Lecture 8: Constituency Parsing

Statistical NLP
Fall 2016

They solved the problem with statistics

Syntax and Semantics

Constituency and Dependency

Analyzing Natural Language

They solved the problem with statistics
Constituency and Dependency

PRON  VERB  DET  NOUN  ADP  NOUN

They solved the problem with statistics

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

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

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):
  - 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

```
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

```
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]
```

• One small change:
A Bottom-Up Parser (CKY)

• Can also organize things bottom-up

\[
\text{bestScore}(s) \\
\text{for } (i : [0,n-1]) \\
\quad \text{for } (X : \text{tags}[s[i]]) \\
\quad \quad \text{score}[X][i][i+1] = \\
\quad \quad \text{tagScore}(X,s[i]) \\
\text{for } (\text{diff} : [2,n]) \\
\text{for } (i : [0,n-\text{diff}]) \\
\quad j = i + \text{diff} \\
\text{for } (X \rightarrow YZ : \text{rule}) \\
\quad \text{for } (k : [i+1, j-1]) \\
\quad \quad \text{score}[X][i][j] = \max \left( \text{score}[X][i][j], \right. \\
\quad \quad \quad \left. \text{score}(X \rightarrow YZ) \times \right. \\
\quad \quad \quad \quad \text{score}[Y][i][k] \times \\
\quad \quad \quad \quad \text{score}[Z][k][j] \right)
\]

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: |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?

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

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

bestScoreB(X, i, j, s)
    return max \(\max_{X \to YZ} \) *
    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 \(\max_{X \to Y} \) *
        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):

```
S \rightarrow NP VP .
NP \rightarrow PRP
NP \rightarrow DT NN
VP \rightarrow VBD NP
PRP \rightarrow She
```

- 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

[Klein & Manning ’03]

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 [Charniak ‘97, Collins ‘97]

- 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

\[
VP(\text{aw}) \rightarrow VBD(\text{aw}) \text{ NP}^\text{C(her)} \text{ NP(} \text{today})
\]

- 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>

Lexicalized CKY

\[
\text{bestScore}(X,i,j,h) = \begin{cases} 
\text{tagScore}(X,s[i]) & \text{if } (j = i+1) \\
\max \left\{ \text{score}(X[h]\rightarrow Y[h] Z[h']) \ast \text{bestScore}(Y,i,k,h) \ast \text{bestScore}(Z,k,j,h') \right\} & \text{else}
\end{cases}
\]

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

The Game of Designing a Grammar

Latent Variable Grammars

[Matsumoto 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.

Refinement of the DT tag

Hierarchical Refinement

Hierarchical Estimation Results
Refinement of the " tag

- Splitting all categories equally is wasteful:

[Diagram showing the refinement process]

Adaptive Splitting

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

[Graph showing adaptive splitting results]

Adaptive Splitting Results

[Graph showing parsing accuracy (F1) against total number of grammar symbols]

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

Number of Phrasal Subcategories

[Bar chart showing number of phrasal subcategories]
Learned Splits

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

- Personal pronouns (PRP):
  - PRP-0: It, He, I
  - PRP-1: it, he, they
  - PRP-2: it, them, him

Learned Splits

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

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

Bayesian Symbol Refined TSG

- Latent Variable Tree-Substitution Grammar

  Hierarchical Generation Process:
  - Joint model of everything
  - Complex sampling scheme needed

[Shindo et al. ’12]
### Spectral Learning for PCFGs

- **EM is a local method**
  - Can never be sure to have the global optimum
  - Significant variance between different runs

- **Spectral methods**
  - Provably find the global optimum
  - Compute SVD of training data
  - Efficient to run
  - But currently not competitive in practice

---

### Objective Functions

#### Generative Objective Function:

\[
\max_{\theta} \mathcal{L}_\theta(\omega_1, \omega_2, \ldots, \omega_n) \quad [\text{Petrov, Barrett, Thibaux & Klein '06}]
\]

#### Discriminative Objective Function:

\[
\max_{\theta} \mathcal{L}_\theta(\omega_1, \omega_2, \ldots, \omega_n) \quad [\text{Petrov & Klein '08, Finkel et. al '08}]
\]

#### Bayesian Objective Function:

\[
\max_{\theta} \mathcal{P}(\theta) \mathcal{L}_\theta(\omega_1, \omega_2, \ldots, \omega_n) \quad [\text{Liang, Petrov, Jordan & Klein '07}]
\]

---

### The Game of Designing a Grammar

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

(Taskar et al. '04, Petrov & Klein '07, Hall et al. '14, Durrett et al. '15)

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

\text{score}(r) = w \cdot f_s(r)

**CRF Parsing Sparse Features**

FirstWord = a & NP \rightarrow NP
PrevWord = gave & NP \rightarrow NP
AfterSplit = on & NP \rightarrow NP
FirstWord = a & NP

**LSTM Parsing**

(Vinyals et al. '15)

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

They solved the problem with statistics

Coarse-to-Fine Inference

NP-21 NP-21

NP-21 NP-21

Prune?

For each chart item X[i,j], compute posterior probability:

\[ \frac{P_{IN}(X, i, j) \cdot P_{OUT}(X, i, j)}{P_{IN}(root, 0, n)} < \text{threshold} \]

E.g. consider the span 5 to 12:

Bracket Posteriors
Hierarchical Pruning

coarse:

split in two:

split in four:

split in eight:

Parsing Times (per sentence)

<table>
<thead>
<tr>
<th></th>
<th>Fine</th>
<th>X-Bar - Fine</th>
<th>Coarse-to-Fine original grammars</th>
<th>Coarse-to-Fine projected grammars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parsing time per sentence (in sec)</td>
<td>62</td>
<td>5</td>
<td>2.1</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Detailed English Results

Multi-Lingual Results