A Taxonomy of Generative Poetry Techniques

Carolyn Lamb, Daniel G. Brown, Charles L.A. Clarke
Cheriton School of Computer Science, University of Waterloo

Abstract

We describe computer-generated poetry techniques in the categories of mere generation, human enhancement, and computer enhancement, and argue that the artificial intelligence techniques used by computer scientists are artistically relevant. We also generalize this taxonomy to computer-generated music.

Introduction

Many people generate poetry using computers, from artists exploring the effects of algorithms on language, to Internet hobbyists, to computer scientists interested in making artificial intelligence creative. Despite these varied authors, and lack of communication between communities, the techniques used to generate such poetry can be boiled down into a few simple categories with well-defined relationships.

We define these categories as follows. In mere generation, a computer produces text based on a random or deterministic algorithm. All generative poetry systems we have come across use some form of mere generation. In the remaining two categories, the results of mere generation are modified and enhanced. This occurs either through interaction with a human (Human Enhancement), or through the use of optimization techniques and/or knowledge bases (Computer Enhancement). The results of mere generation can appear nonsensical, though this is not always a bad thing from an artistic perspective. By bringing in knowledge about words and the world, and by setting artistic goals, both human and computer enhancement drive generative poetry towards coherence and artistic style.

Digital poetics encompasses a wider range of techniques than those described here. They include hypertext poetry, kinetic poetry, chatbots as art, interactive fiction [6], multimedia poetry, and even poetry that presents itself as a game [8]. However, for the purposes of this paper, we are interested only in English poems represented as ASCII text, in which the computer has a meaningful role in determining what the text will be. This is our working definition of "generative poetry".

We are not the first to attempt a taxonomy of generative poetry. Roque [28] classifies poems according to the goals of their creators, while Funkhouser [9] uses the categories of permutational, combinatorial, and template-based generation. Gervas [11] classified four types of artificial intelligence techniques used for poetry. These taxonomies are useful. However, our taxonomy serves needs that others do not. It includes generative poetry from a variety of sources, whether scientific, hobbyist, or artistic, and focuses not on technical processes but on the purposes for which these processes are used. Further, our taxonomy illustrates how generative poetry can move forward both computationally and artistically.

In the rest of this paper, we illustrate the techniques used in our three categories. We explain why scholars have moved away from mere generation, and argue that Computer Enhancement, while pursued primarily by scientists, has the potential to solve *artistic* dilemmas in generative poetry. We then bring up the related field of generative music, to show that our taxonomy can be generalized to other creative tasks.

Methods of Mere Generation

Templates. Template generation, also called slot-filling, has been common since the first generative poetry program, Theo Lutz's "Stochastic Texts". Template generation is one of the simplest means of constructing

a poem. The basic steps are as follows:

1. Create lists of words or phrases in different categories, e.g. nouns or verbs.
2. Create one or more line templates with slots into which a word from a given list can be inserted.
3. Randomly select a word from the appropriate list to fill each slot.

"Stochastic Texts" uses templates and word lists based on lines from Kafka's "The Castle". Template poetry can draw its word lists from existing art, or from dictionaries representing the whole of the language, or the lists can be handcrafted by the programmer.

An issue in template poetry is repeatability. Often, running a template program several times will produce a repetitive effect in which the template's structure becomes obvious, as in this example [16]:

A HOUSE OF STEEL
IN A COLD, WINDY CLIMATE
USING ELECTRICITY
INHABITED BY NEGROES WEARING ALL COLORS
A HOUSE OF SAND
IN SOUTHERN FRANCE
USING ELECTRICITY
INHABITED BY VEGETARIANS

Such repetitiveness may, as in the given example, be intentional. However, most poets want output that looks fresh each time the program is run. One way of achieving this is with templates that change over time. John Morris, for example, uses shifting templates to create haiku [23]:

Frogling, listen, waters
Insatiable, listen,
The still, scarecrow dusk.

Listen: I dreamed, was slain.
Up, battles! Echo these dusk
Battles! Glittering...

Markov chaining. A Markov chain is a statistical model applied to data in a series. Based on the last N entries, an N-order Markov chain calculates the probability distribution for the next entry. The N entries used to make the prediction are referred to as an *n-gram*; no entries before the n-gram need to be considered. For poetry generation, the entries in the n-gram can be characters or words. A probability model is generated either from a broad corpus or a specific work, and the system repeatedly samples from the model to create the next entry. Using words as entries ensures that no novel words appear in the output. Using characters results in many non-words and neologisms, which can be an intended effect [28]:

book her sist be chin up seen a good deal uneasilent for coursation
dropped, and the
litter on,
The Queen was
siliarly with them, the Footmance.

Markov chain poetry is related to Dadaist "cut-ups", in which a text is cut into N-character blocks and rearranged [9]. While Markov chains preserve many features of the input text, they fail to replicate grammar.

Context-free grammars. A context-free grammar constructs sentences by recursively applying generation rules. It is a more flexible generalization of a slot-filling template. For example, a [NOUN PHRASE] slot could be filled with a noun, but also with "the [ADJECTIVE NOUN]", "the [NOUN] that [VERB PHRASE]", or a variety of other grammatical forms, which can continue to recurse and produce more sub-phrases.

By adding Markov chain-like probabilistic reasoning to a context-free grammar, one can construct a stochastic context-free grammar which constructs the most likely sentences based on some input corpus while remaining grammatical. Jim Carpenter's Electronic Text Composition uses probabilistic grammar, resulting in language which is semantically odd, yet more coherent than the output of a Markov chain [3]:

> The important statement, like one advance act.
> A spit goes eastern, showering.
> *it perches on the branching foam*
>
> The statement.

Found poetry. Another method is to skip the generation process and, instead, use a computer to harvest text written by humans. While most generation techniques use existing human language—as a corpus for calculation of N-gram frequencies, for example—found poetry preserves entire human-written sentences without significant modification. The computer's role is to select text which meets some constraint and present it outside of its original context. Examples include Ranjit Bhatnagar's "Pentametron", which constructs sonnets by assembling pairs of rhyming, 10-syllable posts on Twitter [1], and the New York Times Haiku project, which mines phrases with a 5-7-5 syllabic structure from that newspaper [13]:

> Surely that shower
> couldn't have been going since
> yesterday morning.

Miscellaneous methods. Some poetry is generated using other methods: for example, permuting the words in a short phrase [9]. We will not explore these methods in detail.

The Problem With Mere Generation

While the methods described above may seem endlessly flexible, there is a limit to their usefulness. The frequency of words and the rules of grammar can be modeled, but semantic coherence is more difficult. This has caused some pioneers of generative poetry to grow discouraged with the form. Charles Hartman, for example, concluded that even the most syntactically correct programs "did only a little to drive the random words toward sense [9]." Chris Funkhouser, studying poetry generators, says that "even the best of them fatigue the reader with blatant slotted structures and repetition."

Lack of coherence is not necessarily a problem. Dadaist poets are interested precisely in the human response to nonsense [9]. Similarly, Oulipans [9] are interested in inventing poetry techniques, not in the quality of the poetry itself. However, other poets may be discouraged by the limitations of current generation techniques. We will now turn our attention to the two major methods for improving on these techniques.

Human Enhancement

The most obvious way for a human to enhance computer-generated poetry is to edit the poetry generator's output. While this arguably invalidates the generator's usefulness [4], it is an established practice. John Cage, for instance, removed unwanted words from the output of his algorithms [9]. Computational text generation is seen by many as a "jumping-off point" [4] from which they acquire raw material.

One group particularly keen on this is the Flarf movement, which revolves around intentionally lowbrow language from the Internet. One Flarf technique, "Google sculpting", consists of searching the Internet for particular terms, taking phrases from the results (a form of Found Poetry)—and then recombining these phrases however the poet sees fit. The result is a distinctive, over-the-top poetics [12]:

Oddly enough, there is a
"Unicorn Pleasure Ring" in existence.
Research reveals that Hitler lifted
the infamous swastika from a unicorn
emerging from a colorful rainbow.

Even more interesting is Gnoetry, an application for interactive text generation which allows decisions to cycle between the computer and a human user. The computer generates poetry based on n-grams from a user-provided corpus. The user can click on individual words, deciding which words and phrases are worth keeping and which should be generated again. The computer then generates new phrases to replace those the user did not find satisfactory. This repeats as many times as the user would like. Gnoetry turns the generation process into a dialogue between the human and computer. An online community, blog, and several chapbooks exist showcasing the work of various poets using Gnoetry. A favored technique, as with Markov chain poetry, is to mix together the styles of several contrasting works [31].

The notion of human and computer cooperation also appears in the development of creativity support tools. Kantosalo *et al.* , for example, have a system to help elementary school students write poetry, which suggests possible words in a magnetic poetry format [15].

Computer Enhancement

The other way to enhance computational poetry is to add advanced concepts from computer science: not merely generating words, but making sophisticated attempts to optimize the output. This set of methods comes not from the humanities but from scientists in the discipline of computational creativity. While a variety of AI techniques can be brought to bear on these problems, there are two main purposes. One is optimization of the system's output on some metric; the other is connection of the generation apparatus to underlying knowledge about the world.

Importantly, all computational poetry systems we analyzed begin with one of the mere generation techniques discussed above. Template generation is most common [5, 24, 33, 34, 32], but the McGonagall system [21, 19, 20] uses a technique similar to context-free grammar, while DopeLearning [18] (recombining lines from rap songs) and our own TwitSong [17] (recombining lines from Twitter) are found poetry systems. What distinguishes all these systems from mere generation systems is that something—optimization, a knowledge base, or both—is *added* to the basic technique, either following the mere generation step or incorporated into it as guidance.

Data mining and knowledge representation. Merely generated poetry tends towards nonsense; whether artistically desired or not, this is a result of the computer's lack of real-world experience. By representing semantic facts, an artificially intelligent system can attempt to overcome this limitation. Just as Gnoetry puts human enhancement inside the generation process rather than adding it on after generation, a knowledge base can be put inside the generation process to guide its range of output.

One way of representing knowledge is to encode it in propositional logic. One experiment in this vein is Ruli Manurung's McGonagall system [20]. When propositions are explicitly encoded, McGonagall creates very logically consistent poetry, as in the first example below [20]. But when the system is allowed to construct its own propositions, as in the second example, it lapses into nonsense [21]:

The cat is the cat which is dead.
The bread which is gone is the bread.
The cat which consumed
the bread is the cat
which gobbled the bread which is gone.

They play. An expense is a waist.
A lion, he dwells in a dish.
He dwells in a skin.
A sensitive child,
he dwells in a child with a fish.

It is not enough for a computer to be able to represent facts; to be anything other than nonsense, these facts must have some relation to human experience.

Since McGonagall, a few systems in non-English languages have made good strides using semantic knowledge bases, such as ConceptNet, to widen the range of available propositions [27, 30], but English systems have not yet fallowed suit. What *has* been done in English is parsing of word associations. By calculating the co-occurrence of different words in a source text, computers can gain a sense of which words are and aren't related to a given topic. Toivanen *et al.* create poetry using word associations mined from Finnish Wikipedia [32] and news stories [33]. Netzer *et al.* [24] use a list of word associations from psychological testing. Combining these associations with syntactic templates results in plausible haiku [24]:

> cherry tree
> poisonous flowers lie
> blooming

Veale and Yao [35] search Google N-Grams for similes, which contain implicit information about the properties of things in the real world. The Full-FACE system [5] modifies these similes to create poems. Its poetry, while repetitive, is full of comparisons that make sense to a human [5]:

> the wild relentless attack of a snake
> a relentless attack, like a glacier
> the high-level function of eye sockets

Optimization. The other mode of enhancement that computer science has to offer is optimization. Given some formal definition of the desired traits of a poem, a computer system can begin to, in some sense, think critically—testing different possibilities, and choosing the ones which best fit its requirements. While this is a very elementary form of critical thought, it is an important step towards true creativity on the part of computers: being able to understand one's own aesthetic and create work to match.

Optimization techniques applied to poetry include stochastic hill-climbing search [19], generate-and-test [24, 5], genetic algorithms [21], constraint satisfaction [34], case-based reasoning [10], and recurrent neural networks [18]. The details of these methods are relatively unimportant: each involves setting some goal as to the desired properties of a poem, and trying multiple possibilities, often building on previous attempts, until one or more poems are found which are as close to the goal as possible.

What goals do these poetry systems work towards? Many concentrate on basics such as meter, rhyme, and grammaticality [19, 24, 34]. However, a more exciting possibility is setting goals for the subject matter, emotions, or artistic style of a poem—traits which can be measured through natural language processing techniques. ASPERA [10], for example, includes mood as one of its constraints. DopeLearning [18] generates rap lyrics according to a variety of textual measures, including the maximization of complex, internal, and multisyllabic rhymes, and uses a deep learning neural net to represent semantic content. The Full-FACE system [5] chooses between possible goals, including relevance to the poem's topic, emotion, and some stylistic constraints. TwitSong [17] also selects tweets according to their topic relevance, sentiment, and the presence of sensory imagery: this results in poems composed of tweets which, while written by different people at different times, nonetheless come together into something reasonably coherent [17]:

> Hey Nashville...2014 is pretty awesome!
> Happy 2014 friends! Be safe out there!!
> Had a great New Years Eve at Magic Kingdom
> We started off 2014 with a prayer

The science of computational creativity is in its infancy, and these are small steps compared to the

complex goal-setting process of skilled human poets. Nonetheless, further refinements in goal specification and knowledge representation could produce truly interesting generative poetry.

Generalization and Comparison With Music

Our taxonomy is more interesting if its principles can be generalized to other domains. An obvious example would be other domains of digital poetry, such as those described in [6, 8]. Certainly many such poems combine the kind of generation that is of interest to us with hypermedia techniques, which can themselves be seen as a form of either human or computer enhancement, applied to the poems mode of presentation rather than to its words. Nick Montfort and Stephanie Strickland's "Sea and Spar Between", for example, uses a handcrafted grammar (a mere generation technique) to combine phrases from Emily Dickinson and Herman Melville's work, and displays these phrases using a combinatorial framework which could only be accomplished using the calculating power of a computer [22].

However, in our specific research program we are more concerned with whether our taxonomy is generalizable to computational creativity as a whole, outside the domain of text altogether. To investigate this question, we will look very briefly at the field of generative music. Our taxonomy—Mere Generation, Human Enhancement, and Computer Enhancement—does apply to music, with a few interesting modifications.

Mere Generation. Like poetry, music can be composed using a Markov model trained on the note sequences of existing music [7, 26]. Other ways of generating music include cellular automata [37], a random trajectory in a directed graph [29], or even producing music from a visual image as if it were a spectrogram [14]. What these techniques have in common is that, as with mere generation in poetry, the output of the algorithm is not optimized. The program simply produces notes according to its rules.

Human Enhancement. The Human Enhancement category in music is most noticeable in jazz improvisation. Computer Improvisation systems are built to play alongside human improvisers. They need to both process human musical input in real time and respond to that input with novel sequences of notes [2]. A more avant-garde alternative is the creation of digital instruments. An instrument's responses to human input can be non-obvious or shifting, which results in unpredictable interactions between the musician and the instrument [36]. Like Gnoetry, improvisation systems and digital instruments can create new works through interaction which neither the computer nor the human could have created on their own.

Computer Enhancement. As with poetry, the output of mere generation can be fit to hard and soft optimization constraints using techniques such as Answer Set Programming [25] or genetic algorithms [29]. Such constraints can be rooted in music theory [25, 29], or a specific goal, such as making cover songs incorporate features of the original [26]. Due to the non-representational nature of music, it is more difficult to find generative music that incorporates a semantic knowledge base.

The main difference between generative poetry and generative music is that in we see systems in which human and computer enhancement are combined. Systems can, for example, improvise with human partners and use machine learning to optimize that improvisation [2]. There is no *a priori* reason that this could not also be done with poetry; its presence in music is likely the sign of a more mature research field.

Conclusion

In our taxonomy, there are three areas of work in generative poetry: a mere generation technique, human extensions to the technique, and computer extensions to the technique. We believe that when critics such as Funkhouser declare that generative art has reached a plateau, it is because they are looking only at mere generation and not at the more powerful computational techniques which enhance it. Artistic optimization and knowledge representation techniques have not yet reached their full potential, but they have the power

to push generative poetry forward towards the kinds of sense and style that are currently lacking. Far from being a played-out form, generative poetry is just getting started.

References

[1] Ranjit Bhatnagar. Pentametron, 2012. `http://pentametron.com`, accessed December 9, 2015.

[2] Oliver Bown. Player responses to a live algorithm: Conceptualising computational creativity without recourse to human comparisons? In *Proceedings of the Sixth International Conference on Computational Creativity*, pages 126–133, 2015.

[3] Jim Carpenter. Public override void, 2004. `https://slought.org/resources/public_override_void`, accessed December 9, 2015.

[4] Jim Carpenter. etc4 (blog post), 2007. `http://theprostheticimagination.blogspot.ca/2007/07/etc4.html`.

[5] Simon Colton, Jacob Goodwin, and Tony Veale. Full face poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, pages 95–102, 2012.

[6] Jeremy Douglass. Numeracy and electronic poetry. *Journal of Mathematics and the Arts*, 8(1-2):13–23, 2014.

[7] Arne Eigenfeldt. Generative music for live musicians: An unnatural selection. In *Proceedings of the Sixth International Conference on Computational Creativity*, pages 142–149, 2015.

[8] Chris T Funkhouser. *New Directions in Digital Poetry*. A&C Black, 2012.

[9] Christopher Thompson Funkhouser. *Prehistoric digital poetry: an archaeology of forms, 1959-1995*. University Alabama Press, 2007.

[10] Pablo Gervás. An expert system for the composition of formal Spanish poetry. *Knowledge-Based Systems*, 14(3):181–188, 2001.

[11] Pablo Gervás. Exploring quantitative evaluations of the creativity of automatic poets. In *Workshop on Creative Systems, Approaches to Creativity in Artificial Intelligence and Cognitive Science, 15th European Conference on Artificial Intelligence*, 2002.

[12] Nada Gordon. Unicorn believers don't declare fatwas. *Poetry*, July/August 2009.

[13] Jacob Harris. Times haiku: Serendipitous poetry from the New York Times. `http://haiku.nytimes.com/`, accessed December 9, 2015.

[14] Eric Heep and Ajay Kapur. Extracting visual information to generate sonic art installation and performance. In *Proceedings of the 21st International Symposium on Electronic Art*, 2015.

[15] Anna Kantosalo, Jukka M Toivanen, Ping Xiao, and Hannu Toivonen. From isolation to involvement: Adapting machine creativity software to support human-computer co-creation. In *Proceedings of the Fifth International Conference on Computational Creativity*, pages 1–7, 2014.

[16] Alison Knowles and James Tenney. A sheet from 'The House', a computer poem, 1968. qtd. in (Funkhouser, 2007).

[17] Carolyn E Lamb, Daniel G Brown, and Charles LA Clarke. Can human assistance improve a computational poet? In *Proceedings of BRIDGES*, pages 37–44, 2015.

[18] Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. Dopelearning: A computational approach to rap lyrics generation. *arXiv preprint arXiv:1505.04771*, 2015.

[19] Hisar Manurung, Graeme Ritchie, and Henry Thompson. Towards a computational model of poetry generation. Technical report, The University of Edinburgh, 2000.

[20] Hisar Maruli Manurung. Chart generation of rhythmpatterned text. In *Proc. of the First International Workshop on Literature in Cognition and Computers*, pages 15–19, 1999.

[21] Ruli Manurung, Graeme Ritchie, and Henry Thompson. Using genetic algorithms to create meaningful poetic text. *J Exp Theor Artif In*, 24(1):43–64, 2012.

[22] Nick Montfort and Stephanie Strickland. Sea and spar between. *Dear Navigator*, 2, 2010.

[23] John Morris. Haiku—at random, 1973. qtd. in (Funkhouser, 2007).

[24] Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. Gaiku: Generating haiku with word associations norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 32–39. Association for Computational Linguistics, 2009.

[25] Sarah Opolka, Philipp Obermeier, and Torsten Schaub. Automatic genre-dependent composition using answer set programming. In *Proceedings of the 21st International Symposium on Electronic Art*, 2015.

[26] Graham Percival, Satoru Fukayama, and Masataka Goto. Song2quartet: A system for generating string quartet cover songs from polyphonic audio of popular music. In *The International Society of Music Information Retrieval*, pages 114–120, 2015.

[27] Ananth Ramakrishnan A and Sobha Lalitha Devi. An alternate approach towards meaningful lyric generation in tamil. In *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity*, pages 31–39. Association for Computational Linguistics, 2010.

[28] Antonio Roque. Language technology enables a poetics of interactive generation. *Journal of Electronic Publishing*, 14(2), 2011.

[29] Marco Scirea, Peter Eklund, and Julian Togelius. Toward a context sensitive music generator for affective state expression. In *Proceedings of the Sixth International Conference on Computational Creativity*, 2015. Late-breaking abstract.

[30] Von-Wun Soo, Tung-Yi Lai, Kai-Ju Wu, and Yu-Po Hsu. Generate modern style chinese poems based on common sense and evolutionary computation. In *2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 315–322. IEEE, 2015.

[31] eRoGK7 *et al.* Gnoetry daily. `https://gnoetrydaily.wordpress.com`, accessed December 9, 2015.

[32] Jukka Toivanen, Hannu Toivonen, Alessandro Valitutti, Oskar Gross, et al. Corpus-based generation of content and form in poetry. In *Proceedings of the Third International Conference on Computational Creativity*, pages 175–179, 2012.

[33] Jukka M Toivanen, Oskar Gross, Hannu Toivonen, et al. The officer is taller than you, who race yourself! Using document specific word associations in poetry generation. In *Proceedings of the 5th International Conference on Computational Creativity*, pages 355–359, 2014.

[34] Jukka M Toivanen, Matti Järvisalo, Hannu Toivonen, et al. Harnessing constraint programming for poetry composition. In *Proceedings of the Fourth International Conference on Computational Creativity*, pages 160–167, 2013.

[35] Tony Veale and Yanfen Hao. Exploiting readymades in linguistic creativity: A system demonstration of the jigsaw bard. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations*, pages 14–19, 2011.

[36] Victor Zappi and Andrew McPherson. The D-Box: How to rethink a digital musical instrument. In *Proceedings of the 21st International Symposium on Electronic Art*, 2015.

[37] Mo H Zareei, Dale A Carnegie, and Ajay Kapur. Noise square: Physical sonification of cellular automata through mechatronic sound-sculpture. In *Proceedings of the 21st International Symposium on Electronic Art*, 2015.