Statistical Parsing
Language Model vs. Parsing Model

• Language model:
  – interested in string probability:
    \[ P(W) = \text{probability definition using a formula such as} \]
    \[ = \prod_{i=1..n} p(w_i|w_{i-2},w_{i-1}) \quad \text{trigram language model} \]
    \[ = \sum_{s \in S} p(W,s) = \sum_{s \in S} \prod_{r \in s} r \quad \text{PCFG; } r \sim \text{rule used in parse tree} \]

• Parsing model
  – conditional probability of tree given string:
    \[ P(s|W) = \frac{P(W,s)}{P(W)} = \frac{P(s)}{P(W)} \quad !! P(W,s) = P(s) !! \]
    – for argmax, just use \( P(s) \) (\( P(W) \) is constant)
Once again, Lexicalization

- Lexicalized parse tree (~ dependency tree + phrase labels)
- Ex. subtree:

  \[
  \text{PP(with)}
  \quad \xrightarrow{\text{PREP(with)}} \quad \text{N(telescope)}
  \quad \xrightarrow{\text{with}} \quad \text{a\_telescope}
  \]

- Pre-terminals (above leaves): assign the word below
- Recursive step (step up one level): (a) select node, (b) copy word up.
Lexicalized Tree Example

• #1 S → NP VP
• #2 VP → V NP PP
• #3 VP → V NP
• #4 NP → N
• #5 NP → N PP
• #6 PP → PREP N
• #7 N → a_dog
• #8 N → a_cat
• #9 N → a_telescope
• #10 V → saw
• #11 PREP → with a_dog saw a_cat with a_telescope
Using POS Tags

- Head ~ word, tag

```
S(saw, V)
   VP(saw, V)
   NP(a_dog, N) NP(a_cat, N) PP(with, PREP)
     N     V     N     PREP     N
     a_dog saw a_cat with a_telescope
```
Conditioning

• Original PCFG: \( P(\alpha B\gamma D\varepsilon \ldots |A) \)
  – No “lexical” units (words)

• Introducing words:

\[
P( \alpha B(\text{head}_B) \gamma D(\text{head}_D) \varepsilon \ldots |A(\text{head}_A))
\]

where \( \text{head}_A \) is one of the heads on the left

E.g. rule \( \text{VP(saw)} \rightarrow V(\text{saw}) \ NP(\text{a_cat}) \):

\[
P(V(\text{saw}) \ NP(\text{a_cat}) | \text{VP(saw)})
\]
Independence Assumptions

- Too many rules
- Decompose:
  \[ P(\alpha B(\text{head}_B) \gamma D(\text{head}_D) \varepsilon \ldots | A(\text{head}_A)) = \]
- In general (total independence):
  \[ P(\alpha | A(\text{head}_A)) \times P(B(\text{head}_B) | A(\text{head}_A)) \times \]
  \[ \ldots \times P(\varepsilon | A(\text{head}_A)) \]
- Too much independent: need a compromise.
The Decomposition

- Order does not matter; let’s use intuition (“linguistics”):
- Select the head daughter category:
  \[ P_H(H(\text{head}_A) | A(\text{head}_A)) \]
- Select everything to the right:
  \[ P_R(R_i(r_i) | A(\text{head}_A), H) \]
- Also, choose when to finish: \( R_{m+1}(r_{m+1}) = \text{STOP} \)
- Similarly, for the left direction:
  \[ P_L(L_i(l_i) | A(\text{head}_A), H) \]
Example Decomposition

- Order:

```
1  A(head)
```

```
STOP  L₁(head₁)  H(head)R₁(head₁)R₂(head₂)  STOP
3  ---------------------------------  2
```

- Example:

```
VP(saw)
```

```
STOP  V(saw)  NP(a_cat)  PP(with)  STOP
```
More Conditioning: Distance

• Motivation:
  – close words tend to be dependents (or phrases) more likely
  – ex.: walking on a sidewalk on a sunny day without looking on...

• Words: too detailed distribution, though:
  – use more sophisticated (yet more robust) distance measure $d_{r/l}$:
    • distinguish 0 and non-zero distance (2)
    • distinguish if verb is in-between the head and the constituent in question (2)
    • distinguish if there are commas in-between: 0, 1, 2, >2 commas (4).
    • ...total: 16 possibilities added to the condition: $P_R(R_i(r_i) \mid A(head_A),H,d_r)$
    • same to the left: $P_L(L_i(l_i) \mid A(head_A),H,d_l)$
More Conditioning: Complement/Adjunct

- So far: no distinction

- ...but: time NP ≠ subject NP

- also, Subject NP cannot repeat... useful during parsing

[Must be added in training data]
More Conditioning: Subcategorization

• The problem still not solved:
  – two subjects:
  
  \[ S(\text{was}) \]
  \[ \text{NP-C(Johns Hopkins)} \]
  \[ \text{NP-C(the 7th-best)} \]
  \[ \text{VP(\text{was})} \]
  \[ \text{wrong!} \]

• Need: relation among complements.
  – [linguistic observation: adjuncts can repeat freely.]

• Introduce:
  – Left & Right Subcategorization Frames (multisets)
Inserting Subcategorization

- Use head probability as before:
  \[ P_H(H(\text{head}_A)|A(\text{head}_A)) \]
- Then, add left & right subcat frame:
  \[ P_{lc}(LC| A(\text{head}_A),H), P_{rc}(RC| A(\text{head}_A),H) \]
  - LC, RC: list (multiset) of phrase labels (not words)
- Add them to context condition:
  \[ (\text{left}) \ P_L(L_i(l_i) | A(\text{head}_A),H,d_l,LC) \quad [\text{right: similar}] \]
- LC/RC: “dynamic”: remove labels when generated
  - \( P(\text{STOP}|......,LC) = 0 \) if LC non-empty
Smoothing

- Adding conditions... ~ adding parameters
- Sparse data problem as usual (head ~ <word,tag>!)
- Smooth (step-wise):
  \[ P_{\text{smooth-H}}(H(\text{head}_A)|A(\text{head}_A)) = \]
  \[ = \lambda_1 P_H(H(\text{head}_A)|A(\text{head}_A)) + (1-\lambda_1) P_{\text{smooth-H}}(H(\text{head}_A)|A(\text{tag}_A)) \]
  \[ - P_{\text{smooth-H}}(H(\text{head}_A)|A(\text{tag}_A)) = \]
  \[ = \lambda_2 P_H(H(\text{head}_A)|A(\text{tag}_A)) + (1-\lambda_2) P_H(H(\text{head}_A)|A) \]
- Similarly, for \( P_R \) and \( P_L \).
The Parsing Algorithm for a Lexicalized PCFG

- **Bottom-up Chart parsing**
  - Elements of a chart: a pair
    - \(<\text{from-position}, \text{to-position}, \text{label}, \text{head}, \text{distance}), \text{probability}\>\)
    - \text{span} \text{score}
  - Total probability = multiplying elementary probabilities
    \(\Rightarrow\) enables dynamic programming:
    - discard chart element with the same span but lower score.

- **“Score” computation:**
  - joining chart elements: [for 2]: \(<e_1, p_1>, <e_2, p_2>, <e_n, p_n>:\)
    \[ P(e_{\text{new}}) = p_1 \times p_2 \times \ldots \times p_n \times P_H(...) \times \prod P_R(...) \times \prod P_L(...); \]
Results (PCFG)

• English, WSJ, Penn Treebank, 40k sentences
  < 40Words < 100 Words
  – Labeled Recall: 88.1% 87.5%
  – Labeled Precision: 88.6% 88.1%
  – Crossing Brackets (avg): 0.91 1.07
  – Sentences With 0 CBs: 66.4% 63.9%

• Czech, Prague Dependency Treebank, 13k sentences:
  – Dependency Accuracy overall: 80.0% (MST’05: 85%)
   (~ unlabelled precision/recall)