Treebanks

- Treebanks are corpora in which each sentence has been paired with a parse tree
- Hopefully the right one!
- Encodes a particular grammatical framework
- These are generally created:
  - By first parsing the collection with an automatic parser
  - And then having human annotators correct each parse as necessary
- But...
  - Detailed annotation guidelines are needed
  - Explicit instructions for dealing with particular constructions
  - Difficult, but essential, to ensure consistency
  - Starting point for a data-driven approach

Penn Treebank

- Penn TreeBank is a widely used treebank
- 1 million words from the Wall Street Journal
- "Least-common denominator" syntactic annotation, i.e. relatively theory-neutral
- Treebanks implicitly define a grammar for the language
Penn WSJ Non-Terminals (NTs)

- Basic non-terminal tagset (not including pre-terminals)
- Other "function" tags may label constituents, e.g. PP-TMP means temporal PP
- Raw treebank contains empty categories

Why treebanks?

- Treebanks are critical to training statistical parsers
- Also valuable to linguist when investigating phenomena

Grammar Induction

- Extract context-free rules from trees in the treebank
- Context-free rules of the form:
  \[ A \rightarrow B C D E \]
  - where A is the (one and only) 'parent'
  - and B, C, D, and E are the 'children'
  - also refer to left-hand side (LHS): A
  - and right-hand side (RHS): B C D E

CFG Induction

- Local tree: Parent (S), children (NP VP)
- Each local tree represents a context-free rule:
  \[ S \rightarrow NP VP \]

Interpretations of a CFG rule

- For a rule such as \( S \rightarrow NP VP \), there are various interpretations of what this means
- Derivations:
  - An NP and a VP can combine (or compose) to produce an S
  - An S can be split into an NP followed by a VP
- Trees:
  - An S node can generate an NP and a VP node
  - An S node can be the parent of an NP and a VP node

Derivations

- If we have a rule \( A \rightarrow \alpha \), then define a derives relation: \( \beta \vdash \gamma \)
- A string \( w_1, \ldots, w_n \) is in the language of a CFG \( G \) if \( S^+ \Rightarrow w_1, \ldots, w_n \)
- For example, consider these noun compounding rules:
  1. \( N \rightarrow N N \) \( (i) \) \( N \rightarrow \) dog
  2. \( N \rightarrow N N \) \( (ii) \) \( N \rightarrow \) food
  3. \( N \rightarrow N N \) \( (iii) \) \( N \rightarrow \) can
  4. \( N \rightarrow N N \) \( (iv) \) \( N \rightarrow \) food can
- There are many possible derivations, s.t. \( N \Rightarrow \text{dog food can} \)
  1. \( N \Rightarrow N N \Rightarrow N \Rightarrow \text{food can} \Rightarrow \text{dog food can} \)
  2. \( N \Rightarrow N N \Rightarrow N \Rightarrow N \Rightarrow \text{food can} \Rightarrow \text{dog food can} \)
  3. \( N \Rightarrow N N \Rightarrow N N \Rightarrow \text{dog food can} \Rightarrow \text{dog food can} \)
  4. \( N \Rightarrow N N \Rightarrow \text{dog can} \Rightarrow \text{dog food can} \Rightarrow \text{dog food can} \)
- Derivation 1. is the rightmost derivation, always expanding the rightmost non-terminal; derivation 4. is a leftmost derivation
Pushdown automata

- Consider the leftmost derivation:
  \[ N \rightarrow N \rightarrow \text{dog} N \rightarrow \text{dog} N \rightarrow \text{dog food} N \rightarrow \text{dog food can} \]
- We can represent this as an automaton, with a stack at each state:

  ![Pushdown Automaton Diagram]
- Generally cannot be represented with finite-state automaton

Parse Tree, Derivation

- We can represent this as an automaton, with a stack at each state:

  ![Parse Tree Diagram]

Labeled Bracketing

- Another representation of the same tree:
  \[ (S (NP (PRP \text{we})) (VP (VBD \text{helped}) (NP (PRP \text{her})) (VP (VB \text{paint}) (NP (DT \text{the}) (NN \text{house}))))))) \]
- Some terminology (review):
  - Terminals are words.
  - Penn Treebank non-terminal set has 2 disjoint subsets:
    - Pre-terminal (POS) tags rewrite to exactly 1 word.
    - The rest never have terminals as children.

Parse Tree, of speech

- From Switchboard Corpus

Probabilistic CFGs (PCFGs)

- A PCFG is a CFG with a probability assigned to each rule:
  \[ P(S \rightarrow NP \ VP) = P(\text{we} = \{NP \ VP\} | \text{IHS} = S) = P(\text{NP} \ VP | S) \]
- Joint probability of the right-hand side (RHS) can be decomposed using the chain rule:
  \[ P(S \rightarrow NP \ VP) = P(\text{NP} | S) \cdot P(\text{VP} | S, \text{NP}) = P(\text{IHS} | S, \text{NP} \ VP) \]
  where `<r>` is an "end-of-rule" symbol
- Standard PCFG induction approach
  - Count the number of times rules (local trees) occur
  - Use relative frequency estimation for conditional probabilities

CFG Equivalence

- Two CFGs G and G’ are strongly equivalent if they describe the same language, and they produce identical trees for strings, modulo node labels.
- Two CFGs G and G’ are weakly equivalent if they describe the same language.
- Sometimes a grammar G can be transformed to a weakly equivalent grammar G’ that has some beneficial computational properties.
Normal Forms

- Chomsky Normal Form (CNF)
  - A grammar $G = (V, T, P, S)$ is in CNF if all productions in $P$ are in one of two forms:
    - $A \rightarrow BC$ where $A, B, C \in V$ or
    - $A \rightarrow a$ where $A \in V$ and $a \in T$

- Greibach Normal Form (GNF)
  - A grammar $G = (V, T, P, S)$ is in GNF if all productions in $P$ are of the following form:
    - $A \rightarrow aX$ where $A \in V$, $a \in T$, and $X \in V^*$

Every CFG $G$ has weakly equivalent CFGs in CNF or GNF
- Chomsky Normal Form very useful for chart parsing

Penn Treebank CNF

- Disjoint pre-terminal set, so all POS $\rightarrow$ word productions already in CNF
- Left or right factorization removes productions with $> 2$ RHS categories
- Remaining issues:
  - Remove empty categories (0 categories on RHS)
  - Collapse unary productions (1 non-terminal on RHS)

PCFG Induction and Factorization

- Original CFG rules:
  \[
  \hat{p}(A \rightarrow \alpha) = \frac{c(A \rightarrow \alpha)}{\sum_{A \rightarrow \beta \in P} c(A \rightarrow \beta)}
  \]
- Left factorization:
  \[
  \hat{p}(A \rightarrow AB \rightarrow \beta) = \frac{\sum_{A \rightarrow \alpha \in P} c(A \rightarrow \alpha) c(A \rightarrow \beta)}{\sum_{A \rightarrow \alpha \in P} c(A \rightarrow \beta)}
  \]

Sparsity

- We may observe in our corpus the following rule:
  NP $\rightarrow$ DT JJ NN NN NNS
- We may not observe:
  NP $\rightarrow$ DT JJ JJ NN NN NNS
- Does this mean that the second rule should have zero probability?
- A “Markov” grammar is a factored grammar that provides probability mass to unobserved rules
Left Factorization & “Markov” Grammars

- Take a rule from the grammar such as
  \[ NP \rightarrow DT \ JJ \ NN \ NNS \]
- Left factorization:
  \[ NP \rightarrow DT \ NP-DT \]
  \[ NP-DT \rightarrow JJ \ NP-DT, JJ \]
  \[ NP-DT, JJ \rightarrow NN \ NNS \]
- Markov grammar, order 1:
  \[ NP \rightarrow DT \ NP-DT \]
  \[ NP-DT \rightarrow JJ \ NP-JJ \]
  \[ NP-JJ \rightarrow NN \ NP-NN \]
  \[ NP-NN \rightarrow NN \]

Agenda: Summary

- Questions, comments, concerns?
- Context-Free Grammars
  - Treebanks
  - Inducing CFGs from trees
  - Probabilistic CFGs
- Next week: parsing algorithms