Syntax and Parsing I

Constituency Parsing

Slav Petrov – Google

Thanks to:
 Dan Klein, Ryan McDonald, Alexander Rush, Joakim Nivre, Greg Durrett, David Weiss

Lisbon Machine Learning School 2015
Notes for 2016

• Add BerkeleyParser results
They solved the problem with statistics.
They solved the problem with statistics.
They solved the problem with statistics.
They solved the problem with statistics.
Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new savings-and-loan bailout agency can raise capital, creating another potential obstacle to the government's sale of sick thrifts.
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...
- First part of today’s lecture

*new art critics write reviews with computers*
Dependency Parsing

- Directed edges between pairs of word (head, dependent)
- Can handle free word-order languages
- Very efficient decoding algorithms exist
- Second part of today’s lecture
Classical NLP: Parsing

- Write symbolic or logical rules:
  - Minimal grammar on “Fed raises” sentence: 36 parses
  - Real-size grammar: many millions of parses
  - This scaled very badly, didn’t yield broad-coverage tools

- Use deduction systems to prove parses from words

Fed raises interest rates 0.5 percent

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

• I cleaned the dishes from dinner
• I cleaned the dishes with detergent
• I cleaned the dishes in my pajamas
• I cleaned the dishes in the sink
Probabilistic Context-Free Grammars

- A context-free grammar is a tuple \(<N, T, S, R>\)
  - \(N\) : the set of non-terminals
    - Phrasal categories: \(S, NP, VP, ADJP, \text{etc.}\)
    - Parts-of-speech (pre-terminals): \(NN, JJ, DT, VB\)
  - \(T\) : the set of terminals (the words)
  - \(S\) : the start symbol
    - Often written as \(\text{ROOT}\) or \(\text{TOP}\)
    - Not usually the sentence non-terminal \(S\)
  - \(R\) : the set of rules
    - Of the form \(X \rightarrow Y_1 Y_2 \ldots Y_k\), 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 \(P(Y_1 Y_2 \ldots Y_k | X)\)
Treebank Grammars

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

```
S → NP VP . 1.0
NP → PRP 0.5
NP → DT NN 0.5
VP → VBD NP 1.0
PRP → She 1.0
...
```

• 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
Chomsky Normal Form

- **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!
A Recursive Parser

```python
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:

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

- Can also organize things bottom-up

```plaintext
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]
```
Time: Theory

• How much time will it take to parse?

  • For each diff (<= n)
    • For each i (<= 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, or 0.2 sec for an optimized parser
Unary Rules

Unary rules?

\[
\text{bestScore}(X,i,j,s) \\
\quad \text{if } (j = i+1) \\
\quad \quad \text{return } \text{tagScore}(X,s[i]) \\
\quad \text{else} \\
\quad \quad \text{return } \max \ \max \ \text{score}(X \rightarrow YZ) \times \\
\quad \quad \quad \text{bestScore}(Y,i,k) \times \\
\quad \quad \quad \text{bestScore}(Z,k,j) \\
\quad \quad \quad \max \ \text{score}(X \rightarrow Y) \times \\
\quad \quad \quad \quad \text{bestScore}(Y,i,j)
\]
**CNF + Unary Closure**

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

- Alternate unary and binary layers
- Reconstruct unary chains afterwards
Alternating Layers

\[
\text{bestScoreB}(X, i, j, s) = \\
\quad \text{return max max score}(X\to YZ) \times \\
\quad \text{bestScoreU}(Y, i, k) \times \\
\quad \text{bestScoreU}(Z, k, j)
\]

\[
\text{bestScoreU}(X, i, j, s) = \\
\quad \text{if } (j = i+1) \\
\quad \quad \text{return tagScore}(X, s[i]) \\
\quad \text{else} \\
\quad \quad \text{return max max score}(X\to Y) \times \\
\quad \quad \quad \text{bestScoreB}(Y, i, j)
\]
Treebank Grammars

- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn’t work well):
  
  \[
  S \rightarrow NP \ VP . \quad 1.0 \\
  NP \rightarrow PRP \quad 0.5 \\
  NP \rightarrow DT \ NN \quad 0.5 \\
  VP \rightarrow VBD \ NP \quad 1.0 \\
  PRP \rightarrow She \quad 1.0 \\
  \]

- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak ’96</td>
<td>72.0</td>
</tr>
</tbody>
</table>
Conditional Independence?

- Not every NP expansion can fill every NP slot

- A grammar with symbols like “NP” won’t be context-free

- Statistically, conditional independence too strong
Non-Independence

- Independence assumptions are often too strong.

Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).

Also: the subject and object expansions are correlated!
The Game of Designing a Grammar

- Structure Annotation [Johnson ‘98, Klein & Manning ‘03]
- Lexicalization [Collins ‘99, Charniak ‘00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. ‘06]
- (Neural) CRF Parsing [Hall et al. ‘14, Durrett & Klein ‘15]
A Fully Annotated (Unlexicalized) Tree

[Charniak '96: 72.0]

[Klein & Manning '03: 86.3]

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak '96</td>
<td>72.0</td>
</tr>
<tr>
<td>Klein&amp;Manning '03</td>
<td>86.3</td>
</tr>
</tbody>
</table>
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Head lexicalization [Collins ’99, Charniak ’00]
Problems with PCFGs

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

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

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

- Problem: we now have to estimate probabilities like

\[ VP(\text{saw}) \rightarrow VBD(\text{saw}) \ NP-C(\text{her}) \ NP(\text{today}) \]

- Never going to get these atomically off of a treebank

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

- A derivation of a local tree [Collins '99]

Choose a head tag and word

Choose a complement bag

Generate children (incl. adjuncts)

Recursively derive children
Lexicalized Grammars

- **Challenges:**
  - Many parameters to estimate: requires sophisticated smoothing techniques
  - Exact inference is too slow: requires pruning heuristics
  - Difficult to adapt to new languages: At least head rules need to be specified, typically more changes needed

<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>Charniak ’00</td>
<td>90.1</td>
</tr>
</tbody>
</table>
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Automatic clustering
Latent Variable Grammars

[Matsuzaki et al. ’05, Petrov et al. ’06]
Learning Latent Annotations

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

Just like Forward-Backward for HMMs.
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - CRF Parsing (+Neural Network Representations)
Generative vs. Discriminative

**Generative**

Maximize joint likelihood of gold tree **and** sentence

EM-algorithm

EASY: expectations over observed trees

[Matsuzaki et al. ’05, Petrov et al. ’06]

**Discriminative**

Maximize conditional likelihood of gold tree **given** sentence

Gradient-based algorithm

HARD: expectations over all trees

[Petrov & Klein ’07, ’08]
Objective Functions

Generative Objective Function:
\[
\max_{\theta} \mathcal{L}_\theta(y, w_1 \ldots w_n)
\]
[Petrov, Barrett, Thibaux & Klein ’06]

Discriminative Objective Function:
\[
\max_{\theta} \mathcal{L}_\theta(y | w_1 \ldots w_n)
\]
[Petrov & Klein ’08, Finkel et. al ’08]

Bayesian Objective Function:
\[
\max_{\theta} \mathcal{P}(\theta | y) \mathcal{L}_\theta(y, w_1 \ldots w_n)
\]
[Liang, Petrov, Jordan & Klein ’07]
(Neural) CRF Parsing


Score of VP over this span

$w \cdot f_s$

dense neural network

$w \cdot f_s$

sparse log-linear model

He gave a speech

Be a tree

NNP NNP VBD DT NN NNP VBD DT NN
CRF Parsing Sparse Features

\[ P(T|x) \propto \prod_{r \in T} \exp(\text{score}(r)) \]

\[
\text{score}(2\text{NP}_7 \rightarrow 2\text{NP}_4 4\text{PP}_7) = w^\top f(2\text{NP}_7 \rightarrow 2\text{NP}_4 4\text{PP}_7)
\]

FirstWord = a & NP → NP PP
PrevWord = gave & NP → NP PP
AfterSplit = on & NP → NP PP
FirstWord = a & NP

...
Neural CRF Model

\[
\text{score}(2\text{NP}_7 \rightarrow 2\text{NP}_4 4\text{PP}_7) = W \odot \left( f_s(2\text{X}_7 \rightarrow 2\text{X}_4 4\text{X}_7) f_o^T(\text{NP} \rightarrow \text{NP PP}) \right)
\]

\[
f_s = g(Hv)
\]

(arbitrary neural network)

Sparse

He gave a speech on foreign policy.
LSTM Parsing

- Treat parsing as a sequence-to-sequence prediction problem
- Completely ignores tree structure, uses LSTMs as black boxes

\[ P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1}) \]
<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Parser</td>
<td>89.7</td>
</tr>
<tr>
<td>[Charniak et al. '06]</td>
<td>90.1</td>
</tr>
<tr>
<td>[Petrov et al. '06]</td>
<td>91.0</td>
</tr>
<tr>
<td>[Carreras et al. '08]</td>
<td>91.1</td>
</tr>
<tr>
<td>[Hall '12]</td>
<td>91.3</td>
</tr>
<tr>
<td>[Durrett et al. '15]</td>
<td>91.4</td>
</tr>
<tr>
<td>[Zhu et al. '13]</td>
<td>91.6</td>
</tr>
<tr>
<td>[Huang &amp; Harper '08]</td>
<td>91.8</td>
</tr>
<tr>
<td>[Huang &amp; Harper, Petrov '10]</td>
<td>91.7</td>
</tr>
<tr>
<td>[Charniak &amp; Johnson '05]</td>
<td>91.8</td>
</tr>
<tr>
<td>[Huang '08]</td>
<td>92.3</td>
</tr>
<tr>
<td>[McClosky et al. '06]</td>
<td>92.4</td>
</tr>
<tr>
<td>[Petrov '10]</td>
<td>92.4</td>
</tr>
<tr>
<td>[Huang &amp; Harper, Petrov '10]</td>
<td>92.6</td>
</tr>
</tbody>
</table>

### Detailed English Results

**Graph**

- **Single Parser**
  - 89.7
- **Self-Trained**
  - 91.0
- **Reranker**
  - 91.1
- **Product**
  - 91.3
- **Combination**
  - 91.4

- **Charniak '00**
- **Petrov et al. '06**
- **Carreras et al. '08**
- **Durrett et al. '15**
- **Zhu et al. '13**
- **Huang & Harper '08**
- **Huang & Harper, Petrov '10**
- **Charniak & Johnson '05**
- **Huang '08**
- **McClosky et al. '06**
- **Petrov '10**
- **Huang & Harper, Petrov '10**
- **Sagae & Lavie '06**
- **Fossum & Knight '09**
- **Zhang et al. '09**
- **Hall '12**
- **Durrett et al. '15**
- **Zhu et al. '13**
- **Huang & Harper, Petrov '10**
- **Sagae & Lavie '06**
- **Fossum & Knight '09**
- **Zhang et al. '09**
Multi-Lingual Results

Test set F1 all lengths

Hall et al. ’14

Durrett et al. ’15

Arabic
Basque
French
German
Hebrew
Hungarian
Korean
Polish
Swedish
Average

83.4
85.4
81.3
81.0
87.2
88.6
90.7
93.0
83.2
85.1