CSE 447/547
Natural Language Processing
Winter 2018

Parsing (Trees)

Yejin Choi - University of Washington

[Slides from Dan Klein, Michael Collins, Luke Zettlemoyer and Ray Mooney]
Ambiguities
I shot [an elephant] [in my pajamas]

Examples from J&M
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.*
**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
Probabilistic Context Free Grammars
Probabilistic Context-Free Grammars

- A context-free grammar is a tuple $<N, \Sigma, S, R>$
  - $N$ : the set of non-terminals
    - Phrasal categories: S, NP, VP, ADJP, etc.
    - Parts-of-speech (pre-terminals): NN, JJ, DT, VB, etc.
  - $\Sigma$ : 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 $X \rightarrow Y_1 Y_2 \ldots Y_n$, with $X \in N$, $n \geq 0$, $Y_i \in (N \cup \Sigma)$
    - Examples: $S \rightarrow NP \ VP$, $VP \rightarrow VP \ CC \ VP$
- A PCFG adds a distribution $q$:
  - Probability $q(r)$ for each $r \in R$, such that for all $X \in N$:
    $$\sum_{\alpha \rightarrow \beta \in R: \alpha = X} q(\alpha \rightarrow \beta) = 1$$
PCFG Example

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>⇒</td>
<td>NP</td>
<td>VP</td>
</tr>
<tr>
<td>VP</td>
<td>⇒</td>
<td>Vi</td>
<td></td>
</tr>
<tr>
<td>VP</td>
<td>⇒</td>
<td>Vt</td>
<td>NP</td>
</tr>
<tr>
<td>VP</td>
<td>⇒</td>
<td>VP</td>
<td>PP</td>
</tr>
<tr>
<td>NP</td>
<td>⇒</td>
<td>DT</td>
<td>NN</td>
</tr>
<tr>
<td>NP</td>
<td>⇒</td>
<td>NP</td>
<td>PP</td>
</tr>
<tr>
<td>PP</td>
<td>⇒</td>
<td>P</td>
<td>NP</td>
</tr>
<tr>
<td>Vi</td>
<td>⇒</td>
<td>sleeps</td>
<td></td>
</tr>
<tr>
<td>Vt</td>
<td>⇒</td>
<td>saw</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>⇒</td>
<td>man</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>⇒</td>
<td>woman</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>⇒</td>
<td>telescope</td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td>⇒</td>
<td>the</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>⇒</td>
<td>with</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>⇒</td>
<td>in</td>
<td></td>
</tr>
</tbody>
</table>

- Probability of a tree $t$ with rules $\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n$ is

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

where $q(\alpha \rightarrow \beta)$ is the probability for rule $\alpha \rightarrow \beta$. 
### PCFG Example

#### Probabilistic Context-Free Grammar (PCFG)

<table>
<thead>
<tr>
<th>Production</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S ⇒ NP VP</td>
<td>1.0</td>
</tr>
<tr>
<td>VP ⇒ Vi</td>
<td>0.4</td>
</tr>
<tr>
<td>VP ⇒ Vt NP</td>
<td>0.4</td>
</tr>
<tr>
<td>VP ⇒ VP PP</td>
<td>0.2</td>
</tr>
<tr>
<td>NP ⇒ DT NN</td>
<td>0.3</td>
</tr>
<tr>
<td>NP ⇒ NP PP</td>
<td>0.7</td>
</tr>
<tr>
<td>PP ⇒ P NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

#### Productions for Terminal Symbols

<table>
<thead>
<tr>
<th>Production</th>
<th>Symbol</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vi ⇒ sleeps</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>Vt ⇒ saw</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>NN ⇒ man</td>
<td></td>
<td>0.7</td>
</tr>
<tr>
<td>NN ⇒ woman</td>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td>NN ⇒ telescope</td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>DT ⇒ the</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>IN ⇒ with</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>IN ⇒ in</td>
<td></td>
<td>0.5</td>
</tr>
</tbody>
</table>

#### Example Trees

**Tree 1 (t₁):**
- S ⇒ NP VP
- NP ⇒ DT NN
- VP ⇒ Vi

**Example Sentence:** The man sleeps

**Probability:**
\[
p(t₁) = 1.0 \times 0.3 \times 1.0 \times 0.7 \times 0.4 \times 1.0
\]

**Tree 2 (t₂):**
- S ⇒ VP
- VP ⇒ NP
- NP ⇒ DT NN
- VP ⇒ Vt

**Example Sentence:** The man saw the woman with the telescope

**Probability:**
\[
p(t₂) = 1.8 \times 0.3 \times 1.0 \times 0.7 \times 0.2 \times 0.4 \times 1.0 \times 0.3 \times 1.0 \times 0.2 \times 0.4 \times 0.5 \times 0.3 \times 1.0 \times 0.1
\]
PCFGs: Learning and Inference

- **Model**
  - The probability of a tree $t$ with $n$ rules $\alpha_i \rightarrow \beta_i$, $i = 1..n$

  $p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$

- **Learning**
  - Read the rules off of labeled sentences, use ML estimates for probabilities

  $q_{ML}(\alpha \rightarrow \beta) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$

  and use all of our standard smoothing tricks!

- **Inference**
  - For input sentence $s$, define $T(s)$ to be the set of trees whole yield is $s$ (whole leaves, read left to right, match the words in $s$)

  $t^*(s) = \arg \max_{t \in T(s)} p(t)$
Dynamic Programming

- We will store: score of the max parse of \(x_i\) to \(x_j\) with root non-terminal \(X\)
  \[\pi(i, j, X)\]

- So we can compute the most likely parse:
  \[\pi(1, n, S) = \max_{t \in T_G(s)} p(t)\]

- Via the recursion:
  \[\pi(i, j, X) = \max_{X \rightarrow Y Z \in R, s \in \{i \ldots (j-1)\}} (q(X \rightarrow Y Z) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))\]

- With base case:
  \[\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}\]
The CKY Algorithm

- **Input:** a sentence \( s = x_1 \ldots x_n \) and a PCFG = \(<N, \Sigma, S, R, Q>\)
- **Initialization:** For \( i = 1 \ldots n \) and all \( X \) in \( N \)
  \[
  \pi(i, i, X) = \begin{cases} 
  q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\
  0 & \text{otherwise}
  \end{cases}
  \]
- For \( l = 1 \ldots (n-1) \)
  - For \( i = 1 \ldots (n-l) \) and \( j = i+l \)  
    - For all \( X \) in \( N \)
      \[
      \pi(i, j, X) = \max_{X \rightarrow YZ \in R, s \in \{i\ldots(j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
      \]
      also, store back pointers
      \[
      bp(i, j, X) = \arg\max_{X \rightarrow YZ \in R, s \in \{i\ldots(j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
      \]
- [iterate all phrase lengths]
- [iterate all phrases of length \( l \)]
- [iterate all non-terminals]
Probabilistic CKY Parser

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
0.01 0.004 0.006
S → Verb NP
S → VP PP
NP → I | he | she | me
0.1 0.02 0.02 0.06
NP → Houston | NWA
0.16 .04
Det→ the | a | an
0.6 0.1 0.05
NP → Det Nominal
0.6
Nominal → book | flight | meal | money
0.03 0.15 0.06 0.06
Nominal → Nominal Nominal
0.2
Nominal → Nominal PP
0.5
Verb→ book | include | prefer
0.5 0.04 0.06
VP → Verb NP
0.5
VP → VP PP
0.3
Prep → through | to | from
0.2 0.3 0.3
PP → Prep NP
1.0
Probabilistic CKY Parser

Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell.
Probabilistic CKY Parser

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S: .01,</td>
<td>S: .05*.5*.054</td>
<td>VP: .5*.5*.054</td>
<td>None</td>
<td>S: 00001296</td>
</tr>
<tr>
<td>Verb: .5</td>
<td>= .00135</td>
<td>= .0135</td>
<td></td>
<td>S: 0000216</td>
</tr>
<tr>
<td>Nominal: .03</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| VP: .5*.5*.054 | None | VP: .5*.5*.054 | None | S: 00001296 |
|               |      |                |      | S: 0000216 |

| NP: .6*.6*.15 | None | NP: .6*.6* .0024 | None | Nominal: .5*.15*.032 |
|               |      | = .054          |      | = .0024 |
| Det: .6      |      |                 |      |         |

| Nominal: .15 | None | Nominal: .5*.15*.032 | None | PP: 1.0*.2*.16 |
|              |      | = .032           |      | = .032 |

| Prep: .2     | PP: 1.0*.2*.16 | None |         |         |
|              |                |      |         |         |

| PP: 1.0*.2*.16 | None |         |         |         |
|                |      |         |         |         |

| NP: .16 | None |         |         |         |
|         |      |         |         |         |

Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell.
Memory

- How much memory does this require?
  - Have to store the score cache
  - Cache size: $|\text{symbols}| \times n^2$

- Pruning: Beam Search
  - $\text{score}[X][i][j]$ can get too large (when?)
  - Can keep beams (truncated maps $\text{score}[i][j]$) which only store the best $K$ 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 \((j - i) \leq n\)
  - For each \(i \leq n\)
    - For each rule \(X \rightarrow Y Z\)
      - For each split point \(k\)
        Do constant work

- Total time: \(|\text{rules}| \times n^3\)
- Something like 5 sec for an unoptimized parse of a 20-word sentences
Time: 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
Other Dynamic Programs

Can also compute other quantities:

- **Best Inside**: score of the max parse of $w_i$ to $w_j$ with root non-terminal $X$

- **Best Outside**: score of the max parse of $w_0$ to $w_n$ with a gap from $w_i$ to $w_j$ rooted with non-terminal $X$
  - see notes for derivation, it is a bit more complicated

- **Sum Inside/Outside**: Do sums instead of maxes
Why Chomsky Normal Form?

Inference:
- Can we keep N-ary (N > 2) rules and still do dynamic programming?
- Can we keep unary rules and still do dynamic programming?

Learning:
- Can we reconstruct the original trees?
Treebanks
The Penn Treebank: Size

- Penn WSJ Treebank = 50,000 sentences with associated trees
- Usual set-up: 40,000 training sentences, 2400 test sentences

An example tree:
### Table 1.2. The Penn Treebank syntactic tagset

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJP</td>
<td>Adjective phrase</td>
</tr>
<tr>
<td>ADVP</td>
<td>Adverb phrase</td>
</tr>
<tr>
<td>NP</td>
<td>Noun phrase</td>
</tr>
<tr>
<td>PP</td>
<td>Prepositional phrase</td>
</tr>
<tr>
<td>S</td>
<td>Simple declarative clause</td>
</tr>
<tr>
<td>SBAR</td>
<td>Subordinate clause</td>
</tr>
<tr>
<td>SBARQ</td>
<td>Direct question introduced by <em>wh</em>-element</td>
</tr>
<tr>
<td>SINV</td>
<td>Declarative sentence with subject-aux inversion</td>
</tr>
<tr>
<td>SQ</td>
<td>Yes/no questions and subconstituent of SBARQ excluding <em>wh</em>-element</td>
</tr>
<tr>
<td>VP</td>
<td>Verb phrase</td>
</tr>
<tr>
<td>WHADVP</td>
<td>Wh-adverb phrase</td>
</tr>
<tr>
<td>WHNP</td>
<td>Wh-noun phrase</td>
</tr>
<tr>
<td>WHPP</td>
<td>Wh-prepositional phrase</td>
</tr>
<tr>
<td>X</td>
<td>Constituent of unknown or uncertain category</td>
</tr>
<tr>
<td>*</td>
<td>“Understood” subject of infinitive or imperative</td>
</tr>
<tr>
<td>0</td>
<td>Zero variant of <em>that</em> in subordinate clauses</td>
</tr>
<tr>
<td>T</td>
<td>Trace of <em>wh</em>-Constituent</td>
</tr>
</tbody>
</table>
Treebank Grammars

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

```
ROOT → S 1
S → NP VP . 1
NP → PRP 1
VP → VBD ADJP 1
... 1
```

- 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

NP:
Grammar encodings: Non-black states are active, non-white states are accepting, and bold transitions are phrasal. FSAs for a subset of the rules for the category NP.
Typical Experimental Setup

- Corpus: Penn Treebank, WSJ

  ![Training](02-21)
  ![Development](section 22 (here, first 20 files))
  ![Test](section 23)

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

- Here: also size – number of symbols in grammar.
  - Passive / complete symbols: NP, NP^S
  - Active / incomplete symbols: NP → NP CC •
How to Evaluate?

**Correct Tree T**

```
S
  VP
    Verb book
    NP Det the Nominal
      Noun flight
      Prep through
      NP Houston
```

**Computed Tree P**

```
S
  VP
    Verb book
    NP Det the Nominal
      Noun flight
      Prep through
      NP Proper-Noun Houston
```
PARSEVAL Example

Correct Tree T

```
S
  VP
    Verb
    NP
      Det
      Nominal
        Noun
        Prep
        NP
          flight
          through
          Houston
```

# Constituents: 11
# Correct Constituents: 10
Recall = 10/11 = 90.9%

Computed Tree P

```
S
  VP
    Verb
    NP
      Det
      Nominal
        Noun
        Prep
        NP
          flight
          through
          Houston
          Proper-Noun
```

# Constituents: 12

Precision = 10/12 = 83.3%

$F_1 = 87.4\%$
Evaluation Metric

- **PARSEVAL** metrics measure the fraction of the constituents that match between the computed and human parse trees. If $P$ is the system’s parse tree and $T$ is the human parse tree (the “gold standard”):
  - Recall = (# correct constituents in $P$) / (# constituents in $T$)
  - Precision = (# correct constituents in $P$) / (# constituents in $P$)

- Labeled Precision and labeled recall require getting the non-terminal label on the constituent node correct to count as correct.

- $F_1$ is the harmonic mean of precision and recall.
  - $F_1 = (2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$
Performance with Vanilla PCFGs

- Use PCFGs for broad coverage parsing
- Take the grammar right off the trees

**Model** | **F1**
--- | ---
Baseline | 72.0
Grammar Refinements

1. Markovization
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!
Vertical Markovization

- Vertical Markov order: rewrites depend on past $k$ ancestor nodes. (cf. parent annotation)
Horizontal Markovization

Order 1

Order $\infty$

Symbols

Horizontal Markov Orde

Horizontal Markov Order
Vertical and Horizontal

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=h=2v</td>
<td>77.8</td>
<td>7.5K</td>
</tr>
</tbody>
</table>
### Unlexicalized PCFG Grammar Size

<table>
<thead>
<tr>
<th>Vertical Order</th>
<th>$h = 0$</th>
<th>$h = 1$</th>
<th>$h \leq 2$</th>
<th>$h = 2$</th>
<th>$h = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v = 1$ No annotation</td>
<td>71.27 (854)</td>
<td>72.5 (3119)</td>
<td>73.46 (3863)</td>
<td>72.96 (6207)</td>
<td>72.62 (9657)</td>
</tr>
<tr>
<td>$v \leq 2$ Sel. Parents</td>
<td>74.75 (2285)</td>
<td>77.42 (6564)</td>
<td>77.77 (7619)</td>
<td>77.50 (11398)</td>
<td>76.91 (14247)</td>
</tr>
<tr>
<td>$v = 2$ All Parents</td>
<td>74.68 (2984)</td>
<td>77.42 (7312)</td>
<td>77.81 (8367)</td>
<td>77.50 (12132)</td>
<td>76.81 (14666)</td>
</tr>
<tr>
<td>$v \leq 3$ Sel. GParents</td>
<td>76.50 (4943)</td>
<td>78.59 (12374)</td>
<td>79.07 (13627)</td>
<td>78.97 (19545)</td>
<td>78.54 (20123)</td>
</tr>
<tr>
<td>$v = 3$ All GParents</td>
<td>76.74 (7797)</td>
<td>79.18 (15740)</td>
<td>79.74 (16994)</td>
<td>79.07 (22886)</td>
<td>78.72 (22002)</td>
</tr>
</tbody>
</table>

**Figure 2:** Markovizations: $F_1$ and grammar size.
Grammar Refinements

2. Lexicalization
Problems with PCFGs

- These trees differ only in one rule:
  - VP $\rightarrow$ VP PP
  - NP $\rightarrow$ NP PP
- Lexicalization allows us to be sensitive to specific words
Add "headwords" to each phrasal node

- Headship not in (most) treebanks
- Usually use (handwritten) 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

Lexicalize Trees!
Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like
  
  $$\text{VP(saw)} \rightarrow \text{VBD(saw)} \text{ NP-C(her) NP(today)}$$

- Never going to get these atomically off of a treebank

- Solution: break up derivation into smaller steps
Lexical Derivation Steps

Main idea: define a linguistically-motivated Markov process for generating children given the parent

Step 1: Choose a head tag and word

Step 2: Choose a complement bag

Step 3: Generate children (incl. adjuncts)

Step 4: Recursively derive children

[Collins 99]
Lexicalized CKY

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

still cubic time?
Pruning with Beams

- The Collins parser prunes with per-cell beams [Collins 99]
  - Essentially, run the $O(n^5)$ CKY
  - 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)

<table>
<thead>
<tr>
<th>Model</th>
<th>$F1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Treebank Grammar</td>
<td>72.6</td>
</tr>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.6</td>
</tr>
</tbody>
</table>
Grammar Refinements

3. Using Latent Sub-categories
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
Learning Latent Annotations

Latent Annotations:
- Brackets are known
- Base categories are known
- Hidden variables for subcategories

He was right.

Can learn with EM: like Forward-Backward for HMMs.
Final Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>F1 ≤ 40 words</th>
<th>F1 all words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
<td>85.7</td>
</tr>
<tr>
<td>Matsuzaki et al. ’05</td>
<td>86.7</td>
<td>86.1</td>
</tr>
<tr>
<td>Collins ’99</td>
<td>88.6</td>
<td>88.2</td>
</tr>
<tr>
<td>Charniak &amp; Johnson ’05</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td>Petrov et. al. 06</td>
<td>90.2</td>
<td>89.7</td>
</tr>
</tbody>
</table>
Grammar as Foreign Language” (deep learning)

John has a dog ➔

$\text{S} \quad \text{NP} \quad \text{VP} \quad .
\text{NNP} \quad \text{VBZ} \quad \text{NP} \quad \text{DT} \quad \text{NN}$

John has a dog ➔

$(\text{S} \ (\text{NP} \ \text{NNP})_{\text{NP}} \ (\text{VP} \ \text{VBZ} \ (\text{NP} \ \text{DT} \ \text{NN})_{\text{NP}})_{\text{VP}} \ .)_{\text{S}}$

- Linearize a tree into a sequence
- Then parsing problem becomes similar to machine translation
  - Input: sequence
  - Output: sequence (of different length)
- Encoder-decoder LSTMs (Long short-term memory networks)
"Grammar as Foreign Language" (deep learning)

Vinyals et al., 2015

John has a dog ➔

(S (NP NNP) NP (VP VBZ (NP DT NN) NP) VP . ) S

- Penn treebank (~40K sentences) is too small to train LSTMs
- Create a larger training set with 11M sentences automatically parsed by two state-of-the-art parsers (and keep only those sentences for which two parsers agreed)
"Grammar as Foreign Language" (deep learning)

Vinyals et al., 2015

<table>
<thead>
<tr>
<th>Parser</th>
<th>Training Set</th>
<th>WSJ 22</th>
<th>WSJ 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline LSTM+D</td>
<td>WSJ only</td>
<td>&lt; 70</td>
<td>&lt; 70</td>
</tr>
<tr>
<td>LSTM+A+D</td>
<td>WSJ only</td>
<td>88.7</td>
<td>88.3</td>
</tr>
<tr>
<td>LSTM+A+D ensemble</td>
<td>WSJ only</td>
<td>90.7</td>
<td>90.5</td>
</tr>
<tr>
<td>baseline LSTM</td>
<td>BerkeleyParser corpus</td>
<td>91.0</td>
<td>90.5</td>
</tr>
<tr>
<td>LSTM+A</td>
<td>high-confidence corpus</td>
<td>93.3</td>
<td>92.5</td>
</tr>
<tr>
<td>LSTM+A ensemble</td>
<td>high-confidence corpus</td>
<td>93.5</td>
<td>92.8</td>
</tr>
<tr>
<td>Petrov et al. (2006) [12]</td>
<td>WSJ only</td>
<td>91.1</td>
<td>90.4</td>
</tr>
<tr>
<td>Zhu et al. (2013) [13]</td>
<td>WSJ only</td>
<td>N/A</td>
<td>90.4</td>
</tr>
<tr>
<td>Petrov et al. (2010) ensemble [14]</td>
<td>WSJ only</td>
<td>92.5</td>
<td>91.8</td>
</tr>
<tr>
<td>Zhu et al. (2013) [13]</td>
<td>semi-supervised</td>
<td>N/A</td>
<td>91.3</td>
</tr>
<tr>
<td>McClosky et al. (2006) [16]</td>
<td>semi-supervised</td>
<td>92.4</td>
<td>92.1</td>
</tr>
</tbody>
</table>
Supplementary Topics

I. CNF Conversion
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!
### Original Grammar

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → Aux NP VP</td>
<td>0.1</td>
</tr>
<tr>
<td>S → VP</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP → Pronoun</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → Proper-Noun</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → Det Nominal</td>
<td>0.6</td>
</tr>
<tr>
<td>Nominal → Noun</td>
<td>0.3</td>
</tr>
<tr>
<td>Nominal → Nominal Noun</td>
<td>0.2</td>
</tr>
<tr>
<td>Nominal → Nominal PP</td>
<td>0.5</td>
</tr>
<tr>
<td>VP → Verb</td>
<td>0.2</td>
</tr>
<tr>
<td>VP → Verb NP</td>
<td>0.5</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>0.3</td>
</tr>
<tr>
<td>PP → Prep NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

### Lexicon:

<table>
<thead>
<tr>
<th>Part of Speech</th>
<th>Words</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>book</td>
<td>flight</td>
</tr>
<tr>
<td>Verb</td>
<td>book</td>
<td>include</td>
</tr>
<tr>
<td>Det</td>
<td>the</td>
<td>a</td>
</tr>
<tr>
<td>Pronoun</td>
<td>I</td>
<td>he</td>
</tr>
<tr>
<td>Proper-Noun</td>
<td>Houston</td>
<td>NWA</td>
</tr>
<tr>
<td>Aux</td>
<td>does</td>
<td>1.0</td>
</tr>
<tr>
<td>Prep</td>
<td>from</td>
<td>to</td>
</tr>
</tbody>
</table>
### Original Grammar

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → Aux NP VP</td>
<td>0.1</td>
</tr>
<tr>
<td>S → VP</td>
<td>0.1</td>
</tr>
<tr>
<td>NP → Pronoun</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → Proper-Noun</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → Det Nominal</td>
<td>0.6</td>
</tr>
<tr>
<td>Nominal → Noun</td>
<td>0.3</td>
</tr>
<tr>
<td>Nominal → Nominal Noun</td>
<td>0.2</td>
</tr>
<tr>
<td>Nominal → Nominal PP</td>
<td>0.5</td>
</tr>
<tr>
<td>VP → Verb</td>
<td>0.2</td>
</tr>
<tr>
<td>VP → Verb NP</td>
<td>0.5</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>0.3</td>
</tr>
<tr>
<td>PP → Prep NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

### Lexicon (See previous slide for full list):
- **Noun** → book | flight | meal | money
  - 0.1 0.5 0.2 0.2
- **Verb** → book | include | prefer
  - 0.5 0.2 0.3

### Chomsky Normal Form

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → X1 VP</td>
<td>0.1</td>
</tr>
<tr>
<td>X1 → Aux NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>
### Original Grammar

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → Aux NP VP</td>
<td>0.1</td>
</tr>
<tr>
<td>S → VP</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### Chomsky Normal Form

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → X1 VP</td>
<td>0.1</td>
</tr>
<tr>
<td>X1 → Aux NP</td>
<td>1.0</td>
</tr>
<tr>
<td>S → book</td>
<td>include</td>
</tr>
<tr>
<td>S → Verb NP</td>
<td></td>
</tr>
<tr>
<td>S → VP PP</td>
<td></td>
</tr>
</tbody>
</table>

### Lexicon (See previous slide for full list):

**Noun**
- book | flight | meal | money
  - 0.1 0.5 0.2 0.2

**Verb**
- book | include | prefer
  - 0.5 0.2 0.3
## Original Grammar

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \rightarrow NP \ VP )</td>
<td>0.8</td>
<td>( S \rightarrow NP \ VP )</td>
<td>0.8</td>
</tr>
<tr>
<td>( S \rightarrow Aux \ NP \ VP )</td>
<td>0.1</td>
<td>( S \rightarrow X1 \ VP )</td>
<td>0.1</td>
</tr>
<tr>
<td>( X1 \rightarrow Aux \ NP )</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( S \rightarrow VP )</td>
<td>0.1</td>
<td>( S \rightarrow book \mid include \mid prefer )</td>
<td>0.01 0.004 0.006</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( S \rightarrow Verb \ NP )</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( S \rightarrow VP \ PP )</td>
<td>0.03</td>
</tr>
<tr>
<td>( NP \rightarrow Pronoun )</td>
<td>0.2</td>
<td>( NP \rightarrow I \mid he \mid she \mid me )</td>
<td>0.1 0.02 0.02 0.06</td>
</tr>
<tr>
<td>( NP \rightarrow Proper-Noun )</td>
<td>0.2</td>
<td>( NP \rightarrow Houston \mid NWA )</td>
<td>0.16 0.04</td>
</tr>
<tr>
<td>( NP \rightarrow Det Nominal )</td>
<td>0.6</td>
<td>( NP \rightarrow Det Nominal )</td>
<td>0.6</td>
</tr>
<tr>
<td>( Nominal \rightarrow Noun )</td>
<td>0.3</td>
<td>( Nominal \rightarrow book \mid flight \mid meal \mid money )</td>
<td>0.03 0.15 0.06 0.06</td>
</tr>
<tr>
<td>( Nominal \rightarrow Nominal Noun )</td>
<td>0.2</td>
<td>( Nominal \rightarrow Nominal Noun )</td>
<td>0.2</td>
</tr>
<tr>
<td>( Nominal \rightarrow Nominal PP )</td>
<td>0.5</td>
<td>( Nominal \rightarrow Nominal PP )</td>
<td>0.5</td>
</tr>
<tr>
<td>( VP \rightarrow Verb )</td>
<td>0.2</td>
<td>( VP \rightarrow book \mid include \mid prefer )</td>
<td>0.1 0.04 0.06</td>
</tr>
<tr>
<td>( VP \rightarrow Verb NP )</td>
<td>0.5</td>
<td>( VP \rightarrow Verb NP )</td>
<td>0.5</td>
</tr>
<tr>
<td>( VP \rightarrow VP \ PP )</td>
<td>0.3</td>
<td>( VP \rightarrow VP \ PP )</td>
<td>0.3</td>
</tr>
<tr>
<td>( PP \rightarrow Prep \ NP )</td>
<td>1.0</td>
<td>( PP \rightarrow Prep \ NP )</td>
<td>1.0</td>
</tr>
</tbody>
</table>

## Chomsky Normal Form

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \rightarrow NP \ VP )</td>
<td>0.8</td>
<td>( S \rightarrow NP \ VP )</td>
<td>0.8</td>
</tr>
<tr>
<td>( S \rightarrow X1 \ VP )</td>
<td>0.1</td>
<td>( X1 \rightarrow Aux \ NP )</td>
<td>1.0</td>
</tr>
<tr>
<td>( S \rightarrow book \mid include \mid prefer )</td>
<td>0.01 0.004 0.006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( S \rightarrow Verb \ NP )</td>
<td>0.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( S \rightarrow VP \ PP )</td>
<td>0.03</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Lexicon (See previous slide for full list):

<table>
<thead>
<tr>
<th>Category</th>
<th>Items</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>book</td>
<td>0.1 0.5 0.2 0.2</td>
</tr>
<tr>
<td>Verb</td>
<td>book</td>
<td>0.5 0.2 0.3</td>
</tr>
</tbody>
</table>
Advanced Topics

I. CKY with Unary Rules
CNF + Unary Closure

We need unaries to be non-cyclic

- Calculate **closure** $\text{Close}(R)$ for unary rules in $R$
  - Add $X \rightarrow Y$ if there exists a rule chain $X \rightarrow Z_1, Z_1 \rightarrow Z_2, \ldots, Z_k \rightarrow Y$ with $q(X \rightarrow Y) = q(X \rightarrow Z_1) * q(Z_1 \rightarrow Z_2) * \ldots * q(Z_k \rightarrow Y)$
  - If no unary rule exist for $X$, add $X \rightarrow X$ with $q(X \rightarrow X) = 1$ for all $X$ in $N$

- Rather than zero or more unaries, always exactly one
- Alternate unary and binary layers
- What about $X \rightarrow Y$ with different unary paths (and scores)?

**WARNING:** Watch out for unary cycles!
The CKY Algorithm

- **Input:** a sentence \( s = x_1 \ldots x_n \) and a PCFG = \(<N, \Sigma, S, R, q>\)
- **Initialization:** For \( i = 1 \ldots n \) and all \( X \) in \( N \)
  \[
  \pi(i, i, X) = \begin{cases} 
  q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\
  0 & \text{otherwise} 
  \end{cases}
  \]
- For \( l = 1 \ldots (n-1) \)
  - For \( i = 1 \ldots (n-l) \) and \( j = i+l \) [iterate all phrase lengths]
    - For all \( X \) in \( N \) [iterate all non-terminals]
      \[
      \pi(i, j, X) = \max_{X \rightarrow YZ \in R, s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
      \]
- also, store back pointers
  \[
  bp(i, j, X) = \arg \max_{X \rightarrow YZ \in R, s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
  \]
CKY with Unary Closure

- **Input:** a sentence \( s = x_1 \ldots x_n \) and a PCFG = \( <N, \Sigma, S, R, q> \)

- **Initialization:** For \( i = 1 \ldots n \):
  - Step 1: for all \( X \) in \( N \):
    \[
    \pi(i, i, X) = \begin{cases} 
    q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\
    0 & \text{otherwise}
    \end{cases}
    \]
  - Step 2: for all \( X \) in \( N \):
    \[
    \pi_U(i, i, X) = \max_{X \rightarrow Y \in \text{Close}(R)} (q(X \rightarrow Y) \times \pi(i, i, Y))
    \]

- For \( l = 1 \ldots (n-1) \) [iterate all phrase lengths]
  - For \( i = 1 \ldots (n-l) \) and \( j = i+l \) [iterate all phrases of length \( l \)]
    - Step 1: (Binary)
      - For all \( X \) in \( N \) [iterate all non-terminals]
        \[
        \pi_B(i, j, X) = \max_{X \rightarrow YZ \in R, s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi_U(i, s, Y) \times \pi_U(s+1, j, Z))
        \]
    - Step 2: (Unary)
      - For all \( X \) in \( N \) [iterate all non-terminals]
        \[
        \pi_U(i, j, X) = \max_{X \rightarrow Y \in \text{Close}(R)} (q(X \rightarrow Y) \times \pi_B(i, j, Y))
        \]
Advanced Topics

2. Grammar Refinements: Tag Splits
Tag Splits

- Problem: Treebank tags are too coarse.

- Example: Sentential, PP, and other prepositions are all marked IN.

- Partial Solution:
  - Subdivide the IN tag.

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=h=2v</td>
<td>78.3</td>
<td>8.0K</td>
</tr>
<tr>
<td>SPLIT-IN</td>
<td>80.3</td>
<td>8.1K</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>80.4</td>
<td>8.1K</td>
</tr>
<tr>
<td>80.5</td>
<td>8.1K</td>
</tr>
<tr>
<td>81.2</td>
<td>8.5K</td>
</tr>
<tr>
<td>81.6</td>
<td>9.0K</td>
</tr>
<tr>
<td>81.7</td>
<td>9.1K</td>
</tr>
<tr>
<td>81.8</td>
<td>9.3K</td>
</tr>
</tbody>
</table>
A Fully Annotated (Unlex) Tree
Some Test Set Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>LP</th>
<th>LR</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magerman 95</td>
<td>84.9</td>
<td>84.6</td>
<td><strong>84.7</strong></td>
</tr>
<tr>
<td>Collins 96</td>
<td>86.3</td>
<td>85.8</td>
<td><strong>86.0</strong></td>
</tr>
<tr>
<td>Unlexicalized</td>
<td>86.9</td>
<td>85.7</td>
<td><strong>86.3</strong></td>
</tr>
<tr>
<td>Charniak 97</td>
<td>87.4</td>
<td>87.5</td>
<td><strong>87.4</strong></td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.7</td>
<td>88.6</td>
<td><strong>88.6</strong></td>
</tr>
</tbody>
</table>

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