Iperstoria
2024, no. 24
ISSN: 2281-4582

DOI: 10.13136/2281-4582/2024.i24.1547
iperstoria.it

Text Generation and Other Uneasy Human-Machine Collaborations

J. R. Carpenter
University of Leeds
ORCID: https://orcid.org/0000-0002-5701-2796
Email: J.R.Carpenter@leeds.ac.uk

Keywords

Computer-generated text

Variable text

Digital poetry

Poetry

Collaboration

Abstract

Since the rise of mainframe computing, literary authors and critics alike have expressed anxiety about the computer's ability to write narrative prose and poetry as well as or better than humans. With the recent emergence of publicly accessible AI authoring platforms such as ChatGPT these fears may appear to have been well-founded. In this article I will situate contemporary digital literary practices of reading, writing, rewriting, and performing computer-generated variable texts within broader social and historical contexts. Experimentation with generative, permutational, and combinatory text began long before digital computers came into being. How and why does experimentation with generative or in other ways variable text emerge within certain human and machinic generations? How do our attempts to make writing machines help us understand how we write ourselves?

1. Introduction

In human terms, a generation refers both to a group of individuals of approximately the same age having similar ideas and attitudes, and to the period of time between one such a group and the next, which is roughly thirty years. Two and a half human generations have passed since the first generation of mainframe computers emerged in the 1940s. Many more generations of machines have passed since then, each supporting yet more generations of operating systems and softwares capable of being programmed to generate a wide variety of computer-generated texts. This article queries how and why experimentation with generative or in other ways variable text has emerged within certain human and machinic generations. Rather than focus on the literary quality of computer-generated writing, this article offers a two-fold discussion on select, non-comprehensive examples of collaboration between humans engaged in attempting to make writing machines. Firstly, by offering social and historical contextualisation, through literary analysis and archival research, of two well-known first-generation mainframe text generators and their creators from the 1950s. Secondly, by offering a discussion of my own practice-led research methods involving creative experiments with text generation. These experiments are informed by earlier generations of human and machine generators, but they unfold in communication with a new generation of experimental writers operating in the online open-source file-sharing ecology of the 2010s. Rather than pose a narrowly focused research question, this exploration is intended to offer insight from a practice-led research perspective on how past and present attempts to collaborate with machines to generate text might help us understand something about how we write ourselves. Further, this essay attempts to situate these uneasy human and machine collaborations in relation to a wider world.

2. Pre-digital

Experimentation with text generation, permutation, and recombination began long before digital computers came into being. Florian Cramer cites the classical rhetorical figure 'permutatio' as among the earliest Western prototypes of combinatory poetry (2000, 1). Cramer offers as an example *Carmen XXV*, a work of the fourth-century poet Optatianus Porfyrius (2000, 1-2), in which lists of words written in four columns may be arbitrarily combined by the reader to create "1.62 billion possible permutations of the text" (Cramer 2000, 1). Of interest here is the indivisibility of the poem's formal structure, its process, and its contents. According to Cramer, "the poem tells of disharmonic junctions, uneven meters, rough tones and confused words tormenting the singer [of the poem]" (2000, 2). The result is a self-reflexive text which, as Cramer states, "jumbl[es] its own words, performs and confuses itself simultaneously" (2000, 2).

In eighteenth-century England, Jonathan Swift evoked combinatorial textual processes toward critical rather than poetic or performative ends. In Part 3, Chapter V, of *Gulliver's Travels,* Gulliver is taken on a tour of the grand academy of Lagado where a professor demonstrates a machine designed "for improving speculative knowledge by practical and mechanical operations" (Swift 1992 [1726], 137).

> It was twenty feet square [...] composed of several bits of wood [...] linked together by slender wires. These bits of wood were covered, on every square, with paper pasted on them; and on these papers were written all the words of their language, in their several moods, tenses, and declensions; but without any order. [...] The pupils [...] took each of them hold of an iron handle [...] and giving them a sudden turn, the whole disposition of the words was entirely changed. [The professor] then commanded six-and-thirty of the lads, to read the several lines softly, as they appeared upon the frame; and where they found three or four words together that might make part of a sentence, they dictated to the four remaining boys, who were scribes [...] Six hours a day the young students were employed in this labour; and the professor showed me several volumes in large folio, already collected, of broken sentences, which he intended to piece together, and out of those rich materials, to give the world a complete body of all arts and sciences... (1992 [1726], 137-138)

With a machine such as this, Swift goes on to say, "the most ignorant person at a reasonable charge, and with a little bodily labour, may write books in philosophy, poetry, politicks, law, mathematicks [sic] and theology, without the least assistance from genius or study" (1992 [1726], 137). Swift's satire belies a genuine and not entirely unfounded fear that the machine might one day replace human endeavour.

Since the industrial revolution, human labour has been increasingly devalued in large-scale precision work such as weaving, printing, manufacture, assembly, and, indeed, computation. Into the early twentieth-century, the term 'computer' referred to a person who computed, or calculated. The programable digital devices we now call computers do more than that. As Lillian-Yvonne Bertram and Nick Montfort observe in *Output*, what we refer to in English as a computer is called "an *ordinateur* in France and *ordenador* in Spain, highlighting this machine's ability to order or sort things rather than to calculate" (2024, xv). Computers reorder the world.

3. Mainframe experiments

Since the rise of the mainframe computer, literary authors and critics alike have expressed anxiety about the computer's ability to write narrative prose and poetry as well as humans, or better. In 1948, British children's book author Roald Dahl published a short story called "The Great Automatic Grammatizator" in which a machine writes such excellent fiction that its creator soon dominates the field of publishing. In "Can Computers Think," a talk broadcast on BBC Radio in 1951, British computing pioneer Alan Turing appears to express some sympathy for those who might fear being replaced by a computer, imploring: "consider the naive point of view of the man on the street. He hears amazing accounts of what these machines can do: most of them apparently involve technical feats of which he would be incapable" (1951, 6a). Turing does little to quash those fears:

> It might for instance be said that no machine could write good English, or that it could not be influenced by sex-appeal or smoke a pipe. I cannot offer any such comfort, for I believe that no such bounds can be set. [...] Attempts to produce a thinking machine seem to me to be in a different category. The whole thinking process is still rather mysterious, but I believe that the attempt to make a thinking machine will help us greatly in finding out how we think ourselves. (1951, 7)

To détourne Turing's statement slightly, it may also be said that the whole writing process is still rather mysterious, but I believe that the attempt to make writing machines will help us greatly in founding, grounding, and expanding our understanding of how we write. As such, my research places no qualitative bounds on the writing that machines produce. Rather, it focuses on the 'attempt' as an active process and a site of creative thought. To try, in English, is "essayer" in French. This is where the word “essay” comes from. This essay takes a pragmatic approach to reading and writing text generators, one which aims to understand text generation in relation to a larger world.

Text generation was among the earliest forms of creative experimentation with computers, but where did these computers come from? The computer is not one but many inventions taking place on both sides of the Atlantic over the course of a generation. During the Second World War, huge fiscal resources were made available for the development of new generations of machines designed to calculate complex ballistic trajectories and decode encrypted messages at rates much faster that humans ever could. The brute-force Enigma code-breaking Bombes built by Alan Turing and others at Bletchley Park, England, in 1940 – so called because of loud ticking sound they produced – were second-generation machines. They were based on technology that the UK inherited from Poland just before the German invasion in 1939. The Bombes were electromechanical calculators, not computers; they could calculate according to pre-set functions, but they couldn’t be programmed. The Colossus, designed by engineer Tommy Flowers based on plans developed by mathematician Max Newman, built at Bletchley in 1943, was the world’s first digital, programable computer able to handle variable equations. Until 1989, these and other developments in the history of British computing remained obscured by the Official Secrets Act. I offer this historical contextualisation to mainframe computation to emphasise that these are the developments of a generation working towards a common goal, rather than the invention of any one individual.

The two examples of early computer-generated texts which I will now discuss were both programmed in collaboration in the 1950s on mainframe computers barely one human generation removed from a world in which the words ‘calculator’ and ‘computer’ referred to the humans who were assigned to carry out long and laborious computational tasks.

3.1 Stochastic Texts

In *Prehistoric Digital Poetry: An Archaeology of Forms, 1959-1995*, Christopher Funkhouser attributes the creation of the oldest computer-generated poetry to Theo Lutz, who began experimenting with random variation ‘stochastic’ texts in 1959 at the suggestion of his professor, the German writer and philosopher Max Bense (Funkhouser 2007, 37). Lutz’s text generators were based on the logical structures of mathematics. The Zuse Z22 computer he worked on was built in 1956 by Konrad Zuse, a German computer pioneer who had worked entirely independently until he was called into military service. His work was financed by the Nazis during World War II. The Z22 was but one generation in Zuse’s prolific output. The first computer with a core memory based on magnetic storage tape, this machine was particularly well-suited to random variation. Though Funkhouser situates Lutz’s stochastic texts within the literary realm of poetry, in an essay published in Bense's journal *Augenblick* in 1959, Lutz frames his own experiments in scientific terms:

> The Z22 is especially suited to applications in extra-mathematical areas. It is particularly suited to programs with a very logical structure i.e. for programs containing many logical decisions. The machine's ability to be able to print the results immediately, on demand, on a teleprinter is ideal for scientific problems. (Lutz 1959)

In order to situate Lutz's work with text-generation in terms of human generations we must consider two factors. First, the influence of the existential rationalism of Max Bense's philosophy on Lutz's experimentation with computational processes. Bense's print publication of Lutz's essay on his stochastic text experiments helped to canonise Lutz's efforts. It could be said that Lutz put some of Bense's theories into practice.

The second factor we must consider, in relation to the influence Lutz's stochastic texts continue to have on generations of critics and creators of computer-generated texts, is his choice of source text. Lutz's most oft-quoted generator, *The Castle* (1959), basks in a reflected fascination with its source text, *The Castle* (1926), an unusual novel written by hand by Franz Kafka, a now well-known but in his own time unpublished and utterly obscure literary author of a previous generation. The print text of the novel *The Castle* that we now think of as definitive was unfinished at the time of Kafka's death in 1924. Kafka's friend and mentor Max Brod edited the novel heavily before publishing it posthumously, against the author's wishes, in 1926. The formal structure of *The Castle* lends itself to random variation. The unobtainable goal of an ending is inherent in this text. But what if Lutz had selected a different source text by a lesser-known author, or a text from outside of literature altogether? Given the scientific terms he used to describe his own work, would digital literary critics still consider it literary? Although largely rhetorical in nature, one pragmatic approach to following this line of questioning further would be to attempt to apply Lutz's computational processes to a different source text. Towards this end, remix as a research method will be discussed later on in this article.

As its title suggests, Funkhouser's *Prehistoric Digital Poetry* is a history of digital poetry, not inclusive of narrative forms. There is a certain irony in Funkhouser's assertion that the "pursuit of composing poetry by using computer operations began in 1959" (2007, 37) with a text composed "with a very logical structure" (Lutz 1959) from a database comprised of a selection of subjects and titles from a now well-known print novel. The one sample permutation, or output, of Lutz's generator offered by Funkhouser is an English translation (yet another generation removed from the original text). Funkhouser focuses his analysis on "verbal components and [...] what the programs emit" (2007, 32), or, what Bertram and Montfort term output (2024). Cramer argues, “written combinatory literature does not denote the generated text itself, but only a set of formal instructions” (2000, 1).

3.2 Love Letters

Funkhouser makes no mention of Christopher Strachey’s *Love Letter* generator, which, programmed in Manchester, England, in 1952, pre-dates Lutz’s stochastic texts by seven years. In personal correspondence I asked Funkhouser about this omission. He replied: “I was aware of Strachey’s work but did not consider it to be in the realm of poetry – at least as I was willing to define it (which was pretty wide!)” (January 2013).

There is no mention of either Strachey or the Love Letter generator in *Mainframe Experimentalism: Early Computing and the Foundations of the Digital Arts* (Higgins and Kahn 2012). This may be explained, in part, by the avowedly American focus of the scholarship represented in the volume. Further, as the title suggests, this book draws upon and refers to the corpus of digital arts, rather than digital literature. Only one chapter, by Funkhouser, addresses textual practices.

This article aims, in part, to redress this categorically perpetuated omission by contextualising Strachey’s work within the broader conception of ‘the literary’ put forward by N. Katherine Hayles in *Electronic Literature: New Horizons for the Literary* (2008, 45), in which Hayles makes a critical distinction between ‘literature’ and ‘the literary,’ pointing to the latter as having a much broader conceptual framework, within which certain literary hybrids may bridge physical, digital, and performative modes of creation and dissemination through a process Hayles terms ‘intermediation’ (2008, 45). This pragmatic approach takes into account the social as well as historical contexts of the production and public presentation of *Love Letter*, considers the ‘attempt’ or process rather than simply the output, and tests the intelligibility of this early text generator in a contemporary digital literary context.

Strachey was of almost the same generation as Alan Turing, but not quite. Both were brilliant code breakers and code makers, math puzzlers, and playful experimenters, and both were lonely gay men muzzled by post-war retrenchment of homophobic conservative values. In 1952, Turing was prosecuted for homosexual acts, which remained illegal in England and Wales until 1967. Turing wrote the manual for the Manchester University Computer Mark I, for which *Love Letter* was programmed. He gave the manual to Strachey, at Strachey’s request, and, impressed by Strachey’s first attempt to program with it, set into motion the hire of Strachey by the Manchester Computer Laboratory. As Turing’s biographer Andrew Hodges puts it, Turing “handed over the torch” (2012 [1983], 447).

Noah Wardrip-Fruin attributes to Christopher Strachey the “first experiment with digital literature and digital art of any kind” (2011, 302). The texts produced by the *Love Letter* generator fall between forms. Neither poetry nor literary prose, the letters output by this generator sound amateurish, outlandish, and even absurd. The generator uses what Funkhouser terms the "slot" method of generation. Words are chosen from lists, or strings, to appear in set-order sentences. The *Love Letter* generator contains the following variables: Adjectives, Nouns, Adverbs, Verbs, Letter Start. These appear in sentences such as this one: 'You are my (adjective) (noun). My (adjective) (noun) (adverb) (verb) your (adjective) (noun).' The following are two examples of texts generated by *Love Letter:*

> DARLING LOVE
> YOU ARE MY AVID FELLOW FEELING. MY AFFECTION CURIOUSLY CLINGS TO YOUR PASSIONATE WISH. MY LIKING YEARNS FOR YOUR HEART. MY TENDER LIKING. YOU ARE MY WISTFUL SYMPATHY.
> YOURS LOVINGLY,
> M.U.C.
>
> HONEY MOPPET,
> MY FONDEST FERVOUR LONGS FOR YOUR PASSION. MY YEARNING KEENLY LOVES YOUR ENTHUSIASM. MY SWEET YEARNING COVETOUSLY PINES FOR YOUR AFFECTIONATE LONGING. YOU ARE MY ANXIOUS BEING, MY EAGER SYMPATHY.
> YOURS BURNINGLY,
> M.U.C.

The signature, "Yours (adverb), M. U. C. (Manchester University Computer), suggests a deliberate interrogation of the notion of authorship. The posting of print-outs of these love letters on bulletin boards around the Manchester Computer Laboratory constitutes a mode of publication, a social engagement with an audience, or an 'attempt' to, at least. These aspects of the work place it firmly within the literary realm, though the public response was not entirely favourable. Hodges suggests:

> Those doing real men's jobs on the computer, concerned with optics or aerodynamics, thought this silly, but it was as good a way as any of investigating the nature of syntax, and it greatly amused Alan and Christopher Strachey – whose love lives, as it happened, where rather similar too. (2012 [1983], 478)

Wardrip-Fruin cautions, "the resulting letters are not really the interesting part of the project" (2011, 306). Previous to *Love Letter,* Wardrip-Fruin notes, Strachey had created a far more sophisticated draughts-playing program, which is counted among the first computer games. He had the technical skills necessary to program *Love Letter* to generate less-silly-sounding letters, but he chose not to. Instead, Wardrip-Fruin implies, he created a system of deliberate simplicity, a process designed to fail, and to do so humorously.

Basing his analysis solely on the few outputs that have made their way into print circulation, Roberto Simanowski has argued that the *Love Letter* generator "parodied the familiar, sanctioned, conventional ways to express love. The actual meaning of the Love Letter Generator would thus be the deconstruction of love letters" (2011, 94). Based on examination of the source code held in the Strachey Papers at the Bodleian Library, Oxford, Wardrip-Fruin offers a reading based on process rather than output:

> I see the love letter generator, not as a process for producing parodies, but as itself a parody of a process. The letters themselves are not parodies of human-authored letters; rather, the letter production process is a parodic representation of a human letter-writing process. It is not a subtle parody, driven by a complex structures that circuitously but inevitably lead, for example, to the same small set of vapid sentiments stored as data. Rather it is a brutally simple process, representing the authoring of traditional society's love letters as requiring no memory, driven by utterly simple sentence structures, and filled out from a thesaurus. The love letter generator, in other words, was as complex as it needed to be in order to act out a parody. (2011, 316)

Wardrip-Fruin points out, and my own research in the Strachey Papers at the Bodleian confirms, that Strachey had plans for a more elaborate letter generator, which went further toward investigating the letter as a literary form. These plans were abandoned shortly after Turing's death in 1954. Strachey made no further literary experiments. In his classic paper "The 'Thinking' Machine," published that same year, Strachey described the *Love Letter* generator as "a simple type of intelligence test […] The scheme on which it works […] is almost childishly simple" (Strachey 1954, 26). That simplicity does not remove Strachey's 'attempt' to create a writing machine from the realm of the literary.

Strachey's *Love Letter* generator was prefigured by the American novelist Kurt Vonnegut. On 25 November 1950, the American magazine *Collier's Weekly* published a short story by Vonnegut featuring a fictional computer called "EPICAC" which wrote love poetry. The name EPICAC was a direct reference to the ENIAC computer which had been unveiled four years previously, on February 14, 1946, at the University of Pennsylvania. ENIAC cost almost $500,000 to build and was out of date before it was finished. The EPICAC computer reappeared in Vonnegut's novel *Player Piano* which came out in 1952, the same year as Strachey's *Love Letter* generator. It is well within the realm of possibility that Strachey's enigmatic choice of the love letter as a form through which to test the random number facility of the Manchester University Computer was inspired by a work of print literature.

3.2.1 Re-reading Love Letters

Much of the recent interest in Strachey's *Love Letter* generator has come from computer historians and media archaeologists rather than from literary scholars. Writing on *Love Letter* in "There Must Be an Angel: On the Beginnings of the Arithmetics of Rays," German media artist and historian David Link notes that Strachey "performed this experiment a full thirteen years before the appearance of Joseph Weizenbaum's ELIZA, which is commonly – and mistakenly – held to be the earliest example of computer-generated texts" (2006, 16). Neither Funkhouser nor Simanowski mention ELIZA in their discussions of computer-generated texts; it has not generally been considered to be a literary endeavour, but this attitude may shift over time in light of new literary experiments using ChatGPT.

Link's recreation of *Love Letter*, which ran on his own recreation of the Ferranti Mark I, a commercialised version of the Manchester University Computer, was included in the "Old Media" exhibition at Arnolfini, Bristol, England, in September 2010. In that same year, the Museum of Science and Industry (MOSI) in Manchester, England, commissioned Matt Sephton, a Cornwall-based programmer, to create an iPad emulation of *Love Letter*. The press release states: "Visitors at MOSI can now create their own love letters on iPads in the gallery and email them to a loved one" (MOSI 2011). Sephton posted a PHP version of *Love Letter* on his website, and made the source code publicly available on GitHub (Sephton 2010). Examination of Sephton's open-source code revealed a number of inconsistencies between his arrays and those listed in Strachey's notes held at the Bodleian. Firstly, a number of words were missing from Sephton's arrays. Through Twitter, I pointed Sephton toward these, which he then added to his source code. Secondly, I noted that the word 'chickpea' which appeared in Stephton's source code did not appear in Strachey's arrays. Sephton admitted it was his own playful addition; the word remains in his code.

Strachey's technique of using random words selected from the thesaurus prefigures Flarf poetry, which is created by using obscure search terms to mine the internet for found text. Strachey parodies the process of writing love letters full of vapid sentiments by using a thesaurus to fill in formulaic sentence structures. Just as I suggested above – that more might be learned about Lutz's generator by remixing it using a new source text – in an attempt to learn more about the structure of Strachey's *Love Letter,* I followed the formulaic structure of the source code of Sephton's php iteration to create a new text. I replaced all of Strachey's variables with keywords from my PhD thesis, *Writing Coastlines* (Carpenter 2015).

WRITING AND ERASING PASSAGES,
MY PERFORMANCE KEENLY EXAMINES YOUR THESIS. MY THESIS EAGERLY SPEAKS YOUR AIM. YOU ARE MY MEDIA ARCHAEOLOGICAL CONCLUSION, MY DIGITAL LITERARY WORK, MY PRAGMATIC READING.

> YOURS NECESSARILY,
> J.R.C.
>
> NORTH ATLANTIC IN-BETWEEN SPACES,
> MY MEDIATION IDENTIFIES YOUR MARXIST WORK. MY PERFORMANCE RECEIVES YOUR LITERARY EXPERIMENT. MY CHAPTER OUTLINE DILIGENTLY MAKES CLEAR YOUR INTERDISCIPLINARY CORPUS. YOU ARE MY LOCATIVE INTERVENTION, MY COMBINATORY CITATION.
> YOURS ANALYTICALLY,
> J.R.C.

Whether or not scholars of digital literature would consider the output to be silly, or literary, or poetry, the critical intention behind these texts and the computation processes by which they were created can certainly be situated within a practice of text generation which, as evidenced by the examples of Porfyrius and Swift, dates back hundreds of years.

In 2014, Nick Montfort developed browser-based adaptations of a number of early mainframe text generators for *Memory Slam* (2014), including both Lutz's *Stochastic Texts* and Strachey's *Love Letters,* making these literary experiments accessible to a new generation of readers and writers. Montfort terms these 'reading editions' – they do not represent the material aspects of the originals. "These versions are for study, sharing, and learning, too, and you are encouraged to view the source" (2014). A quick peak at the source code of the first generation of Montfort's adaptation of Strachey's *Love Letter* revealed the word 'chickpea,' indicating that Montfort's code had borrowed from Sephton's code. I forwarded Montfort a photography of Strachey's arrays that I had taken during my research at the Bodleian. He removed the word 'chickpea' from his code.

I recount this anecdote by way of demonstrating an instance of collaboration between one generation of humans, softwares, languages, and devices engaged in attempting to reread and remake an earlier generation of text generator. Montfort's open-source 'reading editions' help fill the generation gap between early mainframe experiments and the affordances of contemporary web-programming. Rereading and remixing Strachey and Lutz through these open-source editions helps us understand the way they were written, an invaluable contribution to future scholarship.

4. Python and JavaScript experiments

With the recent proliferation of Artificial Intelligence (AI), it is increasingly common for humans to read computer-generated texts, often with little or no awareness of if, how, or why these texts have been computer-generated. Although a broader discussion of AI is outside the scope of this article, it is important to note that AI systems generate text in a very different way than the text generators under discussion here. In *Atlas of AI* (2021) Kate Crawford articulates this difference through the observation of a critical shift in approach to text generation:

> In the 1970s, artificial intelligence researchers were mainly exploring what's called an expert systems approach: rules-based programming that aims to reduce the field of possible actions by articulating forms of logical reasoning. […] By the mid-1980s, research labs were turning toward probabilistic or brute force approaches […] using lots of computing cycles to calculate as many options as possible to find the optimal result. (2021, 99)

Expert systems approaches were deemed "fragile and impractical in real-world settings, where a rule set was rarely able to handle uncertainty and complexity" (2021, 99). To illustrate this point, Crawford cites yet another example wherein uneasy human-machine relations are explored through a fictional text-generator – a short story by the Polish science fiction writer Stanislaw Lem, called "The First Sally (A), or Trurl's Electronic Bard" first published in 1965. The protagonist, Trurl, attempts a rules-based systems approach, but soon realises that to program an autonomous poetry machine, one needs "to repeat the entire Universe from the beginning – or at least a good piece of it" (Lem 2014, 50). Interestingly, Crawford mistakenly describes Trurl as a man, an assumption easily made from reading this one story in isolation. In the context of the full collection, *The Cyberiad,* it's clear that Trurl is, in fact, a robot – a humanoid machine who travels the galaxy constructing other fantastic machines. As we have seen in each of the other generations of generators discussed in this article, Trurl is not acting alone. His machines are created in competition and thus in dialogue with fellow robot Klapaucius. Trurl's Electronic Bard produces such excellent poetry at such a prodigious rate that the ordinary (human) poets turn against Trurl, though no one else seems to mind. It's the electricity bill that finally prompts Trurl to unplug his invention. This is a prescient plot in light of the vast amount of energy consumed by AI.

4.1 The Two

The remainder of this article will focus on a cluster of examples of modern text generators which take a very different approach to generating literary texts. Rather than "articulating forms of logical reasoning" (Crawford 2021, 99) to strive for optimal results, these generators use rules-based approaches to explore uncertainty itself. Moreover, contrary to the massive scale of Trurl's Electronic Bard, these generators are tiny. These examples are by no means comprehensive; they were selected for discussion because they have been central to my own practice-led research over the past fifteen years.

In 2008, Nick Montfort published "Three 1k Story generators" on *Grand Text Auto,* a group blog about computer narrative, games, poetry, and art (2008b). These tiny programs were initially written in Python and then ported to JavaScript. Each story explores a different approach to text generation. *The Two* (2008a) employs a deliberate formal simplicity reminiscent of Strachey's *Love Letter* generator, albeit executed in an entirely different media ecology, to comment upon the complexity of the operation of gender as a variable. The word 'generation' has the same Latin base as the words 'gender' and 'genre.'

The Two strips the short story down to its most fundamental sections: beginning, middle, and end. As Montfort explains: "A sentence is chosen from a pool of beginnings. A middle is generated by joining 'He' or 'She' to a verb or other middle section and concluding that with 'he' or 'she.' Then, an ending is chosen from a pool of endings" (2008b). Here is one of a near infinite number of possible outputs:

> The police officer nears the alleged perpetrator.
> She berates her.
> Six years later, neither one remembers the incident. (2008a)

Given the power dynamics set out in the first sentence, we may be surprised to learn in the second sentence that both the police officer and the alleged perpetrator are female. Some deeply engrained gender stereotypes may persist from one human generation to the next. From one text generation to the next, however, gender relations may shift radically:

> The police officer nears the alleged perpetrator.
> She berates him.
> Six years later, neither one remembers the incident. (2008a)

Or:

> The police officer nears the alleged perpetrator.
> He berates her.
> Six years later, neither one remembers the incident. (2008a)

Montfort cites a slightly earlier generation of text generators as his inspiration, namely "Nanette Wylde's minimal and clever programs, such as *Storyland* and *about so many things*" (2008b). Wylde's self-described 'Electronic Flipbooks' were created in Director for specific installation contexts between 1998 and 2006. They were not available online until 2012. I first read them on a CD that Wylde sent to me by post. As Wylde describes in the booklet which accompanies this CD, and on her website, "about so many things randomly displays the activities of 'He' and 'She' without bias to gender. That is, the activities are drawn from the same pool of possibilities" (1998, 2012). Anything 'He' can do, 'She' can do. Similarly, *The Two* capitalises on the variability of gender assumptions by making gender a variable: var heshe=['He','She']. The source code of *The Two* is not literally a translation of *about so many things,* but the complexity of the gender variable is born across from one generation of generator to the next.

4.2 TRANS.MISSION [A.DIALOGUE]

In my practice-led research I employ remix, adaptation, and translation as methods for reading as well as writing computer-generated texts. In 2011 I adapted the source code of Montfort's *The Two* to create *TRANS.MISSION [A.DIALOGUE],* a browser-based computer-generated dialogue presented in the form of a conversation, "an encoded discourse propagating across, beyond, and through long-distance communications networks" (Carpenter 2012, 4). Although the nature and form of Montfort's narrative were substantially transformed in the process, the operation of gender as a variable endures as a central narrative imperative in the new text. The line of code "line='The '+choose(operator)+' '+choose(transmit)+'s '+choose(hisher)+' '+choose(condolence)+'s.';" (Carpenter 2011) produces outputs which also play on gender stereotypes:

> The translator conveys her encouragements.
> The administrator relays his congratulations.
> The pilot broadcasts her explanations.
> The receptionist transmits his salutations. (2011)

It must be stressed that we are not dealing with particularly difficult code here. The encoded assumptions about gender alluded to by the stories generated by *The Two* are far more complex than the JavaScript source code which generates them. Roberto Simanowski argues that "the internal problem of this genre of digital literature is its poetics of technology, which replaces a language juggler with a crafter of code" (2011, 91). *TRANS.MISSION [A.DIALOGUE]* thwarts this argument, in so far as the source code was not entirely crafted by me. Technically, my code is crude, a transmutation, a wilful mutilation, a hack (Carpenter 2012). My decision to hack rather than craft code anew was a deliberate one. In *A Hacker Manifesto,* McKenzie Wark argues, "[t]o hack is always to produce the odd difference in the production of information […] by transforming in some way the very process of production" (2004, 222). Just as something of the fragmentary, irresolvable nature of Kafka's *The Castle* underpins Lutz's generator of the same name, something of the uncanny twinning of characters at work in Wylde's *about so many things* and Montfort's *The Two* underpinned my process of production. My hack transforms Montfort's source code into a code medium of sorts. *TRANS.MISSION [A.DIALOGUE]* sends and receives dialogue on and through source code and associated media haunted by generations of past usage.

In 2013, *TRANS.MISSION [A.DIALOGUE]* was translated into French by Ariane Savoie for a special translation issue of *bleuOrange*, a Montreal-based online journal of 'littérature hypermédiatique,' which launched at the Electronic Literature Organization conference *Chercher le texte* in Paris 23-26 September 2013. In personal correspondence Savoie shared thoughts on her process, which I synthesise here. A strict translation of all the English variables into French equivalents would have resulted in subject-verb gender disagreements, the resolution of which would require considerable modification to the source code, which, Savoie felt, would have diminished the variability of the generator and the structure of the piece. Instead, Savoie elected to respect the structure of the source code. Gender conflicts were avoided by the population of strings with variables from only one gender, letting go of any variables that didn't have the exact equivalent in that gender in French. Initially, this resulted in an eradication of the gender variable altogether. Eventually, a compromise was reached in which two versions of certain variable strings were created, that both masculine and feminine proper nouns might be called at different points in the script.

In "Recoding Location and Memory: Finnish Translation of J. R. Carpenter's *TRANS.MISSION [A.DIALOGUE]*," Anne Karhio approaches "the translation process as a case study for considering wider questions of literary translation, particularly in online environments, and in the context of poetic texts in digital interfaces" (2023, 76). For Karhio gender is less of an issue as Finnish has no grammatical gender or gendered pronouns. But, as Karhio notes, "linguistic differences go beyond the technicalities of grammar alone. Natural languages are always historically and culturally situated, and respond to lived experiences and encounters" (2023, 82). In *TRANS.MISSION [A.DIALOGUE]*, place is also a variable. Karhio, Savoie, and I are multi-lingual multi-national women of roughly the same generation. For each of us, the JavaScript imperative '+choose(place)+' will return different results. Here the computer, or *ordinateur* in French, is well-suited to a writing which continuously reorders our relation to place in the world.

5. Conclusions

In querying how and why experimentation with generative or in other ways variable text has emerged within certain human and machinic generations, this article has advocated for moving beyond a literary analysis based solely on the textual output of generators towards a broader consideration of the social and historical context of these textual experiments. It has been argued that a pragmatic approach to reading computer-generated text requires consideration of the complex interrelations between source code and output, between humans and other humans as well as between humans and machines, and between discreet softwares and the global networks of communication and influence within which they operate and propagate. My literary analysis of early text generators has incorporated scholarly and archival research as well as practice-led research methods. This research has been carried out within a generation of like-minded individuals who are not only experimenting with computer-generated text, but also sharing their experiments on the open web. Borrowing from Turing, I have framed these experiments not as attempts to create machines that write, but rather, as collaborations with machines made in an attempt to help us understand how we write ourselves. In the early examples of text generators discussed in this article, human collaborations clustered around specific machines, which themselves emerged in response to specific contexts. In later examples, human collaboration was facilitated by the affordances of the current generation of machines, protocols, and softwares we have at our disposal: a searchable and indexable Internet, open-source code-sharing repositories such as GitHub, the View Page Source function of the modern web browser, and email and other messaging platforms. The hybrid critical approaches and research methods taken in the discussion of these works could be applied to other text generators emerging in other contexts, in other generations, or nations, or language groups, for example. Considering human and machine generations in tandem helps us understand the degree to which the term 'computer-generated text' is a misnomer. However much they operate within the ordered logic of the computer, these are human textual products, publicly circulating within a wide conception of the literary, and so must be written and read in relation to a wider world.

Bionote

J. R. Carpenter is an artist, writer, researcher, and lecturer in Creative Practice at University of Leeds, UK. She has been using the internet as a medium for the creation and dissemination of experimental writing since 1993. She is the author of seven books, four chapbooks, and innumerable zines. For more information visit: luckysoap.com.

Works cited

Bertram, Lillian-Yvonne and Nick Montfort. *OUTPUT: An Anthology of Computer-Generated Text, 1953–2023.* Cambridge: MIT Press, 2024.

Carpenter, J. R. *TRANS.MISSION [A.DIALOGUE].* 2010. http://luckysoap.com/generations/transmission.html. Last visited 24/06/2024.

---. "Translating E-Literature – Translation, Transmutation, Transmediation, and Transmission in TRANS.MISSION [A.DIALOGUE]: [http://luckysoap.com/generations/transmission.html]." *Proceedings of the Conference "Translating E-Literature/Traduire la littérature numérique."* Bibliothèque numérique Paris 8, 2012. https://octaviana.fr/document/COLN0011_2. Last visited 24/06/2024.

---. *Writing Coastlines Letters*. 2013. http://luckysoap.com/generations/coastlinesletter.php. Last visited 24/06/2024.

Cramer, Florian. *Combinatory Poetry and Literature in the Internet*. 2000. http://www.dvara.net/hk/combinatory_poetry.pdf. Last visited 24/06/2024.

Crawford, Kate. *Atlas of AI: Power, Politics, and the Planetary Costs of Artificial Intelligence.* New Haven: Yale University Press, 2021.

Dahl, Roald. *The Great Automatic Grammatizator and Other Stories*. London: Puffin, 2013.

Funkhouser, Christopher T. *Prehistoric Digital Poetry: An Archaeology of Forms, 1959-1995.* Tuscaloosa: University Press Alabama, 2007.

Hayles, N. Katherine. *Electronic Literature: New Horizons for the Literary.* Chicago: University of Notre Dame Press, 2008.

Higgins, Hannah and Douglas Kahn. *Mainframe Experimentalism: Early Computing and the Foundations of the Digital Arts*. Berkeley: University of California Press, 2012.

Hodges, Alan. *Alan Turing: The Enigma*. London: Vintage, 2012.

Karhio, Anne. "Recoding Location and Memory: Finnish Translation of J. R. Carpenter's TRANS.MISSION[A.DIALOGUE]." *Proceedings of the Conference "TRANSLATING E-LIT: WHITHER NOW?"*. Bibliothèque numérique Paris 8, 2020. https://octaviana.fr/document/COLN0025_1. Last visited 11/09/2024.

Lem, Stanislaw. *The Cyberiad: Fables for the Cybernetic Age*. Trans. Michael Kandel. London: Penguin Classics 2014.

Link, David. "There Must Be an Angel: On the Beginnings of the Arithmetics of Rays." *Variantology 2: On Deep Time Relations of Arts, Sciences and Technologies*. Edited by Siegfried Zielinski and David Link. Cologne: König, 2006. 15-42.

Lutz, Theo. "Stochastische Texte." *Augenblick* 4.1 (1959): 3-9. [Eng. Trans. Helen MacCormac, 2005. http://www.stuttgarter-schule.de/lutz_schule_en.htm. Last visited 24/06/2024.]

Montfort, Nick. *Memory Slam*. https://nickm.com/memslam/. 2014. Last visited 24/06/2024.

---. *The Two*. http://nickm.com/poems/the_two.html. 2008a. Last visited 24/06/2024.

---. "Three 1K Story Generators." *Grand Text Auto*. https://grandtextauto.soe.ucsc.edu/2008/11/30/three-1k-story-generators/. 2008b. Last visited 24/06/2024.

Sephton, Matt. "Christopher Strachey 'Loveletters' (1952)." Manchester: Museum of Science and Industry, 2010. http://www.gingerbeardman.com/loveletter/. Last visited 24/06/2024.

Simanowski, Roberto. *Digital Art and Meaning: Reading Kinetic Poetry, Text Machines, Mapping Art, and Interactive Installations.* Minneapolis: University of Minnesota Press, 2011.

Swift, Jonathan. *Gulliver's Travels*. 1726. Ware: Wordsworth Classics, 1992.

Turing, Alan. "Can Computers Think?" *BBC Third Programme* 15 May 1951. [Turing Papers, Kings College Cambridge, AMT B5.]

Vonnegut, Kurt. "EPICAC." *Welcome to the Monkey House.* New York: Dell Publishing Co, 1958. 277-284.

Wardrip-Fruin, Noah. "Digital Media Archaeology: Interpreting Computational Processes." *Media Archaeology: Approaches, Applications, and Implications.* Berkeley: University of California Press, 2010. 302-322.

Wark, McKenzie. *A Hacker Manifesto*. Cambridge: Harvard University Press, 2004.

Wylde, Nanette. *about so many things.* http://preneo.org/nwylde/flipbooks/aboutsomanythings_index.html. 1998, 2012. Last visited 24/06/2024.