CSE 517
Natural Language Processing
Winter 2019

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]
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
  \textit{impractical design requirements}
  \textit{plastic cup holder}

- Multiple gap constructions
  \textit{The chicken is ready to eat.}
  \textit{The contractors are rich enough to sue.}

- Coordination scope:
  \textit{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>NP</td>
<td>VP</td>
<td>1.0</td>
</tr>
<tr>
<td>VP</td>
<td>Vi</td>
<td></td>
<td>0.4</td>
</tr>
<tr>
<td>VP</td>
<td>Vt</td>
<td>NP</td>
<td>0.4</td>
</tr>
<tr>
<td>VP</td>
<td>VP</td>
<td>PP</td>
<td>0.2</td>
</tr>
<tr>
<td>NP</td>
<td>DT</td>
<td>NN</td>
<td>0.3</td>
</tr>
<tr>
<td>NP</td>
<td>NP</td>
<td>PP</td>
<td>0.7</td>
</tr>
<tr>
<td>PP</td>
<td>P</td>
<td>NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

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

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

- **Vi** ⇒ sleeps: 1.0
- **Vt** ⇒ saw: 1.0
- **NN** ⇒ man: 0.7
- **NN** ⇒ woman: 0.2
- **NN** ⇒ telescope: 0.1
- **DT** ⇒ the: 1.0
- **IN** ⇒ with: 0.5
- **IN** ⇒ in: 0.5

The man sleeps

\[
p(t_1) = 1.0 \times 0.3 \times 1.0 \times 0.7 \times 0.4 \times 1.0
\]

The man saw the woman with the telescope

\[
p(t_2) = 1.8 \times 0.3 \times 1.0 \times 0.7 \times 0.2 \times 0.4 \times 1.0 \times 0.3 \times 1.0 \times 0.2 \times 0.4 \times 0.5 \times 0.3 \times 1.0 \times 0.1
\]
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 whose 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 $= \langle N, \Sigma, S, R, q \rangle$
- **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 $N$

\[
\pi(i, j, X) = \max_{X \rightarrow YZ \in R, s \in \{i \ldots (j-1)\}} \left( q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z) \right)
\]

- also, store back pointers

\[
bp(i, j, X) = \arg \max_{X \rightarrow YZ \in R, s \in \{i \ldots (j-1)\}} \left( q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z) \right)
\]
Probabilistic CKY Parser

S \rightarrow NP \ VP \ 0.8
S \rightarrow X1 \ VP \ 0.1
X1 \rightarrow Aux \ NP \ 1.0
S \rightarrow book \mid include \mid prefer \ 0.01 \ 0.004 \ 0.006
S \rightarrow Verb \ NP \ 0.05
S \rightarrow VP PP \ 0.03
NP \rightarrow I \mid he \mid she \mid me \ 0.1 \ 0.02 \ 0.02 \ 0.06
NP \rightarrow Houston \mid NWA \ 0.16 \ 0.04
Det \rightarrow the \mid a \mid an \ 0.6 \ 0.1 \ 0.05
NP \rightarrow Det \ Nominal \ 0.6
Nominal \rightarrow book \mid flight \mid meal \mid money \ 0.03 \ 0.15 \ 0.06 \ 0.06
Nominal \rightarrow Nominal \ Nominal \ 0.2
Nominal \rightarrow Nominal \ PP \ 0.5
Verb \rightarrow book \mid include \mid prefer \ 0.5 \ 0.04 \ 0.06
VP \rightarrow Verb \ NP \ 0.5
VP \rightarrow VP PP \ 0.3
Prep \rightarrow through \mid to \mid from \ 0.2 \ 0.3 \ 0.3
PP \rightarrow Prep \ NP \ 1.0
Probabilistic CKY Parser

Parse Tree #1

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

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
  - score[$X$][$i$][$j$] can get too large (when?)
  - Can keep beams (truncated maps 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:

Canadian Utilities had 1988 revenue of C$ 1.16 billion, mainly from its natural gas and electric utility businesses in Alberta, where the company serves about 800,000 customers.
## Penn Treebank Non-terminals

<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>

*Table 1.2. The Penn Treebank syntactic tagset*
Treebank Grammars

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

```
ROOT
  |  
  S
  |  
  NP
  |  
  VBD ADJP
  |  
  PRP
  |  
  He
  |  
  was
  |  
  JJ
  |  
  right
```

```
ROOT \rightarrow S  
S \rightarrow NP VP .  
NP \rightarrow PRP  
VP \rightarrow VBD ADJP  
.....
```

- 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: sections 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 •
Correct Tree T

`S`

`VP`

`Verb`

`book`

`NP`

`Det`

`the`

`Nominal`

`flight`

`Prep`

`through`

`NP`

`Houston`

Computed Tree P

`S`

`VP`

`Verb`

`book`

`NP`

`Det`

`the`

`Nominal`

`flight`

`Prep`

`through`

`NP`

`Proper-Noun`

`Houston`
PARSEVAL Example

Correct Tree T

Correct Tree T

Computed Tree P

Computed Tree P

# Constituents: 11

# Constituents: 12

# Correct Constituents: 10

Recall = 10/11 = 90.9%

Precision = 10/12 = 83.3%

F_1 = 87.4%
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 = \( \frac{\text{# correct constituents in } P}{\text{# constituents in } T} \)
- Precision = \( \frac{\text{# correct constituents in } P}{\text{# constituents in } P} \)

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

- \( F1 \) is the harmonic mean of precision and recall.
  \[ F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \]
Performance with Vanilla PCFGs

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

Model | F1
--- | ---
Baseline | 72.0

[Charniak 96]
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 $\infty$

Horizontal Markov Orde

Symbols

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>Horizontal Markov Order</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$h = 0$</td>
</tr>
<tr>
<td>$v = 1$ No annotation</td>
<td>71.27</td>
</tr>
<tr>
<td></td>
<td>(854)</td>
</tr>
<tr>
<td>$v \leq 2$ Sel. Parents</td>
<td>74.75</td>
</tr>
<tr>
<td></td>
<td>(2285)</td>
</tr>
<tr>
<td>$v = 2$ All Parents</td>
<td>74.68</td>
</tr>
<tr>
<td></td>
<td>(2984)</td>
</tr>
<tr>
<td>$v \leq 3$ Sel. GParents</td>
<td>76.50</td>
</tr>
<tr>
<td></td>
<td>(4943)</td>
</tr>
<tr>
<td>$v = 3$ All GParents</td>
<td>76.74</td>
</tr>
<tr>
<td></td>
<td>(7797)</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.
Lexicalize Trees!

- 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
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

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

  1. **Step 1:** Choose a head tag and word
  2. **Step 2:** Choose a complement bag
  3. **Step 3:** Generate children (incl. adjuncts)
  4. **Step 4:** Recursively derive children

[Collins 99]
bestScore(i,j,X, h)
if (j = i)
    return tagScore(X,s[i])
else
    return 
        max max score(X[h]->Y[h] Z[h']) * 
            k,h', X->YZ bestScore(i,k,Y, h) * 
                bestScore(k+1,j,Z, h')  
        max score(X[h]->Y[h'] Z[h]) * 
            k,h', X->YZ bestScore(i,k,Y, h') * 
                bestScore(k+1,j,Z, h)
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><strong>90.2</strong></td>
<td><strong>89.7</strong></td>
</tr>
</tbody>
</table>
“Grammar as Foreign Language” (deep learning)  

Vinyals et al., 2015

John has a dog ➔  

(S (NP NNP )<sub>NP</sub> (VP VBZ (NP DT NN )<sub>NP</sub>)<sub>VP</sub>)<sub>S</sub>

- 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 ➞

\[
S \rightarrow NP \rightarrow S \\
/ \ \\ \\
NNP \rightarrow VP \rightarrow NP \\
/ \ \ \ \ \ \ \\
VBZ \rightarrow DT \rightarrow NN
\]

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:
- All rules of the form $X \rightarrow Y Z$ 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

S → NP VP 0.8
S → Aux NP VP 0.1
S → VP 0.1

NP → Pronoun 0.2
NP → Proper-Noun 0.2
NP → Det Nominal 0.6
Nominal → Noun 0.3
Nominal → Nominal Noun 0.2
Nominal → Nominal PP 0.5
VP → Verb 0.2
VP → Verb NP 0.5
VP → VP PP 0.3
PP → Prep NP 1.0

Det → the | a | that | this
      0.6 0.2  0.1  0.1
Pronoun → I | he | she | me
        0.5  0.1 0.1  0.3
Proper-Noun → Houston | NWA
            0.8  0.2
Aux → does
      1.0
Prep → from | to | on | near | through
      0.25 0.25  0.1 0.2  0.2

Lexicon:
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>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>0.8</td>
</tr>
<tr>
<td>$S \rightarrow Aux \ NP \ VP$</td>
<td>0.1</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>0.1</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>0.2</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>0.2</td>
</tr>
<tr>
<td>$NP \rightarrow Det \ Nominal$</td>
<td>0.6</td>
</tr>
<tr>
<td>$Nominal \rightarrow Noun$</td>
<td>0.3</td>
</tr>
<tr>
<td>$Nominal \rightarrow Nominal \ Noun$</td>
<td>0.2</td>
</tr>
<tr>
<td>$Nominal \rightarrow Nominal \ PP$</td>
<td>0.5</td>
</tr>
<tr>
<td>$VP \rightarrow Verb$</td>
<td>0.2</td>
</tr>
<tr>
<td>$VP \rightarrow Verb \ NP$</td>
<td>0.5</td>
</tr>
<tr>
<td>$VP \rightarrow VP \ PP$</td>
<td>0.3</td>
</tr>
<tr>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>0.8</td>
</tr>
<tr>
<td>$S \rightarrow X1 \ VP$</td>
<td>0.1</td>
</tr>
<tr>
<td>$X1 \rightarrow Aux \ NP$</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
## Original Grammar

<table>
<thead>
<tr>
<th>Production</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>Production</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>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Productions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>book</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>Verb</td>
<td>book</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
</tr>
</tbody>
</table>
## Original Grammar

<table>
<thead>
<tr>
<th>Grammar 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>Grammar 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>

## Chomsky Normal Form

<table>
<thead>
<tr>
<th>Grammar 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>0.05</td>
</tr>
<tr>
<td>S → VP PP</td>
<td>0.03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grammar Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP → Houston</td>
<td>NWA</td>
</tr>
<tr>
<td>VP → book</td>
<td>include</td>
</tr>
<tr>
<td>Nominal → book</td>
<td>flight</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 → book</td>
<td>include</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):

<table>
<thead>
<tr>
<th>Noun</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>0.1</td>
</tr>
<tr>
<td>flight</td>
<td>0.5</td>
</tr>
<tr>
<td>meal</td>
<td>0.2</td>
</tr>
<tr>
<td>money</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verb</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>0.5</td>
</tr>
<tr>
<td>include</td>
<td>0.2</td>
</tr>
<tr>
<td>prefer</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Advanced Topics

I. CKY with Unary Rules
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 \( q(X \rightarrow Y) = q(X \rightarrow Z_1) \times q(Z_1 \rightarrow Z_2) \times \ldots \times q(Z_k \rightarrow Y) \)
  - 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 \( l = 1 \ldots (n-1) \) [iterate all phrase lengths]
  - For \( i = 1 \ldots (n-l) \) and \( j = i+l \) [iterate all phrases of length \( l \)]
    - 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 $N$: 
    $$\pi(i, i, X) = \begin{cases} q(X \to x_i) & \text{if } X \to x_i \in R \\ 0 & \text{otherwise} \end{cases}$$
  - Step 2: for all $X$ in $N$: 
    $$\pi_U(i, i, X) = \max_{X \to Y \in \text{Close}(R)} (q(X \to Y) \times \pi(i, i, Y))$$
- For $l = 1 \ldots (n-1)$
  - For $i = 1 \ldots (n-l)$ and $j = i+l$
    - Step 1: (Binary)
      - For all $X$ in $N$ 
        $$\pi_B(i, j, X) = \max_{X \to YZ \in R, s \in \{i \ldots (j-1)} (q(X \to YZ) \times \pi_U(i, s, Y) \times \pi_U(s + 1, j, Z))$$
    - Step 2: (Unary)
      - For all $X$ in $N$ 
        $$\pi_U(i, j, X) = \max_{X \to Y \in \text{Close}(R)} (q(X \to Y) \times \pi_B(i, j, Y))$$
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\(^\text{VP}\))
- **SPLIT-AUX**: mark auxiliary verbs with \(-\text{AUX}\) [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>84.7</td>
</tr>
<tr>
<td>Collins 96</td>
<td>86.3</td>
<td>85.8</td>
<td>86.0</td>
</tr>
<tr>
<td>Unlexicalized</td>
<td>86.9</td>
<td>85.7</td>
<td>86.3</td>
</tr>
<tr>
<td>Charniak 97</td>
<td>87.4</td>
<td>87.5</td>
<td>87.4</td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.7</td>
<td>88.6</td>
<td>88.6</td>
</tr>
</tbody>
</table>

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