Two views of linguistic structure:
1. Constituency (phrase structure)
   - Phrase structure organizes words into nested constituents.
     - Example: Fed raises interest rates

   - How do we know what is a constituent? (Not that linguists don’t argue about some cases.)
     - Distribution: a constituent behaves as a unit that can appear in different places:
       - John talked [to the children] [about drugs].
       - *John talked [about drugs] [to the children].

   - Substitution/expansion/pronoun:
     - I sat [on the box/right on top of the box/here].

Headed phrase structure
- Context-free grammar
  - VP → ... VB* ...
  - NP → ... NN* ...
  - ADJP → ... JJ* ...
  - ADVP → ... RB* ...
  - S → ... NP VP ...

- Plus minor phrase types:
  - QP (quantifier phrase in NP: some people), CONJP (multi word constructions: as well as), INTJ (interjections: aha), etc.
Two views of linguistic structure:
2. Dependency structure
• Dependency structure shows which words depend on (modify or are arguments of) which other words.

The boy put the tortoise on the rug

Phrase Chunking
• Find all non-recursive noun phrases (NPs) and verb phrases (VPs) in a sentence.
  • [NP [NP [NP the spaghetti] [VP with] [NP meatballs]]]
  • [NP [He [NP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only 1.8 billion] [PP in] [NP September]].

Phrase Chunking as Sequence Labeling
• Tag individual words with one of 3 tags
  • B (Begin) word starts new target phrase
  • I (Inside) word is part of target phrase but not the first word
  • O (Other) word is not part of target phrase
• Sample for NP chunking
  • He reckons the current account deficit will narrow to only 1.8 billion in September.

Evaluating Chunking
Per token accuracy does not evaluate finding correct full chunks. Instead use:

\[
\text{Precision} = \frac{\text{Number of correct chunks found}}{\text{Total number of chunks found}}
\]
\[
\text{Recall} = \frac{\text{Number of correct chunks found}}{\text{Total number of actual chunks}}
\]
\[
F = \frac{2PR}{P + R}
\]

Current Chunking Results
• Best system for NP chunking: F1=96%
• Typical results for finding range of chunk types (CONLL 2000 shared task: NP, VP, PP, ADV, S, ADJP) is F1=92–94%
Syntactic Parsing

• Produce the correct syntactic parse tree for a sentence.

Annotated data: The Penn Treebank

13

The rise of annotated data

• Starting off, building a treebank seems a lot slower and less useful than building a grammar

• But a treebank gives us many things
  • Reusability of the labor
    • Many parsers, POS taggers, etc.
  • Valuable resource for linguists
  • Broad coverage
  • Frequencies and distributional information
  • A way to evaluate systems

14

Two problems to solve for parsing:
1. Repeated work...

"Cats scratch people with cats with claws"

"Cats scratch people with cats with claws"

15

Two problems to solve for parsing:
2. Choosing the correct parse

• How do we work out the correct attachment:
  • She saw the man with a telescope
  • Words are good predictors of attachment, even absent full understanding
    • Moscow sent more than 100,000 soldiers into Afghanistan...
    • Sydney Water breached an agreement with NSW Health...
  • Our statistical parsers will try to exploit such statistics.
Statistical parsing applications

Statistical parsers are now robust and widely used in larger NLP applications:

- High precision question answering [Pasca and Herabagiu SIGIR 2001]
- Improving biological named entity finding [Finkel et al. JNLPBA 2004]
- Syntactically based sentence compression [Lin and Wilbur 2007]
- Extracting opinions about products [Booher et al. NAACL 2007]
- Improved interaction in computer games [Collins and Roy 2005]
- Helping linguists find data [Resnik et al. BLS 2005]
- Source sentence analysis for machine translation [Xu et al. 2009]
- Relation extraction systems [Fundel et al. Bioinformatics 2006]

(Probabilistic) Context-Free Grammars

- CFG
- PCFG

Phrase structure grammars = context-free grammars (CFGs)

- $G = (T, N, S, R)$
  - $T$ is a set of terminal symbols
  - $N$ is a set of nonterminal symbols
  - $S$ is the start symbol ($S \in N$)
  - $R$ is a set of rules/productions of the form $X \rightarrow \gamma$
    - $X \in N$ and $\gamma \in (N \cup T)^*$

A phrase structure grammar

- $S \rightarrow NP \ VP$
- $VP \rightarrow V \ NP$
- $VP \rightarrow V \ NP \ PP$
- $NP \rightarrow NP \ PP$
- $NP \rightarrow N$
- $NP \rightarrow e$
- $PP \rightarrow P \ NP$

people fish tanks
people fish with rods

Sentence Generation

- Sentences are generated by recursively rewriting the start symbol using the productions until only terminals symbols remain.
Phrase structure grammars in NLP

- $G = (T, C, N, S, L, R)$
- $T$ is a set of terminal symbols
- $C$ is a set of preterminal symbols
- $N$ is a set of nonterminal symbols
- $S$ is the start symbol ($S \in N$)
- $L$ is the lexicon, a set of items of the form $X \rightarrow x$
  - $X \in C$ and $x \in T$
- $R$ is the grammar, a set of items of the form $X \rightarrow g$
  - $X \in N$ and $g \in (N \cup C)^*$
- By usual convention, $S$ is the start symbol, but in statistical NLP, we usually have an extra node at the top (ROOT, TOP)
- We usually write $e$ for an empty sequence, rather than nothing

A phrase structure grammar

$S \rightarrow$ NP VP
$VP \rightarrow V \ NP$
$NP \rightarrow NP \ PP$
$NP \rightarrow N$
$PP \rightarrow P \ NP$

people fish tanks
people fish with rods

Probabilistic – or stochastic – context-free grammars (PCFGs)

- $G = (T, N, S, R, P)$
  - $T$ is a set of terminal symbols
  - $N$ is a set of nonterminal symbols
  - $S$ is the start symbol ($S \in N$)
  - $R$ is a set of rules/productions of the form $X \rightarrow g$
  - $P$ is a probability function
    - $P: R \rightarrow [0,1]$
    - $\forall X \in N, \sum_{g \in R(X)} P(X \rightarrow g) = 1$
- A grammar $G$ generates a language model $L$.

A PCFG

$S \rightarrow$ NP VP $1.0$
$VP \rightarrow V \ NP$ $0.6$
$NP \rightarrow NP \ PP$ $0.4$
$NP \rightarrow N$ $0.1$
$PP \rightarrow P \ NP$ $1.0$

$N \rightarrow$ people $0.5$
$N \rightarrow$ fish $0.2$
$N \rightarrow$ tanks $0.2$
$N \rightarrow$ rods $0.1$
$V \rightarrow$ people $0.1$
$V \rightarrow$ fish $0.6$
$V \rightarrow$ tanks $0.3$
$P \rightarrow$ with $1.0$

The probability of trees and strings

- $P(t)$ – The probability of a tree $t$ is the product of the probabilities of the rules used to generate it.
- $P(s)$ – The probability of the string $s$ is the sum of the probabilities of the trees which have that string as their yield

$$P(s) = \sum_{t} P(s, t) \text{ where } t \text{ is a parse of } s = \sum_{t} P(t)$$

The probability of trees and strings (cont.)
Tree and String Probabilities

- $s = \text{people fish tanks with rods}$
- $P(t_1) = 1.0 \times 0.7 \times 0.5 \times 0.6 \times 0.7 \\ \times 1.0 \times 0.2 \times 1.0 \times 0.7 \times 0.1 \\ = 0.0008232$
- $P(t_2) = 1.0 \times 0.7 \times 0.6 \times 0.5 \times 0.6 \times 0.2 \\ \times 0.7 \times 1.0 \times 0.2 \times 1.0 \times 0.7 \times 0.1 \\ = 0.00024696$
- $P(s) = P(t_1) + P(t_2) \\ = 0.0008232 + 0.00024696 \\ = 0.00107016$

Chomsky Normal Form

- All rules are of the form $X \rightarrow Y Z$ or $X \rightarrow w$
  - $X, Y, Z \in N$ and $w \in T$
- A transformation to this form doesn’t change the generative capacity of a CFG
  - That is, it recognizes the same language
  - But maybe with different trees
- Empties and unaries are removed recursively
  - $n$-ary rules are divided by introducing new nonterminals ($n > 2$)

A phrase structure grammar

- $S \rightarrow \text{NP VP}$
- $\text{VP} \rightarrow \text{V NP}$
- $\text{VP} \rightarrow \text{V}$
- $\text{NP} \rightarrow \text{NP NP}$
- $\text{NP} \rightarrow \text{NP PP}$
- $\text{NP} \rightarrow N$
- $\text{NP} \rightarrow e$
- $\text{PP} \rightarrow P NP$

- $N \rightarrow \text{people}$
- $N \rightarrow \text{fish}$
- $N \rightarrow \text{tanks}$
- $N \rightarrow \text{rods}$
- $V \rightarrow \text{people}$
- $V \rightarrow \text{fish}$
- $V \rightarrow \text{tanks}$
- $P \rightarrow \text{with}$

Chomsky Normal Form steps

- $S \rightarrow \text{NP VP}$
- $S \rightarrow \text{NP}$
- $\text{VP} \rightarrow \text{V NP}$
- $\text{VP} \rightarrow \text{V}$
- $\text{VP} \rightarrow \text{V NP PP}$
- $\text{VP} \rightarrow \text{V PP}$
- $\text{VP} \rightarrow \text{N}$
- $\text{PP} \rightarrow P NP$
- $\text{PP} \rightarrow P$

- $N \rightarrow \text{people}$
- $N \rightarrow \text{fish}$
- $N \rightarrow \text{tanks}$
- $N \rightarrow \text{rods}$
- $V \rightarrow \text{people}$
- $V \rightarrow \text{fish}$
- $V \rightarrow \text{tanks}$
- $P \rightarrow \text{with}$

Chomsky Normal Form steps

- $S \rightarrow \text{NP VP}$
- $S \rightarrow \text{NP}$
- $\text{VP} \rightarrow \text{V NP}$
- $\text{VP} \rightarrow \text{V}$
- $\text{VP} \rightarrow \text{V NP PP}$
- $\text{VP} \rightarrow \text{V PP}$
- $\text{VP} \rightarrow \text{N}$
- $\text{PP} \rightarrow P NP$
- $\text{PP} \rightarrow P$

- $N \rightarrow \text{people}$
- $N \rightarrow \text{fish}$
- $N \rightarrow \text{tanks}$
- $N \rightarrow \text{rods}$
- $V \rightarrow \text{people}$
- $V \rightarrow \text{fish}$
- $V \rightarrow \text{tanks}$
- $P \rightarrow \text{with}$
Chomsky Normal Form steps

37

Chomsky Normal Form steps

38

Chomsky Normal Form steps

39

Chomsky Normal Form steps

40

An example: before binarization...

41

Before and After binarization on VP

42
Parsing

- Given a string of terminals (e.g. sentences) and a CFG, determine if the string can be generated by the CFG.
- Also return a parse tree for the string
- Also return all possible parse trees for the string
- Must search space of derivations for one that derives the given string.
  - **Top-Down Parsing**: Start searching space of derivations for the start symbol.
  - **Bottom-up Parsing**: Start search space of reverse derivations from the terminal symbols in the string.

Parsing Example

Top Down Parsing

43

44

45

46

47

48
Bottom Up Parsing

Top Down vs. Bottom Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.
- Bottom up never explores options that do not connect to the actual sentence but can explore options that can never lead to a full parse.
- Relative amounts of wasted search depend on how much the grammar branches in each direction.

Two problems to solve for parsing:
1. Repeated work

"Cats scratch people with cats with claws"
Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.
- Caching (memorizing) is critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs.

(Probabilistic) CKY Parsing

Constituency Parsing

Input: a PCFG, and a sentence

Output: a parsing tree

Cocke-Kasami-Younger (CKY)
Constituency Parsing

Reusing local decisions
Reusing local decisions

The CKY algorithm (1960/1965) ... extended to unaries

The CKY algorithm (1960/1965) ... extended to unaries