Algorithmic Ku: Monkeys with Typewriters v0.0.1

Brendan Slater

Although I write poetry I would not describe myself as a poet. I am, however, happy to out myself as a Ph.D. drop-out which, while it lasted, as well as satisfying my curious nature, allowed me three years to experiment with a super-computer called the Cray X1. I used to rise at 4a.m. every morning checking my results and processing my raw data with quick and dirty Perl scripts. Of course, that was dependant on how much I had drunk the day before, and I used to drink a hell of a lot, but this is not a confessional, and the most important thing is that I learnt to code in Perl. And Perl is wonderful for dealing with text, it absolutely adores text, words, characters, sentences, paragraphs. It could be described as the Poet of Programming Languages. For instance, it has in-built functions to deal with sentences, like the "split" function, which will, surprisingly enough, split a variable containing a sentence into its component words. It will in fact split a variable of words separated by, well, almost anything. At the time I was using Perl to process raw data from simulations of fibre-optic systems and did not consider the possibility of using it to generate ku, but when the idea came to me to experiment with algorithms to do so, there was only one tool for the job, and the serendipitous coming together of Perl and I became apparent.

In the title of this essay I use the word "ku". Why ku? Well, it not only refers to haiku but also related forms such as senryu, zappai and monostich. I also think it gives a convenient "out" for people who do not agree on the use of terms like "one-line-haiku", "one-liner", "one-line", and so on. So, for the purposes of this essay, "ku" refers to "one-line-verse". Those who are familiar with the changing styles of English language ku will be noticing more widespread experimentation, something that had been the bastion of the "famous" haiku poets of the latter 20th century. Everyone and their dog seems to be setting up a new journal, ezine or Facebook page for the purpose of exploring this mysterious form of writing—me included. And why not? Ku can be cool, amusing, powerful and nonsense all at the same time. But the one thing I would never call them is boring. I mention "nonsense" when describing ku. I think this is an important property for the coder in me who can see immediately how algorithms could be used to create the form. When a ku, for instance, may end with an article, surely an algorithm would not have to be terribly sophisticated to create one? Well, just for fun, let's see how well my first attempt at an algorithm does. Below are 10 ku. The question is can you tell which were written by hand and which were generated by an algorithm?

the wherewithal an acorn's must

last the ripple birch

on time whenever the moon

complicated the eternal dustbin breeze

basement breasts the infinite necrophilia noon

that cool sudden a sea

unless the man the strolling gallows

the undercellar a tsantsa

until waste the strawberry scum

breeze the silver ocean

Well, the answer is neither. The basic structures were generated by the algorithm and then all but four edited by Sheila Windsor and myself to create the finished ku. The algorithm used to generate the ku contains 77 lines of code of which 23 are necessary to declare variables, set compiler variables and add comments, and 12 are necessary for simple printing: program name, version number, etc., leaving 42 lines to compose the ku, which is quite short for a sequential script, that is one that does not contain any original functions or classes. It employs a Dictionary File, a simple word list which can be made from scratch, dumped from online sources, or even generated using a script. The algorithm chooses words from the Dictionary File, and inserts articles and conjunctions in a pseudo-random manner. The script is a work in progress, and I am aware that very sophisticated algorithms have been written to generate rhyming verses that scan nicely and even stick to certain predetermined subjects and themes. But is that really necessary for ku? Do we really need hundreds and hundreds of lines of code to produce a one line poem? Well, the coder in me tends to think the more complexity the better the results. But the poet in me tends to think, actually, no, keep it simple, keep it raw, be pragmatic. I run batches of 100 ku at a time and capture them in a text file. I then go through the list and pick out the constructions I feel have some potential, sometimes they are ready-made, needing no extra work, but others need a bit of work, maybe a bit of rearrangement or some additional words, or some substitutions, or some trimming. Some even act as inspiration for a completely new poem containing absolutely none of the words from the one generated. A great tool to break through that writers' block.

Now the greatest problem that comes up is that of ethics. A few years ago I wrote a small script in the Basic programming language that generated very crude word combinations which I described as mathematical compositions. A friend of mine, who enjoyed my “handwritten” poetry said “but it's cheating”. Well, is it cheating? Let's look at this

logically:

1. People on Facebook and Twitter regularly post a prompt and people post their poems inspired by that prompt and possibly containing the prompt itself. Is this cheating? No, this is writing. Selecting words from The Universal Set of Words (**Y**), i.e. every word that exists in a given language, for this essay we can assume that to be English, plus the predetermined prompt word, which also exists in **Y**. It could be argued that human selection is actually made from a smaller subset of **Y** ($\upsilon=Y-1$), but the poem, including prompt, could potentially contain any word, and the choice is left to the poet.

2. So what if an algorithm generates a ku, and the poet then uses the content and reorganises it to create a new poem. Is that cheating? I would have to argue no, as all the words used to create the poem are part of a universal set (υ), much smaller than **Y** but all exist in **Y**, i.e. $\upsilon=Y-N$ (where $N \approx Y$). The same process is used, the algorithm is simply generating a small universal set of words and the poet is manipulating them to create a poem. The poet could also decide to introduce new words from **Y**, so again, I call this writing.

3. So what if the poet simply runs an algorithm and then uses the results as a finished poem? Considering that the algorithm uses a small subset $\upsilon=Y-N$ to pick from, it can be argued that the poet has influenced the algorithm by creating υ with the choice of Dictionary File (Δ). We can suppose that no Dictionary File will contain every word in the universal set ($\Delta << Y$), so again it could be argued that the poet has influenced the generated poem, obviously not as greatly as the previous two examples, but influenced nonetheless. And again, I call this writing.

4. But, for argument's sake, let's suppose the Dictionary File Δ contains **Y**, and the poet simply runs an algorithm, makes no changes or choices and uses the first result as a poem. This would seem like the monkey with a typewriter scenario, with the monkey being the algorithm. This, I would have to concede, although writing, would not be the writing of the poet, unless the poet is also the coder. But, does this even matter? A poem needs both a creator and consumer and surely the role of the consumer is equally as important as that of the creator? Maybe, but this is hypothetical, simply because language changes continuously, so once **Y** had been captured in a Dictionary File Δ it would have grown, shrunk or morphed in some way and would therefore contain $Y \pm \nu$ and $\Delta \neq Y$. So again, as in point 3., we can argue that the poet has influenced the algorithm, albeit inadvertently.

So, looking back at the previous 4 points, 1.-3. can arguably be called writing and point 4., which is hypothetical anyway, let's say can arguably be called writing, but only just. Concluding that algorithmically generated ku are actually authored works just like any other, and arguments against using algorithms to create ku cannot simply be "because it's cheating". But what of aesthetics? Well, this is a subjective consideration. I love the work of writer "A" but can't stand anything by writer "B", though writer "C" sometimes hits the spot, and on rainy days writer "D" is my preference, etc., etc. It is beyond the scope of this essay (and my abilities) to discuss grammatical constructs, but I think it is safe to say that a string of 6 nouns would probably not read too well, but not definitely so. So an algorithm needs to follow some grammatical rules. But, as mentioned earlier, ku are the great experimenter of poetry, so the grammatical rules need not be so complicated to hinder an enthusiastic coder from writing an algorithm. Of course, if the coder is also a scholar of English grammar, then, we may well have found the perfect monkey with a typewriter.

Appendix: Ku Pre and Post-Edit

Pre-Edit	Post-Edit
the wherewithal an acorn's must	*
lasting the birch a ripple	*last the ripple birch*
whenever on time the moon	*on time whenever the moon*
complicated breeze the eternal dustbin	*complicated the eternal dustbin breeze*
breasts when basement necrophilia is infinitely noon	*basement breasts the infinite necrophilia noon*
that cool sudden a sea	*
unless the man the strolling gallows	*
the undercellar a tsantsa	*
until waste scum the strawberry	*until waste the strawberry scum*
the breeze the silver autumn ocean	*breeze the silver ocean*