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

Two views of linguistic structure:
1. Constituency (phrase structure)
   • Phrase structure organizes words into nested constituents.

Two views of linguistic structure:
1. Constituency (phrase structure)
   • Phrase structure organizes words into nested constituents.
     • 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].
         • *John talked drugs to the children about.
       • Substitution/expansion/pro-forms:
         • I sat [on the bus/right on top of the bus/there].

Project Progress Report
• What changes you have made for the task compared to the proposal, including problem/task, models, datasets, or evaluation methods? If there is any change, please explain why.
• Describe data preprocessing process. This includes data cleaning, selection, feature generation or other representation you have used, etc.
• What methods or models have you tried towards the project goal? And why do you choose the methods (you can including related work on similar task or relevant tasks)?
• What results you have achieved up to now based on your proposed evaluation methods? What is working or What is wrong with the model?
• How can you improve your models? What are the next steps?
• Grading: For 2-5, each aspect will take about 25 points.
• Length: Length: 2 page (or more if necessary)
Headed phrase structure

- Context-free grammar
  - VP → ... VB* ...
  - NP → ... NN* ...
  - ADJP → ... JJ* ...
  - ADVP → ... RB* ...
  - SBAR(Q) → S|SINV|SQ → ... NP VP ...

- Plus minor phrase types:
  - QP (quantifier phrase in NP), CONJP (multi-word constructions, as we'll see), INJ (interjections), 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

Two views of linguistic structure:
2. Dependency structure

A Brief Parsing History

Pre 1990 ("Classical") NLP Parsing

- Wrote symbolic grammar (CFG or often richer) and lexicon
  - S → NP VP
  - NP → (DT) NN
  - NP → NN NN
  - NP → NNP
  - VP → V NP
  - VP → V NP
- Used grammar proof systems to prove parses from words
- This scaled very badly and didn't give coverage. For sentence:
  - Fed raises interest rates 0.5% in effort to control inflation
    - Minimal grammar: 36 parses
    - Simple 10 rule grammar: 550 parses
    - Real-size broad coverage grammar: millions of parses

Classical NLP Parsing:
The problem and its solution

- Categorical constraints can be added to grammars to limit unlikely/weird parses for sentences
  - But the attempt make the grammars not robust
    - In traditional systems, commonly 30% of sentences in even an edited text would have no parse.
- A less constrained grammar can parse more sentences
  - But simple sentences end up with ever more parses with no way to choose between them
- We need mechanisms that allow us to find the most likely parse(s) for a sentence
  - Statistical parsing lets us work with very loose grammars
    - That admit millions of parses for sentences but still quickly find the best parse(s)
The rise of annotated data:
The Penn Treebank

- 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 linguistics
  - Broad coverage
  - Frequencies and distributional information
  - A way to evaluate systems

Statistical parsing applications

Statistical parsers are now robust and widely used in larger NLP applications:
- High precision question answering [Puzia and Havasi SIGIR 2001]
- Improving biological named entity finding [Hsieh et al. NLPNA 2004]
- Syntactically based sentence compression [Sun and Wu NAACL 2007]
- Extracting opinions about products [Shokr et al. AAAI 2007]
- Improved interaction in computer games [Gorinsky and Roy 2005]
- Helping linguists find data [Kreml et al. IJIS 2005]
- Source sentence analysis for machine translation [Yu et al. 2009]
- Relation extraction systems [Yan et al. Bioinformatics 2004]

Attachment ambiguities

- A key parsing decision is how we ‘attach’ various constituents
  - PPs, adverbial or participial phrases, infinitives, coordinations, etc.

The board approved [its acquisition] [by Royal Trustco Ltd.]
[for $27 a share]
[at its monthly meeting].

The rise of annotated data

An exponential number of attachments

Attachment ambiguities

- A key parsing decision is how we ‘attach’ various constituents
  - PPs, adverbial or participial phrases, infinitives, coordinations, etc.
Attachment ambiguities

- A key parsing decision is how we 'attach' various constituents
  - PPs, adverbial or participial phrases, infinitives, coordinations, etc.

The board approved [its acquisition] [by Royal Trustco Ltd.]

[for $27 a share]

[as its monthly meeting]

- Catalan numbers: $C_n = \frac{(2n)!}{n!(n+1)!}$
  - An exponentially growing series, which arises in many tree-like contexts
    - E.g., the number of possible triangulations of a polygon with n+2 sides
    - Turns up in triangulation of probabilistic graphical models...

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

Two problems to solve:
2. Choosing the correct parse

- How do we work out the correct attachment:
  - She saw the man with a telescope
  - Is the problem 'AI complete'? Yes, but...
  - 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.

(Probabilistic) Context-Free Grammars

- CFG
- PCFG

A phrase structure grammar

\[
S \rightarrow NP \ VP \\
NP \rightarrow V \ NP \\
NP \rightarrow NP \ NP \\
NP \rightarrow NP \ PP \\
NP \rightarrow N \\
NP \rightarrow \# \\
PP \rightarrow P \ NP \\
people \ fish \ tanks \\
people \ fish \ with \ rods
\]

\[
N \rightarrow people \\
N \rightarrow fish \\
N \rightarrow tanks \\
N \rightarrow rods \\
V \rightarrow people \\
V \rightarrow fish \\
V \rightarrow tanks \\
P \rightarrow with
\]
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 grammar $G$ generates a language $L$.

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 $X \rightarrow \gamma$
    - $X \in N$ and $\gamma \in (N \cup C)^*$

- 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

- $S \rightarrow NP \ VP$
- $VP \rightarrow V \ NP$
- $VP \rightarrow V \ NP \ PP$
- $NP \rightarrow NP \ NP$
- $NP \rightarrow NP \ PP$
- $NP \rightarrow e$
- $PP \rightarrow P \ NP$
- people fish tanks
- people fish with rods
- $N \rightarrow people$
- $N \rightarrow fish$
- $N \rightarrow tanks$
- $N \rightarrow rods$
- $V \rightarrow people$
- $V \rightarrow fish$
- $V \rightarrow tanks$
- $P \rightarrow with$

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 $X \rightarrow \gamma$
  - $P$ is a probability function
    - $P: R \rightarrow [0,1]$
  - $\forall X \in N, \sum_{\gamma} P(X \rightarrow \gamma) = 1$
  - $\gamma \in T^*$

- A grammar $G$ generates a language model $L$.
  - $\sum_{\gamma} P(\gamma|t) = 1$

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
  - $P(s) = \sum_t P(t, s)$ where $t$ is a parse of $s$
  - $= \sum_t P(t)$

A PCFG

- $S \rightarrow NP \ VP \ 1.0$
- $VP \rightarrow V \ NP \ 0.6$
- $VP \rightarrow V \ NP \ PP \ 0.4$
- $NP \rightarrow NP \ NP \ 0.1$
- $NP \rightarrow NP \ PP \ 0.2$
- $NP \rightarrow N \ 0.7$
- $PP \rightarrow P \ NP \ 1.0$
- $N \rightarrow people \ 0.5$
- $N \rightarrow fish \ 0.2$
- $N \rightarrow tanks \ 0.2$
- $N \rightarrow rods \ 0.1$
- $V \rightarrow people \ 0.1$
- $V \rightarrow fish \ 0.6$
- $V \rightarrow tanks \ 0.3$
- $P \rightarrow with \ 1.0$

[With empty NP removed to less ambiguous]
Tree and String Probabilities

- $s = \text{people fish tanks with rods}$
- $P(t_1) = 1.0 \times 0.7 \times 0.4 \times 0.5 \times 0.6 \times 0.7 \times 1.0 \times 0.2 \times 1.0 \times 0.7 \times 0.1 = 0.0008232$
- $P(t_2) = 1.0 \times 0.7 \times 0.6 \times 0.5 \times 0.6 \times 0.2 \times 0.7 \times 1.0 \times 0.2 \times 1.0 \times 0.7 \times 0.1 = 0.00024696$
- $P(s) = P(t_1) + P(t_2) = 0.0008232 + 0.00024696 = 0.00107016$

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 weak 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

\[
\begin{align*}
S & \to NP \ VP \\
VP & \to V \ NP \\
VP & \to V \ NP \ PP \\
NP & \to NP \ NP \\
NP & \to N \\
NP & \to e \\
PP & \to P \ NP
\end{align*}
\]

Chomsky Normal Form steps

\[
\begin{align*}
S & \to NP \ VP \\
VP & \to V \ NP \\
V & \to N \\
N & \to e \\
P & \to with
\end{align*}
\]
Chomsky Normal Form steps

- $S \rightarrow NP \ VP$
- $VP \rightarrow V \ NP$
- $S_NP \rightarrow V \ NP$
- $S \rightarrow V \ NP$
- $VP \rightarrow NP \ NP$
- $VP \rightarrow V \ PP$
- $NP \rightarrow P \ NP$
- $NP \rightarrow NP \ NP$
- $NP \rightarrow NP \ PP$
- $NP \rightarrow P \ PP$
- $NP \rightarrow NP \ PP$
- $V \rightarrow people$
- $V \rightarrow fish$
- $V \rightarrow tanks$
- $V \rightarrow rods$
- $NP \rightarrow people$
- $NP \rightarrow fish$
- $NP \rightarrow tanks$
- $P \rightarrow with$
- $PP \rightarrow with$

Chomsky Normal Form

- You should think of this as a transformation for efficient parsing
- With some extra book-keeping in symbol names, you can even reconstruct the same trees with a detransform
- In practice full Chomsky Normal Form is a pain
  - Reconstructing unaries is easy
  - Reconstructing unaries/empties is trickier
- **Binarization** is crucial for cubic time CFG parsing
  - The rest isn’t necessary; it just makes the algorithms cleaner and a bit quicker

An example: before binarization...

After binarization...

Treebank: empties and unaries

CKY Parsing
Constituency Parsing

PCFG

Rule Fish R
S → NP VP, B_s
NP → NP NP, B_r
NP → N, B_n
V → fish, B_v
N → people, B_p
V → fish, B_v

Viterbi (Max) Scores

S → NP VP 0.9
S → VP 0.1
VP → V NP 0.5
VP → V 0.1
VP → VP 0.3
 VP → VP , V 0.1
 NP → NP NP 0.1
 NP → NP VP 0.3
 NP → N 0.7
 PP → P NP 1.0

Extended CKY parsing

• Unaries can be incorporated into the algorithm
• Messy, but doesn’t increase algorithmic complexity
• Empties can be incorporated
• Use fenceposts
• Doesn’t increase complexity; essentially like unaries
• Binarization is vital
  • Without binarization, you don’t get parsing cubic in the length of the sentence and in the number of nonterminals in the grammar