Statistical Parsing

Language Model vs. Parsing Model

- Language model:
  - interested in string probability:

    P(W) = probability definition using a formula such as

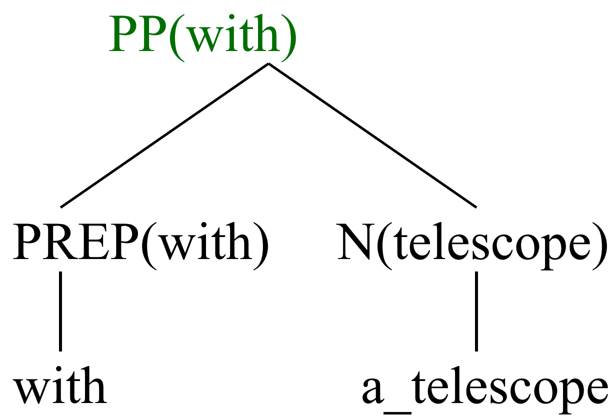
    $= \prod_{i=1..n} p(w_i|w_{i-2},w_{i-1})$ trigram language model

    $= \sum_{s\in S} p(W,s) = \sum_{s\in S} \prod_{r\in s} r$ PCFG; r ~ rule used in parse tree
- Parsing model
  - conditional probability of tree given string:

    $P(s|W) = P(W,s) / P(W) = P(s) / P(W)$!! $P(W,s) = P(s)$!!
  - for argmax, just use P(s) (P(W) is constant)

Once again, Lexicalization

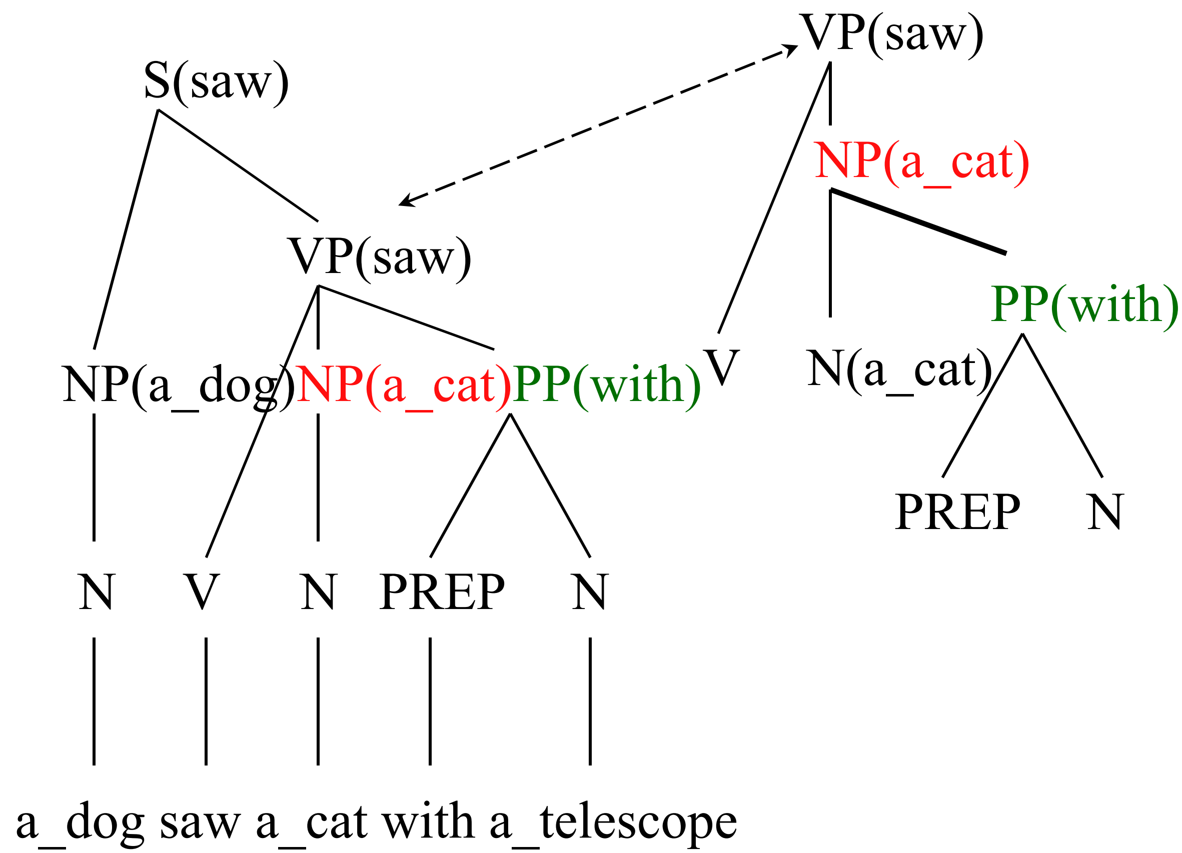
- Lexicalized parse tree (~ dependency tree+phrase labels)
- Ex. subtree:



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

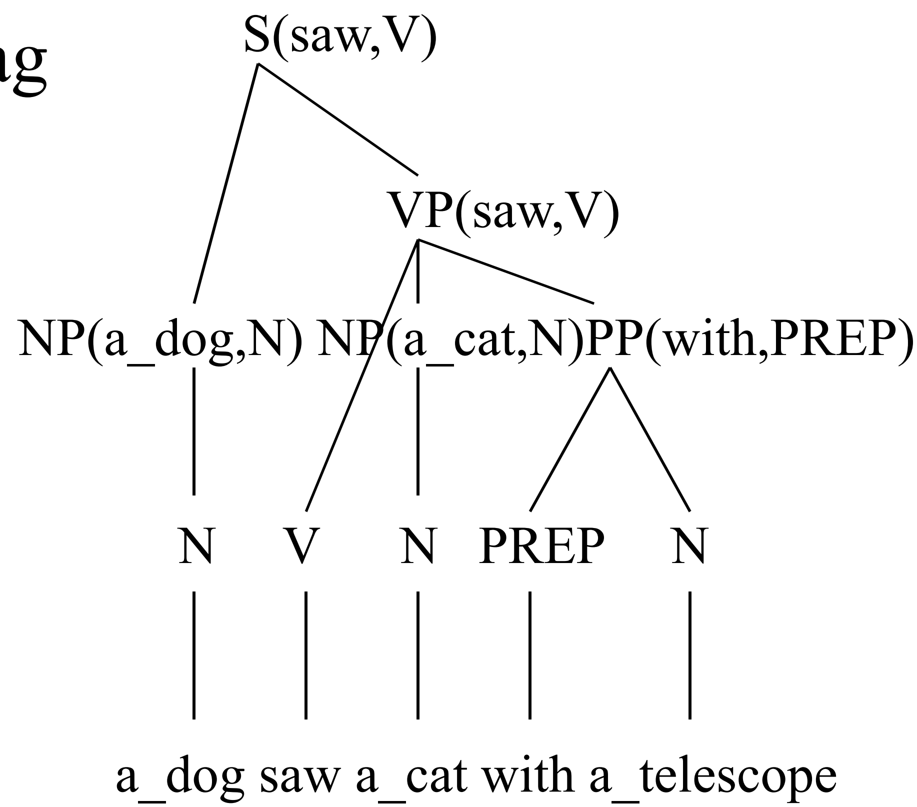
Lexicalized Tree Example

- #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



Using POS Tags

- Head ~ word,tag



Conditioning

- Original PCFG: $P(\alpha B \gamma D \varepsilon ...|A)$
  - No "lexical" units (words)
- Introducing words:

$$P(\ \alpha\ B(head_B)\ \gamma\ D(head_D)\ \varepsilon\ ...\ |A(head_A))$$

where $head_A$ is one of the heads on the left

E.g. rule VP(saw) $\rightarrow$ V(saw) NP(a_cat):

P(V(saw) NP(a_cat) | VP(saw))

Independence Assumptions

- Too many rules
- Decompose:

  $$P(\ \alpha\ B(head_B)\ \gamma\ D(head_D)\ \varepsilon\ ...\ |A(head_A)) =$$

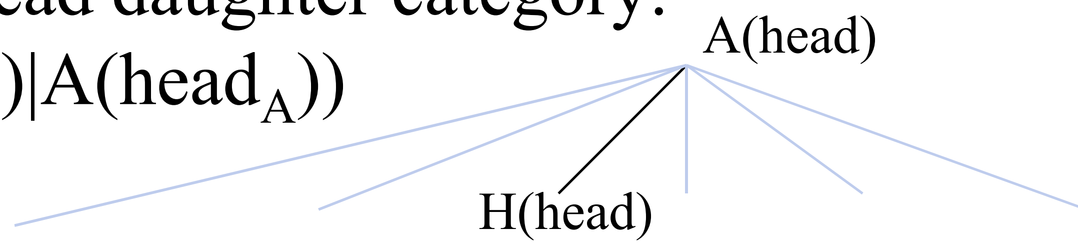
- In general (total independence):

  $$P(\alpha|A(head_A)) \times P(B(head_B)|A(head_A)) \times$$
  $$... \times P(\varepsilon|A(head_A))$$

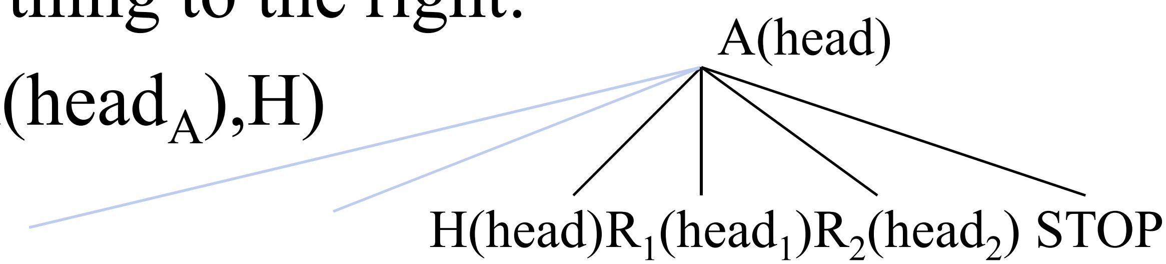
- Too much independent: need a compromise.

The Decomposition

- Order does not matter; let's use intuition ("linguistics"):
- Select the head daughter category: $P_H(H(head_A)|A(head_A))$



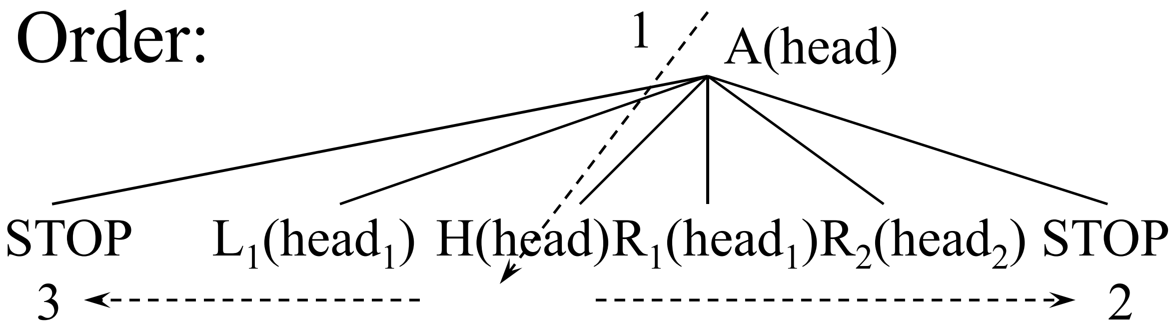
- Select everything to the right: $P_R(R_i(r_i) \mid A(head_A),H)$



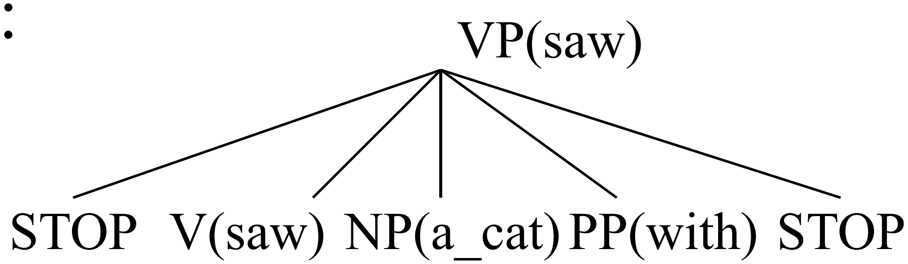
- Also, choose when to finish: $R_{m+1}(r_{m+1}) = STOP$
- Similarly, for the left direction: $P_L(L_i(l_i) \mid A(head_A),H)$

Example Decomposition

- Order:



- Example:

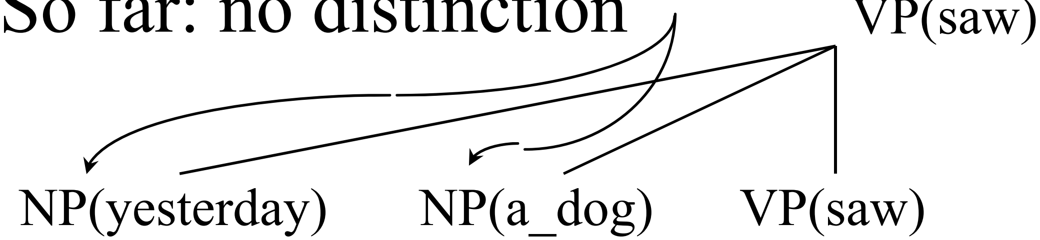


More Conditioning: Distance

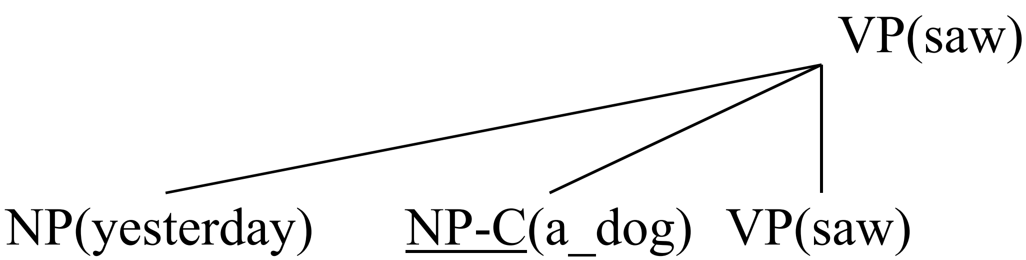
- Motivation:
  - close words tend to be dependents (or phrases) more likely
  - ex.: <u>walking on</u> a sidewalk <u>on</u> a sunny day without looking <u>on</u>...
- Words: too detailed distribution, though:
  - use more sophisticated (yet more robust) distance measure $d_{r/l}$:
    - distinguish 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



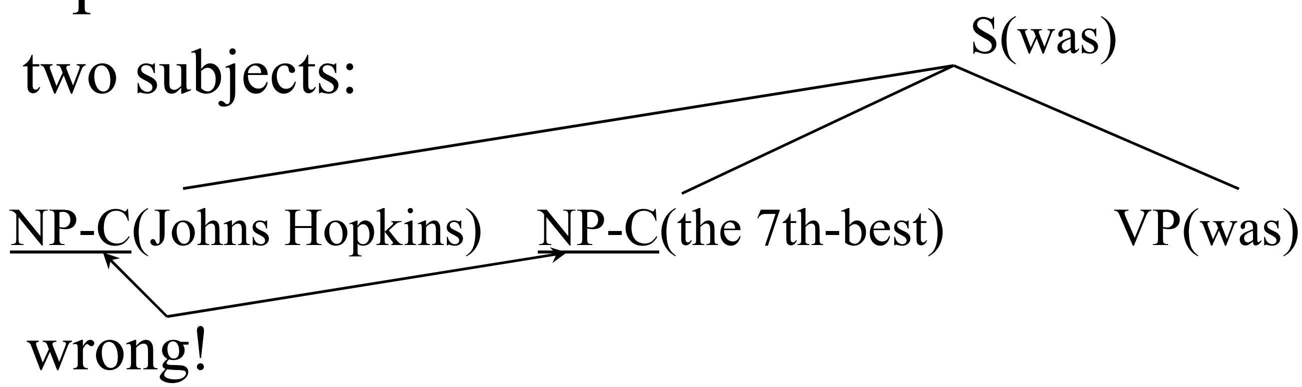
- ...but: time NP ≠ subject NP
- also, Subject NP cannot repeat... useful <u>during</u> parsing



[Must be added in training data]

More Conditioning: Subcategorization

- The problem still not solved:
  - two subjects:



- Need: relation among complements.
  - [linguistic observation: adjuncts can repeat freely.]
- Introduce:
  - Left & Right Subcategorization Frames (multisets)

Inserting Subcategorization

- Use head probability as before:

  $P_H(H(head_A)|A(head_A))$

- Then, add left & right subcat frame:

  $P_{lc}(LC|\ A(head_A),H)$, $P_{rc}(RC|\ A(head_A),H)$

  – LC, RC: list (multiset) of phrase labels (not words)

- Add them to context condition:

  (left) $P_L(L_i(l_i)\ |\ A(head_A),H,d_l,LC)$ [right: similar]

- LC/RC: "dynamic": remove labels when generated

  – $P(STOP|.....,LC) = 0$ if LC non-empty

Smoothing

- Adding conditions... ~ adding parameters
- Sparse data problem as usual (head ~ <word,tag>!)
- Smooth (step-wise):
  - $P_{smooth\text{-}H}(H(head_A)|A(head_A)) =$
    $= \lambda_1 P_H(H(head_A)|A(head_A)) + (1-\lambda_1)P_{smooth\text{-}H}(H(head_A)|A(tag_A))$
  - $P_{smooth\text{-}H}(H(head_A)|A(tag_A)) =$
    $= \lambda_2 P_H(H(head_A)|A(tag_A)) + (1-\lambda_2)P_H(H(head_A)|A)$
- Similarly, for P_R and P_L.

The Parsing Algorithm for a Lexicalized PCFG

- Bottom-up Chart parsing
  - Elements of a chart: a pair
    - <(from-position,to-position,label,head,distance), probability>

      span 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]: $<e_1, p_1>$, $<e_2, p_2>$, $<e_n,p_n>$:

$$P(e_{new}) = p_1 \times p_2 \times ... \times p_n \times P_H(...) \times \prod P_R(...) \times \prod P_L(...);$$

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)