Natural Language Parsing with Context-Free Grammars

SPFLODD

September 10, 2013
What is “Parsing”?  

• General answer: analyze text with respect to some theory.

• Usually it means **syntactic analysis**.

• **Syntax**: branch of linguistics dealing with how words and phrases are *ordered* to create well-formed sentences.
  
  – As in programming languages, syntax is understood as relevant to the mapping from strings to their meanings.

• Different theories of syntax → different kinds of parsing.
  
  – Today we’ll talk about context-free syntax
  
  – Thursday we’ll talk about dependency syntax
Formal Stuff First
Context-Free Grammars

- Chomsky hierarchy:
- Informally, CFGs can represent center-embedding, which regular grammars can’t.
- Classic argument from Chomsky (1956): NL is not regular.
  - Pumping lemma-type argument on (the Noun)^n (Verb-past)^n-1 VP
Context-Free Grammars

• Alphabet $\Sigma$
• Set of variables $N$
• Start symbol $S \in N$
• Rewrite rules: $X \rightarrow \alpha$, where $X \in N$ and $\alpha \in (N \cup \Sigma)^*$

• CNF: Assume $\alpha \in N^2 \cup \Sigma$. Can always convert to CNF.

• Grammars for NL usually have nonterminals like $S$, $NP$, $VP$, $PP$, and preterminals like $N$, $V$, $Adj$, $Adv$, ...
  – Tokens of labeled spans are called *constituents*. 
Probabilistic Context-Free Grammar

• Associate a multinomial distribution over right-hand sides to the set of rules sharing a left-hand side.
  – Conditional probability of “children” given “parent.”

• Generative story:
  1. Instantiate the start symbol S as a single red node.
  2. While there are any red symbols:
     1. Choose a red node X and color it white.
     2. Draw \( \alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_k \rangle \) according to \( p(*) \mid X \).
     3. Add \( \langle \alpha_1, \alpha_2, \ldots, \alpha_k \rangle \) to the tree as the sequence of children of the node X you selected.
     4. For any \( \alpha_i \) that are nonterminals, color them red; color the terminals white.
Like “Branching” Bayesian Networks

• Everything in a subtree is conditionally independent of everything else given its parent.
• A node’s label is conditionally independent of its descendents given its children.

• But not easy to capture in a Bayesian network:
  – variable length derivations of the grammar
  – joint model of tree structure and labels
  – direct dependency between any span’s label (or lack of label) and any potential parent, child, or sibling
HMMs are Special PCFGs

• Alphabet $\Sigma$
• $N = \text{HMM states } Q$
• Start state $q_0$
• Rules
  
  $q \rightarrow x \ q' \text{ with probability } p_{\text{emit}}(x \mid q) \ p_{\text{trans}}(q' \mid q)$
  
  $q \rightarrow \varepsilon \text{ with probability } p_{\text{trans}}(\text{stop} \mid q)$
Weighted Context-Free Grammar

• Don’t need a generative story; just assign weights to rules.
  – Can featurize

• Like a Markov network, but representing a WCFG as a MN is not elegant.
Parsing Natural Language
Penn Treebank (Marcus et al., 1993)

• A million words (40K sentences) of *Wall Street Journal* text (late 1980s).
  – This is important to remember!
• Parsed by experts; consensus parse for each sentence was published.
• The structure is basically what you’d expect from a PCFG.
  – Tends to be “flat” where there’s controversy.
  – Some “traces” for extraposed elements.
• Attempts to be theory-neutral, probably more accurate to say that it represents its own syntactic theory.
• Many other treebanks now available in other languages.
Example

(S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken))
    (, ,)
  (ADJP
    (NP (CD 61) (NNS years))
    (JJ old))
    (, ,)
  )
  (VP (MD will)
    (VP (VB join)
      (NP (DT the) (NN board))
      (PP-CLR (IN as)
        (NP (DT a) (JJ nonexecutive) (NN director)))
      (NP-TMP (NNP Nov.) (CD 29)))
    (. .))
  )
)
Example

(S
  (NP-SBJ-1
    (NP (NNP Rudolph) (NNP Agnew) )
    (, ,)
  )
  (UCP
    (ADJP
      (NP (CD 55) (NNS years) )
      (JJ old) )
    (CC and)
    (NP
      (NP (JJ former) (NN chairman) )
      (PP (IN of)
        (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC) )))
    (, ,)
  )
  (VP (VBD was)
    (VP (VBN named)
      (S
        (NP-SBJ (-NONE- *-1) )
        (NP-PRD
          (NP (DT a) (JJ nonexecutive) (NN director) )
          (PP (IN of)
            (NP (DT this) (JJ British) (JJ industrial) (NN conglomerate) )))
        (, . ))))
Evaluation

• Take a sentence from the test set.
• Use your parser to propose a hypothesis parse.
• Treebank gives you the correct parse.
• Precision and recall on labeled (or unlabeled) constituents.
  – Also, average number of crossing brackets (compared to correct parses) in your hypotheses.
• The training/development/test split has been held constant for a long time; possibly a cause for concern.
Basic Algorithms
CFG Parsing

• Given a treebank of reasonable size, the grammar we extract will be ambiguous.
  – Algorithms used for programming languages will not work.

• The most common approaches are based on two dynamic programming algorithms:
  – Cocke-Kasami-Younger (CKY) algorithm
  – Earley’s algorithm

• Originally these were not weighted, but today we assume rules have weights.
CKY: Weighted Logic Program

- \text{constit}(X, I, I) \ \text{max} = \text{word}(W, I) \times \text{unary}(X, W).
- \text{constit}(X, I, K) \ \text{max} = \text{constit}(Y, I, J)
  \times \text{constit}(Z, J+1, K)
  \times \text{binary}(X, Y, Z).
- \text{goal} \ \text{max} = \text{constit}(S, 1, N)
  \times \text{length}(N) \times \text{startsymbol}(S).
Visualizing Probabilistic CKY

\[ X \rightarrow w_i \]

\[ X \rightarrow Y Z \]

\[ X \rightarrow w_i \]

\[ X \rightarrow Y Z \]

\[ X \rightarrow Y Z \]

\[ X \rightarrow Y Z \]

\[ X \rightarrow w_i \]

\[ X \rightarrow Y Z \]

\[ X \rightarrow Y Z \]

\[ X \rightarrow Y Z \]

\[ X \rightarrow Y Z \]

\[ X \rightarrow Y Z \]

\[ X \rightarrow Y Z \]
Visualizing Probabilistic CKY
Visualizing Probabilistic CKY

How do we fill in $C(1,2)$?
How do we fill in $C(1,2)$?

Put together $C(1,1)$ and $C(2,2)$. 

Visualizing Probabilistic CKY
Visualizing Probabilistic CKY

How do we fill in $C(1,3)$?
Visualizing Probabilistic CKY

How do we fill in \( C(1,3) \)?

One way ...
How do we fill in $C(1,3)$?

One way ...
Another way.
Visualizing Probabilistic CKY

How do we fill in $C(1, n)$?
Visualizing Probabilistic CKY

How do we fill in $C(1,n)$?

$n - 1$ ways!
Visualizing Probabilistic CKY

$O(|N|n^2)$ cells to fill
$O(|N|^2 n)$ ways to fill each
Earley’s Algorithm

need(X, I) max = constit(_/Xα, _, I).
need(S, 0) max = \text{startsymbol}(S).
constit(X/α, I, I) max = \text{rewrite}(X, α) \text{ whenever } need(X, I).
\text{predict}
constit(X/α, I, J+1) max = constit(X/W α, I, J) \times \text{word}(W, J + 1).
\text{scan}
constit(X/α, I, K) max = constit(X/Yα, I, J) \times constit(Y/ε, J, K).
\text{complete}
goal max = constit(S/ε, 0, N) \times \text{length}(N) \times \text{startsymbol}(S).
Visualizing Probabilistic Earley’s
CKY vs. Earley’s

• Both $O(n^3)$ runtime, $O(n^2)$ space
• Neither requires weights to be probabilities, just like Viterbi.
• Earley’s doesn’t require the grammar to be in CNF
• Proof structures in Earley’s “move” left-to-right; CKY “moves” bottom-to-top.
• Earley’s ≈ on-the-fly binarization + CKY
• If you’re into logic programming, there are interesting ways to derive each of these from the other.
 Parsing in Reality

• Generally speaking, few industrial-strength parsers actually call CKY or Earley’s.

• Extensions to the basic CFG model (next topic) make reduction to CFG expensive.

• Standard techniques:
  – Beam search
  – Agenda-based approximations with pruning and/or A*
  – “Coarse-to-fine”
  – “Cube pruning” that makes use of local k-best lists (Huang and Chiang, 2005)
  – Shift-reduce-style algorithms with search
Better CFGs
Training Parsers In Practice

• Transformations on trees
  • Some of these are generally taken to be crucial
  • Some are widely debated
  • Lately, people have started learning these transformations
• Smoothing is crucial; the grammars that result from transformed trees have lots more rules and therefore more parameters.
from Johnson (1998)
Parent Annotation

NP $\rightarrow^p$ NP PP

NP $\rightarrow^q$ NP PP PP
Parent Annotation

\[ \text{NP}^{\text{VP}} \rightarrow \rho \text{NP}^{\text{NP}} \text{PP}^{\text{NP}} \]

\[ \text{NP}^{\text{NP}} \rightarrow \rho \text{NP}^{\text{NP}} \text{PP}^{\text{NP}} \]

\[ \text{NP}^{\text{VP}} \rightarrow q \text{NP}^{\text{NP}} \text{PP}^{\text{NP}} \text{PP}^{\text{NP}} \]
Parent Annotation

• Another way to think about it ...

• Before: 
  \[ p(\text{tree}) = \prod_{n \in \text{nodes(tree)}} \rho(\text{childsequence}(n) \mid n) \]

• Now: 
  \[ p(\text{tree}) = \prod_{n \in \text{nodes(tree)}} \rho(\text{childsequence}(n) \mid n, \text{parent}(n)) \]

• This could conceivably **help** performance (weaker independence assumptions)

• This could conceivably **hurt** performance (data sparseness)
Parent Annotation

• From Johnson (1998):
  • PCFG from WSJ Treebank: 14,962 rules
    • Of those, 1,327 would *always* be subsumed!
  • After parent annotation: 22,773 rules
    • Only 965 would always be subsumed!
• Recall 69.7% → 79.2%; precision 73.5% → 80.0%
• Trick: check for subsumed rules, remove them from the grammar → faster parsing.
Head Annotation

• “I love all my children, but one of them is special.”

\[
S \rightarrow NP \ VP
\]

\[
VP \rightarrow \text{VBD} \ NP
\]

\[
NP \rightarrow DT \ NNS \ PP
\]

• Heads not in the Treebank.
• Usually people use **deterministic head rules** (Magerman, 1995).
Lexicalization

• Every nonterminal node is annotated with a word from its yield; such that
  • \( \text{lex}(n) = \text{lex}(\text{head}(n)) \)
Lexicalization

• Every nonterminal node is annotated with a word from its yield; such that
  • \( \text{lex}(n) = \text{lex}(\text{head}(n)) \)

• What might this allow?
• What might we worry about?
Algorithms

• These “decorations” affect our parser’s runtime.
  – Why?
  – Any ideas about how to get around this?
Some Famous Parsers
Collins Model 1 (1997)

- Trees are headed and lexicalized
  - What’s the difference?

- Huge number of rules!
  - \( VP_{saw} \rightarrow V_{saw} \ NP_{man} \ PP_{through} \)
  - \( VP_{saw} \rightarrow V_{saw} \ NP_{man} \ PP_{with} \)
  - \( VP_{saw} \rightarrow V_{saw} \ NP_{woman} \ PP_{through} \)
  - \( VP_{saw} \rightarrow V_{saw} \ NP_{man} \)

- Key: factor probabilities within rule.
Collins Model 1 (1997)

• Everything factors down to rules, then further. We’re given the parent nonterminal and head word.
Collins Model 1 (1997)

• Everything factors down to rules, then further. We’re given the parent nonterminal and head word.

• Randomly generate the head child’s nonterminal.
Collins Model 1 (1997)

• Everything factors down to rules, then further. We’re given the parent nonterminal and head word.

• Randomly generate the head child’s nonterminal.

• Generate a sequence of left children.
Collins Model 1 (1997)

• Everything factors down to rules, then further. We’re given the parent nonterminal and head word.

• Randomly generate the head child’s nonterminal.

• Generate a sequence of left children.

\[
\text{VP}_{\text{saw}} \rightarrow \text{Adv}_{\text{somehow}} \rightarrow \text{V}_{\text{saw}}
\]
Collins Model 1 (1997)

• Everything factors down to rules, then further. We’re given the parent nonterminal and head word.
• Randomly generate the head child’s nonterminal.
• Generate a sequence of left children.
• Then right.
Collins Model 1 (1997)

• Everything factors down to rules, then further. We’re given the parent nonterminal and head word.

• Randomly generate the head child’s nonterminal.

• Generate a sequence of left children.

• Then right.
Collins Model 1 (1997)

- Everything factors down to rules, then further. We’re given the parent nonterminal and head word.
- Randomly generate the head child’s nonterminal.
- Generate a sequence of left children.
- Then right.
Interesting twist: want to model the **distance** between head constituent and child constituent. How?
Collins Model 1 (1997)

- Interesting twist: want to model the distance between head constituent and child constituent. How?
- Depth-first recursion.
Collins Model 1 (1997)

• Interesting twist: want to model the **distance** between head constituent and child constituent. How?

• Depth-first recursion.

• Condition next child on features of the parent’s yield so far.

![Diagram of syntactic tree]

- **VP**: VP
  - **Adv**: Adv
    - **V**: V
  - **NP**: NP
    - **PP**: PP
  - **generate these ...**
  - **... before this**
Collins Model 1 (1997)

• Interesting twist: want to model the distance between head constituent and child constituent. How?

• Depth-first recursion.

• Condition next child on features of the parent’s yield so far.

\[
p(PP_{\text{with}} \mid VP_{\text{saw}}, \text{right, “the cat who liked milk”}) \approx p(PP_{\text{with}} \mid VP_{\text{saw}}, \text{right, length } > 0, +\text{verb})
\]

\[
p(L_n, u_n, L_{n-1}, u_{n-1}, \ldots, L_1, u_1, H, w, R_1, v_1, R_2, v_2, \ldots, R_m, v_m \mid P, w)
\]

\[
= p(H \mid P, w) \cdot \prod_{i=1}^{n} p(L_i, u_i \mid P, w, H, \text{left}, \Delta_i) \cdot p(\text{stop} \mid P, w, H, \text{left}, \Delta_{n+1}) \cdot \prod_{i=1}^{m} p(R_i, v_i \mid P, w, H, \text{right}, \Delta_i') \cdot p(\text{stop} \mid P, w, H, \text{right}, \Delta_{n+1}')
\]
Collins Models 2 and 3 (1997)

• Model 2: Complements, adjuncts and subcategorization frames
  – Treebank decoration: -C on specifiers and arguments
  – Probability model: first pick set of complements (side-wise), must ensure they are all generated
  – *the issue was a bill funding Congress*

• Model 3: Wh-movement and extraction
  – Treebank decoration: “gap feature”
  – Probability model: gap feature “passed around the tree,” must be “discharged” as a trace element.
  – *the store that IBM bought last week*
Other Points

• Unknown words at test time: any training word with count < 6 becomes UNK
• Smoothing: deleted interpolation
• Tagging is just part of parsing (not a separate stage)
• Markov order increased in special cases:
  – within base noun phrases (NPBs) - first order
  – conjunctions ("and") predicted together with second conjunct
  – punctuation (details in 2003 paper)
Practical Notes

• Collins parser is freely available
• Dan Bikel replicated the Collins parser cleanly in Java
  – Easier to re-train
  – Easier to plug-and-play with different options
  – Multilingual support
  – May be faster (with current Java) - I’m not sure
Charniak (1997) - in brief

- Generally similar to Collins
- Key differences:
  - Used an additional 30 million words of unparsed text in training
  - Rules not fully markovized: pick full nonterminal sequence, then lexicalize each child independently
Charniak (1997) - in brief

VP\textsubscript{saw}
Charniak (1997) - in brief

$$\text{VP}_{saw} \rightarrow \text{Adv} \_ \_ V \_ \_ \text{NP} \_ \_ \text{PP}$$
Charniak (1997) - in brief

\[ p(\text{somehow I } \text{VP}_{\text{saw}}, \text{Adv}) \]
Charniak (1997) - in brief

\[ p(\text{cat} \mid \text{VP}_{\text{saw}}, \text{NP}) \]
Charniak (1997) - in brief

\[ p(\text{with | } \text{VP}_{\text{saw}}, \text{PP}) \]
Charniak (2000)

• Uses grandparents (Johnson ’98 transformation)
• Markovized children (like Collins)
• Bizarre probability model:
  — Smoothed estimates at many backoff levels
  — Multiply them together
  — “Maximum entropy inspired”
  — Kind of a product of experts (untrained)
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>labeled recall</th>
<th>labeled precision</th>
<th>average crossing brackets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collins</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model 1</td>
<td>87.5</td>
<td>87.7</td>
<td>1.09</td>
</tr>
<tr>
<td>Model 2</td>
<td>88.1</td>
<td>88.3</td>
<td>1.06</td>
</tr>
<tr>
<td>Model 3</td>
<td>88.0</td>
<td>88.3</td>
<td>1.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charniak</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1997</td>
<td>86.7</td>
<td>86.6</td>
<td>1.20</td>
</tr>
<tr>
<td>2000</td>
<td>89.6</td>
<td>89.5</td>
<td>0.88</td>
</tr>
</tbody>
</table>
By now, lexicalization was kind of controversial
– So many probabilities, such expensive parsing: is it necessary?

Goal: reasonable unlexicalized baseline
– What tree transformations make sense?
– Markovization (what order?)
– Add all kinds of information to each node in the treebank

Performance close to Collins model, much better than earlier unlexicalized models
I hit the cats on mats with bats.
Markovization

horizontal: 1
vertical: 1

VP

VP[VB ... PP]

VP[VB ... NP]

VP[VB]

NP

PP

l

hit

the
cats

on

mats

with

bats

VP[VB] → VB

VP[VB ... NP] → VP[VB] NP

VP[VB ... PP] → VP[VB ... NP] PP
Markovization

S

VPs

VPs

VBvp

NPvp

VPs → VBvp NPvp ppvp

horizontal: ∞
vertical: 2

I

hit

the
cats

on

mats

with

bats
Markovization

• More vertical Markovization is better
  – Consistent with Johnson (1998)
• Horizontal 1 or 2 beats 0 or $\infty$
• Used (2, 2), but if sparse “back off” to 1
Other Tree Decorations

• Mark nodes with only 1 child as UNARY
• Mark DTs (determiners), RBs (adverbs) when they are only children
• Annotate POS tags with their parents
• Split IN (prepositions; 6 ways), AUX, CC, %
• NPs: temporal, possessive, base
• VPs annotated with head tag (finite vs. others)
• DOMINATES-V
• RIGHT-RECURSIVE NP
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>labeled recall</th>
<th>labeled precision</th>
<th>average crossing brackets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collins</td>
<td>87.5</td>
<td>87.7</td>
<td>1.09</td>
</tr>
<tr>
<td>Model 1</td>
<td>88.1</td>
<td>88.3</td>
<td>1.06</td>
</tr>
<tr>
<td>Model 2</td>
<td>88.0</td>
<td>88.3</td>
<td>1.05</td>
</tr>
<tr>
<td>Model 3</td>
<td>86.7</td>
<td>86.6</td>
<td>1.20</td>
</tr>
<tr>
<td>Charniak</td>
<td>89.6</td>
<td>89.5</td>
<td>0.88</td>
</tr>
<tr>
<td>1997</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>86.3</td>
<td>85.1</td>
<td>1.31</td>
</tr>
<tr>
<td>K&amp;M</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>