CS460/626 : Natural Language Processing/Speech, NLP and the Web
(Lecture 11, 12– Parsing Algorithms)

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay
30th and 31st Jan, 2012
Need for parsing

Sentences are linear structures

But there is a hierarchy - a tree - hidden behind the linear structure

There are constituents and branches

(Note: things to come back to - POS tagset; MEMM and CRF for POS)
Parsing

- Essentially a Yes/No problem (belongingness)
- Parse Tree is a side effect
- 2 general ways – Expand from grammar or Resolve through data

Language Representation

Intrinsic Representation

Grammar for STR:

\[ S \rightarrow aSb \mid \epsilon \]

- Compact representation
- Less Kolomogorov Complexity

Extrinsic Representation

STR: \( \{ \epsilon, ab, aabb, \ldots, a^n b^n \} \)

- Enumerated Representation
Points to consider:
- Should POS Tagging precede parsing?
- Is POS Tagging necessary for Parsing?

→ POS Tagging increases implementation efficiency

Data

**People**  

**laugh**

Lexicon

People- Noun, Verb  

laugh- Noun, Verb

Grammar

- Going back again and again to the lexicon isn’t required when POS Tagging has been done before Parsing
Two issues are at the crux of parsing:
- Ambiguity in Data
- Ambiguity in Grammar

Parsing Algorithms:
- Top-Down Parsing
  - Predictive Parsing, Expectation Driven Parsing, Theory Driven Parsing, Grammar Driven Parsing
  - Suffers from Left-recursion
- Bottom-Up Parsing
  - Data Driven parsing
  - Ambiguity on POS Tags can lead to useless steps while parsing

Chart Parsing
Example sentence:

1. People 2 laugh 3 loudly 4

- Multiple parse trees possible for ambiguous sentences
  - The man saw a boy with a telescope

- Partial parses are also important
  - Text entailment
  - Question Answering
    - People laugh loudly → Who laughs? People laugh
Grammar and Parsing Algorithms
A simplified grammar

- $S \rightarrow NP \ VP$
- $NP \rightarrow DT \ N \mid N$
- $VP \rightarrow V \ ADV \mid V$
Example Sentence

People laugh

Lexicon:
People - N, V
Laugh - N, V

These are positions

This indicate that both Noun and Verb is possible for the word “People”
## Top-Down Parsing

<table>
<thead>
<tr>
<th>State</th>
<th>Backup State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ((S) 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. ((NP VP)1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3a. ((DT N VP)1)</td>
<td>((N VP) 1)</td>
<td></td>
</tr>
<tr>
<td>3b. ((N VP)1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. ((VP)2)</td>
<td></td>
<td>Consume “People”</td>
</tr>
<tr>
<td>5a. ((V ADV)2)</td>
<td>((V)2)</td>
<td></td>
</tr>
<tr>
<td>6. ((ADV)3)</td>
<td>((V)2)</td>
<td>Consume “laugh”</td>
</tr>
<tr>
<td>5b. ((V)2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. ((.)3)</td>
<td></td>
<td>Consume “laugh”</td>
</tr>
</tbody>
</table>

Termination Condition: All inputs over. No symbols remaining.

Note: Input symbols can be pushed back.
Discussion for Top-Down Parsing

- This kind of searching is goal driven.
- Gives importance to textual precedence (rule precedence).
- No regard for data, a priori (useless expansions made).
Bottom-Up Parsing

Some conventions:

$N_{12}$  
Represents positions

$S_{1?} \rightarrow NP_{12} \cdot VP_{2?}$
End position unknown  
Work on the LHS done, while the work on RHS remaining
Bottom-Up Parsing (pictorial representation)

1. People
   - N_{12}
   - V_{12}
   - NP_{12} \rightarrow N_{12}°
   - VP_{12} \rightarrow V_{12}°
   - S_1? \rightarrow NP_{12}° VP_{22}°

2. Laugh
   - N_{23}
   - V_{23}
   - NP_{23} \rightarrow N_{23}°
   - VP_{23} \rightarrow V_{23}°

3. S \rightarrow NP_{12} VP_{23}°
Problem with Top-Down Parsing

- Left Recursion
  - Suppose you have A -> AB rule.
    Then we will have the expansion as follows:
    - ((A)K) -> ((AB)K) -> ((ABB)K) .......
Combining top-down and bottom-up strategies
Top-Down Bottom-Up Chart Parsing

- Combines advantages of top-down & bottom-up parsing.
- Does not work in case of left recursion.
  - *e.g.* – “People laugh”
    - People – noun, verb
    - Laugh – noun, verb
  - **Grammar**
    - \[ S \rightarrow \text{NP} \text{ VP} \]
    - \[ \text{NP} \rightarrow \text{DT N} | \text{N} \]
    - \[ \text{VP} \rightarrow \text{V ADV} | \text{V} \]
Transitive Closure

1

S → NP VP
NP → DT N
NP → N

2

NP → N
S → NP VP
VP → V ADV
VP → V

3

VP → V
S → NP VP
success
Arrows in Parsing

- Each arc represents a chart which records
  - Completed work (left of •)
  - Expected work (right of •)
Example

People

1 2 3 4

S →• NP VP
NP →• DT N
NP →• N
VP →• V ADV
VP →• V

laugh

VP → V
VP → V•ADV
S → NP VP•

loudly

VP → V• ADV•
S → NP VP•
Dealing With Structural Ambiguity

- Multiple parses for a sentence
  - The man saw the boy with a telescope.
  - The man saw the mountain with a telescope.
  - The man saw the boy with the ponytail.

At the level of syntax, all these sentences are ambiguous. But semantics can disambiguate 2nd & 3rd sentence.
Prepositional Phrase (PP) Attachment Problem

V – NP₁ – P – NP₂

(Here P means preposition)
NP₂ attaches to NP₁?
or NP₂ attaches to V?
Parse Trees for a Structurally Ambiguous Sentence

Let the grammar be –

\[ S \rightarrow NP \text{ VP} \]
\[ NP \rightarrow DT \text{ N} \mid DT \text{ N PP} \]
\[ PP \rightarrow P \text{ NP} \]
\[ VP \rightarrow V \text{ NP PP} \mid V \text{ NP} \]

For the sentence,

“I saw a boy with a telescope”
I saw a boy with a telescope.
I saw a boy with a telescope.