Syntactic Parsing Introduction

Niranjan Balasubramanian
Stony Brook University

Aug 27, 2015

Some slides adapted from these fantastic researchers: Ray Mooney, Michael Collins, and Chris Manning.
• Syntactic Parsing
  – What is it?
  – A simple formalism (PCFG)
  – Issues w/ Vanilla PCFG
  – Hints at how these are typically addressed.
• What is syntactic parsing?
  – Identifying *syntactic* structure underlying a sentence.
  – Assumes that there is a set of rules that underlie language.

• Why is it useful?
  – Serves as a model that explains the observed language string.
  – Use it to:
    • Predict or complete sentences.
    • Re-organize, simplify sentences.
    • Learn semantic phenomenon identifiable via syntactic patterns.

• What are the big issues in syntactic parsing?
  – Dependence on semantics
  – Lexicalization helps but hurts generalization.
  – Speed. Parsing is a $O(N^3)$ business w/ a constant the size of grammar.
A tree representing nested compositional structure.

Leaves are words.
Nodes one-level above leaves are POS tags or Internal nodes are syntactic categories e.g. Noun phrase (NP).
A tree representing pairwise syntactic relations.

Nodes are words.
Edges are grammatical relations.
Constituency vs. Dependency Grammars

- What is the difference?
  - Constituency parse groups words that act as a unit.
  - Constituents are typically “headed” by a particular type of word.
    - E.g., noun phrases are headed by a noun, verb phrases by a verb.
  - Dependency parse directly specifies relations between heads and their dependents.

- A deterministic procedure can transform constituency into dependency.

- Dependency parses are more compact than constituency parses.
  - [How does this impact the automatic parsing?]
Why is syntactic parsing hard?

Ambiguity:
- PP attachment
- Noun pre-modifiers
What is so hard about this?

>> Write down a grammar in a formal language that has sufficient representation power.

Noam Chomsky tried this in his thesis!

Turns out this is not simple.

Ambiguity and coverage make it hard.

**Fundamental trade-off:**
1) Smaller grammars have limited coverage. No parses for many sentences.
2) Large grammars improve coverage but are ambiguous and yield more parses.
Main Questions in Parsing

• What is the formalism for the grammar?
  – Constituency (Phrase-structure) vs. Dependency Grammar

• How does one get the grammar?
  – We are not writing it down!
  – To address ambiguity, we need probabilities attached to the grammar.

• How does one parse sentences given a grammar?
Parsers done three ways!

- Probabilistic Context Free Grammars
- Transition-based Parsing (Next class)
- Graph-based methods (Next week)
What are context-free grammars?

- Grammars are a way to encode rules that can generate strings in a language.
  - Recall formal languages such as regular, context free, context-sensitive etc.

- What grammar generates the following strings?
  - A, AA, AAA, AAA, …?
  - AB, ABA, ABBA, …?
  - ABCD, AABBCDD, …?

- What kind of language is English?
  - Not regular.
  - Not necessarily context-free.

[Can you come up with sentences that show this?]
Context-Free Grammars -- Formally

- $N$ a set of \textit{non-terminal symbols} (or \textit{variables})
- $\Sigma$ a set of \textit{terminal symbols} (disjoint from $N$)
- $R$ a set of \textit{productions} or \textit{rules} of the form:
  \[ A \rightarrow \beta, \]
  where $A$ is a non-terminal and
  \[ \beta \text{ is a string of symbols from } (\Sigma \cup N)^* \]
- $S$, a designated non-terminal called the \textit{start symbol}

Strings that can be generated by applying a sequence of rules from $R$ are said to be in the language of the grammar.

Parsing becomes the task of identifying if a string is generated by the grammar (and recovering the sequence of rules that generated it).
### Probabilistic Context Free Grammars (PCFGs)

Probabilistic Context Free Grammars are CFGs + Probabilities

<table>
<thead>
<tr>
<th>S</th>
<th>NP</th>
<th>VP</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<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>

<table>
<thead>
<tr>
<th>Vt</th>
<th>saw</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>man</td>
<td>0.7</td>
</tr>
<tr>
<td>NN</td>
<td>woman</td>
<td>0.2</td>
</tr>
<tr>
<td>NN</td>
<td>telescope</td>
<td>0.1</td>
</tr>
<tr>
<td>DT</td>
<td>the</td>
<td>1.0</td>
</tr>
<tr>
<td>IN</td>
<td>with</td>
<td>0.5</td>
</tr>
<tr>
<td>IN</td>
<td>in</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Modeling Assumption:
PCFG rules applied recursively derive sentences.

What about ambiguity?
The product of the probabilities of the rules scores each parse.

**What independence assumptions justify this factoring?**

<table>
<thead>
<tr>
<th>DERIVATION</th>
<th>RULES USED</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S $\rightarrow$ NP VP</td>
<td>1.0</td>
</tr>
<tr>
<td>NP VP</td>
<td>NP $\rightarrow$ DT N</td>
<td>0.3</td>
</tr>
<tr>
<td>DT N VP</td>
<td>DT $\rightarrow$ the</td>
<td>1.0</td>
</tr>
<tr>
<td>the N VP</td>
<td>N $\rightarrow$ dog</td>
<td>0.1</td>
</tr>
<tr>
<td>the dog VP</td>
<td>VP $\rightarrow$ VB</td>
<td>0.4</td>
</tr>
<tr>
<td>the dog VB</td>
<td>VB $\rightarrow$ laughs</td>
<td>0.5</td>
</tr>
<tr>
<td>the dog laughs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TOTAL PROBABILITY** = $1.0 \times 0.3 \times 1.0 \times 0.1 \times 0.4 \times 0.5$
How to parse given a PCFG?

- Exhaustive search of the space of derivations that can produce the input sentence is *infeasible*.

- Why? What is inefficient about this approach?

- So how can we make it better?
Idea: Do not explore paths that cannot lead to the sentence.
Top Down Parsing

```
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP
```

```
Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
...
```

```
book that flight
```

```
S    
/    
NP   VP
    /    
    Pronoun
```

Top Down Parsing

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

book that flight

S
  NP   VP
     /
   Pronoun
    /
   book

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does

....
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does

…. 
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does

....
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does

…. 
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does

....
Top Down Parsing

book that flight

S \to NP \ VP
S \to Aux \ NP \ VP
S \to VP

NP \to Pronoun
NP \to Proper-Noun
NP \to Det Nominal

Nominal \to Noun
Nominal \to Nominal Noun
Nominal \to Nominal PP

VP \to Verb
VP \to Verb NP
VP \to VP PP

PP \to Prep NP

Det \to the \mid a \mid that \mid this
Noun \to book \mid flight \mid meal
Verb \to book \mid prefer
Pronoun \to I \mid he \mid she \mid me
Aux \to does

....
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

S

| VP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does

....
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does

....
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
....
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does

....
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does

…. 
Top Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does

....
And on it goes until ....

Rules Used:

S -> VP
VP -> Verb NP
Verb -> book
NP -> Det Nominal
Det -> that
Nominal -> Noun
Noun -> flight
Efficiency of Parsing

• Top down (and bottom up) are quite bad.
  – Asymptotic complexity is exponential in the length of the sentence \((N)\).

• Dynamic programming approaches bring the complexity down to \(O(N^3)\)
  – E.g. Cocke-Young-Kasami algorithm

• Remember grammar size also affects runtime by a constant factor.

[Learn CYK algorithm and understand the impact of grammar size]
How to learn a PCFG?

• Assume you are given example sentences and their parses (generated by humans).

• You can get the rules by inspecting the parses.

• Obtain probabilities by simple maximum Likelihood estimates:

\[ P_{ML}(\alpha \rightarrow \beta | \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)} \]

• What could be potential pitfalls with this estimation approach?
  – Unseen words.
  – Unseen constructions.
  – Infrequent combinations.
Issues with PCFGs

• Makes strong independence assumptions about language
  – Lexical independence
  – Structural independence
Lexical Dependence: PP Attachment Ambiguity

S
  NP
    NNS
      workers
    VP
      VBD
        dumped
      NP
        NNS
          sacks
      IN
        into
      NP
        DT
          a
          NN
            bin

S
  NP
    NNS
      workers
    VP
      VBD
        dumped
      NP
        NNS
          sacks
      IN
        into
      DT
        a
          NN
            bin
Lexical Dependence:
PP Attachment Ambiguity

\[ \text{Prob(Tree 1, Sentence)} = \ldots \times \text{Prob(VP -> VP PP | VP)} \times \ldots \]
Lexical Dependence:
PP Attachment Ambiguity

\[ \text{Prob(Tree 2, Sentence)} = \ldots \times \text{Prob(NP -> NP PP | NP)} \times \ldots \]
Lexical Dependence: PP Attachment Ambiguity

<table>
<thead>
<tr>
<th>Rules</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>S → NP VP</td>
</tr>
<tr>
<td>NP → NNS</td>
<td>NP → NNS</td>
</tr>
<tr>
<td><strong>VP → VP PP</strong></td>
<td><strong>NP → NP PP</strong></td>
</tr>
<tr>
<td>VP → VBD NP</td>
<td>VP → VBD NP</td>
</tr>
<tr>
<td>NP → NNS</td>
<td>NP → NNS</td>
</tr>
<tr>
<td>PP → IN NP</td>
<td>PP → IN NP</td>
</tr>
<tr>
<td>NP → DT NN</td>
<td>NP → DT NN</td>
</tr>
<tr>
<td>NNS → workers</td>
<td>NNS → workers</td>
</tr>
<tr>
<td>VBD → dumped</td>
<td>VBD → dumped</td>
</tr>
<tr>
<td>NNS → sacks</td>
<td>NNS → sacks</td>
</tr>
<tr>
<td>IN → into</td>
<td>IN → into</td>
</tr>
<tr>
<td>DT → a</td>
<td>DT → a</td>
</tr>
<tr>
<td>NN → bin</td>
<td>NN → bin</td>
</tr>
</tbody>
</table>

Prob(Tree 1, Sentence) > Prob(Tree 2, Sentence)

If Prob(VP -> VP PP | VP) > Prob(NP -> NP PP | NP)
Lexical Dependence:
Co-ordination Ambiguity

```
 NP
  | NP
  |   | CC
  | NP
  |   |   | NNS
  | NP
  |   |   |   | and
  | NNS
  |   |   | dogs
  | IN
  |   |   | houses

NP
  | NP
  |   | PP
  | NP
  |   |   | NNS
  | IN
  |   |   | dogs
  | NNS
  |   |   | cats
```
Lexical Dependence:
Co-ordination Ambiguity
Lexical Dependence: Co-ordination Ambiguity
Lexical Dependence: Co-ordination Ambiguity
Lexical Dependence:
Co-ordination Ambiguity

<table>
<thead>
<tr>
<th>Rules</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP → NP CC NP</td>
<td>NP → NP CC NP</td>
</tr>
<tr>
<td>NP → NP PP</td>
<td>NP → NP PP</td>
</tr>
<tr>
<td>NP → NNS</td>
<td>NP → NNS</td>
</tr>
<tr>
<td>PP → IN NP</td>
<td>PP → IN NP</td>
</tr>
<tr>
<td>NP → NNS</td>
<td>NP → NNS</td>
</tr>
<tr>
<td>NP → NNS</td>
<td>NP → NNS</td>
</tr>
<tr>
<td>NNS → dogs</td>
<td>NNS → dogs</td>
</tr>
<tr>
<td>IN → in</td>
<td>IN → in</td>
</tr>
<tr>
<td>NNS → houses</td>
<td>NNS → houses</td>
</tr>
<tr>
<td>CC → and</td>
<td>CC → and</td>
</tr>
<tr>
<td>NNS → cats</td>
<td>NNS → cats</td>
</tr>
</tbody>
</table>
Structural Preferences

- Same rules applied in both cases. Tree probability is the same.
- Left structure is twice as likely in Wall Street Journal.
- There are similar issues when PPs can attach to multiple verbs.
Structural Dependence

- All NPs:
  - NP PP: 11%
  - DT NN: 9%
  - PRP: 6%

- NPs under S:
  - NP PP: 9%
  - DT NN: 9%
  - PRP: 21%

- NPs under VP:
  - NP PP: 23%
  - DT NN: 7%
  - PRP: 4%
• Specific verbs take some types of arguments but not others.
  – Intransitive, transitive, and di-transitive
  – Finite vs. Non-finite verbs.

• A generic VP label hides the different argument preferences of the various sub-categories.
How to address these issues?

• Lexical dependence
  – Introduce lexical items into the tree.
  – Use headwords as part of the node-label. [Charniak 1997]

• Structural Dependence
  – Add more information to non-terminal categories [state splitting]
    • Include information about parents. [Johnson 1998]
    • Include fine-grained information (mark possessives for example)

• Sub-categorization
  – Add information to the non-terminal categories (state splitting)
  – E.g., S -> NP_firstpersonsingular VP_firstpersonsingular

Trade-off: Adding lexical information and fine-grained categories:

a) Increases sparsity -- Need appropriate smoothing.
b) Adds more rules -- Can affect parsing speed.
A Summary of the Issues

So what are all these issues essentially pointing out?

• Syntactic categories have different attachment preferences depending on their context.
  • Lexical or otherwise.
• Adding this context results in estimation issues due to sparsity.

This is a central challenge in NLP. Many phenomena have this characteristic.

Typical remedial actions include:

1) Using lexicalization but with generalization or dimensionality reduction.
2) Using carefully constructed features that leverages “expert intuitions” thereby avoiding sparsity issues.
• Next class.
  – Research summary and presentation templates.
  – An un-lexicalized parser.
  – Switch a bit for transition based parsing.
  – Annotate some sentences in class.

• Starting next week
  – Student presentations
  – Research reports due before class on Thursday.

• Will add “additional readings”.