

GUIE

Grammar-based Universal Imitation Engine

Sam Wintermute, EECS 595, Fall 2005

Background / Motivation

- Lots of very domain-specific text imitation generators exist
 - SCIGen for comp. sci. papers
 - Postmodern paper generator
- But nothing general-purpose
- How realistic can we make generated text if the source corpus is arbitrary?

Project Overview

- Given a parsed document, try to derive a context-free grammar that can imitate it.
- Do this without plagiarizing from the corpus.
- Try to combine grammars from diverse corpora in interesting ways.
 - Use the syntax from one document, the words from another.
- Try and keep sentences as grammatical as possible
 - Not even worrying about meaning, this is very hard, especially given the inaccuracy of the parser.
 - Parsed output has no information on verb tense, number, etc

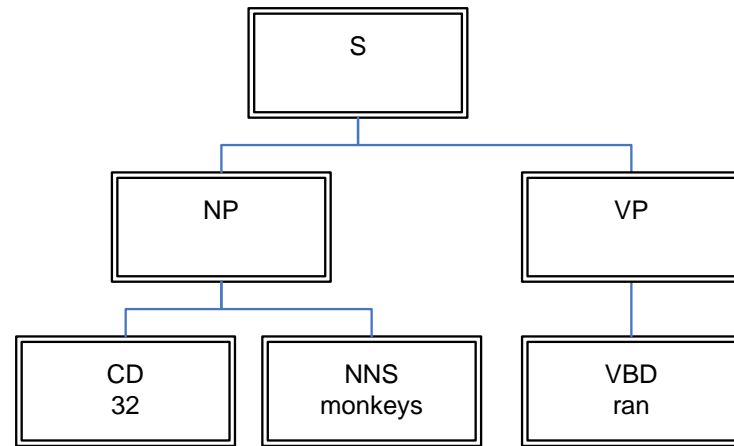
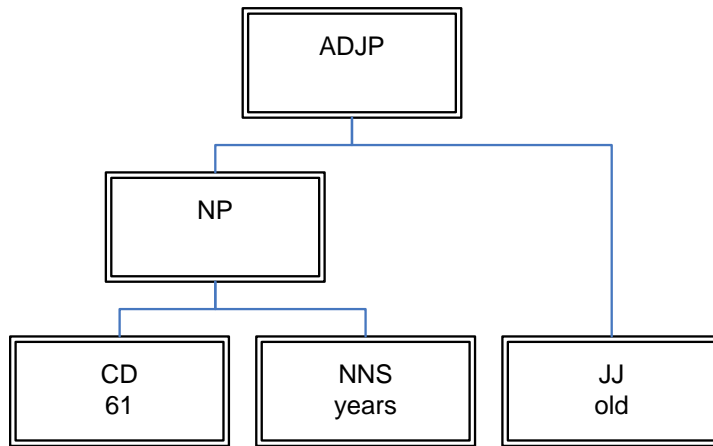
Two basic approaches

- A. Simply take the parse trees from the Charniak parser output, and derive a probabilistic CFG.
 - High randomness, low grammaticality, very low chance of copying from the corpus.
- B. Just allow sentences with the same parse tree to exchange words
 - "The man walked to the store."
 - + "A dog ran in the park."
 - = "A man ran in the store."
 - Low randomness, high grammaticality, very high chance of copying from the corpus
 - Any time a parse is unique, it can only be copied

Combining the approaches

- Approach B (from above) can also be implemented like a CFG
 - If every constituent name has all of the rules above it in the tree encoded in it, that is what will happen
 - Instead of $S \rightarrow NP VP$, have $S \rightarrow NP_0 VP_0$, where the 0 means “originated from $S \rightarrow NP VP$ ”
 - Essentially, for every constituent, there is only one possible way for it to be generated
- We want to do this not for the whole tree, but just the parts of it that are common
 - In other words, try to ensure that the CFG will not have paths where each node only has one possible expansion
- In general, look through the parsed corpus, and if any rule occurs more than once, make its children unique from all other nodes
 - Tack a number on the end
- Doing this for multiple passes will result in a grammar that is very specific, but doesn't get stuck in ruts (copying the source file).

Example- initial parse



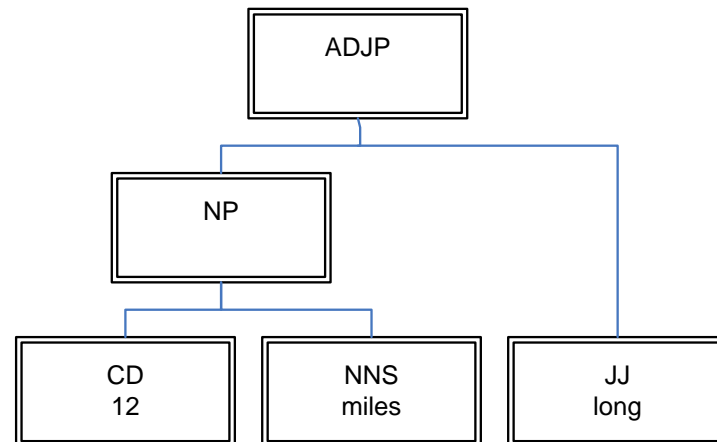
Rules:

S -> NP VP

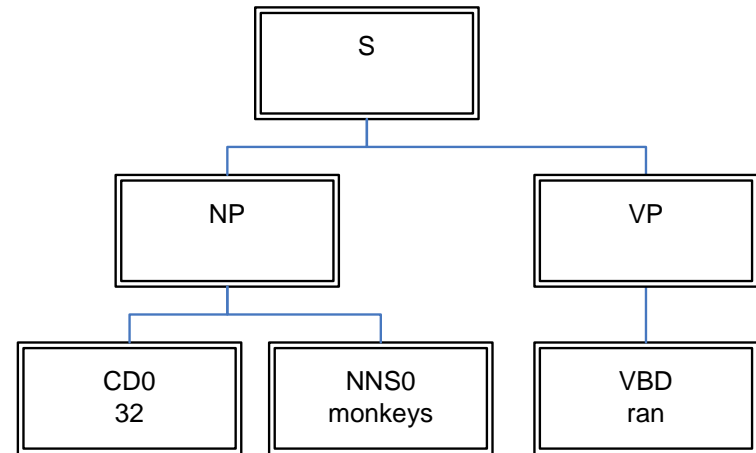
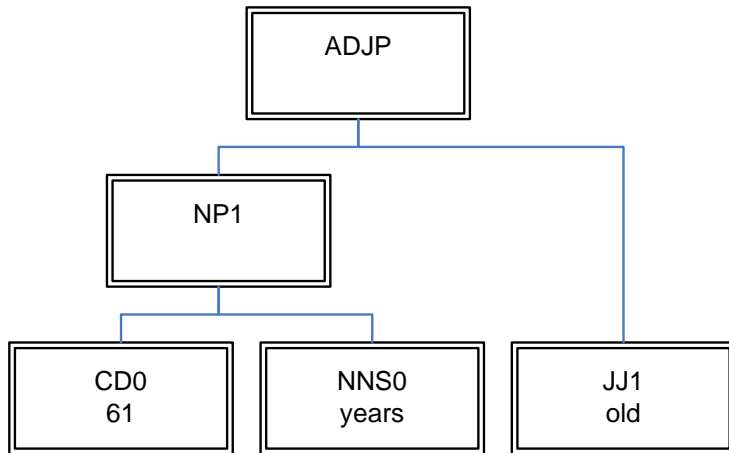
ADJP -> NP JJ (*2)

NP -> CD NNS (*3)

VP -> VBD



Example- after first pass



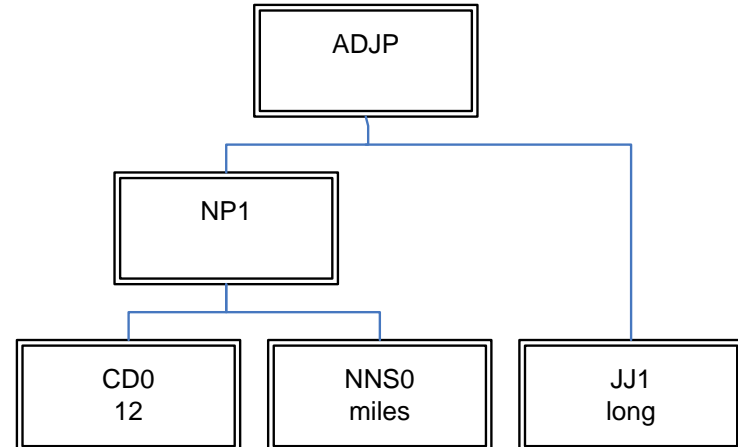
Rules:

S -> NP VP

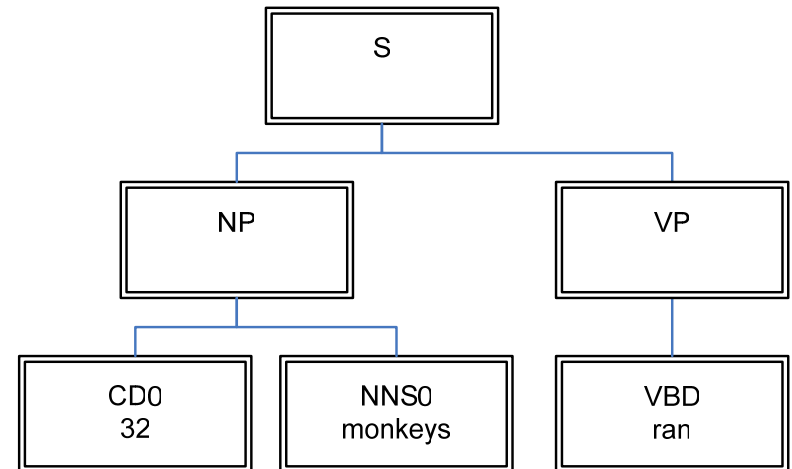
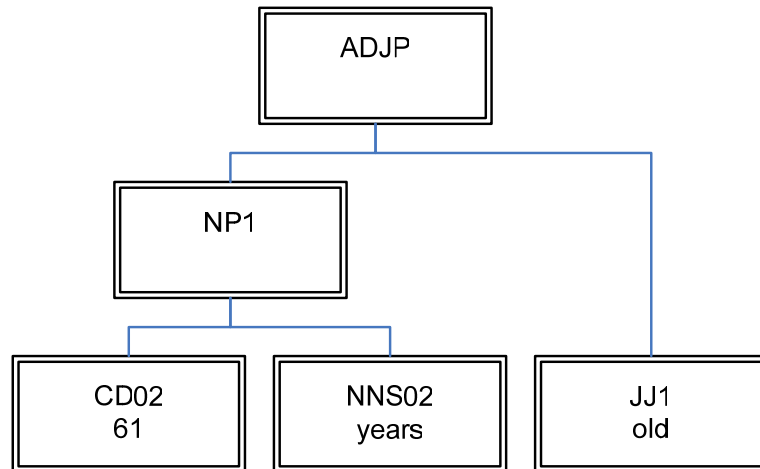
ADJP -> NP1 JJ1 (*2)

NP -> CD0 NNS0 (*3)

VP -> VBD



Example- after second pass



Rules:

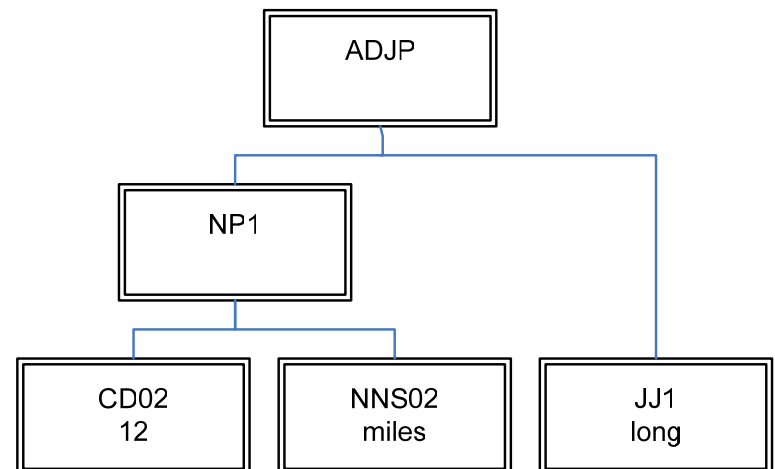
S -> NP VP

ADJP -> NP1 JJ1 (*2)

NP -> CD0 NNS0

NP1 -> CD02 NNS02 (*2)

VP -> VBD



Results of example

- We now have an NP type that is specific to adjective phrases
- We now have CD and NNS types that are specific for the NP inside an ADJP
- Do this enough times, and we can get some very interesting types:
 - Example: a VBD type in a parse of the Wall Street Journal (“child:weight” format)
said:38 added:5 asserted:1 found:1
announced:1 noted:1 reported:1
explained:1

Result of this approach

- 10 random sentences with the same structure:
 - Who are wrong of asking at more credit?
 - Who is wrong of leading for easier credit?
 - Who is wrong with asking about easier money?
 - What was wrong with focusing at easier money?
 - Who's wrong of asking at easier credit?
 - What is dreadful with asking at more credit?
 - Who is peculiar with coming at more money?
 - What was dreadful with asking about easier money?
 - Who is wrong with leading on easier credit?
 - Who was peculiar of leading for easier money?
- Notes:
 - “credit” and “money” are seen as the same type
 - “who” and “what” are seen as the same type
 - Likewise with “dreadful”, “wrong”, and “peculiar”
 - Others not as good: asking == leading == coming == focusing?

Adding a Language Model

- To help improve more, partial sentences are evaluated against a language model as they are generated
- Ideally, this would cause “What is wrong with asking for more money?” to be generated above
- Usually doesn't work quite that well..
 - Slightly improves things
 - At a huge computational cost, though

Merging two documents

- Goal: Get the syntax style from document A, and the words from document B.
- Hard to combine this with the above strategy
 - Rules tend to become unique to their source corpus
- Must restrict what can be made unique, to the detriment of quality
- Tweaking the weights in the CFG does most of the work

Conclusions / Lessons

- Generating realistic text is hard!
- Depends a lot on the consistency of the source corpus, and the accuracy of the parser.
- Things like quotes, footnotes, and formulas are easy to trip up on
 - See the Einstein results
- This approach cannot solve every problem, even with a perfect parse
 - It could never discover male vs female names, for example