Natural Language Processing
CSCI 4152/6509 — Lecture 24
Typical Phrase Structure of English

Instructors: Vlado Keselj
Time and date: 16:05 – 17:25, 30-Nov-2023
Location: Rowe 1011

Previous Lecture

- Definite Clause Grammar (DCG)
  - Basic DCG example
  - Building a parse tree in DCG
  - Agreement example in DCG
  - Embedded code in DCG
- Probabilistic Context-Free Grammars (PCFG)
- PCFG definition
- PCFG as a probabilistic model

Typical Phrase Structure Rules in English

- We will cover some typical phrase structure rules
- Specific to English but also generalizable to other languages
- **Not** *all rules are covered*, but the general principles should be adopted

Typical Sentence Rules (S)

```
S -> NP VP          Declarative sentences, e.g.:
               I want a flight from Halifax to Chicago.

S -> VP             Imperative sentences, e.g.:
                    Show the lowest fare.

S -> Aux NP VP      Yes-no questions, e.g.:
                   Do any of these flights have stops?
           Can you give me some information for United?

S -> Wh-NP VP      Wh-subject questions, e.g.:
                   What airlines fly from Halifax?

S -> Wh-NP Aux NP VP   Wh-non-subject questions, e.g.:
                  What flights do you have on Tuesday?
```

Noun Phrase (NP)

- typically: pronouns, proper nouns, or determiner-nominal construction
- some typical rules

```
NP -> PRP                            e.g.: you
NP -> NNP | NNPS                     e.g.: Halifax
NP -> PDT? DT JJ* NN PP*
NP -> NN NN                          e.g.: computer science
```

- in the last rule, we use regular expression notation to describe a set of different rules
- example: all the various flights from Halifax to Toronto
- determiners and nominals
- modifiers before head noun and after head noun
- postmodifier phrases `NP -> DT JJ* NN RelC`

Relative Clauses

- RelC — relative clause
- clause (sentence-like phrase) following a noun phrase
- example: gerundive relative clause:
  flights arriving after 5pm
- example: infinitive relative clause:
  flights to arrive tomorrow
- example: restrictive relative clause:
  flight that was canceled yesterday

Verb Phrase (VP)

- organizes arguments around the verb
- typical rules

```
VP -> Verb              intransitive verbs;
                        e.g.: disappear
VP -> Verb NP           transitive verbs:
                        e.g.: prefer a morning flight
VP -> Verb NP NP        ditransitive verbs:
                        e.g.: send me an email
VP -> Verb PP*          sentential complements
VP -> Verb NP PP*
VP -> Verb NP NP PP*
```

- sentential complements, e.g.:
  You said these were two flights that were the cheapest.

Prepositional Phrase (PP)

- Preposition (IN) relates a noun phrase to other word or phrase
- Prepositional Phrase (PP) consists of a preposition and the noun phrase which is an object of that preposition
- There is typically only one rule for the prepositional phrase: `PP -> IN NP`
- examples: from Halifax, before tomorrow, in the city
- PP-attachment ambiguity

Adjective Phrase (ADJP)

- less common
- examples:
  - She is *very sure of herself.*
  - . . . the *least expensive* fare . . .

Adverbial Phrase (ADVP)

- Example:
```
(S (NP preliminary findings)
    (VP were reported
        (ADVP (NP a year) ago)))
```
- another example: years ago

About Typical Rules

- Only some typical rules are presented
- For example: We see the cat, and you see a dog.
- The sentence could be described with: `S -> S CC S`
- Relative clauses are labeled in Penn treebank using SBAR ($\bar{S}$) non-terminal; e.g.: `(S (NP (NP Lorillard Inc.)`

```
          ,
          (NP (NP the unit)
              (PP of (NP (ADJP New York-based)
                         Loews Corp.)))
          (SBAR that
                (S (NP *gap*)
                   (VP makes (NP Kent cigarettes))))
          ,)
      (VP stopped (VP using (NP crocidolite))))
```

Heads and Dependency

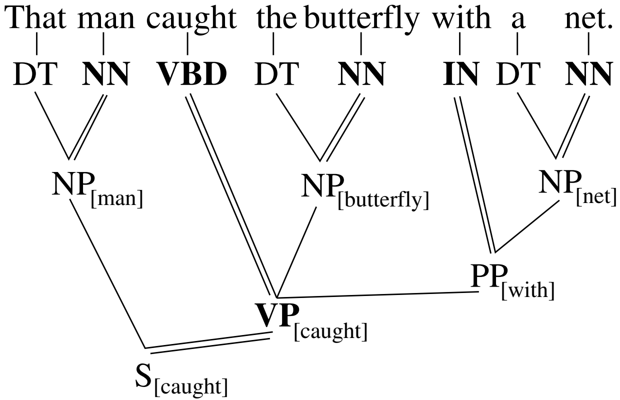
- a phrase typically has a central word called *head*, while other words are direct or indirect *dependents*
- a head is also called a *governor,* although sometimes these concepts are considered somewhat different
- phases are usually called by their head; e.g., the head of a noun phrase is a noun

Example with Heads and Dependencies

That man caught the butterfly with a net.

Example with Heads and Dependencies

- the parse tree of "That man caught the butterfly with a net."
- annotate dependencies, head words



Head-feature Principle

- Head Feature Principle:
  It is a principle that a set of characteristic features of a head word are transferred to the containing phrase.
- Examples of annotating head in a context-free rule:

$$NP \to DT\ NN_H$$

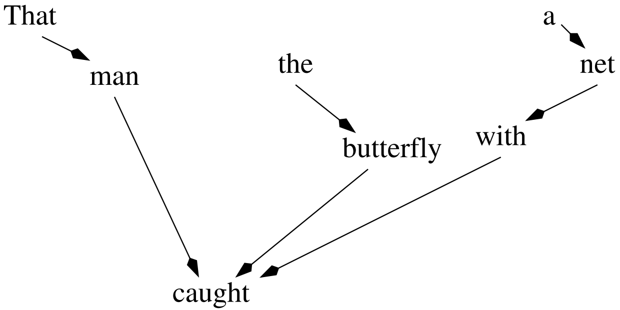
- or

$$[NP] \to [DT]\ H[NN]$$

- HPSG—Head-driven Phrase Structure Grammars

Dependency Tree

- dependency grammar
- example with "That man caught the butterfly with a net."



Arguments and Adjuncts

- There ar two kinds of dependents:
  1. **arguments,** which are required dependents, e.g.,
     We deprived him <u>of food.</u>
  2. **adjuncts,** which are not required;
     - ⋆ they have a "less tight" link to the head, and
     - ⋆ can be moved around more easily

     Example:
     We deprived him of food <u>yesterday</u> <u>in the restaurant</u>.

Efficient Inference in PCFG Model

- Using backtracking is not efficient approach
- Chart parsing is an efficient approach
- We will take a look at the CYK chart parsing algorithm

CYK Chart Parsing Algorithm

- When parsing NLP, there are generally two approaches:
  1. Backtracking to find all parse trees
  2. Chart parsing
- CYK algorithm: a simple chart parsing algorithm
- CYK: Cocke-Younger-Kasami algorithm
- CYK can be applied only to a CNF grammar

Chomsky Normal Form

- all rules are in one of the forms:
  1. $A \to B\,C$, where A, B, and C are nonterminals, or
  2. $A \to w$, where A is a nonterminal and w is a terminal
- If a grammar is not in CNF, it can be converted to it

Is the following grammar in CNF?

$$
\begin{array}{rcl}
\mathrm{S} & \to & \mathrm{NP\ VP} \\
\mathrm{NP} & \to & \mathrm{N} \\
\mathrm{NP} & \to & \mathrm{N\ N} \\
\mathrm{NP} & \to & \mathrm{D\ N}
\end{array}
\qquad
\begin{array}{rcl}
\mathrm{VP} & \to & \mathrm{V\ NP} \\
\mathrm{VP} & \to & \mathrm{V\ PP} \\
\mathrm{PP} & \to & \mathrm{P\ NP}
\end{array}
\qquad
\begin{array}{rcl}
\mathrm{N} & \to & \mathrm{time} \\
\mathrm{N} & \to & \mathrm{arrow} \\
\mathrm{N} & \to & \mathrm{flies} \\
\mathrm{D} & \to & \mathrm{an}
\end{array}
\qquad
\begin{array}{rcl}
\mathrm{V} & \to & \mathrm{like} \\
\mathrm{V} & \to & \mathrm{flies} \\
\mathrm{P} & \to & \mathrm{like}
\end{array}
$$

How about this grammar? (Is it in CNF?)

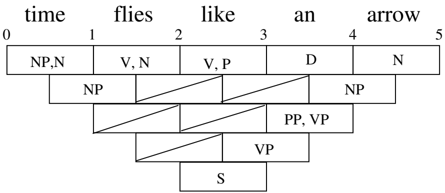
$$
\begin{array}{lll}
S & \rightarrow & NP\ VP \\
NP & \rightarrow & time \\
NP & \rightarrow & N\ N \\
NP & \rightarrow & D\ N
\end{array}
\qquad
\begin{array}{lll}
VP & \rightarrow & V\ NP \\
VP & \rightarrow & V\ PP \\
PP & \rightarrow & P\ NP
\end{array}
\qquad
\begin{array}{lll}
N & \rightarrow & time \\
N & \rightarrow & arrow \\
N & \rightarrow & flies \\
D & \rightarrow & an
\end{array}
\qquad
\begin{array}{lll}
V & \rightarrow & like \\
V & \rightarrow & flies \\
P & \rightarrow & like
\end{array}
$$

CYK Example: time flies like an arrow

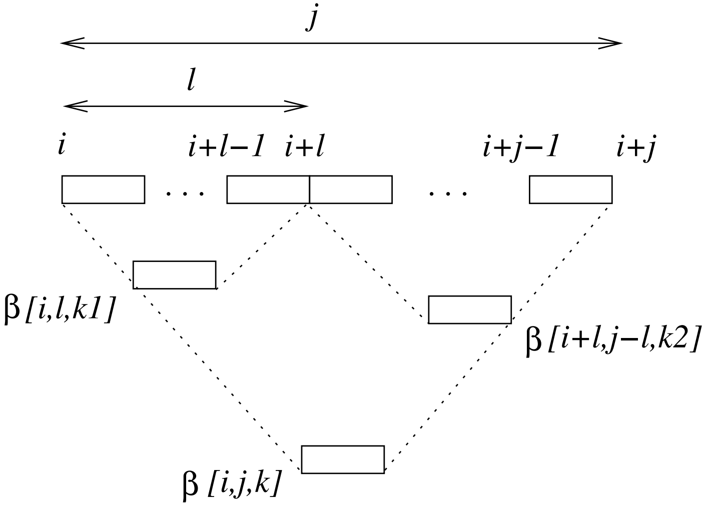
CYK Example

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

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

- and the first-chart-row line:

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

with

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

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

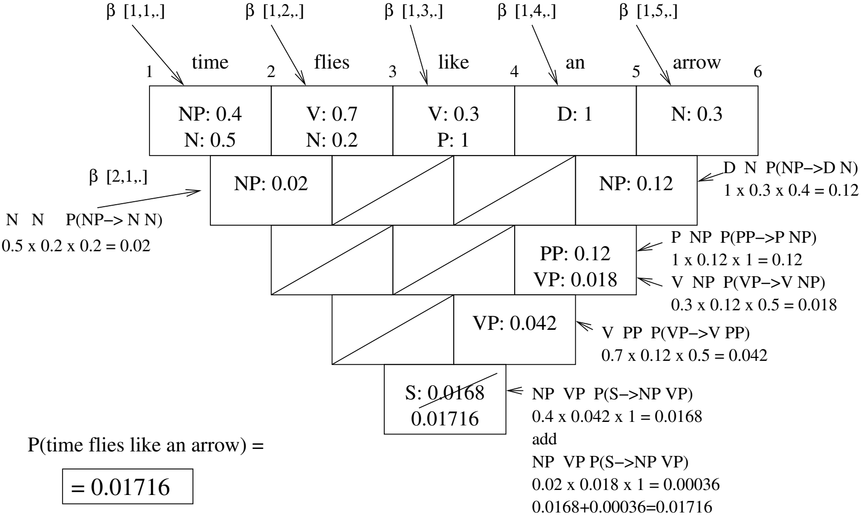
Ensure: P(sentence) is returned

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 $N^k \to w_i$ **do**
        $\beta[i, 1, k] \leftarrow \mathrm{P}(N^k \to w_i)$
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 $N^k \to N^{k_1} N^{k_2}$ **do**
                $\beta[i, j, k] \leftarrow \beta[i, j, k] + \mathrm{P}(N^k \to N^{k_1} N^{k_2}) \cdot \beta[i, l, k_1] \cdot \beta[i + l, j - l, k_2]$
return $\beta[1, n, 1]$

PCFG Marginalization Example (grammar)

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

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

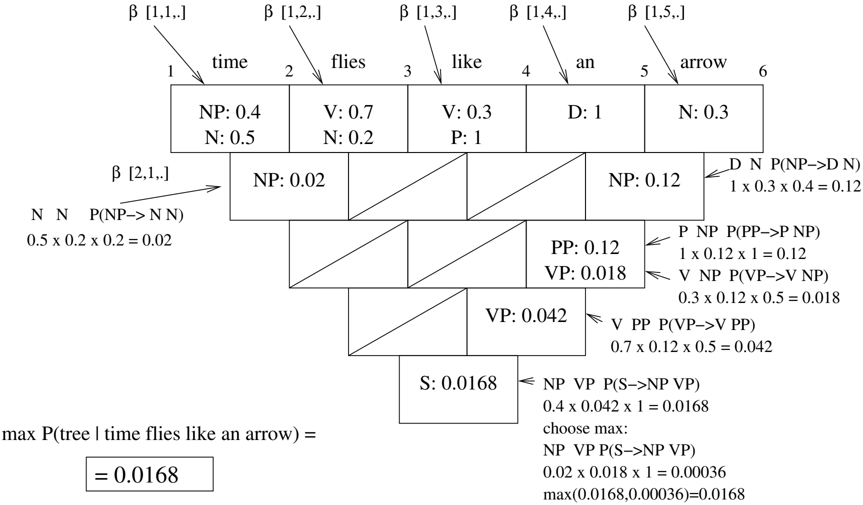
Ensure: a most probable tree with root N^k and leaves $w_i \ldots w_{i+j-1}$ is returned

```
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
```

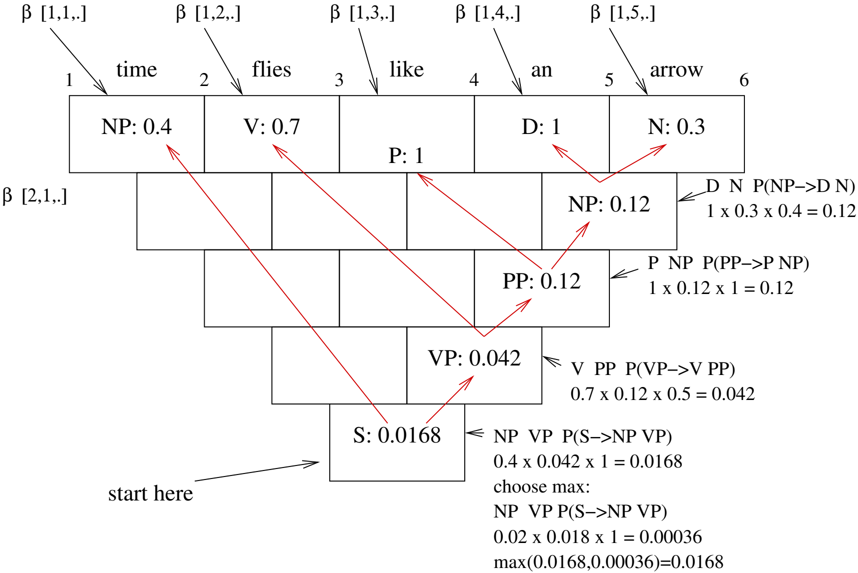
PCFG Completion Example (grammar)

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

PCFG Completion Example (chart)

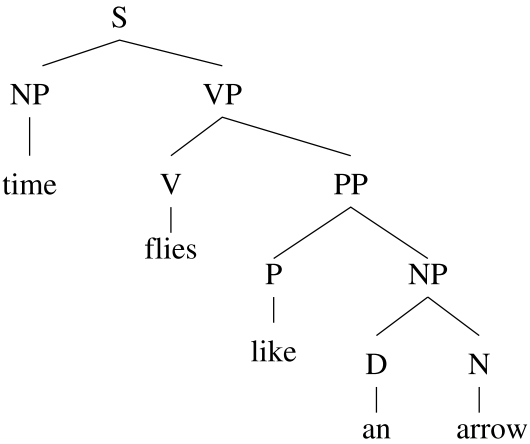


PCFG Completion Example (tree reconstruction)



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 $\rightarrow$ PRP and NP $\rightarrow$ 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., $\mathrm{NP} \rightarrow \mathrm{NP\ PP}$, $\mathrm{VP} \rightarrow \mathrm{VBD\ NP}$, and $\mathrm{VP} \rightarrow \mathrm{VBD\ NP\ PP}$
   - 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:
  - $\mathrm{NP} \rightarrow \mathrm{NP\ PP}$
  - $\mathrm{VP} \rightarrow \mathrm{VBD\ NP}$
  - $\mathrm{VP} \rightarrow \mathrm{VBD\ NP\ PP}$

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

Are Natural Languages Context-Free?

- Can we use CFG directly to model the syntax?
- Surprisingly effective in many cases
- However, not considered sufficient
- Some NL are provably not context-free due to $ww = w^2$ forms
- Additionally, NL Phenomena

Natural Language Phenomena

Three well-known phenomena:

- Agreement
- Movement
- Subcategorization

Agreement

- Phenomenon which requires that constituents must agree on some features before being combined to larger constituents
- Example: "This book" vs. "These book"*, or "He works" vs. "He work"*
- The relevant features are propagated from child nodes to parent nodes; e.g., consider examples:
  These problems usually persist.
  This problem usually persists.

Agreement Examples

- subject-verb agreement
  For example, "I work." and "He works." vs. *"I works." and *"He work."
- specifier-head agreement
  For example, "This book." and "These books." vs. *"This books." and "These book."

Agreement can be a non-local dependency, e.g:
The **women** who found the wallet **were** given a reward.

Movement

- *movement:* an natural language phenomenon, in which a constituent in a grammatically valid sentence, can sometimes be moved to another position and the new sentence remains grammatically valid
- example: "Are you well?" from "You are well."

Movement Examples

E.g, wh-movement

Which book should Peter buy ?
filler *gap*

Another example:

```
(S (NP (NP Air Canada) ,
       (NP (NP-*filler* one of many airline companies)
           (SBAR that (S (NP-*gap*)
                         (VP flies from Halifax
                                to Toronto))
           )) ,
    (VP canceled the flights yesterday) ) . )
```

Subcategorization

- **Subcategorization** phenomenon: tendency of verbs to prefer or require certain types of arguments
- Example, correct sentences:
  The defendant **disappeared***.*
  The defendant **denied** *the accusation.*
- but the following sentences are not correct:
  The defendant **denied***.*
  The defendant **disappeared** *the accusation.*
- The verbs 'deny' and 'disappear' belong to different subcategories.
- For example, some verbs do not take a noun-phrase object, and some do (direct and indirect objects)

Parser Evaluation

- PARSEVAL measures are used to evaluate context-free parsing performance
- Precision and recall of labeled and unlabeled constituents

$$\begin{aligned}
\text{labelled recall} &= \frac{\text{number of correct LC in PT}}{\text{number of LC in GT}} \\
\text{labelled precision} &= \frac{\text{number of correct LC in PT}}{\text{number of LC in PT}} \\
\text{F-measure} &= \frac{2 \cdot (\text{labelled precision}) \cdot (\text{labelled recall})}{(\text{labelled precision}) + (\text{labelled recall})}
\end{aligned}$$

- Labeled constituent: (span-start, span-end, non-terminal)
- Example: (0, 2, NP)

Example: PARSEVAL Measures

Let us consider the following two sentences:

Time flies like an arrow.

and

He ate the cake with a spoon.

Gold standard

```
(S (NP (NN time) (NN flies))
   (VP (VB like)
       (NP (DT an) (NN arrow))))

(S (NP (PRP he))
   (VP (VBD ate) (NP (DT the)
                     (NN cake))
       (PP (IN with)
           (NP (DT a) (NN spoon)))))
```

Parser result

```
(S (NP (NN time))
   (VP (VB flies)
       (PP (IN like)
           (NP (DT an) (NN arrow)))))

(S (NP (PRP he))
   (VP (VBD ate)
       (NP (DT the) (NN cake)
           (PP (IN with)
               (NP (DT a) (NN spoon))))))
```

```
  time flies like an arrow
0     1     2    3  4     5
```

```
Gold standard:                   Parser result:
S  0,5 time flies like an arrow S  0,5 time flies like an arrow
NP 0,2 time flies               NP 0,1 time
NN 0,1 time                     NN 0,1 time
NN 1,2 flies                    VP 1,5 flies like an arrow
VP 2,5 like an arrow            VB 1,2 flies
VB 2,3 like                     PP 2,5 like an arrow
NP 3,5 an arrow                 IN 2,3 like
DT 3,4 an                       NP 3,5 an arrow
NN 4,5 arrow                    DT 3,4 an
                                NN 4,5 arrow
```

```
 he ate the cake with a spoon
0  1   2   3    4    5 6    7
```

Gold standard:

```
S   0,7 he ate ...ke with a spoon
NP  0,1 he
PRP 0,1 he
VP  1,7 ate the cake with a spoon
VBD 1,2 ate
NP  2,4 the cake
DT  2,3 the
NN  3,4 cake
PP  4,7 with a spoon
IN  4,5 with
NP  5,7 a spoon
DT  5,6 a
NN  6,7 spoon
```

Parser result:

```
S   0,7 he ate the cake with a spoon
NP  0,1 he
PRP 0,1 he
VP  1,7 ate the cake with a spoon
VBD 1,2 ate
NP  2,7 the cake with a spoon
DT  2,3 the
NN  3,4 cake
PP  4,7 with a spoon
IN  4,5 with
NP  5,7 a spoon
DT  5,6 a
NN  6,7 spoon
```

Precision$= \frac{17}{23} \approx 0.739130434782609$
Recall $= \frac{17}{22} \approx 0.772727272727273.$