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 2016
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:**

  
  - VBD       VB
  - VBN       VBZ
  - NNP       NNS      NN      NNS      CD      NN

  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>VBP → interest</td>
<td></td>
</tr>
<tr>
<td>NNS → raises</td>
<td></td>
</tr>
<tr>
<td>VBD → raises</td>
<td></td>
</tr>
</tbody>
</table>

- **Use deduction systems to prove parses from words**
  - 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
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: \( S, NP, VP, ADJP \), etc.
    - Parts-of-speech (pre-terminals): \( NN, JJ, DT, VB \)
  - \( T \) : the set of terminals (the words)
  - \( S \) : the start symbol
    - Often written as ROOT or TOP
    - Not usually the sentence non-terminal \( S \)
  - \( R \) : the set of rules
    - Of the form \( X \rightarrow Y_1 Y_2 \ldots Y_k \), 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 $X \rightarrow YZ$ or $X \rightarrow w$
  • In principle, this is no limitation on the space of (P)CFGs
    • N-ary rules introduce new non-terminals

  • Unaries / empties are “promoted”

  • In practice it’s kind of a pain:
    • Reconstructing n-aries is easy
    • Reconstructing unaries is trickier
    • The straightforward transformations don’t preserve tree scores

  • Makes parsing algorithms simpler!
A Recursive Parser

```
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:

```java
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

```python
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 → Y Z
        • For each split point k
          Do constant work
    
  • Total time: |rules|*n^3
  • Something like 5 sec for an unoptimized parse of a 20-word sentences, or 0.2sec for an optimized parser
Unary Rules

- Unary rules?

```python
bestScore(X, i, j, s)
    if (j = i+1)
        return tagScore(X, s[i])
    else
        return max max score(X→YZ) * 
                bestScore(Y, i, k) * 
                bestScore(Z, k, j)
        max score(X→Y) * 
            bestScore(Y, i, j)
```
CNF + Unary Closure

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

- Alternate unary and binary layers
- Reconstruct unary chains afterwards
bestScoreB(X, i, j, s)
    return max max score(X->YZ) *
    bestScoreU(Y, i, k) *
    bestScoreU(Z, k, j)

bestScoreU(X, i, j, s)
    if (j = i+1)
        return tagScore(X, s[i])
    else
        return max max score(X->Y) *
            bestScoreB(Y, i, j)
Treebank Grammars

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

- 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

[Model: Klein & Manning '03]

F1

Charniak '96 72.0
Klein & Manning '03 86.3
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:
  - $\text{VP} \rightarrow \text{VP PP}$
  - $\text{NP} \rightarrow \text{NP PP}$
- 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

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

• 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]
  
  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>
Lexicalized CKY

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

Diagram:

- \((VP \rightarrow VBD \ldots NP \cdot)[saw]\)
- \((VP \rightarrow VBD \cdot)[saw] \ NP[her]\)
- \((X[h]), (Y[h]), (Z[h'])\)
- \(i, h, k, h', j\)

Example:

- \((VP \rightarrow VBD \ldots NP \cdot)[saw]\)
- \((VP \rightarrow VBD \cdot)[saw] \ NP[her]\)
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak ’00</td>
<td>90.1</td>
</tr>
<tr>
<td>Petrov et al. ’06</td>
<td>90.6</td>
</tr>
</tbody>
</table>
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

W₁ W₂ ... Wₙ

W₁ W₂ ... Wₙ

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

W₁ W₂ ... Wₙ

HARD: expectations over all trees

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

Generative Objective Function:
\[
\max_{\theta} \mathcal{L}_\theta (\mathcal{Y}, w_1...w_n) \quad [\text{Petrov, Barrett, Thibaux & Klein '06}]
\]

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

Bayesian Objective Function:
\[
\max_{\theta} \mathcal{P}(\theta | \mathcal{Y}) \mathcal{L}_\theta (\mathcal{Y}, w_1...w_n) \quad [\text{Liang, Petrov, Jordan & Klein '07}]
\]
He gave a speech

[Taskar et al. ‘04, Petrov & Klein ’07, Hall et al. ‘14, Durrett et al. ’15]

(Neural) CRF Parsing

Score of VP over this span

w · f_s
dense neural network

w · f_s
sparse log-linear model

Be a tree

He gave a speech
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

score($2NP_7 \rightarrow 2NP_{4\ 4}PP_7$) =

$$W \odot \left( f_s(2X_7 \rightarrow 2X_{4\ 4}X_7)f_o^T(NP \rightarrow NP\ PP) \right)$$

$$f_s = g(Hv)$$

(artificial neural network)

He gave a speech on foreign policy.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Petrov et al. ’06</td>
<td>90.6</td>
</tr>
<tr>
<td>Durrett et al. ’16</td>
<td>91.3</td>
</tr>
</tbody>
</table>
LSTM Parsing [Vinyals et al. ’15]

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

\[
P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})
\]
Detailed English Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Single Parser</th>
<th>Self-Trained</th>
<th>Reranker</th>
<th>Product</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Charniak '00]</td>
<td>89.7</td>
<td>90.1</td>
<td>91.3</td>
<td>91.8</td>
<td>92.4</td>
</tr>
<tr>
<td>[Petrov et al. '06]</td>
<td>89.2</td>
<td>91.1</td>
<td>91.6</td>
<td>91.8</td>
<td>92.4</td>
</tr>
<tr>
<td>[Carreras et al. '08]</td>
<td>91.0</td>
<td>91.1</td>
<td>91.4</td>
<td>91.7</td>
<td>92.3</td>
</tr>
<tr>
<td>[Durrett et al. '15]</td>
<td>91.0</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[Zhu et al. '13]</td>
<td>91.1</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[Huang &amp; Harper '08]</td>
<td>91.4</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[Huang &amp; Harper, Petrov '10]</td>
<td>91.6</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[Charniak &amp; Johnson '05]</td>
<td>91.0</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[McClosky et al. '06]</td>
<td>91.0</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[Petrov '10]</td>
<td>91.0</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[Huang &amp; Harper, Petrov '10]</td>
<td>91.8</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[Huang &amp; Harper, Petrov '10]</td>
<td>91.8</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[Sagae &amp; Lavie '06]</td>
<td>91.0</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[Fossum &amp; Knight '09]</td>
<td>91.0</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
<tr>
<td>[Zhang et al. '09]</td>
<td>91.0</td>
<td>91.3</td>
<td>91.5</td>
<td>91.7</td>
<td>92.0</td>
</tr>
</tbody>
</table>