

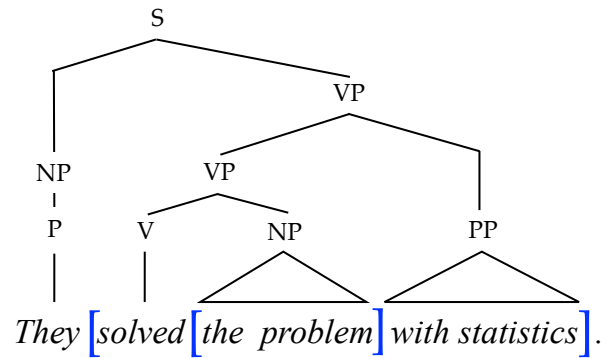
Statistical NLP

Fall 2011

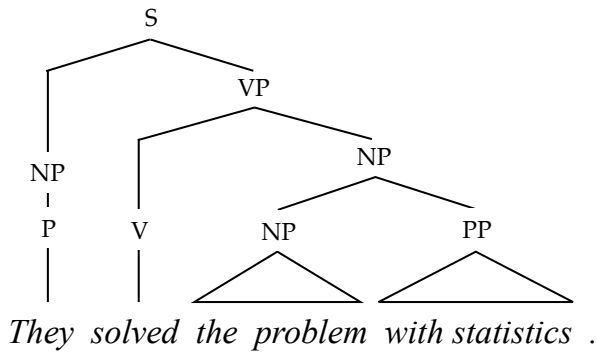
Lecture 9: Syntactic Parsing

Slav Petrov – Google Research

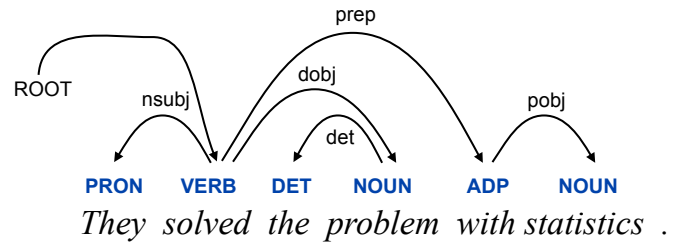
Analyzing Natural Language



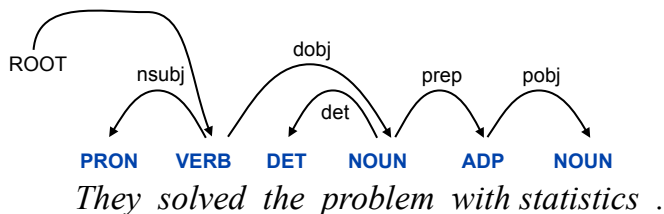
Syntax and Semantics



Constituency and Dependency

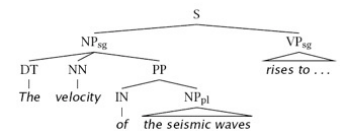


Constituency and Dependency



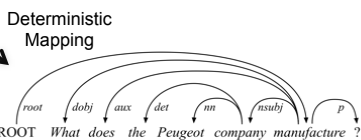
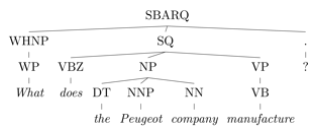
Phrase Structure Parsing

- Phrase structure parsing organizes syntax into *constituents or brackets*
- In general, this involves nested trees
- Linguists can, and do, argue about details
- Lots of ambiguity
- Not the only kind of syntax...



new art critics write reviews with computers

Dependency Parsing



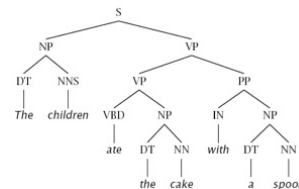
- Directed edges between pairs of word (head, dependent)
- Can handle free word-order languages
- Very efficient decoding algorithms exist

Constituency Tests

- How do we know what nodes go in the tree?

Classic constituency tests:

- Substitution by *proform*
- Question answers
- Semantic grounds
 - Coherence
 - Reference
 - Idioms
- Dislocation
- Conjunction



- Cross-linguistic arguments, too

Conflicting Tests

Constituency isn't always clear

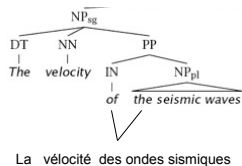
- Units of transfer:
 - think about ~ penser à
 - talk about ~ hablar de

Phonological reduction:

- I will go → I'll go
- I want to go → I wanna go
- a le centre → au centre

Coordination

- He went to and came from the store.



Classical NLP: Parsing

Write symbolic or logical rules:

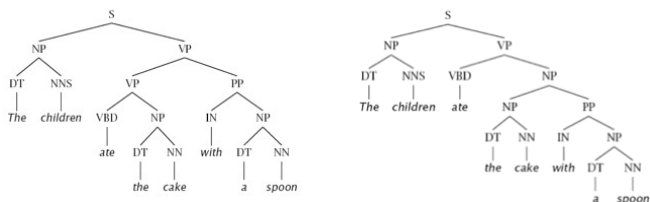
Grammar (CFG)		Lexicon
ROOT → S	NP → NP PP	NN → interest
S → NP VP	VP → VBP NP	NNS → raises
NP → DT NN	VP → VBP NP PP	VBP → interest
NP → NN NNS	PP → IN NP	VBZ → raises
		...

Use deduction systems to prove parses from words

- Minimal grammar on "Fed raises" sentence: 36 parses
- Simple 10-rule grammar: 592 parses
- Real-size grammar: many millions of parses

- This scaled very badly, didn't yield broad-coverage tools

Ambiguities: PP Attachment



Attachments

- I cleaned the dishes from dinner
- I cleaned the dishes with detergent
- I cleaned the dishes in my pajamas
- I cleaned the dishes in the sink

Syntactic Ambiguities I

- **Prepositional phrases:**
They cooked the beans in the pot on the stove with handles.
- **Particle vs. preposition:**
The puppy tore up the staircase.
- **Complement structures**
The tourists objected to the guide that they couldn't hear.
She knows you like the back of her hand.
- **Gerund vs. participial adjective**
Visiting relatives can be boring.
Changing schedules frequently confused passengers.

Syntactic Ambiguities II

- **Modifier scope within NPs**
impractical design requirements
plastic cup holder
- **Multiple gap constructions**
The chicken is ready to eat.
The contractors are rich enough to sue.
- **Coordination scope:**
Small rats and mice can squeeze into holes or cracks in the wall.

Probabilistic Context-Free Grammars

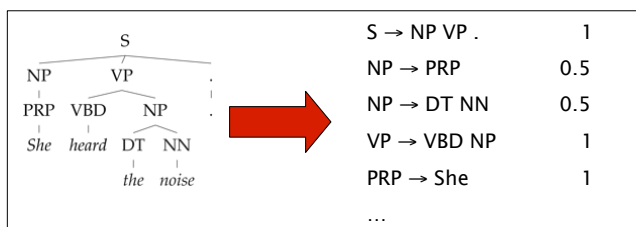
- **A context-free grammar is a tuple $\langle N, T, S, R \rangle$**
 - N : the set of non-terminals
 - Phrasal categories: S, NP, VP, ADJP, etc.
 - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
 - T : the set of terminals (the words)
 - S : the start symbol
 - Often written as ROOT or TOP
 - Not usually the sentence non-terminal S
 - R : the set of rules
 - Of the form $X \rightarrow Y_1 Y_2 \dots Y_k$, with $X, Y_i \in N$
 - Examples: $S \rightarrow NP VP$, $VP \rightarrow VP CC VP$
 - Also called rewrites, productions, or local trees
- **A PCFG adds:**
 - **A top-down production probability per rule $P(Y_1 Y_2 \dots Y_k | X)$**

Trebank Sentences

```
( (S (NP-SBJ The move)
  (VP followed
    (NP (NP a round)
      (PP of
        (NP (NP similar increases)
          (PP by
            (NP other lenders)))
          (PP against
            (NP Arizona real estate loans))))))
  (S-ADV (NP-SBJ *)
    (VP reflecting
      (NP (NP a continuing decline)
        (PP-LOC in
          (NP that market))))))
  .))
```

Trebank Grammars

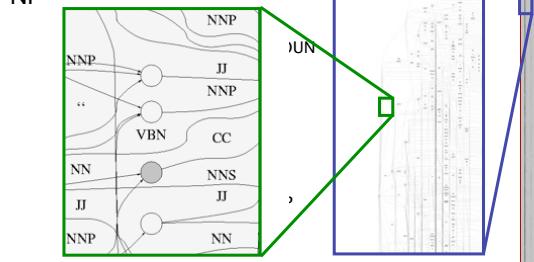
- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn't work well):



- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers without lexicalization.

Trebank Grammar Scale

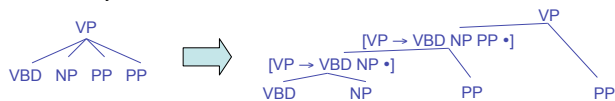
- **Trebank grammars can be enormous**
 - As FSAs, the raw grammar has ~10K states, excluding the lexicon
 - Better parsers usually make the grammars larger, not smaller



Chomsky Normal Form

Chomsky normal form:

- All rules of the form $X \rightarrow YZ$ or $X \rightarrow w$
- In principle, this is no limitation on the space of (P)CFGs
 - N-ary rules introduce new non-terminals



- Unaries / empties are "promoted"
- In practice it's kind of a pain:
 - Reconstructing n-aries is easy
 - Reconstructing unaries is trickier
 - The straightforward transformations don't preserve tree scores
- Makes parsing algorithms simpler!

A Recursive Parser

```
bestScore(X,i,j,s)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return max score(X->YZ) *
              bestScore(Y,i,k) *
              bestScore(Z,k,j)
```

- Will this parser work?
- Why or why not?
- Memory requirements?

A Memoized Parser

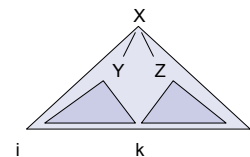
One small change:

```
bestScore(X,i,j,s)
  if (scores[X][i][j] == null)
    if (j = i+1)
      score = tagScore(X,s[i])
    else
      score = max score(X->YZ) *
              bestScore(Y,i,k) *
              bestScore(Z,k,j)
    scores[X][i][j] = score
  return scores[X][i][j]
```

A Bottom-Up Parser (CKY)

- Can also organize things bottom-up

```
bestScore(s)
  for (i : [0,n-1])
    for (X : tags[s[i]])
      score[X][i][i+1] =
        tagScore(X,s[i])
  for (diff : [2,n])
    for (i : [0,n-diff])
      j = i + diff
      for (X->YZ : rule)
        for (k : [i+1, j-1])
          score[X][i][j] = max score[X][i][j],
                              score(X->YZ) *
                              score[Y][i][k] *
                              score[Z][k][j]
```



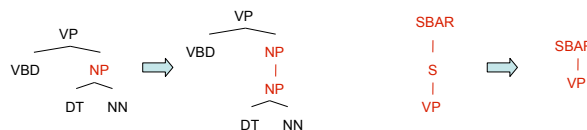
Unary Rules

Unary rules?

```
bestScore(X,i,j,s)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return max max score(X->YZ) *
              bestScore(Y,i,k) *
              bestScore(Z,k,j)
              max score(X->Y) *
              bestScore(Y,i,j)
```

CNF + Unary Closure

- We need unaries to be non-cyclic
 - Can address by pre-calculating the *unary closure*
 - Rather than having zero or more unaries, always have exactly one



- Alternate unary and binary layers
- Reconstruct unary chains afterwards

Alternating Layers

```
bestScoreB(X,i,j,s)
  return max max score(X->YZ) *
                bestScoreU(Y,i,k) *
                bestScoreU(Z,k,j)
```

```
bestScoreU(X,i,j,s)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return max max score(X->Y) *
                bestScoreB(Y,i,j)
```

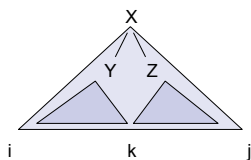
Memory

- How much memory does this require?
 - Have to store the score cache
 - Cache size: $|\text{symbols}|^2 n^2$ doubles
 - For the plain treebank grammar:
 - $X \sim 20K$, $n = 40$, double ~ 8 bytes = $\sim 256\text{MB}$
 - Big, but workable.
- Pruning: Beams
 - $\text{score}[X][i][j]$ can get too large (when?)
 - Can keep beams (truncated maps $\text{score}[i][j]$) which only store the best few scores for the span $[i,j]$
- Pruning: Coarse-to-Fine
 - Use a smaller grammar to rule out most $X[i,j]$
 - Much more on this later...

Time: Theory

- How much time will it take to parse?

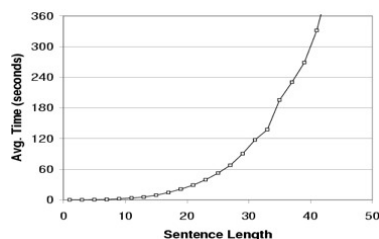
- For each diff ($\leq n$)
 - For each i ($\leq n$)
 - For each rule $X \rightarrow YZ$
 - For each split point k
 - Do constant work



- Total time: $|\text{rules}| \cdot n^3$
- Something like 5 sec for an unoptimized parse of a 20-word sentences

Time: Practice

- Parsing with the vanilla treebank grammar:



$\sim 20K$ Rules
(not an optimized parser!)
Observed exponent:
3.6

- Why's it worse in practice?
 - Longer sentences "unlock" more of the grammar
 - All kinds of systems issues don't scale

Efficient CKY

- Lots of tricks to make CKY efficient
 - Most of them are little engineering details:
 - E.g., first choose k , then enumerate through the $Y:[i,k]$ which are non-zero, then loop through rules by left child.
 - Optimal layout of the dynamic program depends on grammar, input, even system details.
 - Another kind is more critical:
 - Many $X:[i,j]$ can be suppressed on the basis of the input string
 - We'll see this next class as figures-of-merit or A^* heuristics

Human Processing

- Garden pathing:

The cotton clothing is made of grows in Mississippi.

The author wrote the novel was likely to be a best-seller.

The complex houses married and single soldiers and their families.

- Ambiguity maintenance

Have the police . . . eaten their supper?
come in and look around.
taken out and shot.

Agenda-Based Parsing

- Agenda-based parsing is like graph search (but over a hypergraph)
- Concepts:
 - Numbering: we number fenceposts between words
 - "Edges" or items: spans with labels, e.g. PP[3,5], represent the sets of trees over those words rooted at that label (cf. search states)
 - A chart: records edges we've expanded (cf. closed set)
 - An agenda: a queue which holds edges (cf. a fringe or open set)



Word Items

- Building an item for the first time is called discovery. Items go into the agenda on discovery.
- To initialize, we discover all word items (with score 1.0).

AGENDA

critics[0, 1], write[1,2], reviews[2,3], with[3,4], computers[4,5]

CHART [EMPTY]



Item Successors

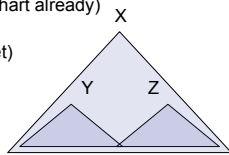
- When we pop items off of the agenda:
 - Graph successors: unary projections (NNS → critics, NP → NNS)

$Y[i,j]$ with $X \rightarrow Y$ forms $X[i,j]$

- Hypergraph successors: combine with items already in our chart

$Y[i,j]$ and $Z[j,k]$ with $X \rightarrow YZ$ form $X[i,k]$

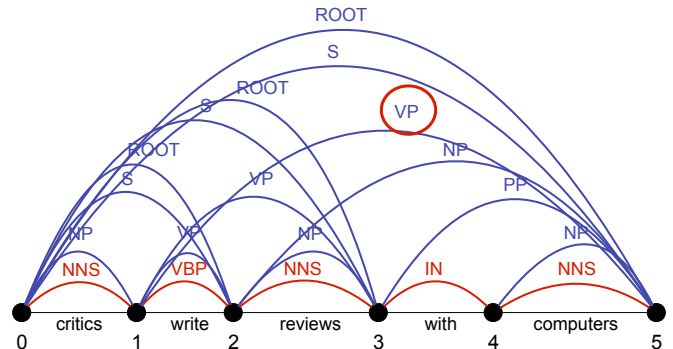
- Enqueue / promote resulting items (if not in chart already)
- Record backtraces as appropriate
- Stick the popped edge in the chart (closed set)



- Queries a chart must support:
 - Is edge $X[i,j]$ in the chart? (What score?)
 - What edges with label Y end at position j ?
 - What edges with label Z start at position i ?

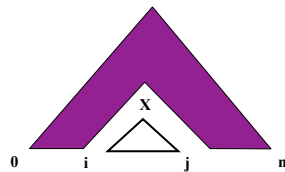
An Example

NNS[0,1] VBP[1,2] NNS[2,3] IN[3,4] NNS[3,4] NP[0,1] VP[1,2] NP[2,3] NP[4,5] S[0,2]
 VP[1,3] PP[3,5] ROOT[0,2] S[0,3] VP[1,5] NP[2,5] ROOT[0,3] S[0,5] ROOT[0,5]



Uniform Cost Search / A*

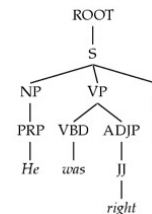
- With weighted edges, order matters
 - Must expand optimal parse from bottom up (subparses first)
 - CKY does this by processing smaller spans before larger ones
 - UCS pops items off the agenda in order of decreasing Viterbi score
 - A* search also well defined



- You can also speed up the search without sacrificing optimality
 - Can select which items to process first
 - Can do with any "figure of merit" [Charniak 98]
 - If your figure-of-merit is a valid A* heuristic, no loss of optimality [Klein and Manning 03]

Trebank PCFGs [Charniak '96]

- Use PCFGs for broad coverage parsing
- Can take a grammar right off the trees (doesn't work well):

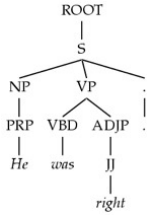


ROOT → S	1.0
S → NP VP .	1.0
NP → PRP	1.0
VP → VBD ADJP	1.0
.....	

Model	F1
Baseline	72.0

Treebank PCFGs [Charniak '96]

- Use PCFGs for broad coverage parsing
- Can take a grammar right off the trees (doesn't work well):

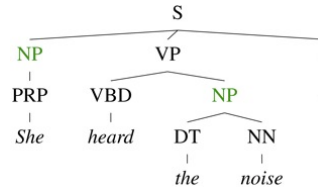


ROOT → S	1.0
S → NP VP .	1.0
NP → PRP	1.0
VP → VBD ADJP	1.0
.....	

Model	F1
Baseline	72.0

Conditional Independence?

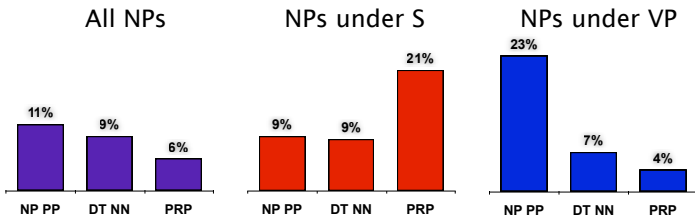
- Not every NP expansion can fill every NP slot



- A grammar with symbols like "NP" won't be context-free
- Statistically, conditional independence too strong

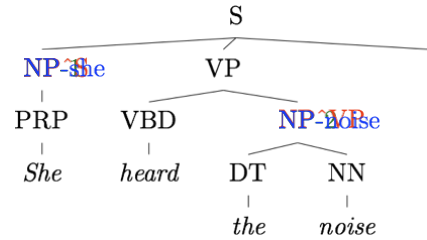
Non-Independence

- Independence assumptions are often too strong.



- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).
- Also: the subject and object expansions are correlated!

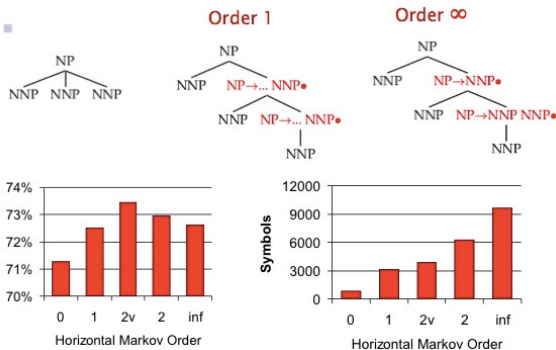
The Game of Designing a Grammar



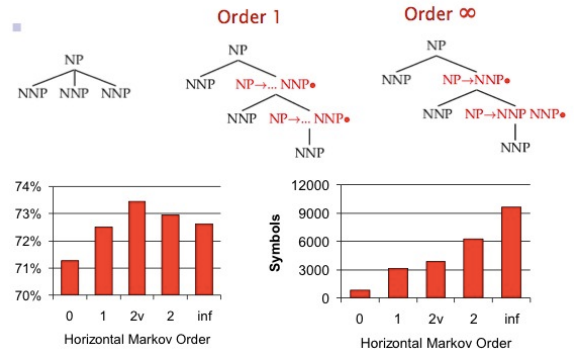
- Structure Annotation [Johnson '98, Klein&Manning '03]
- Lexicalization [Collins '99, Charniak '00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. '06]

Vertical Markovization

[Johnson '98]

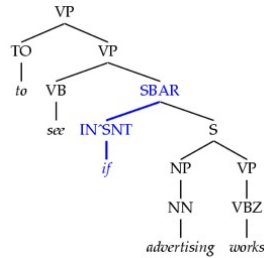


Horizontal Markovization



Tag Splits

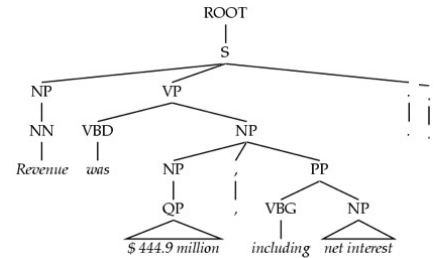
- Problem: Treebank tags are too coarse.
- Example: Sentential, PP, and other prepositions are all marked IN.
- Partial Solution:
 - Subdivide the IN tag.



Annotation	F1	Size
Previous	78.3	8.0K
SPLIT-IN	80.3	8.1K

Unary Splits ^[Klein&Manning '03]

- Problem: unary rewrites used to transmute categories so a high-probability rule can be used.
- Solution: Mark unary rewrite sites with -U



Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K

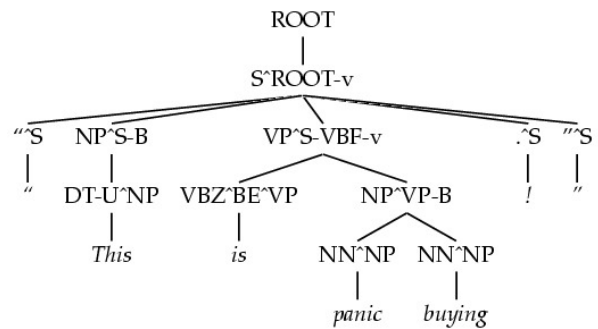
Other Tag Splits

- UNARY-DT: mark demonstratives as DT^U ("the X" vs. "those")
- UNARY-RB: mark phrasal adverbs as RB^U ("quickly" vs. "very")
- TAG-PA: mark tags with non-canonical parents ("not" is an RB^VP)
- SPLIT-AUX: mark auxiliary verbs with -AUX [cf. Charniak 97]
- SPLIT-CC: separate "but" and "&" from other conjunctions
- SPLIT-%: "%" gets its own tag.

	F1	Size
UNARY-DT	80.4	8.1K
UNARY-RB	80.5	8.1K
TAG-PA	81.2	8.5K
SPLIT-AUX	81.6	9.0K
SPLIT-CC	81.7	9.1K
SPLIT-%	81.8	9.3K

A Fully Annotated (Unlexicalized) Tree

[Klein & Manning '03]

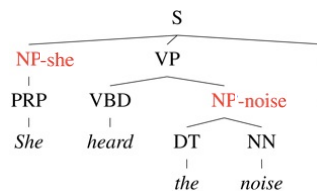


Some Test Set Results

Parser	LP	LR	F1	CB	0 CB
Magerman 95	84.9	84.6	84.7	1.26	56.6
Collins 96	86.3	85.8	86.0	1.14	59.9
Unlexicalized	86.9	85.7	86.3	1.10	60.3
Charniak 97	87.4	87.5	87.4	1.00	62.1
Collins 99	88.7	88.6	88.6	0.90	67.1

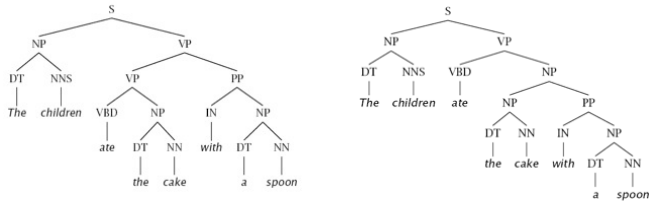
- Beats "first generation" lexicalized parsers.
- Lots of room to improve – more complex models next.

The Game of Designing a Grammar



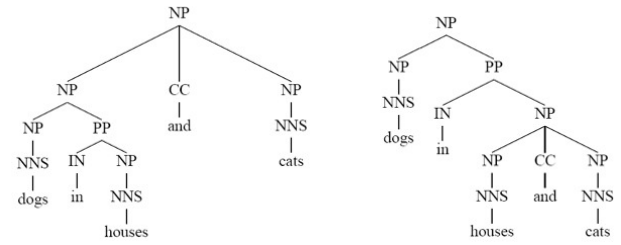
- Annotation refines base treebank symbols to improve statistical fit of the grammar
 - Structural annotation [Johnson '98, Klein and Manning 03]
 - Head lexicalization [Collins '99, Charniak '00]

Problems with PCFGs



- If we do no annotation, these trees differ only in one rule:
 - VP → VP PP
 - NP → NP PP
- Parse will go one way or the other, regardless of words
- We addressed this in one way with unlexicalized grammars (how?)
- Lexicalization allows us to be sensitive to specific words

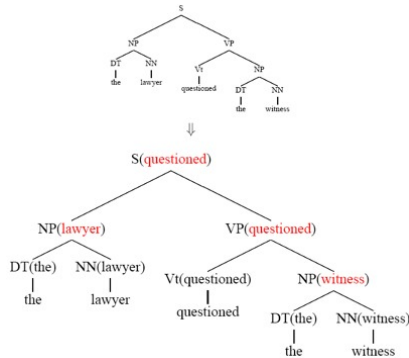
Problems with PCFGs



- What's different between basic PCFG scores here?
- What (lexical) correlations need to be scored?

Lexicalized Trees [Charniak '97, Collins '97]

- Add "headwords" to each phrasal node
 - Syntactic vs. semantic heads
 - Headship not in (most) treebanks
 - Usually use head rules, e.g.:
 - NP:
 - Take leftmost NP
 - Take rightmost N*
 - Take rightmost JJ
 - Take right child
 - VP:
 - Take leftmost VB*
 - Take leftmost VP
 - Take left child



Lexicalized PCFGs?

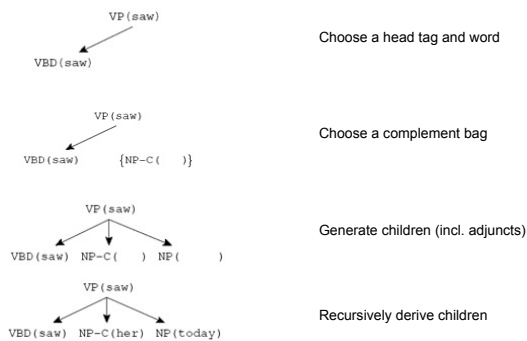
- Problem: we now have to estimate probabilities like

$$VP(\text{saw}) \rightarrow VBD(\text{saw}) NP-C(\text{her}) NP(\text{today})$$
- Never going to get these atomically off of a treebank
- Solution: break up derivation into smaller steps



Lexical Derivation Steps

- A derivation of a local tree [Collins 99]



Lexicalized CKY

$$(VP \rightarrow VBD \dots NP^*) [saw]$$

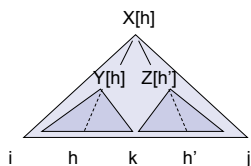
$$(VP \rightarrow VBD^*) [saw] \quad NP[her]$$

```

bestScore(X, i, j, h)
if (j = i+1)
    return tagScore(X, s[i])
else
    return
    max_k, h', X->Y2
    max score(X[h] -> Y[h] Z[h']) *
        bestScore(Y, i, k, h) *
        bestScore(Z, k, j, h')
    max_k, h', X->Y2
    max score(X[h] -> Y[h'] Z[h]) *
        bestScore(Y, i, k, h') *
        bestScore(Z, k, j, h)
    
```

Pruning with Beams

- The Collins parser prunes with per-cell beams [Collins 99]
 - Essentially, run the $O(n^5)$ CKY
 - Remember only a few hypotheses for each span $\langle i, j \rangle$.
 - If we keep K hypotheses at each span, then we do at most $O(nK^2)$ work per span (why?)
 - Keeps things more or less cubic
- Also: certain spans are forbidden entirely on the basis of punctuation (crucial for speed)

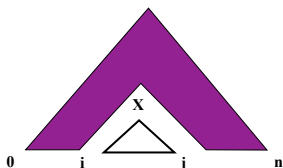


Pruning with a PCFG

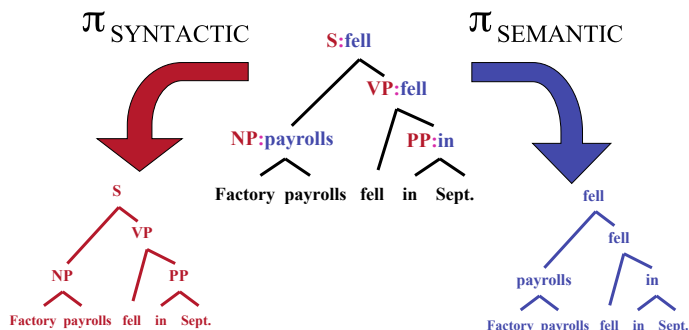
- The Charniak parser prunes using a two-pass approach [Charniak 97+]
 - First, parse with the base grammar
 - For each $X:[i, j]$ calculate $P(X|i, j, s)$
 - This isn't trivial, and there are clever speed ups
 - Second, do the full $O(n^5)$ CKY
 - Skip any $X:[i, j]$ which had low (say, < 0.0001) posterior
 - Avoids almost all work in the second phase!
- Charniak et al 06: can use more passes
- Petrov et al 07: can use many more passes

Pruning with A*

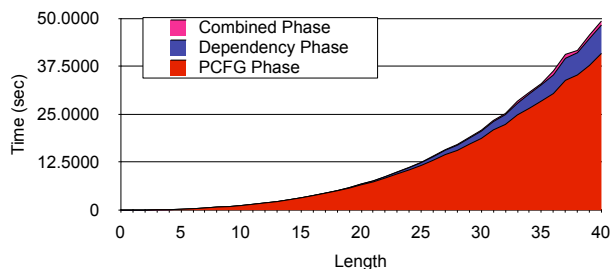
- You can also speed up the search without sacrificing optimality
- For agenda-based parsers:
 - Can select which items to process first
 - Can do with any "figure of merit" [Charniak 98]
 - If your figure-of-merit is a valid A* heuristic, no loss of optimality [Klein and Manning 03]



Projection-Based A*



A* Speedup



- Total time dominated by calculation of A* tables in each projection... $O(n^3)$