow N \ VP$

Preterminal rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$
CKY in action

<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>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N, V$</td>
<td>$N, M$</td>
<td>$N, V$</td>
</tr>
<tr>
<td>$NP, VP$</td>
<td>$NP$</td>
<td>$NP, VP$</td>
</tr>
</tbody>
</table>

$S \rightarrow NP \ VP$

$VP \rightarrow M \ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N \ NP$

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Preterminal rules

Inner rules
CKY in action

<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

<table>
<thead>
<tr>
<th>Preterminal rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
</tr>
<tr>
<td>$VP \rightarrow M \ V$</td>
</tr>
<tr>
<td>$VP \rightarrow V$</td>
</tr>
</tbody>
</table>

Inner rules

<table>
<thead>
<tr>
<th>Inner rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$NP \rightarrow N$</td>
</tr>
<tr>
<td>$NP \rightarrow N \ NP$</td>
</tr>
<tr>
<td>$N \rightarrow \text{can}$</td>
</tr>
<tr>
<td>$N \rightarrow \text{lead}$</td>
</tr>
<tr>
<td>$N \rightarrow \text{poison}$</td>
</tr>
<tr>
<td>$M \rightarrow \text{can}$</td>
</tr>
<tr>
<td>$M \rightarrow \text{must}$</td>
</tr>
<tr>
<td>$V \rightarrow \text{poison}$</td>
</tr>
<tr>
<td>$V \rightarrow \text{lead}$</td>
</tr>
</tbody>
</table>
CKY in action

<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>

min = 0

min = 1

min = 2

max = 0

max = 1

max = 2

max = 3

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

Inner rules
CKY in action

<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>

Check about unary rules: no unary rules here

```
min = 0
N, V
NP, VP
```

```
min = 1
N, M
NP
```

```
min = 2
N, V
NP, VP
```

```
max = 1
S \rightarrow NP \ VP
```

```
max = 2
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
```

```
Inner rules
M \rightarrow can
M \rightarrow must
```

```
V \rightarrow poison
V \rightarrow lead
```
CKY in action

<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>

min = 0

\[
\begin{array}{|c|c|c|}
\hline
& 1 & 4 \\
\hline
1 & N, V & N, M \\
& N, V & N \\
\hline
\end{array}
\]

max = 1

\[
\begin{array}{|c|c|c|}
\hline
& 2 & 5 \\
\hline
2 & N, M & ? \\
& N & N, V \\
\hline
\end{array}
\]

max = 2

\[
\begin{array}{|c|c|c|}
\hline
& 3 & \\
\hline
3 & N, V & \\
& N, V & \\
\hline
\end{array}
\]

max = 3

Preterminal rules

- \( S \rightarrow NP \ VP \)

Inner rules

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

Terminal rules

- \( VP \rightarrow M \ V \)
- \( VP \rightarrow V \)
### CKY in action

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

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

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

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

- $N \rightarrow can$
- $N \rightarrow lead$
- $N \rightarrow poison$
- $M \rightarrow can$
- $M \rightarrow must$
- $V \rightarrow poison$
- $V \rightarrow lead$
**CKY in action**

<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>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>( S, V, P, N P )</td>
</tr>
<tr>
<td>2</td>
<td>( N P )</td>
<td>( N P )</td>
<td>( N P, V P )</td>
</tr>
<tr>
<td>3</td>
<td>( N, V )</td>
<td>( N P, V P )</td>
<td></td>
</tr>
</tbody>
</table>

**Inner rules**

- \( S \rightarrow N P \ V P \)
- \( V P \rightarrow M \ V \)
- \( V P \rightarrow V \)
- \( N P \rightarrow N \)
- \( N P \rightarrow N \ N P \)

**Preterminal rules**

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

Check about unary rules: no unary rules here
### CKY in action

<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>2</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>max = 1</th>
<th>max = 2</th>
<th>max = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</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**

- $N \rightarrow can$
- $N \rightarrow lead$
- $N \rightarrow poison$
- $M \rightarrow can$
- $M \rightarrow must$
- $V \rightarrow poison$
- $V \rightarrow lead$
CKY in action

\[
\begin{array}{|c|c|c|c|}
\hline
\text{lead} & \text{can} & \text{poison} \\
\hline
0 & 1 & 2 & 3 \\
\hline
\end{array}
\]

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

\[
\text{max} = 1 \quad \text{max} = 2 \quad \text{max} = 3
\]

Inner rules

- \( S \rightarrow \text{NP VP} \)
- \( \text{VP} \rightarrow \text{M V} \)
- \( \text{VP} \rightarrow \text{V} \)
- \( \text{NP} \rightarrow \text{N} \)
- \( \text{NP} \rightarrow \text{N NP} \)

Preterminal rules

- \( \text{N} \rightarrow \text{can} \)
- \( \text{N} \rightarrow \text{lead} \)
- \( \text{N} \rightarrow \text{poison} \)
- \( \text{M} \rightarrow \text{can} \)
- \( \text{M} \rightarrow \text{must} \)
- \( \text{V} \rightarrow \text{poison} \)
- \( \text{V} \rightarrow \text{lead} \)
CKY in action

<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>

Min = 0

Min = 1

Min = 2

Mid = 1

Max = 1

Max = 2

Max = 3

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

Inner rules
CKY in action

<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>

\[
\begin{array}{c|c|c|c}
\text{min} & \text{max} & \text{max} & \text{max} \\
0 & 1 & 2 & 3 \\
\end{array}
\]

Preterminal rules

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

Inner rules

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

- \( N, V \)

\( NP, VP \)

\( S, NP \)

\( S(?!?) \)

\( N, M \)

\( NP \)

\( S, VP, NP \)

\( N, V \)

\( NP, VP \)
Apparently the sentence is ambiguous for the grammar: (as the grammar overgenerates)
No subject-verb agreement, and *poison* used as an intransitive verb

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

- Chart can be represented by a Boolean array \( \text{chart}[\text{min}][\text{max}][\text{C}] \)
- Relevant entries have \( 0 < \text{min} < \text{max} \leq n \)
- \( \text{chart}[\text{min}][\text{max}][\text{C}] = \text{true} \) if the signature \((\text{min}, \text{max}, \text{C})\) is already added to the chart; \( \text{false} \) otherwise.

Here we assume that labels \((\text{C})\) are integer indices

In the assignment code we use a class Chart but its access methods are similar
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
Implementation: unary rules

```plaintext
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 -> C₁

      if chart[min][max][C₁] then

        chart[min][max][C] = true

But we forgot something!
```
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 reflexive transitive closure

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

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

Convenient for programming reasons in the PCFG case
Implementation: skeleton

// int n = number of words in the sequence
// int m = number of syntactic categories in the grammar
// int s = the (number of the) grammar’s start symbol

boolean[][][] chart = new boolean[n + 1][n + 1][m]

// Recognize all parse trees built with preterminal rules.
// Recognize all parse trees built with inner rules.

return chart[0][n][s]
Algorithm analysis

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

  \[\theta(n^3|R|)\], where \(|R|\) is the number of rules in the grammar

- There exist algorithms with better asymptotical time complexity but the `constant' makes them slower in practice (in general)
Practical time complexity

- Time complexity? (for the PCFG version)

\[ \sim n^{3.6} \]

[Plot by Dan Klein]
Today

- Parsing algorithms for CFGs
  - Recap, Chomsky Normal Form (CNF)
  - A dynamic programming algorithm for parsing (CKY)
  - Extension of CKY to support unary inner rules

- Parsing for PCFGs
  - Extension of CKY for parsing with PCFGs

- Parser evaluation (if we have time)
Probabilistic parsing

- We discussed the recognition problem:
  - check if a sentence is parsable with a CFG

- Now we consider parsing with PCFGs
  - Recognition with PCFGs: what is the probability of the most probable parse tree?
  - Parsing with PCFGs: What is the most probable parse tree?
Distribution over trees

- Let us denote by $G(x)$ the set of derivations for the sentence $x$.
- The probability distribution defines the scoring $P(T)$ over the trees $T \in G(x)$.
- Finding the best parse for the sentence according to PCFG:

$$\arg \max_{T \in G(x)} P(T)$$
CKY with PCFGs

- Chart is represented by a **double array** `chart[min][max][C]`
  - 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

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.
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 -> 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
Unary (reflexive transitive) closure

\[ A \rightarrow B \quad 0.1 \quad \Rightarrow \quad A \rightarrow B \quad 0.1 \quad A \rightarrow A \quad 1 \]
\[ B \rightarrow C \quad 0.2 \quad B \rightarrow C \quad 0.2 \quad B \rightarrow B \quad 1 \]
\[ A \rightarrow C \quad 0.2 \times 0.1 \quad C \rightarrow C \quad 1 \]

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

\[ A \rightarrow B \quad 0.1 \quad A \rightarrow B \quad 0.1 \quad A \rightarrow A \quad 1 \]
\[ B \rightarrow C \quad 0.2 \quad B \rightarrow C \quad 0.1 \quad B \rightarrow B \quad 1 \]
\[ A \rightarrow C \quad 1.e - 5 \quad A \rightarrow C \quad 0.02 \quad C \rightarrow C \quad 1 \]

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 (e.g., rule and, for binary rules, midpoint)
  - start recovering from $[0, n, S]$

- 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 (approximate search)

- **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.
Today

- Parsing algorithms for CFGs
  - Recap, Chomsky Normal Form (CNF)
  - A dynamic programming algorithm for parsing (CKY)
  - Extension of CKY to support unary inner rules
- Parsing for PCFGs
  - Extension of CKY for parsing with PCFGs
- Parser 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
- **Crossing brackets**: percentage of phrases boundaries crossing
- **Dependency metrics**: scores dependency structure corresponding to the constituent tree (percentage of correctly identified heads)

The most standard measure; we will focus on it.
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
Bracketing notation

The same tree as a bracketed sequence

(S
  (NP (PN My) (N Dog) )
  (VP (V ate)
    (NP (D a ) (N sausage) )
  )
)

(S
  (NP (PN My) (N Dog) )
  (VP (V ate)
    (NP (D a ) (N sausage) )
  )
)}
Brackets scores

\[ Pr = \frac{\text{number of brackets the parser and annotation agree on}}{\text{number of brackets predicted by the parser}} \]

\[ Re = \frac{\text{number of brackets the parser and annotation agree on}}{\text{number of brackets in annotation}} \]

\[ F1 = \frac{2 \times Pr \times Re}{Pr + Re} \]

Harmonic mean of precision and recall
We will introduce these models next time.