As I understand it, folks are on board with the general structure of my proposal, but would like (eventually) to use AI to come up with a score for how likely a given rule is to apply to a given base word. It would seem then to make sense to do the project in two phases: first as described in my original document, but using the simple algorithm where we don’t look at definitions, then with the algorithm replaced by a neural network created with PyTorch (which has a bad Wikipedia article). Here are is a proposed list of attributes of words which we might provide as inputs to the network.

* Language in etymology
* Number of letters
* Number of syllables
* Prominent words in definition
* Secondary words in definition
* Minor words in definition
* Number of senses
* Other parts of speech
* Earliest use
* Most recent use
* Wordnet categories

I suggest that a rule consists of the following pieces of information:

* Name OUT
* Abbreviation OUT
* Orthographic description of rule (OUT\*)
* Part of speech to which it applies verb FPS
* Part of speech of generated word verb TPS

The items below might be considered either Booleans or additional weights.

* Can it be used on capitalized words? .5 CAP
* For phrases, does it apply to the first word or the last word, or is it inapplicable? 0 PF PL
* (optional) regular expression on orthography. Examples might be /[^sx]$/ to get all base words which don’t end in S or X, or /^[a-z]+ and [a-z]+$/ to get certain phrases. RX
* Filter based on pronunciation (do we want this?)
* Possible successor rules (if a rule is its own successor, it is applied a maximum of two times, whether consecutively or not).

For now, we can create rules in a Google spreadsheet and migrate them to a database once we have a UI for that.

google\_count (word)

gooch\_count (word)

get\_rule (id)

candidate\_sql (rule)

transform\_sql (rule, expr)

<https://stackoverflow.com/questions/4082966/what-are-the-alternatives-now-that-the-google-web-search-api-has-been-deprecated>

blippex

contextualweb

blekko

opensearch