Parsing

- Given a grammar G and a sentence $x = (x_1, x_2, ..., x_n)$, find the best parse tree.

- We’re not going to simply build it step by step; we need to entertain many partial possibilities in parallel.

First View: Parsing as Search

Trees break into pieces (partial trees), which can be used to define a search space.
Dynamic Programming

- Where did we see dynamic programming before?
- “Solve sub-problems, store results in a table.”
- Reuse work you’ve done already (runtime)
- Pack sub-problems efficiently (memory)

- First dynamic programming parser: CKY

Chomsky Normal Form

- \( G = (\Sigma, N, S, R) \)
- \( \Sigma \): Vocabulary of terminal symbols
- \( N \): set of nonterminal symbols (AKA variables)
- \( S \in N \): special start symbol
- \( R \): Production rules of the form
  \[ X \rightarrow \alpha \]
  where \( X \in N \) (a nonterminal symbol) and \( \alpha \in N^2 \cup \Sigma \)

Conversion to CNF

- \( S \rightarrow \text{WhNP \ Aux \ VP} \ ? \)
- \( S \rightarrow \text{WhNP \ Z}_{\text{Aux VP}} \ ? \)
- \( Z_{\text{Aux VP}} \ ? \rightarrow \text{Aux \ Z}_{\text{VP}} \ ? \)
- \( Z_{\text{VP}} \ ? \rightarrow \text{VP \ Z} \ ? \)
- \( Z \ ? \rightarrow \ ? \)

- After parsing, “flatten” the \( Z \) phrases.
Conversion to CNF

- VP → V

- Find all instances of VP on the right-hand side.
  - S → NP VP

- Create new rules with V substituted for V
  - S → NP V

- Keep old rules, too.

CKY Schema

- Data structure: chart.

- In each cell, store the possible phrase types.

- Visit each cell once.

- Start with narrow spans, progress to longer ones.

CKY Example

SLP p. 442
CKY Schema

CKY Pseudocode

- Input: x, G
- Output: C
- for i = 1 ... n
  • C[i-1, i] = {A | A → x_i}
- for ℓ = 2 ... n
  • for i = 0 ... n - ℓ
    • k = i + ℓ
    • for j = i + 1 ... k - 1
      • C[i, k] = C[i, j] ∪ {X | X → YZ, Y ∈ C[i, j], Z ∈ C[j, k]}
- return true if S ∈ C[0, n]

CKY Equations

\[

c[i - 1, i, x] = \text{true}
\]
\[
c[i - 1, i, V] = \begin{cases} 
\text{true} & \text{if } V \rightarrow x, \\
\text{false} & \text{otherwise}
\end{cases}
\]
\[
c[i, j, V] = \begin{cases} 
\text{true} & \text{if } |3; j, Y, Z \text{ such that } V \rightarrow YZ \text{ and } C[i, j, V] \text{ and } C[j, k, Z]| \\
\text{false} & \text{otherwise}
\end{cases}
\]
\[
goal = C[0, n, S]
\]
What Else?

- So far, this only does recognition.
- Can we recover parse trees? (Need to store some more information.)
- Remember that we may need to undo our transformation to CNF.
- Flatten the newly created rules
- Extend unary chains

Classical NLP: Parsing

Fed raises interest rates 0.5 percent

- Write symbolic or logical rules:

<table>
<thead>
<tr>
<th>Grammar (CFG)</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT → S</td>
<td>NN → interest</td>
</tr>
<tr>
<td>S → NP VP</td>
<td>VBP → raises</td>
</tr>
<tr>
<td>NP → DT NN</td>
<td>VBZ → raises</td>
</tr>
<tr>
<td>VP → VBP NP</td>
<td>VBP → interest</td>
</tr>
<tr>
<td>NP → NN NNS</td>
<td>PP → IN NP</td>
</tr>
<tr>
<td>PP → IN NP</td>
<td>...</td>
</tr>
</tbody>
</table>

- 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

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
Treebanks

New Parsing Topic

• Where do the rules come from?
• Writing the rules by hand grows tedious!
• Neat idea from 1993: Treebank
• Penn Treebank: 1 million words of Wall Street Journal articles from the late 1980s, parsed by experts

Example Tree

( (S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken) )
    (.,) )
  (ADJP
    (NP (CD 61) (NNS years) )
    (JJ old) )
  (.,) )
  (VP (MD will)
    (VP (VB join)
      (NP (DT the) (NN board) )
      (PP-CLR (IN as)
        (NP (DT a) (JJ nonexecutive) (NN director) ))
      (NP-TMP (NNP Nov.) (CD 29) ))
    (.,) )
  (.,) ))
Extracting Rules from the Treebank

- Each nonterminal node in each tree is a rule.
- Simply traverse the trees and store the rules you see.
- “Treebank grammar”
  - Huge number of rules, many rather large
  - But all are known to be “good” in the sense that they “can happen”
  - Want some notion of “frequent rules”

Some Rules

40717 PP -> IN NP
33803 S -> NP-SBJ/VP
22513 NP-SBJ -> -NONE-
21877 NP -> NP PP
20740 NP -> DT NN
19597 S -> NP-SBJ VP...
100 VP -> VBD VP-PRD
100 PRN -> - NP-
100 NP -> DT JJ...
100 NP -> NP CLR -> NN
100 NP-SBJ -> DT A -> (JJ nonexecutive) (NN director)
100 PP -> VBN PP-LOC -> IN NP
100 NP-SBJ -> DT
100 NP-SBJ -> NN
10 WH-NP-1 -> WRB JJ
10 VP -> VP CC VP ADVP-MNR
10 VP -> VBD PP-LOC -> IN NP
10 NP-SBJ -> DT
10 VP -> VBD S ADVP-TMP
10 VP -> VBD S ADVP-TMP

1 NP -> NP , NP , NP , NP , NP , NP , NP , NP , NP , NP , NP , NP , NP , NP , NP , NP , NP , NP , NP CC NP
CFG Rules in the Penn Treebank

rules in the training section: 32,728 (+ 52,257 lexicon)

rules in the dev section: 4,021 (78%)

Rule Distribution (Training Sections)

One Use of Treebanks: Parser Evaluation

- Use (part of) a treebank as a test set.
- How many sentences are parsed correctly?
- What's wrong?
Parseval Scores

- Gold standard tree can be seen as a set of tuples (label, start position, end position)
- Hypothesis tree, too.
- Compare these on precision and recall:

```
constituents in gold standard trees
```

```
constituents in parser output
```

Probabilistic (or Stochastic) CFG

- $G = (\Sigma, N, S, R)$
- $\Sigma$: Vocabulary of terminal symbols
- $N$: set of nonterminal symbols (AKA variables)
- $S \in N$: special start symbol
- $R$: Production rules of the form $X \rightarrow \alpha$
  where $X \in N$ (a nonterminal symbol) and $\alpha \in (N \cup \Sigma)^*$ (a sequence of terminals and nonterminals)

- Each rule gets a probability.
- Rules with the same left side sum to 1.
- Each nonterminal has a "die."

PCFGs as Generators

- Start out by writing down the start symbol.
- While there are unexpanded nonterminals:
  - Pick one arbitrarily.
  - Roll that nonterminal’s die to decide how to rewrite it, choosing a production rule.
  - Execute the production rule.

- This is a stochastic process for generating sentences; we saw this before with N-gram and hidden Markov models!
a executive foundation to persuade the OOV

• (START (S (NP (DT a) (JJ executive) (NN foundation)) (VP (TO to) (VP (VB persuade) (NP (DT the) (NN OOV)))))

of least major minorities Labor requiring by That Irish C$ <N> <N> binge about

• (START (S (PP (IN of) (NP (JJS least) (JJ major) (NNS minorities))) (NP (NNP Labor)) (VP (VBG requiring) (PP (IN by) (NP (DT That) (JJ Irish) (ADJP ($ C$) (CD <N>) (CD <N>)) (NN binge))) (ADVP (RB about)))))

tell that him move The amount brought you of or assertions For another city of who OOV Senate of he It of Manufacturers Mr. in Sumitomo 's Co. Texaco Dec. to Mr. Chancellor hold bonds or its cost p.m. has analog such kind SEC Raleigh to have Soup ? to recent prices had the profit a industrial year ?

• (START (SINV (VP (VBD fell) (SBAR (WHNP (WDT that)) (S (NP (PRP him)) (VP (VB move) (NP (NP (DT The) (NN amount)) (VP (VB brought) (NP (PRP you)) (PP (IN of) (S (CC or) (NP (NP (NP (NNP assertions)) (PP (IN For) (NP (DT another) (NN city)) (PP (IN of) (SBAR (WHNP (WP who) (S (INP (NNP OOV) (NNP Senate)) (PP (IN of) (NP (NP (NP (NP (NNP he) (NP (NNP (NNP Manufacturers) (NNNP Mt.)) (PP (IN in) (NP (NP (NP (NNNP Sumitomo) (POS 's)) (NNNP Co.) (NNNP Texaco) (NNNP Dec.) (PP (TO to) (NP (NP Mt.) (NNP Chancellor)))) (VP (VB hold) (NP (NNP bonds)))) (CC or) (NP (PRP its) (NN cost)) (ADVP (RB p.m.)) (VP (VBZ has) (SBAR (S (NP (JJ analog) (JJ such)) (NN kind)) (NNP SEC (NNP Raleigh)) (VP (TO to) (NP (VB have) (NP (NP (NNP Soup)))) (., ?)) (PP (TO to) (NP (JJ recent) (NNP prices)))) (VP (VBZ had) (NP (DT the) (NN profit))) (NP (DT a) (JJ industrial) (NN year))), (?))))) (PP (TO to) (NP (JJ recent) (NNP prices)))) (VP (VBZ had) (NP (DT the) (NN profit))) (NP (DT a) (JJ industrial) (NN year))), (?))))))

• (START (S (PP (IN of) (NP (JJ least) (JJ major) (NNS minorities))) (NP (NNP Labor)) (VP (VBG requiring) (PP (IN by) (NP (DT That) (JJ Irish) (ADJP ($ C$) (CD <N>) (CD <N>)) (NN binge))) (ADVP (RB about)))))
Statistical Parsing

- Originally, parsing was verification (is w in the language?)
- Then it was meant to recover structures of the sentence.
- Now: find the most probable structure.

CKY Equations

\[
C[i-1, i, x_i] = \text{true}
\]
\[
C[i-1, i, V] = \begin{cases} 
\text{true} & \text{if } V \rightarrow x_i, \\
\text{false} & \text{otherwise}
\end{cases}
\]
\[
C[i, k, V] = \begin{cases} 
\text{true} & \text{if } \exists j, Y, Z \text{ such that } V \rightarrow YZ \text{ and } C[i, j, Y] \text{ and } C[j, k, Z], \\
\text{false} & \text{otherwise}
\end{cases}
\]
goal = \(C[0, n, S]\)

Probabilistic CKY

\[
C[i-1, i, x_i] = \text{true}
\]
\[
C[i-1, i, V] = \begin{cases} 
\text{true} & \text{if } V \rightarrow x_i, \\
\text{false} & \text{otherwise}
\end{cases}
\]
\[
C[i, k, V] = \begin{cases} 
\text{true} & \text{if } \exists j, Y, Z \text{ such that } V \rightarrow YZ \text{ and } C[i, j, Y] \text{ and } C[j, k, Z], \\
\text{false} & \text{otherwise}
\end{cases}
\]
goal = \(C[0, n, S]\)

\[
C[i-1, i, x_i] = 1
\]
\[
C[j-1, i, V] = p(V \rightarrow x_i)
\]
\[
C[j, k, V] = \max_{\psi(i-1, j, k) \in \mathcal{X}, \mathcal{A} \in \mathcal{A}} p(V \rightarrow YZ) \times C[i, j, Y] \times C[j, k, Z]
\]
\[
p(* | \{x_1 \ldots x_n\}) = C[0, n, S]\]
**Probabilistic CKY Pseudocode**

- **Input:** $x, G$
- **Output:** $C$
- **for** $i = 1 \ldots n$
  - $C[A, i-1, i] = p(A \rightarrow x)$
- **for** $f = 2 \ldots n$
  - **for** $i = 0 \ldots n - f$
    - $k = i + f$
    - **for** $j = i + 1 \ldots k - 1$
      - $C[X, i, k] = \max(C[X, i, k], p(X \mid X \rightarrow YZ) \times C[Y, i, j] \times C[Z, j, k])$
- return $C[S, 0, n]$

**Probabilistic CKY**

- Exploits shared substructure to optimize over an exponentially large set of trees, in polynomial time.

- Similar to Viterbi algorithm, minimum edit distance, ...

**Typical Experimental Setup**

- **Corpus:** Penn Treebank, WSJ
  
  - Training: sections 02-21
  - Development: section 22 (here, first 20 files)
  - Test: section 23

- **Accuracy – F1:** harmonic mean of per-node labeled precision and recall.
Treebank PCFGs

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

```
ROOT  
  S   
     NP  VP  
       PRP VBD ADJP  
         Me  was  right
```

Model  F1
Baseline 72.0

Problems with PCFGs

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

The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
- Head lexicalization [Collins '99, Charniak '00]
Lexicalized Trees

- Add "headwords" to each phrasal node
- 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
  $$ VP(aw) \rightarrow YBD(aw) \ YD(her) \ NP(today) $$
- 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]
Grammar Refinement

- Lexicalization [Collins ‘99, Charniak ‘00]
- Structure Annotation [Johnson ‘98, Klein&Manning ‘03]

Conditional Independence?

- Not every NP expansion can fill every NP slot
- 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!
Grammar Refinement

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

Manual Annotation

- Manually split categories
  - NP: subject vs object
  - DT: determiners vs demonstratives
  - IN: sentential vs prepositional

  Advantages:
  - Fairly compact grammar
  - Linguistic motivations

  Disadvantages:
  - Performance leveled out
  - Manually annotated

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Treebank Grammar</td>
<td>72.6</td>
</tr>
<tr>
<td>Klein &amp; Manning '03</td>
<td>86.3</td>
</tr>
</tbody>
</table>

Automatic Annotation Induction

- Advantages:
  - Automatically learned:
    - Label all nodes with latent variables.
    - Same number $k$ of subcategories for all categories.

- Disadvantages:
  - Grammar gets too large
  - Most categories are oversplit while others are undersplit.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning '03</td>
<td>86.3</td>
</tr>
<tr>
<td>Matsuzaki et al. '05</td>
<td>86.7</td>
</tr>
</tbody>
</table>
Latent Variable Grammars

Parse Tree

Sentence $T$

Derivations $t : T$

Parameters $\theta$

Learning Latent Annotations

EM algorithm:
- Brackets are known
- Base categories are known
- Only induce subcategories

$S \rightarrow NP, VP$
$NP \rightarrow NP, VBP$
$VBP \rightarrow VBD, VP$
$PRP \rightarrow PRP$
$PRP \rightarrow PRP$

He was right.

Refinement of the DT tag

DT

the (0.50)
the (0.80)
this (0.39)
some (0.20)

a (0.11)
the (0.19)
this (0.9)
some (0.20)

DT-1

DT-2

DT-3

DT-4
Hierarchical refinement

Hierarchical Estimation Results

Refinement of the , tag

- Splitting all categories equally is wasteful:
Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful

Evaluate loss in likelihood from removing each split =
\[
\frac{\text{Data likelihood with split reversed}}{\text{Data likelihood with split}}
\]

No loss in accuracy when 50% of the splits are reversed.

Adaptive Splitting Results

<table>
<thead>
<tr>
<th>Total Number of grammar symbols</th>
<th>Parsing accuracy (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>74.0000</td>
</tr>
<tr>
<td>500</td>
<td>78.2500</td>
</tr>
<tr>
<td>900</td>
<td>82.5000</td>
</tr>
<tr>
<td>1300</td>
<td>86.7500</td>
</tr>
<tr>
<td>1700</td>
<td>91.0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>88.4</td>
</tr>
<tr>
<td>With 50% Merging</td>
<td>89.5</td>
</tr>
</tbody>
</table>
Learned Splits

- **Proper Nouns (NNP):**
  
  | NNP-12 | John | Robert | James |
  | NNP-2  | J.   | E.    | L.    |
  | NNP-1  | Bush | Noriega | Peters |
  | NNP-15 | New  | San   | Wall  |
  | NNP-3  | York | Francisco | Street |

- **Personal pronouns (PRP):**

  | PRP-0 | it | He | I |
  | PRP-1 | it | he | they |
  | PRP-2 | it | them | him |
Learned Splits

- Relative adverbs (RBR):
  - RBR-0: further, lower, higher
  - RBR-1: more, less, More
  - RBR-2: earlier, Earlier, later

- Cardinal Numbers (CD):
  - CD-7: one, two, Three
  - CD-11: million, billion, trillion
  - CD-0: 1, 50, 100
  - CD-3: 1, 30, 31
  - CD-9: 78, 58, 34

Final Results (Accuracy)

<table>
<thead>
<tr>
<th>Language</th>
<th>Charniak&amp;Johnson '05 (generative)</th>
<th>Split / Merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>≤40 words F1</td>
<td>all F1</td>
</tr>
<tr>
<td>ENS</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td>GER</td>
<td>Split / Merge</td>
<td>90.6</td>
</tr>
<tr>
<td>C</td>
<td>Dubey '05</td>
<td>80.8</td>
</tr>
<tr>
<td>CHN</td>
<td>Split / Merge</td>
<td>86.3</td>
</tr>
</tbody>
</table>

Still higher numbers from reranking / self-training methods