CS388: Natural Language Processing

Lecture 11: Syntax I

Greg Durrett

Some slides adapted from Dan Klein, UC Berkeley

Administivia

- Mini 2 due Tuesday
- Project 1 back tomorrow
- Final project spec posted

Final Project

- Done in pairs or alone
- Compute: allocation on TACC (Maverick2). 4 1080 Ti / 2 V100 / 2 P100 per machine
- Topic: see spec for suggestions
- Proposal due October 15, in-class presentations December 3/5, final report due December 13

This Lecture

- Constituency formalism
- Context-free grammars and the CKY algorithm
- Refining grammars
- Discriminative parsers
Constituency

Syntax

- Study of word order and how words form sentences
- Why do we care about syntax?
  - Multiple interpretations of words (noun or verb?)
  - Recognize verb-argument structures (who is doing what to whom?)
  - Higher level of abstraction beyond words: some languages are SVO, some are VSO, some are SOV, parsing can canonicalize

Constituency Parsing

- Tree-structured syntactic analyses of sentences
- Common things: noun phrases, verb phrases, prepositional phrases
- Bottom layer is POS tags
- Examples will be in English. Constituency makes sense for a lot of languages but not all
Constituency Parsing

The rat the cat chased squeaked

I raced to Indianapolis, unimpeded by traffic

Challenges

- PP attachment

![Diagram showing PP attachment]

same parse as “the cake with some icing”

Constituency

- How do we know what the constituents are?

Constituency tests:
- Substitution by proform (e.g., pronoun)
- Clefting (It was with a spoon that...)
- Answer ellipsis (What did they eat? the cake)
  (How? with a spoon)

- Sometimes constituency is not clear, e.g., coordination: she went to and bought food at the store

Challenges

- NP internal structure: tags + depth of analysis

![Diagram showing NP internal structure]

same parse as “the cake with some icing”
Context-Free Grammars, CKY

- Context-free grammar: symbols which rewrite as one or more symbols

- Lexicon consists of “preterminals” (POS tags) rewriting as terminals (words)

- CFG is a tuple (N, T, S, R): N = nonterminals, T = terminals, S = start symbol (generally a special ROOT symbol), R = rules

- PCFG: probabilities associated with rewrites, normalize by source symbol

### Estimating PCFGs
- Tree $T$ is a series of rule applications $r$. $P(T) = \prod_{r \in T} P(r | \text{parent}(r))$

- Maximum likelihood PCFG: count and normalize! Same as HMMs / Naive Bayes

### Binarization
- To parse efficiently, we need our PCFGs to be at most binary (not CNF)

- Lossless:

  ```
  VP
  /\  \\/
  VBD NP PP PP
  \\/
  NP VP-
  \\/
  PP-
  \\/
  PP-
  
  P(VP → VBD NP PP PP) = 0.2
  ```

- Lossy:

  ```
  VP
  /\  \\
  VBD NP VP-
  \\
  PP-
  
  P(VP → VBZ PP) = 0.1
  ```
Binarization

- Lossless:
  \[
  \begin{align*}
  & \text{VP} \\
  & \text{VBD} \\
  & \text{VP-}[\text{NP PP PP}] \\
  & \text{NP} \\
  & \text{VP-}[\text{PP PP}] \\
  & \text{PP} \\
  & \text{PP}
  \end{align*}
  \]
  
  \[
  P(\text{VP} \rightarrow \text{VBD} \text{ VP-}[\text{NP PP PP}]) = 0.2
  \]
  
  \[
  P(\text{VP-}[\text{NP PP PP}] \rightarrow \text{NP} \text{ VP-}[\text{PP PP}]) = 1.0
  \]
  
  \[
  P(\text{VP-}[\text{PP PP}] \rightarrow \text{PP PP}) = 1.0
  \]
  
  Deterministic symbols make this the same as before

- Lossy:
  \[
  \begin{align*}
  & \text{VP} \\
  & \text{VBD} \\
  & \text{VP} \\
  & \text{NP} \\
  & \text{PP} \\
  & \text{VP} \\
  & \text{PP} \\
  & \text{PP}
  \end{align*}
  \]
  
  \[
  P(\text{VP} \rightarrow \text{VBD}) = 0.2
  \]
  
  \[
  P(\text{VP} \rightarrow \text{NP}) = 0.03
  \]
  
  \[
  P(\text{VP} \rightarrow \text{PP}) = 0.001
  \]
  
  Makes different independent assumptions, not the same PCFG

CKY

- Find argmax \( P(T|x) = \arg\max P(T, x) \)
- Dynamic programming: chart maintains the best way of building symbol \( X \) over span \((i, j)\)
- CKY = Viterbi, there is also an algorithm called inside-outside = forward-backward

\[
\begin{align*}
\text{He} & \quad \text{wrote} \\
\text{a} & \quad \text{long report} \\
\text{on} & \quad \text{Mars}
\end{align*}
\]

Cocke-Kasami-Younger

CKY

- Chart: \( T[i,j,X] = \text{best score} \)
- Base: \( T[i,i+1,X] = \log P(X \rightarrow w_i) \)
- Loop over all split points \( k \), apply rules \( X \rightarrow Y Z \) to build \( X \) in every possible way
- Recurrence:
  \[
  T[i,j,X] = \max_{k} \max_{r} T[i,k,X1] + T[k,j,X2] + \log P(X \rightarrow X1 X2)
  \]
- Runtime: \( O(n^3G) \) \( G = \text{grammar constant} \)

Unary Rules

- Unary productions in treebank need to be dealt with by parsers
- Binary trees over \( n \) words have at most \( n-1 \) nodes, but you can have unlimited numbers of nodes with unaries \( (S \rightarrow \text{SBAR} \rightarrow \text{NP} \rightarrow S \rightarrow ...) \)
- In practice: enforce at most one unary over each span, modify CKY accordingly
**Parser Evaluation**

- **Precision**: number of correct brackets / num pred brackets = 2/3
- **Recall**: number of correct brackets / num of gold brackets = 2/4
- **F1**: harmonic mean of precision and recall = \( \frac{1}{2} * \left( \frac{2}{4} \cdot \frac{1}{3} + \frac{2}{3} \cdot \frac{1}{4} \right) \) = 0.57

**Results**

- Standard dataset for English: Penn Treebank (Marcus et al., 1993)
- Evaluation: F1 over labeled constituents of the sentence
- Vanilla PCFG: ~75 F1
- Best PCFGs for English: ~90 F1
- SOTA (discriminative models): 95 F1
- Other languages: results vary widely depending on annotation + complexity of the grammar

Klein and Manning (2003)

**Refining Generative Grammars**

- Language is not context-free: NPs in different contexts rewrite differently
- Can we make the grammar “less context-free”?

**PCFG Independence Assumptions**

- |    | All NPs | NPs under S | NPs under VP |
- |----|---------|-------------|--------------|
- | NP PP | 11% | 9% | 23% |
- | DT NN | 9% | 9% | 7% |
- | PRP | 6% | 9% | 4% |

- Language is not context-free: NPs in different contexts rewrite differently
- Can we make the grammar “less context-free”?
**Vertical Markovization**

- Basic tree (v = 0)
- V = 1 Markovization

- Why is this a good idea?

**Horizontal Markovization**

- h = 0: VP
- h = 1: VP [... VBZ]
- h = 2: VP [... <S> VBZ]

- Changes amount of context remembered in binarization process

**Tag Splits**

- Can do some other ad hoc tag splits
- Sentential prepositions behave differently from other prepositions

- 75 F1 with basic PCFG => 86.3 F1 with a highly customized PCFG (v = 2, h = 2, other hacks like this)

**Annotated Tree**

- 75 F1 with basic PCFG => 86.3 F1 with this highly customized PCFG (SOTA was 90 F1 at the time, but with more complex methods)

---

*Klein and Manning (2003)*
Lexicalized Parsers

- Even with parent annotation, these trees have the same rules. Need to use the words.

Lexicalized Parsers

- Annotate each grammar symbol with its “head word”: most important word of that constituent.
- Rules for identifying headwords (e.g., the last word of an NP before a preposition is typically the head).
- Collins and Charniak (late 90s): ~89 F1 with these.

Discriminative Parsers

CRF Parsing

He wrote a long report on Mars. He wrote a long report on Mars.
CRF Parsing

\[
\text{score} \left( \begin{array}{c} \text{NP} \\ \text{NP} \\ \text{PP} \end{array} \right) = w^T f \left( \begin{array}{c} \text{NP} \\ \text{NP} \\ \text{PP} \end{array} \right)
\]

He wrote a long report on Mars.

\[ f \left( \begin{array}{c} \text{NP} \\ \text{NP} \\ \text{NP} \end{array} \right) = \begin{array}{c} \text{report} \\ \text{NP} \\ \text{PP} \end{array} \]

Can learn that we report [PP], which is common due to reporting on things.

Can “neuralize” this as well like neural CRFs for NER.

Taskar et al. (2004)
Hall, Durrett, and Klein (2014)
Durrett and Klein (2015)

Joint Discrete and Continuous Parsing

- Chart remains discrete!
- Discrete + Continuous
- Discrete + Continuous...
- Feedforward pass on nets
- Discrete feature computation
- Run CKY dynamic program

Durrett and Klein (ACL 2015)

Parsing with ELMo

<table>
<thead>
<tr>
<th>Encoder Architecture</th>
<th>F1 (dev)</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM (Gaddy et al., 2018)</td>
<td>92.24</td>
<td>-0.43</td>
</tr>
<tr>
<td>Self-attentive (Section 2)</td>
<td>92.67</td>
<td>0.00</td>
</tr>
<tr>
<td>+ Factored (Section 3)</td>
<td>93.15</td>
<td>0.48</td>
</tr>
<tr>
<td>+ CharLSTM (Section 5.1)</td>
<td>93.61</td>
<td>0.94</td>
</tr>
<tr>
<td>+ ELMo (Section 5.2)</td>
<td>95.21</td>
<td>2.54</td>
</tr>
</tbody>
</table>

- Improves the neural CRF by using a transformer layer (self-attentive), character-level modeling, and ELMo

Top-down Parsing

- Greedily predict bracketing at next stage of the tree. Like a neural CRF but with no dynamic program (CKY) pass

Kitaev and Klein (2018)
Stern et al. (2017)
Takeaways

- PCFGs estimated generatively can perform well if sufficiently engineered
- Neural CRFs work well for constituency parsing
- Next time: revisit lexicalized parsing as *dependency parsing*