CS 6740/IS 6300

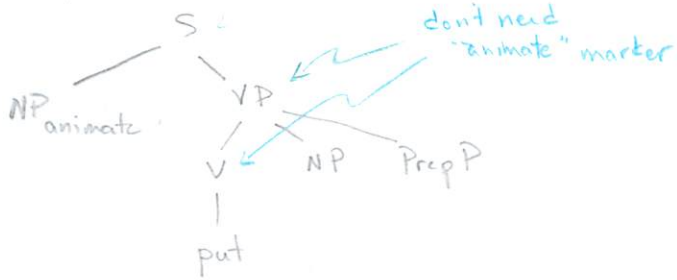
Lecture 5: Tree-adjoining grammars

1. Recall: every CFG is a tree-substitution grammar is a tree-adjoining grammar, but not the other way around

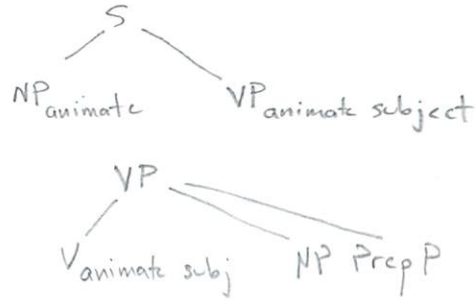
TSGs vs. CFGs:

- "more elegant" handling of long-distance dependencies

1(a) TSG tree:



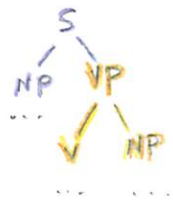
1(b) CFG rules:



- derivation tree gives better account of sentence semantics

2. Modification:

2(a) derived tree

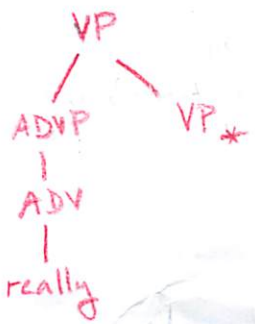


2(b) derived tree

→ in a TSG, need two trees for everything that can be optionally modified.

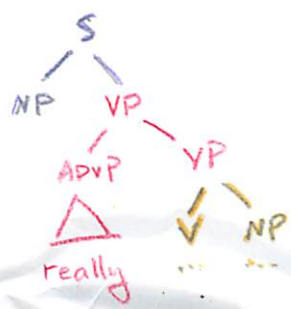
3. Auxiliary tree: exactly one of the non-root leaves whose label matches the root's is distinguished as the foot (conventionally marked with a *)

3(a)



The tree 2(b) looks like 2(a) had 3(a) "spliced in":

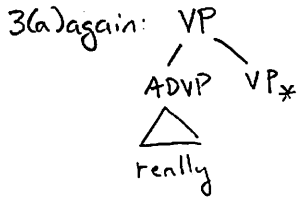
3(b)



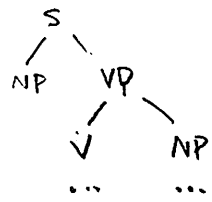
4. <see #5 on the handout from last time, except: replace "into an initial tree α" with "into a tree γ".

Exercise: what derived tree do you get by doing this?

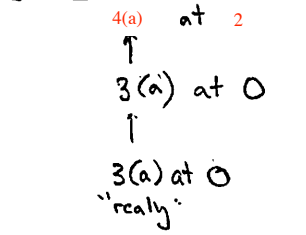
3(a) again:



4(a):



so derivation tree:



---

5. Notes re: Assignment 1

a. You don't need to worry about features

b. NLTK (python NLP toolkit, link on course homepage) OK for tree representations

c. Make simple test cases and consider how to handle them

d. Handle substitution first, maybe, before considering adjunction? Or, will it be easier to handle adjunction first, and then make "backwards compatible" with substitution?

e. You can make simplifications, but document them!