1 Remotivation

- Weak generative capacity: Grammar’s ability to decide whether or not a sentence is English
  - Grammar checking
  - Language modeling for speech recognition, machine translation

- Strong generative capacity: Grammar’s ability to assign structural descriptions to grammatical English sentences
  - Interface to semantic analysis
  - Structural description can serve as a poor man’s semantics in machine translation, question answering, lexical semantics, etc.

2 Long-distance dependencies

Raising

- Some more examples involving what are traditionally called *raising* verbs:
  1. The dog seems to bite the man.
  2. The dog seems to exist.

- A CFG analysis
  3. $\text{VP} \rightarrow \text{V[raising]} \rightarrow \text{VP[inf]}$
  4. $\text{V[raising]} \rightarrow \text{seems}$
  5. $\text{VP[inf]} \rightarrow \text{V[inf}, \text{subcat} = \langle \text{NP} \rangle] \text{ NP}$ etc.
  6. $\text{V[inf]} \rightarrow \text{bite}$ etc.

- Recall our CFG analysis of expletive ‘there’ from last time. How does it interact with ‘seems’?
  7. There seems to exist a dog.
  8. *The dog seems to exist a cat.
  9. *There seems to exist.
Historical note

- Traditionally analyzed using movement of ‘there’ from one position to another

(10)

- Deletion and insertion were also part of the system. But Peters and Ritchie (1973) showed that the transformational grammar of their day could generate sets that are recursively enumerable but not decidable!

- Stabler has formalized the most recent version of transformational grammar in a polynomial-time parsable framework

• CFG solution: make the raising verb “transmit” the there feature

(11) 

\[ \text{VP}[\text{there}] \rightarrow \text{V}[\text{raising}] \text{ to VP}[\text{there}, \text{inf}] \]

• Something similar happens with idioms:

(12) The cat seems to be out of the bag.

Relative clauses

• How about:

(13) The man who bit the dog exists.

(14) The man whom the dog bit exists.

(15) *The man who the cat bit the dog exists.

(16) *The man whom the dog bit the cat exists.

• First attempt:

(17) \[ \text{NP} \rightarrow \text{NP} \; S'[\text{relative}] \]

(18) \[ S'[\text{relative}] \rightarrow \text{who}\; S \]

But this doesn’t explain the facts right.
- Refinement: make the clause “transmit” information about the relativization

\[(19)\] \[S'[\text{relative}] \rightarrow \text{who} S[\text{gap} = \text{who}]\]

\[(20)\] \[S[\text{gap} = \text{who}] \rightarrow \text{VP}\] note missing subject

\[(21)\] \[S'[\text{relative}] \rightarrow \text{whom} S[\text{gap} = \text{whom}]\]

\[(22)\] \[S[\text{gap} = \text{whom}] \rightarrow \text{NP} \text{VP}[\text{gap} = \text{whom}]\]

\[(23)\] \[\text{VP}[\text{gap} = \text{whom}] \rightarrow \text{V[\text{subcat} = \langle \text{NP} \rangle]}\] note missing object

- Exercises:

1. Write a CFG to generate the sentences

\[(24)\] The man whom a man hired hired another man.

\[(25)\] The man (whom a man)\(^n\) hired\(^n\) hired another man.

Write the derivation (as a sequence of rewrites or as a tree) for \(n = 2\). Explain briefly why we couldn’t do this with a finite-state machine.

2. Write a CFG to generate the sentences

\[(26)\] The man who John said bit the dog exists.

\[(27)\] The man who John said Mary said bit the dog exists.

\[(28)\] The man who (John said Mary said)\(^n\) bit the dog exists.

as well as

\[(29)\] The man whom (John said Mary said)\(^n\) the dog bit exists.

Your analysis must be in the spirit of the analysis given above. It’s okay to generate other sentences as long as they’re good English sentences. Write the derivation for (28) for \(n = 2\).

Cross-serial dependencies

- Now we move on to Dutch!

\[(30)\] . . .dat Jan Piet de kinderen zag helpen zwemmen

. . .that John Peter the children saw help swim

‘. . .that John saw Peter help the children swim.’

- Can’t be generated by a CFG with the right tree structure

- Swiss-German version is more airtight: set of strings that cannot be generated by a CFG with any tree structure

3 Tree-adjoining grammar

Definition

- Terminal alphabet

- Nonterminal alphabet
• Initial trees: leaf nodes can have nonterminal labels, in which case they are called *substitution nodes.*

• Auxiliary trees: one leaf node has the same label as the root node and is called the *foot node.*

A derivation starts with an initial tree and proceeds by a series of rewriting operations, which can be either of:

• *substitution:* a substitution node labeled $X$ is rewritten using an initial tree whose root label is $X$

• *adjunction:* a node labeled $X$ is rewritten using an auxiliary tree whose root/foot label is $X$

Long-distance dependencies

• Adjunction stretches trees, obviating the need to “transmit” features

• Raising:

\[(31)\]

\[
\begin{aligned}
S & \quad S & \quad VP \\
NP \downarrow & \quad NP & \quad VP \\
to & \quad to & \quad seems \\
VP & \quad VP & \quad NP \downarrow \\
exist & \quad exist & \quad a \text{ dog}
\end{aligned}
\]

Preventing overgeneration will take a bit more work

• Relative clauses: left as an exercise

3. Do exercise 2 using a tree-adjoining grammar. You shouldn’t need to use any features, just the basic categories $S, S', NP, VP, V.$
Cross-serial dependencies

- A TAG for the Dutch example (treat the $t$ symbols as empty strings):

(32)

- Result of adjoining $\beta_1$ into $\alpha$:

(33)
• Result of adjoining $\beta_2$ into that:

(34)