Parse Reranking

- Assume the number of parses is very small
- We can represent each parse \( T \) as an arbitrary feature vector \( \phi(T) \)
  - Typically, all local rules are features
  - Also non-local features, like how right-branching the overall tree is
    - [Charniak and Johnson 05] gives a rich set of features

Inside and Outside Scores

K-Best Parsing

- Lexicalized parsers can be seen as producing dependency trees
- Each local binary tree corresponds to an attachment in the dependency graph
Dependency Parsing

- Pure dependency parsing is only cubic [Eisner 99]

- Some work on non-projective dependencies
  - Common in, e.g. Czech parsing
  - Can do with MST algorithms [McDonald and Pereira 05]

Shift-Reduce Parsers

- Another way to derive a tree:

Data-oriented parsing:

- Rewrite large (possibly lexicalized) subtrees in a single step

- Formally, a tree-insertion grammar
- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable parse is NP-complete

TIG: Insertion

- Start with local trees
- Can insert structure with adjunction operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- as well as other weird stuff that CFGs don’t capture well (e.g. cross-serial dependencies)
CCG Parsing

- Combinatory Categorial Grammar
  - Fully (mono-) lexicalized grammar
  - Categories encode argument sequences
  - Very closely related to the lambda calculus (more later)
  - Can have spurious ambiguities (why?)

  \[
  \begin{align*}
  John & \rightarrow \text{NP} \\
  shares & \rightarrow \text{NP} \\
  buys & \rightarrow (S\backslash\text{NP})/\text{NP} \\
  sleeps & \rightarrow S/\text{NP} \\
  well & \rightarrow (S\backslash\text{NP})\backslash(S\backslash\text{NP}) \\
  \end{align*}
  \]

Syntax-Based MT

- synchronous context-free grammars (SCFGs)
- context-free grammar in two dimensions
- generating pairs of strings/trees simultaneously
- co-indexed nonterminal further rewritten as a unit

  \[
  \begin{align*}
  \text{VP} & \rightarrow \text{pp}^{(1)} \text{vp}^{(2)}, \\
  \text{VP} & \rightarrow \text{pp}^{(1)} \text{pp}^{(2)}, \\
  \text{PP} & \rightarrow \text{pp}^{(1)} \text{pp}^{(2)} \\
  \text{PP} & \rightarrow \text{vp}^{(1)} \text{vp}^{(2)}, \\
  \text{PP} & \rightarrow \text{vp}^{(1)} \text{pp}^{(2)} \\
  \text{PP} & \rightarrow \text{pp}^{(1)} \text{vp}^{(2)} \\
  \text{PP} & \rightarrow \text{pp}^{(2)} \text{vp}^{(1)} \\
  \end{align*}
  \]

Translation by Parsing

- translation with SCFGs => monolingual parsing
- parse the source input with the source projection
- build the corresponding target sub-strings in parallel

Translation by Parsing

- translation with SCFGs => monolingual parsing
- parse the source input with the source projection
- build the corresponding target sub-strings in parallel

Extracting syntactic rules

- syntax-directed, English to Chinese (Huang, Knight, Joch, 2006)
- first parse input, and then recursively transfer synchronous tree-substitution grammars (STSG) (Galley et al., 2004; Eber, 2003)
Rules can...

- capture phrasal translation
- reorder parts of the tree
- traverse the tree without reordering
- insert (and delete) words

Bad alignments make bad rules

This isn't very good, but let's look at a worse example...

Sometimes they're really bad

One bad link makes a totally unusable rule!