

Statistical NLP

Spring 2008

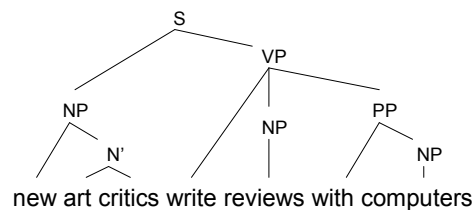
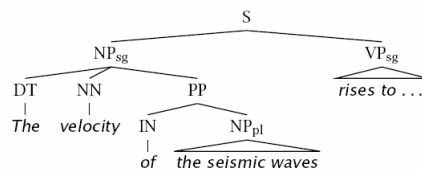


Lecture 14: Parsing I

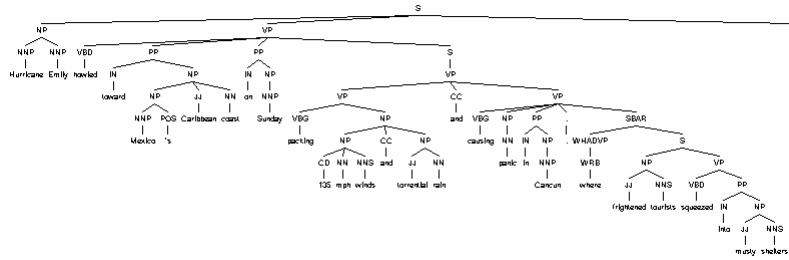
Dan Klein – UC Berkeley

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...



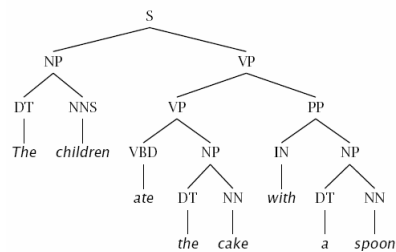
Example Parse



Hurricane Emily howled toward Mexico 's Caribbean coast on Sunday packing 135 mph winds and torrential rain and causing panic in Cancun , where frightened tourists squeezed into musty shelters .

Constituency Tests

- How do we know what nodes go in the tree?
- Classic constituency tests:
 - Substitution by *proform*
 - Question answers
 - Semantic reference
 - Dislocation
- Cross-linguistic arguments, too



Conflicting Tests

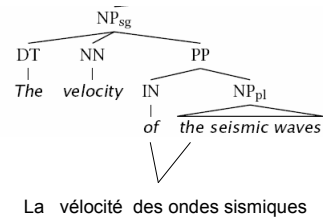
- Constituency isn't always clear

- 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



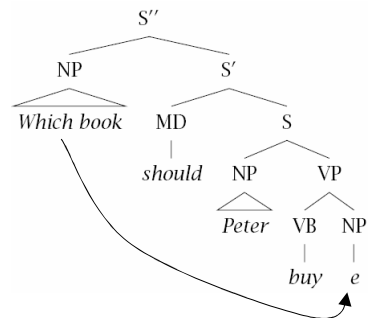
Non-Local Phenomena

- Dislocation / gapping

- Why did the postman think that the neighbors were home?
- A debate arose which continued until the election.

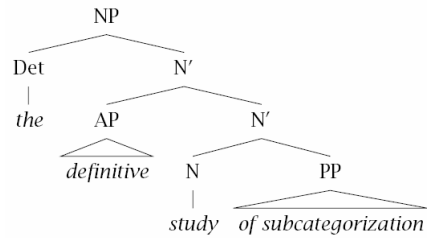
- Binding

- Reference
 - The IRS audits itself
- Control
 - I want to go
 - I want you to go

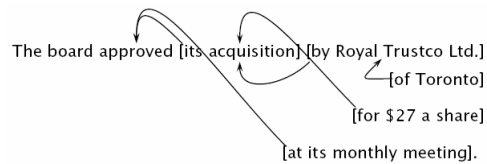
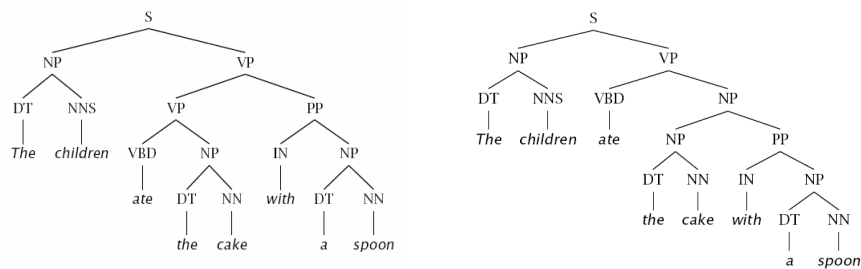


Regularity of Rules

- Argumentation
- Adjunction
- Coordination
- X' Theory



PP Attachment



PP Attachment

V	N1	P	N2	Attachment
join	board	as	director	V
is	chairman	of	N.V.	N
using	crocidolite	in	filters	V
bring	attention	to	problem	V
is	asbestos	in	products	N
making	paper	for	filters	N
including	three	with	cancer	N

Method	Accuracy
Always noun attachment	59.0
Most likely for each preposition	72.2
Average Human (4 head words only)	88.2
Average Human (whole sentence)	93.2

Attachments

- I cleaned the dishes from dinner
- I cleaned the dishes with detergent
- 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:**
*A good pharmacist dispenses with accuracy.
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.

Human Processing

- Garden pathing:

the man who hunts ducks out on weekends

the cotton shirts are made from grows in Mississippi

the daughter of the king's son loves himself

- Ambiguity maintenance

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

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

Probabilistic Context-Free Grammars

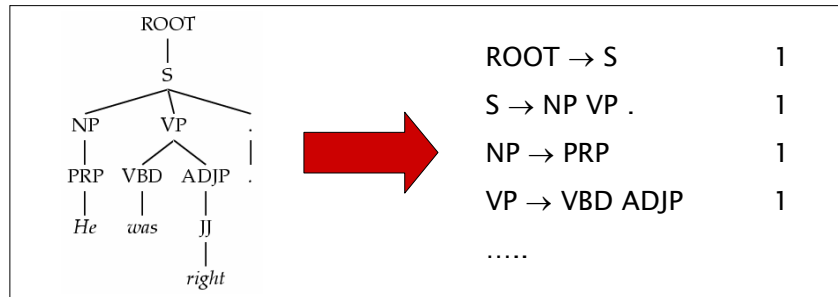
- A context-free grammar is a tuple $\langle N, T, S, R \rangle$
 - N : the set of non-terminals
 - Phrasal categories: S, NP, VP, ADJP, etc.
 - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
 - T : the set of terminals (the words)
 - S : the start symbol
 - Often written as ROOT or TOP
 - *Not* usually 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 $P(Y_1 Y_2 \dots Y_k | X)$

Treebank 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))))))
  .))
```


Treebank Grammars

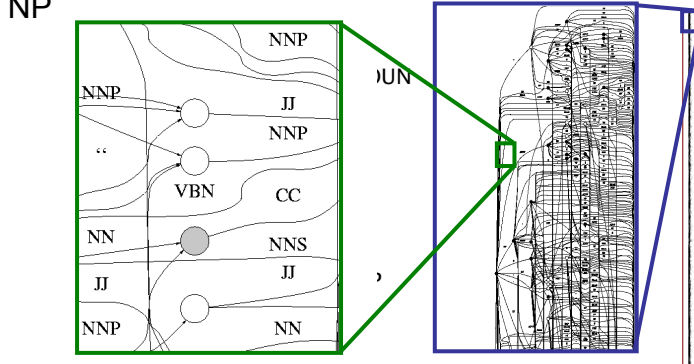
- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn't work well):



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

Treebank Grammar Scale

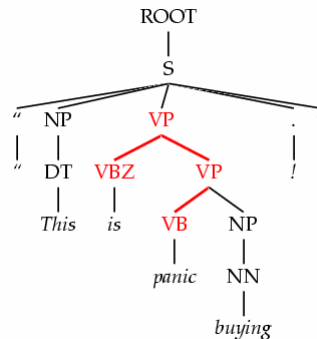
- Treebank grammars can be enormous
 - As FSAs, the raw grammar has ~10K states, excluding the lexicon
 - Better parsers usually make the grammars larger, not smaller



Dark Ambiguities

- *Dark ambiguities*: most analyses are shockingly bad (meaning, they don't have an interpretation you can get your mind around)

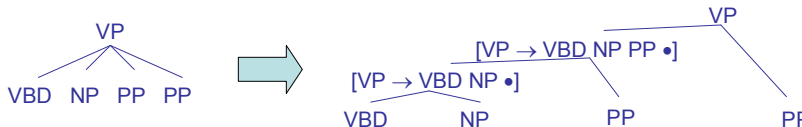
This analysis corresponds to the correct parse of
 "This will panic buyers !"



- **Unknown words and new usages**
- **Solution**: We need mechanisms to focus attention on the best ones, probabilistic techniques do this

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

- Here's a recursive (CNF) parser:

```
bestParse(X,i,j,s)
  if (j = i+1)
    return X -> s[i]
  (X->YZ,k) = argmax score(X->YZ) *
                    bestScore(Y,i,k,s) *
                    bestScore(Z,k,j,s)
  parse.parent = X
  parse.leftChild = bestParse(Y,i,k,s)
  parse.rightChild = bestParse(Z,k,j,s)
  return parse
```

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]
```

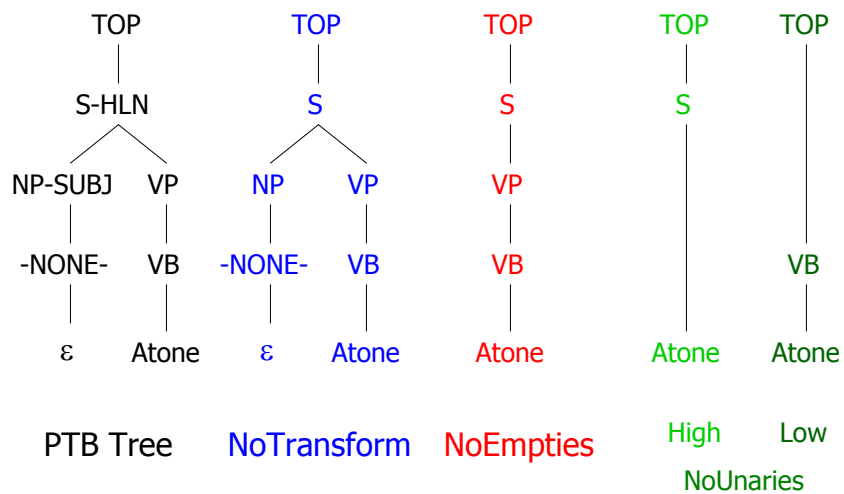
Memory: Theory

- How much memory does this require?
 - Have to store the score cache
 - Cache size: |symbols|*n² doubles
 - For the plain treebank grammar:
 - X ~ 20K, n = 40, double ~ 8 bytes = ~ 256MB
 - Big, but workable.
- What about sparsity?

Time: Theory

- How much time will it take to parse?
 - Have to fill each cache element (at worst)
 - Each time the cache fails, we have to:
 - Iterate over each rule $X \rightarrow Y Z$ and split point k
 - Do constant work for the recursive calls
 - Total time: $|\text{rules}| * n^3$
 - Cubic time
 - Something like 5 sec for an unoptimized parse of a 20-word sentences

Unaries in Grammars



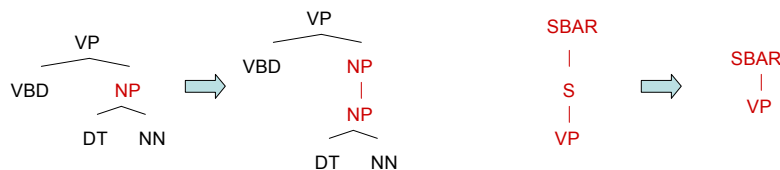
Unary Rules

- 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)
```

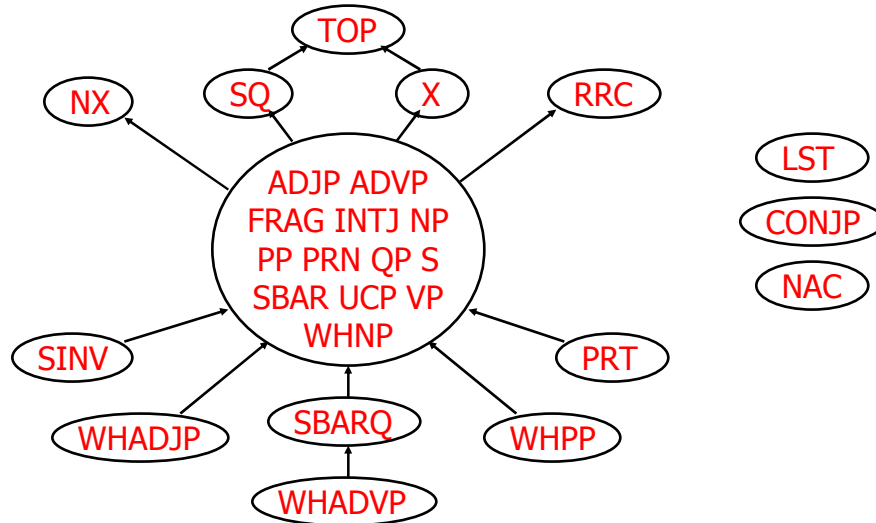
CNF + Unary Closure

- 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

Same-Span Reachability



Alternating Layers

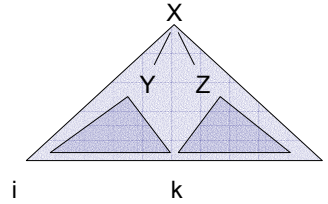
```
bestScoreB(X,i,j,s)
    return max max score(X->YZ) *
                bestScoreU(Y,i,k) *
                bestScoreU(Z,k,j)

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)
```

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]
```



Efficient CKY

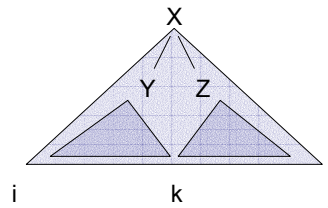
- Lots of tricks to make CKY efficient
 - Most of them are little engineering details:
 - E.g., first choose k, then enumerate through the Y:[i,k] which are non-zero, then loop through rules by left child.
 - Optimal layout of the dynamic program depends on grammar, input, even system details.
 - Another 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

Memory: Practice

- **Memory:**
 - Still requires memory to hold the score table
- **Pruning:**
 - $\text{score}[X][i][j]$ can get too large (when?)
 - can instead keep beams scores $\text{scores}[i][j]$ which only record scores for the top K symbols found to date for the span $[i,j]$

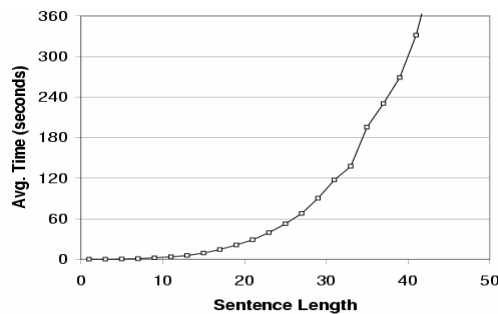
Time: Theory

- **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: $|\text{rules}| * n^3$



Runtime: Practice

- Parsing with the vanilla treebank grammar:



~ 20K Rules

(not an optimized parser!)

Observed exponent:

3.6

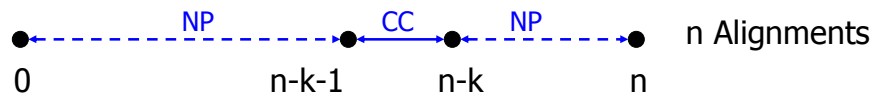
- Why's it worse in practice?
 - Longer sentences “unlock” more of the grammar
 - All kinds of systems issues don't scale

Rule State Reachability

Example: NP CC •



Example: NP CC NP •



- Many states are more likely to match larger spans!