Lecture 16: Syntax I

Recap: POS Tagging

- Layer of shallow syntactic analysis

- One way to model it: Hidden Markov Models, generative models of $P(y, x)$ from which we compute the posterior $P(y \mid x)$ (+ use Viterbi to max)

- Can also use conditional random fields (discriminative) or even neural CRFs — better for tasks like named entity recognition

This Lecture

- Constituency formalism
- Context-free grammars and the CKY algorithm
- Refining grammars
- Dependency grammar
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

- 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](image-url)
Constituency Parsing

A refund that the court estimated

Challenges

- PP attachment

```
NP          VP          PP
DT  NNS VBD VPD DT NNS IN DT NN
The children ate the cake with a spoon
```

same parse as “the cake with some icing”

Challenges: NP Internal Structure

```
NP
NN NN NN
plastic cup holder
```

```
NP
NN NN
plastic cup
```

What is a plastic cup holder?

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

- 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 for a set of labeled trees: 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
  - $P(VP \rightarrow \text{VP-DP}) = 0.2$

- Lossy:

- VP
  - $P(VP \rightarrow \text{VP-DP}) = 0.1$

- $P(\text{VP-DP} \rightarrow \text{VP-DP}) = ...$

- $P(\text{NP} \rightarrow \text{VP-DP}) = ...$

**Grammar (CFG) | Lexicon**

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT $\rightarrow$ S</td>
<td>1.0</td>
</tr>
<tr>
<td>S $\rightarrow$ NP VP</td>
<td>1.0</td>
</tr>
<tr>
<td>NP $\rightarrow$ DT NN</td>
<td>0.2</td>
</tr>
<tr>
<td>NP $\rightarrow$ NN NNS</td>
<td>0.5</td>
</tr>
</tbody>
</table>
- Find $\arg\max P(T|x) = \arg\max P(T, x)$

- 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

**CKY**

Chart:

$$T[i,j,X] = \text{best score for } X \text{ over } (i, j)$$

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:

$$T[i,j,X] = \max_{k: x \rightarrow x_1 x_2} \max_{k: x \rightarrow x_1 x_2} T[i,k,X_1] + T[k,j,X_2] + \log P(X \rightarrow X_1 X_2)$$

Runtime: $O(n^3 G)$ $G = \text{grammar constant}$

**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 ($S \rightarrow SBAR \rightarrow NP \rightarrow S \rightarrow ...$)

- In practice: enforce at most one unary over each span, modify CKY accordingly

**CKY Example**

```
the    child    raises    it
DT -> the    1  VBP -> raises    1  S -> NP VP    1
NN -> child    1  PRP -> it    1  NP -> DT NN    1/2  VP -> VBP PRP    1
NNS -> raises    1  NP -> NN NNS    1/2

Recurrence:
$$T[i,j,X] = \max_{k: x \rightarrow x_1 x_2} \max_{k: x \rightarrow x_1 x_2} T[i,k,X_1] + T[k,j,X_2] + \log P(X \rightarrow X_1 X_2)$$
```
Parser Evaluation

She saw it

NP  PRP  NN  PRP
  0    1    2    3

S(0,3), NP(0,2), NP(2,3), PRP(0,1), NN(1,2), PRP(2,3)

S(0,3), NP(0,1), VP(1,3), NP(2,3), PRP(0,1), VBD(1,2), PRP(2,3)

- 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 = (1/2 * ((2/4)^-1 + (2/3)^-1))^-1
  = 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

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  DT  NN  PRP  NP  PP  DT  NN  PRP  NP  PP  DT  NN  PRP
11%  9%  6%  9%  9%  21%  23%  7%  4%
Vertical Markovization

```
S
  NP   VP
    PRP  VBD PRP
She saw it
```

- Why is this a good idea?

```
S^ROOT
  NP^S   VP^S
    PRP^NP VBD^VP PRP^VP
She saw it
```

v = 1 Markovization

Horizontal Markovization

```
VP
  VP
    sold books to her for $50
```

- Changes amount of context remembered in binarization process

```
VP
  VP
    sold books to her for $50
```

h = 0: VP
h = 1: VP [... VBZ]

```
VP [... VBZ]
```

h = 1: VP [... NP]

```
VP [... NP]
```

h = 2: VP [... VBZ NP]

Annotated Tree

```
ROOT
  S^ROOT-v
  NP^S-B
    DT-U\'NP
      This
  VP^S-VBF-v
    is
      NN\'NP
        panic
        buying
```

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

Klein and Manning (2003)

Lexicalized Parsers

```
NP
  CC
    NP
      NNS
        dogs
        in
        NNS
        houses
      and
      NNS
        cats
```

- 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

CRF Parsing

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

- 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!

He wrote a long report on Mars.

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

Durrett and Klein (ACL 2015)
Pre-trained Models

- Improves the neural CRF by using a transformer layer (self-attentive), character-level modeling, and ELMo
- 95.21 on Penn Treebank dev set — much better than past parsers! (~92-93)
- This constituency parser with BERT is one of the strongest today, or use a transition-based version due to Kitaev and Klein (2020)

Dependentsyntactic structure is defined by these arcs
- Head (parent, governor) connected to dependent (child, modifier)
- Each word has exactly one parent except for the ROOT symbol, dependencies must form a directed acyclic graph
- POS tags same as before, usually run a tagger first as preprocessing
Dependency Parsing

- Still a notion of hierarchy! Subtrees often align with constituents

```
DT the
NN dog
TO to
NN house
VBD ran
```

- Can label dependencies according to syntactic function

```
DT the
NN dog
VBD ran
TO to
DT the
NN house
```

Major source of ambiguity is in the structure, so we focus on that more (labeling separately with a classifier works pretty well)

Dependency vs. Constituency: PP Attachment

- Constituency: several rule productions need to change

```
NP The children
VP ate
NP the cake
IN with
PP a spoon
```

- Dependency: one word (with) assigned a different parent

```
the children ate the cake with a spoon
```

- More predicate-argument focused view of syntax

```
“What’s the main verb of the sentence? What is its subject and object?”
— easier to answer under dependency parsing
```
**Dependency vs. Constituency: Coordination**

- **Constituency:** ternary rule NP -> NP CC NP

  ![Diagram showing ternary rule NP -> NP CC NP]

- **Dependency:** first item is the head

  ![Diagram showing dependency with first item as head]

- Coordination is decomposed across a few arcs as opposed to being a single rule production as in constituency
- Can also choose *and* to be the head
- In both cases, headword doesn’t really represent the phrase — constituency representation makes more sense

**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*