Recall: CNNs vs. LSTMs

- Both LSTMs and convolutional layers transform the input using context.
- LSTM: “globally” looks at the entire sentence (but local for many problems).
- CNN: local depending on filter width + number of layers.

Recall: CNNs

- $P(y|x)$
- Projection + softmax
- c-dimensional vector
  - Max pooling over the sentence
  - $O(n) \times c$
  - c filters, $m \times k$ each
  - $O(n \times k)$

- The movie was good

Recall: Neural CRFs

1) Compute $f(x)$
2) Run forward-backward
3) Compute error signal
4) Backprop (no knowledge of sequential structure required)

Barack Obama will travel to Hangzhou today for the G20 meeting.
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? *Fed raises*... example)
  - 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

same parse as “the cake with some icing”

Challenges

- NP internal structure: tags + depth of analysis
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

Context-Free Grammars, CKY

CFGs and PCFGs

- Grammar (CFG)
- Lexicon

<table>
<thead>
<tr>
<th>Grammar (CFG)</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT → S</td>
<td>1.0</td>
</tr>
<tr>
<td>S → NP VP</td>
<td>1.0</td>
</tr>
<tr>
<td>NP → DT NN</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → NN NNS</td>
<td>0.5</td>
</tr>
<tr>
<td>VP → VBP NP</td>
<td>0.3</td>
</tr>
<tr>
<td>PP → IN NP</td>
<td>1.0</td>
</tr>
<tr>
<td>NN → interest</td>
<td>1.0</td>
</tr>
<tr>
<td>NNS → raises</td>
<td>1.0</td>
</tr>
<tr>
<td>VBP → interest</td>
<td>1.0</td>
</tr>
<tr>
<td>VBJ → raises</td>
<td>1.0</td>
</tr>
</tbody>
</table>

- 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 $r$. $P(T) = \prod_{r \in T} P(r \vert \text{parent}(r))$

- Maximum likelihood PCFG: count and normalize! Same as HMMs / Naive Bayes
To parse efficiently, we need our PCFGs to be at most binary (not CNF)

\[
P(VP \rightarrow VBD NP PP) = 0.2
\]
\[
P(VP \rightarrow VBZ PP) = 0.1
\]

Lossless:

```
VP
  /\  \\
VBD PP
  /\  \\
NP PP
```

Lossy:

```
VP
  /\   /
VBD VP-[NP PP PP]
      /
NP PP
```

Pseudo-Chomsky Normal Form

\[
P(VP \rightarrow VBD VP-[NP PP PP]) = 0.2
\]
\[
P(VP-[NP PP PP] \rightarrow NP VP-[PP PP]) = 1.0
\]

Deterministic symbols make this the same as before

Lossless:

```
VP
  /\  \\
VBD VP
    /
NP PP
```

Lossy:

```
VP
  /\  \\
VBD VP
    /
NP PP
```

P(VP \rightarrow VBD VP) = 0.2
P(VP \rightarrow NP VP) = 0.03
P(VP \rightarrow PP PP) = 0.001

In practice: enforce at most one unary over each span, modify CKY accordingly
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: 95 F1
- Other languages: results vary widely depending on annotation + complexity of the grammar

Klein and Manning (2003)

Refining Generative Grammars

PCFG Independence Assumptions

- Like a trigram HMM tagger, incorporates more context
- Vertical (parent) annotation: add the parent symbol to each node, can do grandparents too
- Horizontal annotation: remember the states of multi-arity rules during binarization

Rule Annotation
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)

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

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
CRF Parsing

He wrote a long report on Mars.

My report—on Mars

Can learn that we report [PP], which is common due to reporting on things
Can “neuralize” this as well like neural CRFs for NER

Joint Discrete and Continuous Parsing

Chart remains discrete!

Parsing a sentence:
- Feedforward pass on nets
- Discrete feature computation
- Run CKY dynamic program

Neural CRF Parsing

Simpler version: score constituents rather than rule applications

Use BiLSTMs (Stern) or self-attention (Kitaev) to compute span embeddings
91-93 F1, 95 F1 with ELMo (SOTA). Great on other langs too!
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*

Survey

- Write one thing you like about the class
- Write one thing you don’t like about the class