Generative Poetry: The Intelligence of Art

Emma Hyde and Amy D'Entremont

May 2016, under Dr. Hava Siegelmann

What does it mean to be human? What separates humans from other animals, and what are the fundamental qualities of that distinction? These questions have spurred debate in many areas, particularly in psychology and philosophy, for hundreds of years. With the modern age of computing and the widespread growth and consequential effect of artificial functioning, we must redefine these questions by considering artificial intelligence. Simple things like laughter, depression, and involved thinking seem as unique to the human race as do complex developments like technology, religion, and self-awareness. But can they be developed by artificial intelligence?

Other species have been known to create some form of art, but our definition of art as an intentional production makes it exclusive to us. If another intelligence could conform their artistic production with our interpretation, would that redefine the rules? Art is separated by distinct manifestations, both as an action and as an experience: is a produced art given meaning by the intention behind it, or simply by its existence? Likewise, most species have a sort of basic communication, but our refined use of language is uniquely human. Human languages are complex verbal systems complete with grammar rules and recursive syntax, and we regard ourselves as the lone champions of this development. But if another entity could create language within these restrictions, could the entity be said to produce language in a human way? Would the language generated really be "artificial"? As engineers, we pursued the intelligence of artistic language production, poetry, in an attempt to deconstruct its human exclusivity as a whole, as well as create an intelligence worthy of its name.

In 2010, Zackary Scholl, an undergraduate student at Duke University, asked many of the same questions. He wanted to create an artificial intelligence that could "create poetry indistinguishable from real poets," *(Scholl, Raspberry Pi AI).*

In 2011, he submitted some of his generated poems to poetry websites, blogs, and forums. Most of the responses he received were "quite flattering... and overwhelmingly positive", which encouraged him to submit his work to poetry or literary journals as a "most stringent test of authenticity," *(Scholl, Raspberry Pi AI).* This was Scholl's simple Turing test: if someone reads a poem and thinks that it is in of itself a poem, let alone a good one, does that not classify it as a human poem, by the sheer nature of poetry being a human art? Indeed, many people found these poems to pass their own perceptual Turing tests, and by September, Duke University's "The Archive" had accepted his poem "For The Bristlecone Snag" for their publication, unaware that it had been generated by an artificial intelligence. The poem reads:

A home transformed by the lightning
the balanced alcoves smother
this insatiable earth of a planet, Earth.
They attacked it with mechanical horns
because they love you, love, in fire and wind.
You say, what is the time waiting for in its spring?
I tell you it is waiting for your branch that flows,
because you are a sweet smelling diamond architecture
that does not know why it grows.

Scholl's poetry generator works with with a context-free grammar using Backus-Naur form. A Context Free Grammar (CFG) is a formal language theory model that defines a formal grammar by recursively mapping nonterminals, intermediary symbols representing multiple possible strings, to terminals, strings that "fill in" the content of the sentence, like a mad-lib. These grammar rules are called "productions". Upon text input, a production is added to the context-free grammar for each word, mapping the part-of-speech (POS) nonterminal to the word as a terminal. Sentences accepted by a grammar can be generated using its context-free grammar by starting with one non-terminal and nondeterministically following appropriate productions to create a string of terminals. Rules may be defined as:

POS	**Noun**	**Fruit**
Noun	Fruit	Grape

Where fruit is the terminal for that particular branch, and each rule leading recursively backward from fruit is a non-terminal that selects it for the correct context. Backus-Naur form is one of the two main notation techniques for context-free grammars, which works primarily with parts of speech and syntax, and reads quite like the above example.

The generator also implements a word type classification that he created in order to make the poems more cohesive iteratively within themselves: it separates positive words (e.g. love) from negative or neutral words (e.g. thorn). When the poem is generated, it can be generated as a positive or negative poem based on these binary classifications. Scholl's generative structure relies on the basis that "every word in the English language is either 'positive' or 'negative'," *(Readme.md, GitHub).*

These implementations are a solid foundation for a program that creates generative art, but some of the grammar rules within the notation were incredibly specific and the context-free grammar was, ultimately, underutilized. One exemplifying line reads, `''The <pnoun> knows this, <p> that life in it's <ptexture> boxes is as endless as the <pnoun>''`. The hardcoded grammar caused Scholl's implementation to be misleadingly simple and relatively unintelligent. As we began to create the basis of our intelligence, we decided that we wanted it to generate its own grammar and syntax from a dynamic, undetermined input of words. This would be a more significant contribution to the world of artistic generation, and a more robust, competitive, and natural associative intelligence. The Python Natural Language Tool Kit (NLTK) allowed the syntactical decision making that would enable this kind of implementation and provide the scope we wanted.

Our original design of the poetry generator also used a CFG model. We defined the CFG to randomly choose a poem-like sentence structure and randomly map the POS tags to appropriate words pulled from the input text. While this selection is completely random, a positive consequence of this was that words that were more frequently used in the input text were more likely to be in the poem, since each word in the input text is added to the CFG, including repeats. This increased overall "topicality", which we defined within the context of this project as the ability for the poem to have a coherent theme, topic, and subject.

NLTK was the main processing tool and method that we used, and is commonly found in computational linguistics and many natural language programs. It classifies parts of speech automatically as a program executes, and reclassifies them with every iteration. We used context-free grammar rules to select those parts of speech. While Scholl's implementation used mostly terminals with a couple non-terminal grammar rules and many hardcoded syntaxes, we tried to use as little hardcoding as possible in order to generate truly unique poems that vary greatly with different inputs. Longevity and variation were important to us, and we wanted very little repetition between poems. NLTK also classifies each word as a new entry to a given part of speech, so words that are more commonly used in an input will be more commonly seen in an output, suggesting that our poems would more accurately resemble their input as opposed to being of a similar structure to each other, as seen in Scholl's generator.

While NLTK has many benefits, it came with its fair share of limitations, most of which we were exposed to after implementation. The primary issue we faced was the way that it classifies parts of speech and its inflexibility post-classification. For instance, "she" and "her" are in the same classification, which leads to many uncorrectable grammatical errors due to possession, and plurality (such as "all" and "a") is also troublesome.

Our findings over time indicated that NLTK's descriptors are often not distinct enough. Without hardcoding the removal of a particular word from a particular part of speech and putting it into a distinct, new classification, there was no way to remedy as the toolkit could not utilize any alternative initial classification code we might attempt to write. Other part of speech misclassifications we encountered included various tense misclassifications, and "is" being classified along with present tense verbs, leading to overuse and incorrect usage of the word "is" (e.g. "She is and is" as opposed to "She looks and waits").

The context-free grammar solution that we chose for our logic also lead to an inability to implement rhyming, due to the recursive nature of the generation. Given more time, we would ideally create our own grammar that has rhyming subsets and parts of speech classification within it, but this would require that we write our own parts of speech classification code. We noticed that NLTK is better equipped for natural language processing than generation, as it implements a version of a depth-first search that locates a word and identifies the nature of a sentence by its various branching paths. Unfortunately, this lead to generations that were far less random than we desired. Since there is such a wide scope of possible productions within a single tree, we found that even generating poems of a hundred lines in a single iteration would yield a hundred sentences that all start with the same word. Writing our own generative randomization code was necessary in order to avoid this.

The last major limitations of NLTK that we wrestled with was the absence of any inter-line relation and inter-iteration classification: the program would generate each line individually with no simple way to consistently choose a tense or a subject between lines, and once the program had generated all lines, created the poem, and ended its process, all possibility for the next execution to be consecutively "better" is eliminated. We could, in a sense, have gotten around this by having one iteration generate more than one poem (much like a genetic algorithm) and have it compare each successive poem to a database of real poems for grammatical validity; but NLTK's inflexibility, particularly its lack of custom classification integration, restricted the feasibility of this within our time restrictions.

The results were satisfying and, for the most part, grammatically correct, after hardcoding the classification changes: the largest inconsistency was between lines instead of within them. We tried a variety of inputs, some producing better results than others. The findings indicated that smaller inputs (e.g. excerpts instead of books), more consistent subject inputs (e.g. textbooks, nonfiction instead of fiction), and more consistent tense inputs were better, as the lines would be more cohesive due to general topic relevance across the input. Artistically the results varied greatly across poems, due to the random nature of the production rule selection; sometimes, extremely simple comparative similes would eclipse the human impact of the poem; other times more interesting statements would seem to metaphorically suggest something. Generally, non-fiction tended to be more reliably sensical, and fiction had a greater risk-reward scope. Here is a relatively well developed poem from *Harry Potter and the Sorcerer's Stone*:

They are as invisible as waiting footsteps
All cold, all tired
He is as dead as safe
He was holding out

There the cloak has been from the heart
She has been going
Insides were presents

She said, the honest smile is an illegal fortune

And here is an example of the kind of reliability that non-fiction tended to provide, from *Artificial Intelligence: A Modern Approach*, a textbook:

She is a teacher of the statistical relationships
He is general and simple
She has been using like each various plasticity
They were networks of the neurobiological information
An approach is usually supervised
A block is delayed and synaptic
She was a complex voltage
The neural bias was as adaptive as special
The synaptic input is a digital interconnection
An whereas note was a nature in input benefits
The supervised types have been learning
It is training

These poems, and many others generated in the process (see Appendix A for a sampling), embody an artistic value that is difficult to invalidate, and within the human definition of art it is consistent that they possess human value and contribute to human art as a whole. Despite the fact that no meaning is intended by the generator, meaning is found; similarly, intelligence can both be inherent, yet also perceived and defined by that perception. Many animals create "art": elephants and gorillas paint, birds use color and shape to organize objects in what is seen to us as a "deliberately artistic ordering", even cows create unique salt-lick patterns *(Endler, Telondis, UCL)*. Part of the generation is internal to the reader, as is much of poetry and, expansively, art. Whether poetry is defined by the intended meaning, the interpretation of the audience, or perhaps more reasonably, both, this artificial intelligence repurposes human input to create conceptualizations that resonate with humanity, thus highlighting the distinction between art as a creation and art as an interpretation. Regardless of the problematic variety of definitions of art, including the common argument that defining it in of itself restricts meaning, if an observer finds meaning in a production, this fulfills evocation, one of the core fundamentals of art as an input instead of an output: a noun instead of a verb. An intelligence is realized by this production, regardless of intention.

Given extra time to work with this project under the wing of Dr. Hava Siegelmann, we began the second version of this project from the ground up with the knowledge of these limitations and the same amount of time, hoping to find a way to alter the NLTK material to focus on generation instead of analysis. More flexibility would lead to a creation that was more topical and categorical, and thusly be a more dynamic "intelligence". We noted early on that we could add many syntactical rules resembling Scholl's hard-coded grammar (e.g. `''I <verb> the canvas, and sink deeply into the <noun>`") to ensure that every poem is more likely to be grammatically correct, but this would distance our creation from our vision: that our AI would natively and correctly know its grammar enough to do this dynamically without our guidance (e.g. `''I <verb> the <noun>, and <verb> <adverb> <preposition> the <noun>''` natively generating the aforementioned example). The idiosyncrasies of NLTK were frustrating and limiting, albeit sometimes entertaining (continually classified "prizes" as an exclamation), and we hoped to find more ways to circumvent this.

We also hoped to stray from the CFG model entirely. However, as the project progressed we realized the framework of it was a very important base of our program, as there was no way we could not have a poem-like sentence structure hard-coded. For example, if we tried to make the poem function like a Bayes-Network probability, with `<verb>` mapping to `<noun>` with some probability, and `<verb>` mapping to `<adverb>` some probability, this would create too much grammatical randomization. Since randomization of topic and structure selection is already dealt with, we figured it would be step back despite any steps forward to randomize the structure itself by removing the CFG model from our implementation.

We shifted our focus to developing more context throughout each poem and increasing topicality, as previously defined within our scope. In order to do this, we changed the CFG to map only POS tags, instead randomly selecting applicable POS words themselves. In effect, this generated a poem "skeleton" instead of generating the poem itself. To fill the skeleton of the poem in with words (or "frame" as it is classified in the code), we first search the input text for two-word collocations, which are words that frequently appear next to each other (e.g. red wine). We add any collocations that match the POS pairings in our poem skeleton, in order to further represent more relevant word pairings, such as nouns that are frequently described by a certain adjective. Next, we find words that are longer than 6 characters and occur more than twice in the text and add them to the skeleton if possible, in order to intelligently increase topicality. Finally, in order to create more of an overall theme over the lines, we choose a noun that occurs close to the beginning of the poem and repeat it throughout the poem where possible, at most once per line. The idea here is that a word like I, she, he, or it will be chosen and then frequently appear at the beginning of the lines of the poem. Overall, this dynamically and reliably generates a subject or narrator for the poem.

To create more context in the individual lines, we loop through the skeleton multiple times and fill in words before or after words that are already filled in to the skeleton that appear before or after the given word in the input text. Like the collocation, this increases the probability that pairs of words will be more appropriate in the poem. However, there was a drawback that we found to this method. Consider the example with a word, a POS, then a word. The generator will fill the POS in with a word that appears after the first word, but this is not necessarily a word that should logically appear in front of the second word. Since this process does not necessarily fill in all words, we randomly insert a valid unfilled POS afterwards, repeat any possible nouns again, fill in any POS tags we can based on context, and randomly fill in any POS tags that may remain.

Despite not completely removing the CFG model from our implementation, we made changes to the CFG in this version of the generator. In poetry, grammatical structures (e.g. "He has lost, he has lived, he has seen") are often repeated. When the skeleton is generated, it occasionally chooses two lines at random to have the same grammatical structure.

The poems that had this repeated sentence structure, as well as the "subjectivity" of choosing a singular noun to be repeated, typically seem to be the most "human". However, when we relied on this feature too heavily by repeating more and more lines, it ultimately appeared to be forced and contrived.

With these changes, the second version of our generator functions quite differently than the first; it works better with larger inputs since collocations are more likely to occur and there are more options for contextual adding. The original generator worked better with smaller excerpts, since topicality was automatic and all words were chosen at random. This is overall an improvement, as with any tailored input to a random word-to-grammar generator, a user can usually create and define their own topicality; our latter implementation leaves topicality to the implementation, increasing its overall intelligence as a program.

We were pleased with the development of our generator and have created countless new poems. While some are nonsensical, we have been quite happy with the results overall. It functions best as a tool for writing such as an automated, poetic brainstormer or inspiration generator, based on a large input. See Appendix B for a sampling of poems from the second generative implementation.

With the year coming to a close, we shifted our focus to a web implementation in the hope that the public could use it with creative intention. We used Flask, a Python framework, in order to prepare a local environment. Its current functionality is that the front-end interface propositions the user to upload a large text file with a simple file-browser upload. As you submit, the front-end calls the back-end generation Python application. Instead of returning the poem to the console, it returns it to a basic HTML page with the resulting generation embedded. This simple implementation, unfortunately, creates too much CPU overhead in order to be implemented on a free web service such as Heroku, as any back-end request that is predicted to take more than 30 seconds is immediately aborted. In order to fully implement our web interface, we would need to purchase server space that does not have CPU limitations, or develop it in such a way that the back-end processes passively, such as the back-end returns a verification that the user will receive an email when it is completed, and completes all of these tasks successively. The Flask implementation functions fully on a local server using venv.

Can computers produce intentional art, and can they think enough to do so? Many creations of the modern programming world have sought to provide insight or answers to this question. In 2015, Google notably contributed the Deep Dream project to the world. Conceptually, this project is an image recognition neural network, reverse engineered. “Neural networks that were trained to discriminate between different kinds of images have quite a bit of the information needed to generate images too,” Google employees wrote on their research blog. Instead of taking the trained program and using it to classify parts of an image, it takes an input and uses it to generate a more developed image. Each layer of the neural network consecutively produces a “higher and higher-level” interpretation of what the last layer developed based on a number of constraints (e.g., “neighboring pixel correlation”). Edges and corners, shapes or components, then complex objects like faces or animals may be represented at the final layer. Google breaks it down into layman’s terms: “If a cloud looks a little bit like a bird, the network will make it look more like a bird...until a highly detailed bird appears, seemingly out of nowhere,” *(Inceptionism, Google).*

Our poetry generator’s ideal form, with limitless time and effort to spend, would borrow many concepts from Google’s Deep Dream neural network. Using a massive database of existing poetry, prose, and literature, it would learn to associate nearby words together and, post-generation, retroactively compare those associations to existing phrases and pairings, testing their grammatical validity and impact using a comparative rating system. It would then record those associative ratings in a memory system to be called upon during the next generation. Many of these preferences and wishes were realized as we worked on the project as-is, making this an involved learning and research experience. The value of a system that learns across executions is unparalleled in the artificial intelligence world, as it trains itself and grows in intelligence and accuracy with very little manual override or maintenance. In this sense, the generator truly would be an intelligent, growing form, and would function very similarly, albeit with a specialization, to human intelligence.

Whether growth and specialization are enough to create a genuinely “intentional” work of art remains to be seen. The general societal reaction spurred as it develops will validate and invalidate this definition of art and intelligence time and time again. The role of consciousness in intelligence at large leads to broad philosophical questions that arguably cannot be answered: long after artificial intelligence displays some degree of consciousness, its validity will constantly be tried and debated. An AI that generates some work meaningful enough to set forward human movement could be quickly invalidated by a critic who says the AI only recomposed human work. Yet, every genuine human work follows this same process: taking in input as inspiration and recomposing it as a new, unique output. This is how the human brain functions. This is how the human brain dreams at night, and thinks throughout the day. Truly original concepts are arguably non-existent in the modern era. Every human tradition, religion, conceptualization, culture, language, historical event, and work of art, is built by deconstructing old concepts and re-purposing them to find new meaning within them. Whether "generation" is a means to an end, or the meaning is defined by the very means, we will continue to attempt to answer these questions for the rest of conscious time, as we attempt to explain and define our own awareness and importance in the world.

Harry Potter and the Prisoner of Azkaban

They are remaining here like the dirt
The first voices have been as silver as the sky
Scattering like feet after the first class
The cheery one starts to sob
Has been bumping away and heaving
A clock in the distance
The mud behind the wooden bucket sets
The pumpkins have been watching only for my commentary
And a lone man lies down

One Flew Over the Cuckoo's Nest

It started slow
He had been caught
The biggest fish was as red as the other
There, the wind pushed against a smooth water
All thousand were shouting finally
The water was excited
Yammering, something was hissing

Electric Kool Aid Acid Test

This trip, in the costume of space, is done
The projectors have been white
The brightest energy will have been just leaving
A huge fantasy has been waiting like a psychic show

He has been standing there forever
Taking it like a thought up in another message
The group, in the big jungle, was asleep

You came off of this first
He's good with cash
Don't need to show it

Please note that punctuation has been added manually.

Electric Kool Aid Acid Test

I have been as unwelcome as the morning
I have been as mysterious as the nothing
Another something is up because nothing has been shooting
The binoculars have been sticking to his face
These possessions, I have been wearing like it's nothing,
Staring quietly like the whisper outside

and

Marijuana bought over outside a mescaline college
You have been graphic like newspapers
You have been straight like shoulders
A gathering without America
It is a black movement
It has been low like a strange ceiling
She is hanging like a shadow in the theater

and

Another beatnik
Plastic between the windows, a message on the bullshit
Fluorescent like Francisco
The sky freaking painted the plastic
Something should have been telling
There the bathroom was balling
Wearing like Saturday, up in the open
Beautiful like the metaphor
He could have been watching
He is instead drawing by some current theater

Please note that punctuation has been added manually.

1984

Astonishing like another half-forgotten statement
It has been questioning me and failing
The invention wore out the preliminary rewrite
A corrupt presence through it persisted without another process
The cylinders became the individual
The unconnected answered ERROR
We were various
We were many
It has been pressing back for years

and

She has been a comparison without a "dislike" preparation
The greatest intensity exposed down another level
It has been anywhere, grasping like another unimaginable approach
All happiness tolerated by the majority
That portion beneath the totalitarian has been trickling up
A picture has been fascinated like a technical contempt
You are a telescreen torture
He is staying throughout the histories

and

Another falsification has been swallowed like a remembered consumption
The reasons were immortal heresies
We have been a doorway to a living-room
The glorious trumpet battles overthrow the changes
Another speakwrite catapault represented the instinctive details
The feelings are grasping without the features
The records are as non-existent as peaceful

Please note that punctuation has been added manually.

Slaughterhouse-Five

She was married
it happened there
the biggest bedroom visited another girl
she was possible
a trouble against him
he could have been weeping
another ordinary machine was powerful
the present ventilators were clean like a corrective kitchen

The Wind-Up Bird Chronicle

There, the school has been relative, like another western keyboard
She was a political vehicle
Another dragged birdcatcher was slightly mentioned
A civilian inspection has been conducted
The secrets were outstanding like the officers
The answers underwear the people
She clicked between the content
It is an unfamiliar explanation
The strangest business has been twisting like a required darkness

and

Neither nonsense settled certainly
It exactly remained
The spirits apologized
I was as patient as the make-believe religion
The automatic hypothesis was included
Features were replaced and broadcast with another grateful businesslike
It would have been pouring

Please note that punctuation has been added manually.

"Schollz/poetry-generator." GitHub, n.d. Web. 13 Dec. 2015.

Scholl, Zackary. "How My Poetry Generator Passed the Turing Test." *Raspberry Pi AI.* N.p., 24 Apr. 2015. Web. 13 Dec. 2015.

"Inceptionism: Going Deeper into Neural Networks." *Research Blog.* Google, 17 June 2015. Web. 13 Dec. 2015.

Merchant, Brian. " The Poem That Passed the Turing Test." *Motherboard.* Motherboard, 5 Feb. 2015. Web. 13 Dec. 2015.

Sipser, Michael. *Introduction to the Theory of Computation.* 3rd ed. Australia: Course Technology Cengage Learning, 2013. Print.

Telondis, Taki. "The Joy Of Salt Licking: Contest Turns Farm Animals Into Fine Artists." *NPR.* NPR, 22 Dec 2012. Web. 14 Dec. 2015.

"Art by Animals Comes to London." *UCL News.* UCL, 27 Jan 2012. Web. 14 Dec. 2015.

Endler, John A. "Bowerbirds, Art and Aesthetics: Are Bowerbirds Artists and Do They Have an Aesthetic Sense?" *Communicative & Integrative Biology.* Landes Bioscience, 1 May 2012. Web. 14 Dec. 2015.