Generating Words and Language

Lorem ipsum dolor sit amet.

Generating random words and sentences has been an interest of mine for a while now. I’ve taken a few different approaches would like to share what I’ve come up with.

One of the first, and simplest methods is just to generate a list of what letters are allowed to be followed by what other letters. You could do it all manually, but it’s much faster to pre-parse a dictionary such as this one [dictionary.txt] (1.7MB, 170,000 words). Interestingly, once the dictionary was processed, it became a mere 10KB rulebook.

[\_\_length\_\_]

[Generate random word from letters] *loading…*

Elin  
xtea  
mosaderd

An alternative approach is to break word down into chunks, larger than single letters. For this experiment, I took the words in the dictionary and split them around vowels and consonants, so flagellum would become fl\*a\*g\*e\*ll\*u\*m, and then grouped them into clusters of 4, as: flage, agell, gellu, ellum. From there, the algorithm builds words by matching the start of a new fragment to the end of the existing word. This seems to produce more readable and pronounceable words but the rulebook ends up being significantly larger at 1.2MB.

[\_\_overlap\_\_] [\_\_probability\_\_]

[generate random word from fragments] *loading…*

One particularly fun and successful approach was to take a list of latin prefixes, roots, and suffixes, and then mix and match. In the case of consonants at the fragment boundaries, I added an extra i, o, or u. As a bonus, each word comes with a definition!

[Generate random latin word]

A couple more here, just for giggles. They both use custom specialized dictionaries to generate their results. The main issue with their results is that they are based on grammar rules and word types (noun, adverb, etc) but the results don’t always make sense. Not all nouns are capable of performing all actions, but the programs don’t distinguish between types of nouns in that way. Mirrors can reflect, but cannot swim, airplanes can fly, but cannot skydive.

[Generate random intrigue…]

This second one lets you build a basic sentence, following a simple syntax.

N = noun  
V = verb  
PRE = preposition

[\_\_N PRE N V N\_\_]

Generate sentence

So the question is how do you classify words in such a way as to be able to make sense? Maybe this program needs to pre-parse a few books to determine what words come after what other words, or what groups of words commonly show up together. It’s definitely a question I plan to revisit at some point.