Natural Language Processing CSCI 4152/6509 — Lecture 21 CYK Algorithm and PCFGs

Instructors: Vlado Keselj
Time and date: 16:05 – 17:25, 27-Nov-2024
Location: Carleton Tupper Building Theatre C

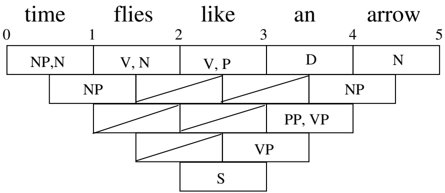
Previous Lecture

- Phrase structure in English (continued):
  - NP, VP, PP, ADJP, ADVP
- Heads and dependency, dependency tree
- **CYK Chart Parsing Algorithm**
- Chomsky Normal Form (CNF)
- CYK algorithm example

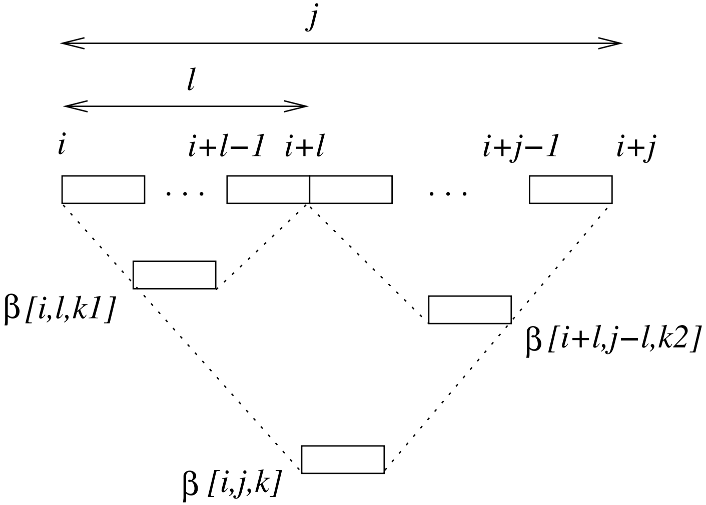
CYK Example

The following grammar in CNF is given:

S	$\rightarrow$	NP VP	VP	$\rightarrow$	V NP	N	$\rightarrow$	time	V	$\rightarrow$	like
NP	$\rightarrow$	time	VP	$\rightarrow$	V PP	N	$\rightarrow$	arrow	V	$\rightarrow$	flies
NP	$\rightarrow$	N N	PP	$\rightarrow$	P NP	N	$\rightarrow$	flies	P	$\rightarrow$	like
NP	$\rightarrow$	D N				D	$\rightarrow$	an			



Explanation of Index Use in CYK



CYK Algorithm

Require: sentence $= w_1 \dots w_n$, and a CFG in CNF with nonterminals $N^1 \dots N^m$,
N^1 is the start symbol

Ensure: parsed sentence

allocate matrix $\beta \in \{0, 1\}^{n \times n \times m}$ and initialize all entries to 0
for $i \leftarrow 1$ to n **do**
    for all rules $N^k \rightarrow w_i$ **do**
        $\beta[i, 1, k] \leftarrow 1$
for $j \leftarrow 2$ to n **do**
    for $i \leftarrow 1$ to $n - j + 1$ **do**
        for $l \leftarrow 1$ to $j - 1$ **do**
            for all rules $N^k \rightarrow N^{k_1} N^{k_2}$ **do**
                $\beta[i, j, k] \leftarrow \beta[i, j, k]$ OR $(\beta[i, l, k_1]$ AND $\beta[i + l, j - l, k_2])$
return $\beta[1, n, 1]$

Efficient Inference in PCFG Model

- consider marginalization task:

$\mathrm{P}(\text{sentence}) =?$

- or: $\mathrm{P}(\text{sentence}) = \mathrm{P}(w_1 w_2 \ldots w_n | S)$
- One way to compute:

$$\mathrm{P}(\text{sentence}) = \sum_{t \in T} \mathrm{P}(t),$$

- Likely inefficient; need a parsing algorithm

Efficient PCFG Marginalization

- Idea: adapt CYK algorithm to store marginal probabilities
- Replace algorithm line:

$$\beta[i,j,k] \leftarrow \beta[i,j,k] \text{ OR } (\beta[i,l,k_1] \text{ AND } \beta[i+l,j-l,k_2])$$

with

$$\beta[i,j,k] \leftarrow \beta[i,j,k] + \mathrm{P}(N^k \to N^{k_1} N^{k_2}) \cdot \beta[i,l,k_1] \cdot \beta[i+l,j-l,k_2]$$

- and the first-chart-row line:

$$\beta[i,1,k] \leftarrow 1$$

with

$$\beta[i,1,k] \leftarrow \mathrm{P}(N^k \to w_i)$$

Probabilistic CYK for Marginalization

Require: sentence $= w_1 \dots w_n$, and a PCFG in CNF with nonterminals $N^1 \dots N^m$, N^1 is the start symbol

Ensure: P(sentence) is returned

allocate $\beta \in \mathbb{R}^{n \times n \times m}$ and initialize all entries to 0

for $i \leftarrow 1$ to n **do**

  for all rules $N^k \rightarrow w_i$ **do**

    $\beta[i, 1, k] \leftarrow \mathrm{P}(N^k \rightarrow w_i)$

for $j \leftarrow 2$ to n **do**

  for $i \leftarrow 1$ to $n - j + 1$ **do**

    for $l \leftarrow 1$ to $j - 1$ **do**

      for all rules $N^k \rightarrow N^{k_1} N^{k_2}$ **do**

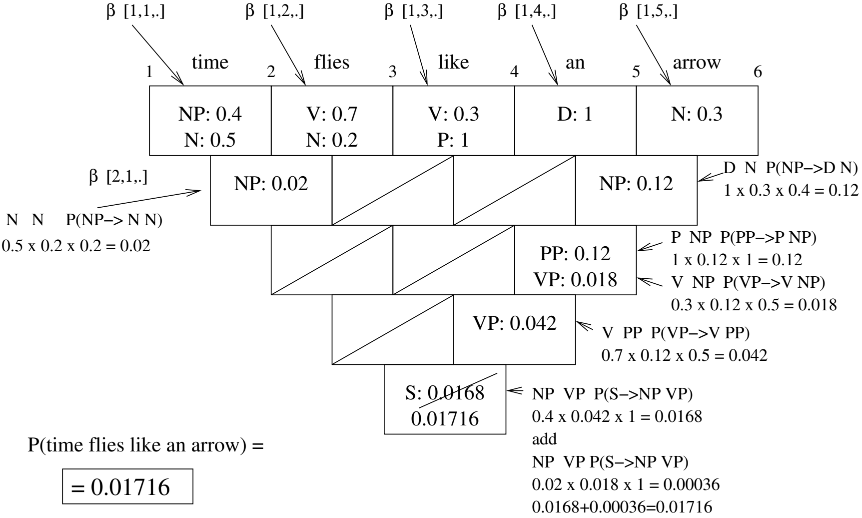
        $\beta[i, j, k] \leftarrow \beta[i, j, k] +$
        $\mathrm{P}(N^k \rightarrow N^{k_1} N^{k_2}) \cdot \beta[i, l, k_1] \cdot \beta[i + l, j - l, k_2]$

return $\beta[1, n, 1]$

PCFG Marginalization Example (grammar)

$$
\begin{array}{rcll}
S & \rightarrow & NP\ VP & /1 \\
NP & \rightarrow & time & /.4 \\
NP & \rightarrow & N\ N & /.2 \\
NP & \rightarrow & D\ N & /.4 \\
V & \rightarrow & like & /.3 \\
V & \rightarrow & flies & /.7 \\
P & \rightarrow & like & /1
\end{array}
\qquad
\begin{array}{rcll}
VP & \rightarrow & V\ NP & /.5 \\
VP & \rightarrow & V\ PP & /.5 \\
PP & \rightarrow & P\ NP & /1
\end{array}
\qquad
\begin{array}{rcll}
N & \rightarrow & time & /.5 \\
N & \rightarrow & arrow & /.3 \\
N & \rightarrow & flies & /.2 \\
D & \rightarrow & an & /1
\end{array}
$$

PCFG Marginalization Example (chart)



Conditioning

- Conditioning in the PCFG model: $\mathrm{P}(\mathrm{tree}|\mathrm{sentence})$
- Use the formula:

$$\mathrm{P}(\mathrm{tree}|\mathrm{sentence}) = \frac{\mathrm{P}(\mathrm{tree}, \mathrm{sentence})}{\mathrm{P}(\mathrm{sentence})} = \frac{\mathrm{P}(\mathrm{tree})}{\mathrm{P}(\mathrm{sentence})}$$

- $\mathrm{P}(\mathrm{tree})$ — directly evaluated
- $\mathrm{P}(\mathrm{sentence})$ — marginalization

Completion

- Finding the most likely parse tree of a sentence:

$$\arg\max_{\text{tree}} \mathrm{P}(\text{tree}|\text{sentence})$$

- Use the CYK algorithm in which line 9 is replaced with:

  9: $\beta[i,j,k] \leftarrow \max(\beta[i,j,k], \mathrm{P}(N^k \rightarrow N^{k_1} N^{k_2}) \cdot \beta[i,l,k_1] \cdot \beta[i+l, j-l, k_2])$

- Return the most likely tree

CYK-based Completion Algorithm

Require: sentence $= w_1 \ldots w_n$, and a PCFG in CNF with nonterminals $N^1 \ldots N^m$, N^1 is the start symbol

Ensure: The most likely parse tree is returned

allocate $\beta \in \mathbb{R}^{n \times n \times m}$ and initialize all entries to 0
for $i \leftarrow 1$ to n **do**
  for all rules $N^k \rightarrow w_i$ **do**
    $\beta[i, 1, k] \leftarrow \mathrm{P}(N^k \rightarrow w_i)$
for $j \leftarrow 2$ to n **do**
  for $i \leftarrow 1$ to $n - j + 1$ **do**
    for $l \leftarrow 1$ to $j - 1$ **do**
      for all rules $N^k \rightarrow N^{k_1} N^{k_2}$ **do**
        $\beta[i, j, k] \leftarrow \max(\beta[i, j, k], \mathrm{P}(N^k \rightarrow N^{k_1} N^{k_2}) \cdot \beta[i, l, k_1] \cdot \beta[i + l, j - l, k_2])$
return Reconstruct$(1, n, 1, \beta)$

Algorithm: Reconstruct(i, j, k, β)

Require: β — table from CYK, i — index of the first word, j — length of sub-string sentence, k — index of non-terminal

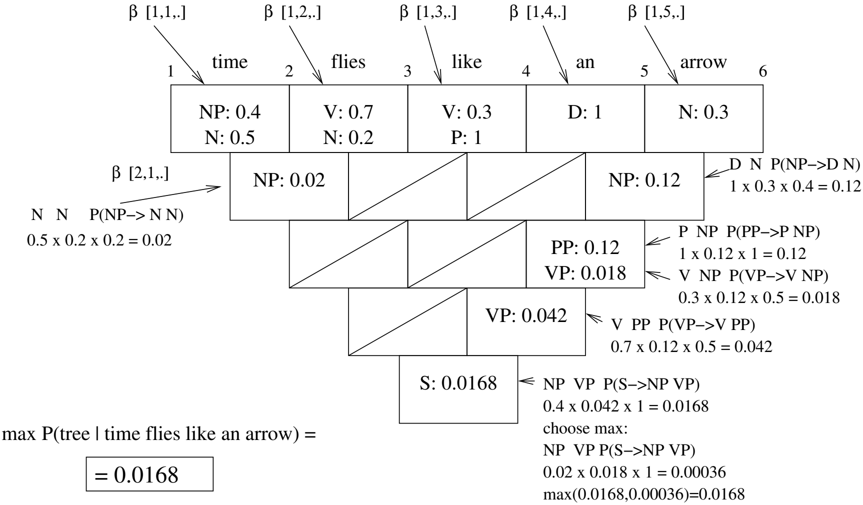
Ensure: a most probable tree with root N^k and leaves $w_i \ldots w_{i+j-1}$ is returned

```
if j = 1 then
    return tree with root N^k and child w_i
for l ← 1 to j − 1 do
    for all rules N^k → N^{k1} N^{k2} do
        if β[i, j, k] = P(N^k → N^{k1} N^{k2}) · β[i, l, k1] · β[i + l, j − l, k2] then
            create a tree t with root N^k
            t.left_child ← Reconstruct(i, l, k1, β)
            t.right_child ← Reconstruct(i + l, j − l, k2, β)
            return t
```

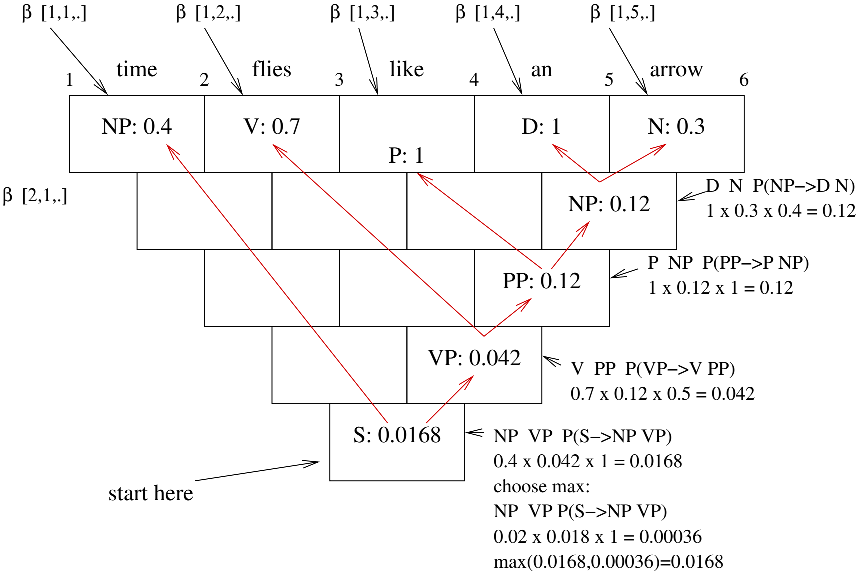
PCFG Completion Example (grammar)

$$
\begin{array}{llll}
S & \rightarrow & NP\ VP & /1 \\
NP & \rightarrow & time & /.4 \\
NP & \rightarrow & N\ N & /.2 \\
NP & \rightarrow & D\ N & /.4 \\
V & \rightarrow & like & /.3 \\
V & \rightarrow & flies & /.7 \\
P & \rightarrow & like & /1
\end{array}
\quad
\begin{array}{llll}
VP & \rightarrow & V\ NP & /.5 \\
VP & \rightarrow & V\ PP & /.5 \\
PP & \rightarrow & P\ NP & /1
\end{array}
\quad
\begin{array}{llll}
N & \rightarrow & time & /.5 \\
N & \rightarrow & arrow & /.3 \\
N & \rightarrow & flies & /.2 \\
D & \rightarrow & an & /1
\end{array}
$$

PCFG Completion Example (chart)

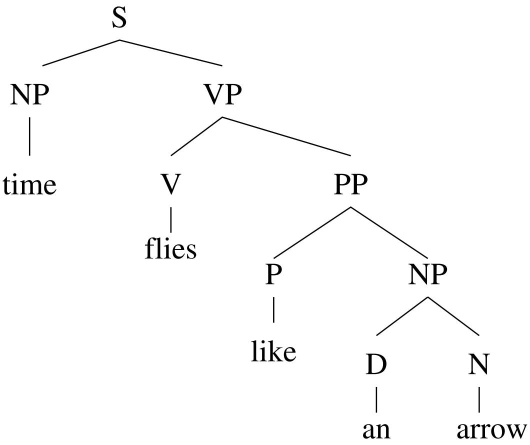


PCFG Completion Example (tree reconstruction)



PCFG Completion Example (final tree)

The most probable three:



Issues with PCFGs

1. Structural dependencies
   - Dependency on position in a tree
   - Example: consider rules NP → PRP and NP → DT NN
   - PRP is more likely as a subject than an object
   - NL parse trees are usually deeper on their right side
2. Lexical dependencies
   - Example: PP-attachment problem
   - In a PCFG, decided using probabilities for higher level rules; e.g., $\mathrm{NP} \rightarrow \mathrm{NP}\ \mathrm{PP}$, $\mathrm{VP} \rightarrow \mathrm{VBD}\ \mathrm{NP}$, and $\mathrm{VP} \rightarrow \mathrm{VBD}\ \mathrm{NP}\ \mathrm{PP}$
   - Actually, they frequently depend on the actual words

PP-Attachment Example

- Consider sentences:
  - "Workers dumped sacks into a bin." and
  - "Workers dumped sacks of fish."
- and rules:
  - $\mathrm{NP} \to \mathrm{NP}\ \mathrm{PP}$
  - $\mathrm{VP} \to \mathrm{VBD}\ \mathrm{NP}$
  - $\mathrm{VP} \to \mathrm{VBD}\ \mathrm{NP}\ \mathrm{PP}$

A Solution: Probabilistic Lexicalized CFGs

- use heads of phrases
- expanded set of rules, e.g.:
  VP(dumped) $\rightarrow$ VBD(dumped) NP(sacks) PP(into)
- large number of new rules
- sparse data problem
- solution: new independence assumptions
- proposed solutions by Charniak, Collins, etc. around 1999