Open Library of Humanities

Part of the Ubiquity Partner Network

Digital Studies / Le champ numérique

Research

How to Cite: Tan, Czander. 2020. "The Poetics of Computer Code: Tracing Digital Inscription in Ada Lovelace's England." *Digital Studies/Le champ numérique* 10(1): 3, pp. 1–30. DOI: https://doi.org/10.16995/dscn.355

Published: 03 July 2020

Peer Review:

This is a peer-reviewed article in *Digital Studies/Le champ numérique*, a journal published by the Open Library of Humanities.

Copyright:

Open Access:

Digital Studies/Le champ numérique is a peer-reviewed open access journal.

Digital Preservation:

The Open Library of Humanities and all its journals are digitally preserved in the CLOCKSS scholarly archive service.

Digital Studies /
Le champ numérique

Tan, Czander. 2020. "The Poetics of Computer Code: Tracing Digital Inscription in Ada Lovelace's England." *Digital Studies/Le champ numérique* 10(1): 3, pp. 1–30. DOI: https://doi.org/10.16995/dscn.355

RESEARCH

The Poetics of Computer Code: Tracing Digital Inscription in Ada Lovelace's England

Czander Tan
University of Oregon, US
czandert@uoregon.edu

In this essay, I investigate the development of algorithms from a digital paradigm in Victorian England, specifically through the work of Ada Lovelace and the influences of the Jacquard loom. I consider Lovelace's algorithms through the framework of poetics, that is, how meaning is made and materialized through symbolic inscription. Within the discursive contexts of industrial manufacturing and Romanticism, I find that an algorithmic mode of production emerges from the consideration and inscription of memory. Since the ramifications of inscription and memory echo throughout contemporary computing and the Digital Humanities, examining the logics and paradigms that computational inscriptions reproduce are increasingly vital today. Thus I ultimately argue for the poetic analyses of algorithms and computer code.

Dans cet article, j'enquête sur le développement d'algorithmes venant d'un paradigme numérique en Angleterre victorienne, spécifiquement à travers l'œuvre d'Ada Lovelace et les influences du métier à tisser Jacquard. Je considère les algorithmes de Lovelace par le biais du cadre poétique, c'est-à-dire, comment l'inscription symbolique crée et matérialise le sens. Dans les contextes discursifs de la fabrication industrielle et du Romantisme, je trouve qu'un mode de production algorithmique émerge de la considération et de l'inscription de mémoire. Puisque les ramifications d'inscriptions et de mémoire résonnent au sein de l'informatique contemporaine et des Humanités numériques, examiner les logiques et les paradigmes que les inscriptions computationnelles reproduisent devient actuellement de plus en plus primordial. Ainsi, je plaide en fin de compte en faveur des analyses poétiques d'algorithmes et de codes informatiques.

Keywords: digital humanities; computer code; poetry; poetics; ada lovelace; analytical engine; digital; jacquard loom; romanticism; victorian

When we think about the digital, that is, in "digital technology" or "digital humanities," what often come to mind are computers, speed and efficiency, and the electronic ethereality of such entities as the Internet and the Cloud. However, at its root, what "digital" refers to is how we handle information. To be digital is to handle information digitally, that is, in digits; specifically for us, those are ones and zeros. Calling our century the "Information Age" hints at this attention to information, though the label points more to our economic basis rather than our digital paradigm. After all, we have always handled information and attempted to make meaning from it; we have language and we have poetry. But what does the digital paradigm entail? What meaning do we make from ones and zeros? And how might it have developed?

In 1804, the textile industry was revolutionized by the Jacquard loom, a machine capable of interpreting holes punched on paper cards. These holes articulated patterns that were then visualized on woven textiles. Besides speeding up manufacturing processes, automation by the Jacquard loom also expressed a digital paradigm: the information of textile patterns was inscribed into a series of values based on whether or not a punched hole was present in a certain read order. An inscription based on "whether or not" is the foundation of the binary language. Paradigmatically speaking, information was encoded into a logical series of presences/absences and true/false. Our algorithms, and subsequently computer code, thus developed from this paradigm for handling information.

The Jacquard loom influenced the work of Charles Babbage and Ada Lovelace to design what is now recognized as the first computer and first computational algorithm, respectively. Lovelace's Notes, which contain this algorithm, explicitly discusses the mechanism of the Jacquard loom and its punched cards (See Appendix A for an image of Lovelace's algorithms, in which she writes specifiable steps for the Analytical Engine to calculate the Bernoulli numbers). But is the punched card, or its informational paradigm, the only thread linking the loom and the computer? Or does that historic tie evoke a more complex and culturally nuanced relationship that speaks through the punches on the card? In order to answer these questions, we would have to look at the discourses that place machines and their algorithms

within the context of sociocultural forces, such as the industries involving looms and textiles. I propose that considering the contexts that surrounded Lovelace's work, namely the mechanization of design and Romantic theories on the materiality of language, can give us insight into how computational algorithms developed from a digital paradigm. By looking at the material and sociocultural contexts surrounding algorithms, my essay also seeks to show the affinity between computer code and poetics.

After situating algorithms within the field of Digital Humanities, I consider the mechanization of weaving in early nineteenth-century England and the production of Kashmiri shawls with Jacquard looms. From there I consider the modes of shawl production in relation to Romantic theories of Logos, proposing this combination as a cultural context with which Lovelace's work would have been conversant. Logos, or incarnational language, was a poetic project of the time, pursued particularly by Coleridge, thus providing a theoretical parallel to Lovelace's poetical science. I argue that Babbage's Analytical Engine gave Lovelace the impetus to apply mathematics in a way that shifted engagement with machines from a paradigm of calculation to that of computation.[1] This shift came from Lovelace's use of indexing, which effectively inscribed the material *memory* of the machine. By noting memory, that is, the storage of information, Lovelace took the digital paradigm of punched cards further and effectively developed computational algorithms. Information was not only encoded into binary but was then inscribed into specifiable steps. The designing of steps necessarily takes into account the matter of time (for one step must always come *before* or *after* another), and the way Lovelace dealt with time was with indices. As my conceptual basis for indexing, I draw from Susan Stewart's take on *deixis*, which she grounds in poetic theory. I then analyze the use of indexing in the third canto

[1] That is, Lovelace's mathematics present a "set of specifiable steps that produce an output" (Schmidt 2016). In algebra, calculative steps do not need to be specifiable because they were not being performed by a machine. Rather, a student of mathematics learned the principles behind a mathematical process, which made the specification of steps redundant. For a machine, however, computation follows a series of steps, regardless of whether it understands the underlying principle of those steps.

of Lord Byron's *Childe Harold's Pilgrimmage* to provide a foundation for comparison with Lovelace's mathematical indexing. The inscription of memory through indexing shows the shift in how information was handled during Lovelace's time, thereby linking the digital language of punched cards with algorithmic productions in such current commonplaces as text messaging and social media.

In our century, the inscription of memory has been veiled by layers of code (e.g. binary, machine code, compiled, etc.) unintelligible to the general public. Practically speaking, however, we can consider these inscriptions of memory in the kinds of activities algorithms currently produce, from simple email and word processing documents, to weather and dating apps, to the complex infrastructure of the World Wide Web. All these forces shape our experiences, and, in turn, are shaped by the algorithms we write. But how much do we understand about the operations of coding languages—when we swipe down to scroll on our Instagram, when we speak a command to Siri, when we follow our GPS systems—that these are all algorithmic operations expressed by computer code? Do we understand how our experiences have been shaped by a digital paradigm, by languages we never see or acknowledge?

In her book *Coding Literacy: How Computer Programming is Changing Writing*, Annette Vee urges us to develop basic literacy of coding languages. She considers code "a socially situated, symbolic system that enables new kinds of expressions as well as the scaling up of preexisting forms of communication" (Vee 2017, 3). Put another way, computer code is a language derived from our contemporary social contexts, itself operating in a new medium of expression that also forms other expressive media. To disregard coding literacy, then, would be to concentrate that knowledge base among the few who have the power to use it. Thus, understanding the relations between algorithmic operations and our lived experiences is a critical issue that can result in wider socioeconomic gaps if political or pedagogic action is not taken. Benjamin M. Schmidt agrees that computer users should develop some basic literacy of coding languages and algorithms that drive human interactions on the computer. In his words, algorithms are a "set of precisely specifiable steps that produce an output" (Schmidt 2016, n.p.). And because the process by which these steps come about are conceptual, they precede inscription into algorithmic language,

thereby making them independent of the language.[2] Put differently, Schmidt proposes that algorithms are developed independently of the coding languages used to express them. According to Schmidt, we need only be literate in those conceptual processes, not necessarily the coding languages themselves. Vee, however, asserts that other kinds of technological literacies (such as software or online network literacy) are built on our literacy in computer code. That is, our understanding of how various software and networks function and interact with each other is intrinsically tied to our understanding of their constituent language. In essence, Vee gives us the sociocultural impetus to become literate in coding languages, and Schmidt argues that one way to become literate in code is to develop a conceptual knowledge of what algorithms do.

My line of questioning attempts to integrate the claims of Vee and Schmidt to argue that algorithmic thinking not only produces coding languages, but is concurrently reified and further developed by coding languages. The relationship between the symbolic language of algorithms and its material effects and contexts is a poetic one: a way to make meaning of our technologies. I take Vee's proposition of code as "a socially situated, symbolic system" (Vee 2017, 3) and investigate its situatedness as grounds for meaning-making, beginning with the algorithm that Lovelace developed for Babbage's Analytical Engine. To be clear, I am not implying that Lovelace's work led to modern day computation.[3] We would be hard-pressed to trace a direct line from Lovelace and Babbage's work to Howard Aiken's Harvard Mark

[2] For example, a common method in distant reading practices is "term frequency and inverse document frequency," or tf-idf. This method receives a text and returns the frequency certain words appear in the text, as well as the relationship this frequency has to other words in the text. If a word, say, "computer" is frequent in a certain part of a text but not in another, the researcher might find this significant. Schmidt's point is that digital humanists need to understand that tf-idf transforms the information of a given text in this way, but that they do not necessarily need to understand the calculations and code that underlie this transformation.

[3] Much scholarship on Lovelace's work has focused on the biographical circumstances that surround her work (Dorothy Stein, Doron Swade, Betty Alexandra Toole, James Essinger, Benjamin Woolley, Joan Baum, Christopher Hollings). While my essay takes Lovelace's biography into account, it also attempts to give credence to the sociocultural contexts in England at the time. Lovelace's Notes, in this essay, is as much a work of intention as it is a symptom of her milieu--it is a "case": a set of choices made with and against its contexts.

1 in 1944. If Lovelace's Notes contain the first recorded computational algorithm, however, we can still consider the contexts in which it emerged, making generative comparisons with the contexts surrounding the development of code in the 1940s and new media today. Why is it that algorithms do what they do? Why do they, as Vee suggests, "[enable] new kinds of expression" (3)? And how does this happen?

To be sure, the most immediate context for the development of algorithms is the Industrial Revolution.[4] But, more specifically, how did Lovelace's mathematical language adapt to the Analytical Engine within her context? To answer these questions, we need to locate particular material shifts during the Industrial Revolution to understand how they affected symbolic systems. In other words, we must investigate the poetics of algorithms. Dennis Tenen, in *Plain Text: The Poetics of Computation*, defines the method of computational poetics as a "strategy of interpretation capable of reaching past surface content to reveal platforms and infrastructures that stage the construction of meaning" in a computer environment (Tenen 2017, 6). Ultimately, Tenen's main analysis remains with digital texts and how its underlying computational infrastructure affects the way users read and interpret digital productions (which we inherit from print culture).[5] I take his project further by considering the algorithmic paradigm that underlies our computer code,

[4] As R.C. Allen (2011) argues, "the eighteenth century does represent a decisive break in the history of technology and the economy. The famous inventions--the spinning jenny, the steam engine, coke smelting, and so forth…mark the start of a process that has carried the west, at least, to the mass prosperity of the twenty-first century" (357). Particularly, he finds that "Britain's unique wage and price structure was the pivot around which the industrial revolution swung…prompt[ing] the technological breakthroughs of the eighteenth century by increasing the demand for inventions that substituted capital and energy for labour, and by generating a population that was exceptionally able to respond to those incentives due to its high rates of literacy, numeracy, and craft skills" (359). Ultimately, Allen shows these elements to be catalysts that altered the social fabric in England in such a way that reality during the Victorian era was quite distinct from the late eighteenth century.

[5] Tenen's (2017) project is similar to Matthew Kirschenbaum's consideration of how word processing software affects the production of texts in *Track Changes: A Literary History of Word Processing*. In a review of Tenen's book, N. Katherine Hayles considers the implication of whether literary scholars must now be competent programmers to analyze digital objects. She complicates the issue from a simple "yes or no" to proposing that Tenen's contribution is ultimately a methodology with which digital texts can be analyzed more in-depth. To be sure, I do not take digital texts to be my objects of analysis (as do Tenen and Kirschenbaum). Instead, I consider computer code itself as literary text, my project in this essay being to trace where its poetics manifested.

applying print-culture reading methods to digital language. Examining Lovelace's mathematics in relation to its contexts, my focus on the lens of poetics establishes a framework through which to consider the relationship between symbolic inscription and material effect, that is, the relationship between form and function. My hope with establishing this framework for reading the poetics of code is not just to encourage poetry readers to become more literate in coding languages, as Vee warrants; nor is it to understand what algorithms do, as Schmidt suggests. Rather, my argument is that a study of the development of algorithms in specific cultural and material contexts (Britain in the first half of the nineteenth century, for example) would make us more discerning of the limitations and affordances of computer code in our own historical moment. Since the languages we code affect the ways we think, at least computationally, we ought to think about the codes we language.

On one hand, this essay takes seriously Schmidt's call for recognizing the role of algorithms in research, but on the other deviates from his recommendation for digital humanists to "understand the transformations that algorithms attempt to bring about" (Schmidt 2016, n.p.). Understanding the transformations alone is not enough; after all, the language we use to inscribe transformations shape those transformations just as much as they work to produce those transformations. In other words, I deviate from Schmidt's premise that the concept of the transformation precedes the writing of the algorithm by considering instead how writing and conceptual transformations are mutually constitutive. To do this work we must understand why those transformations came about in the first place, the relationship between their symbolic inscriptions and material contexts, that is, their poetics. Ultimately, I attempt to establish another angle from which Digital Humanities can move through the "black box" of computation from such methods as distant reading or sentiment analysis. If sentiment analysis developed from the methods of scientific computing practices, to give a crude example, what are its limitations in answering questions from the humanities? Considering sentiment analysis through a framework of poetics would consider the relationship its outputs have with its algorithms, giving us a way to interpret and question the choices made within the contexts of its development. In terms of computer science, a poetics lens would make

coders more aware of the material effects their symbolic inscriptions have on both the machines that read their work as well as the people with which those inscriptions will eventually interface.

Weaving digital threads

Ada Lovelace seems to be the most natural point to begin with, not just because she famously wrote the first computer program, but also due to her historical situation. The technological industrialization that Britain experienced in the first half of the nineteenth century shifted its cultures of production in a fashion much like the influence of digital technology and the Internet today. One example of this shift in Regency-era Britain manifests in productions of the Kashmiri shawl. Historian Chritralekha Zutshi considers the Kashmiri shawl at the intersection between industrialized manufacturing in Britain, interactions with imperial subjects, and domestic responses through fashion and literature. British colonists would travel to parts of south and central Asia to govern imperial subjects or extend economic relations, often returning with gifts, such as the Kashmiri shawl. These luxurious items were romanticized by travel narratives, which made them more valuable due to their exoticism.

The increasing demand for these so imagined exotic Kashmiri shawls, especially among members of Lovelace's aristocratic circles, incentivized British factories to manufacture imitations. While the domestic textile industry attempted to compete with the authentic fabrics brought back through marketing rhetoric, the factory owners found ways to cut production costs so that their imitations could be sold at lower prices. As Zutshi (2009) notes, the English town of "Paisley was most successful in this endeavor, since its weavers used new weaving mechanisms, such as the harness and later the Jacquard loom" (424). In addition to technological innovation, adopting machinery for commercial growth was also a result of the Industrial Revolution. In that case, the revolution was not so much in favor of societal progress as it was advocating for cheaper labor costs.[6] This aspect of British society was not

[6] R.C. Allen (2011) argues that during the Industrial Revolution, "[t]he price and wage structure affected the *demand* for technology by giving British businesses an exceptional incentive to invent technology

lost on Lovelace since her mother, Lady Byron, had investments in such business. Lovelace even accompanied Lady Byron in 1834, as Essinger (2014) explains, "on a tour of the industrial heartland of northern England, visiting many factories and seeing with their own eyes the immense potential of machinery. They saw a Jacquard loom in action, and Lady Byron even drew a picture of a punched card used to control the loom's operation" (121). At this point, the Jacquard loom was a manufacturing commonplace; there were several hundred in Britain by the mid-1830s. The presence of Jacquard looms in the country had less to do with their intrinsic innovative value than with their ability to meet market demand for products, such as imitation Kashmiri shawls. Along with romanticized imperial narratives that fed upon and intensified this demand, the changing economic reality would provide the weft from which a new form of expression would be spun through the threads of Lovelace's computational mathematics.

Although the intended function of the Jacquard loom was to automate pattern-weaving for textiles, the form by which this was done (that is, through punched cards) shaped the way information was written and understood. Information was reduced to whether or not a hole was present on a given card: a value either true or false: binary. In the context of textiles, however, a pattern woven by the Jacquard loom did not just use binarized information, it also proposed that patterns were made up of binary codes. In other words, the symbolic inscriptions of the punched cards encoded material patterns such that the relationship between symbol and material produced a digital paradigm. As Essinger points out, "The portrait of Jacquard…was essentially a digitised image. Made using 24,000 punched cards, it wove an image of the inventor of the loom on which it was woven" (132). That is not to say this true/false logic was wholly novel; philosophers and logicians had been writing about it since Plato. It is the use of binary logic in such material endeavors that was new, a

---

that substituted capital and energy for labour. The high real wage also stimulated product innovation since it meant that Britain had a broader mass market for 'luxury' consumer goods including imports from east Asia. The *supply* of technology was also augmented by the high real wage. It meant that the population at large was better placed to buy education and training than their counterparts elsewhere in the world…[which] contributed to invention and innovation" (358).

cultural blueprint for the Boolean algebra and computational methods that would follow (In fact, George Boole wrote *The Mathematical Analysis of Logic*, the work that first introduced Boolean logic, in 1847, five years after Lovelace had written her Notes).

In one sense, the context of the Jacquard loom, that is, automated and machine-made weaving governed by human-made patterns, gave concrete boundaries for innovation. The inventor and its adopters refined it for simple and practical purposes. On the other hand, the context of weaving also provided boundaries against which to push. Innovators such as Babbage and Lovelace saw the potential of the mechanics of the Jacquard loom for handling other industrial processes, leading to the design of the Analytical Engine. For Lovelace, however, the context of weaving held a metaphorical sway as well. What kinds of information would the Engine be able to process and manipulate? What kinds of patterns would the Engine be able to weave? These "patterns" need not be relegated to aesthetics, that is, architectural designs or musical compositions (the latter of which Lovelace would also come to imagine). Rather, these "patterns" come from the process of weaving itself. The significance of the Jacquard loom on Lovelace's work was not just in its potential for automated production, but the relationship between this automation (how it processed the holes in punched cards) and the context of weaving.[7] Just as the Jacquard loom was not a weaver of clothes or shawls but was in fact a weaver of threads and holes, the Engine would not merely be a weaver of music or poetry (or apps and software). Instead, the Engine (especially as we consider its successors today) is a weaver of words, of language.[8]

---

[7] Regarding historical usages of algorithms, Caroline A. Jones writes: "For Victorian engineer Charles Babbage…difference was to be automated for minimum error and maximum utility in the struggle to produce invariant numeric tables for the use of scientists, navigators, and surveyors, but of most burning necessity for actuaries in the burgeoning new commercial insurance trade" (Jones and Aranda 2009, 28).

[8] Essinger (2014) argues that this was the significant difference between Lovelace's perspective and Babbage's, whose focus was largely on the mechanics of the Engine. He writes: "When we look carefully at Babbage's writing style compared with Ada's, we are driven to the conclusion that *he* saw the world, and mechanisms, in a much more literal, factual and – indeed – *analytical* way than she did…But the real point – and this explains why Ada's contribution to the idea of the Analytical Engine

My point here is that the loom was not just significant in the history of computing due to its mechanics, but also due to the connotations that surround weaving. The significance of the Jacquard loom differs from machines before it, such as the printing press or the steam engine, because the craft of weaving bears a metaphorical affinity to the process of *poiesis*. As Lovelace herself has often been quoted writing, "[w]e may say most aptly, that the Analytical Engine *weaves algebraical patterns* just as the Jacquard loom weaves flowers and leaves" (Toole 1991, 248). We could trace the connotations of weaving from mythologies such as Zhinü, a heavenly maiden from Chinese folklore who wove robes out of clouds for her father, the Jade Emperor, to when Penelope wove and unwove the burial shroud to delay suitors in the *Odyssey*. If we connected these connotations to the basis of automated weaving, we could similarly think of computer coding as a kind of weaving and unweaving. Just as woven images and poetry can be read and interpreted, so can the punched cards, Lovelace's mathematics, and our resultant computer code. Thus, the expressions of code through, say, a smartphone application, can also be read and interpreted within their material contexts.

From Victorian England, Zutshi gives us more concrete examples of these kinds of interpretations through the discourses surrounding the Kashmiri shawl. The marketing strategy of the textile industry capitalized on the distinction between hand-woven Kashmiri fabrics and machine-woven British imitations. One kind of weaving was pitted against the other.[9] On one hand, shawl narratives, such as Charles White's novel, *The Cashmere Shawl: An Eastern Fiction* (1841), presents a narrative about the origins of the popular commodity. White's novel, specifically, has the Kashmiri shawl as the narrator, telling "the story of its own becoming: from the hair on the underbelly of a peacefully grazing shawl goat in the mountains of

---

is so important – is that *the brilliance of the conception of the Analytical Engine requires both a scientific and emotive perception if it is to be fully understood and expressed*" (141).

[9] For example, William Moorcroft, a consultant and agent for the East India Company, concluded that "the techniques involved in the production of shawls could not be allowed to remain in the hands of Kashmiri weavers--whom he described as ingenious but oppressed and fraudulent--they had to be systematized into scientific knowledge through an imperial mediator so as to be more effectively utilized by British industry" (Zutshi 2009, 428).

central Asia to fleece in the markets of Yarkhand, to yarn woven into an exquisite shawl in Kashmir, to the shawl making its way through kingdoms in British India" (Zutshi 2009, 430). Taken this way, the shawl does not merely find its authenticity by being hand-woven, but through its ability to integrate, or weave together, multiple geographical locations. The fact that this process could be replicated by European machinery, such that one could hardly tell the difference between a Kashmiri shawl and a British imitation, was presented as proof that the mysteries of weaving had been mastered by industrial ingenuity. For example, Harriet Martineau, touted as the first female sociologist, wrote an essay titled "Shawls" in 1852 to argue for the superiority of British shawls precisely because of their industrial origins.[10] The civilized British were argued to have had mastery over what the primitive Asiatics did not, represented by automated machinery. Speed and efficiency of production were merely the result of the ability to command machines, such as the Jacquard loom, to do the user's bidding. In other words, what made British weaving superior to that from their Asian counterparts (reinforced by the imperial context), was control over matter.

We have seen that this control was deployed through symbolic inscription, specifically through punched cards.[11] One discourse that explicitly theorized about the relation between inscription and matter, however, was the poetics of Romanticism. Coleridge, particularly, drew his ideas from the notion of Logos as God, that is, the word as creator. On this basis, he believed that matter was made up of

[10] Zutshi (2009) writes: "While she described the Kashmiri weaver as toiling over a primitive loom, weaving shawls with patterns that had been passed down to him through generations, Martineau's description of the shawl production process in Paisley evoked a sense of constant motion, demonstrating the genius of English industrial innovation" (434).

[11] Within an imperial context the argument could be extended to legal and bureaucratic discourses as well. Control over matter manifested in imperial rule and the bureaucratic governance of British colonies, reinforced by the industrial innovations that were produced by such a "rational" culture (as opposed to their colonial counterparts, who were stuck in old traditional ways). If we transferred this conversation onto the issue of coding literacy, as was mentioned through the work of Vee previously, we might see a similar power dynamic between the technologically literate and those who are not. In these cases, the distinctions are not made on national lines so much as corporate and material ones. In other words, Apple is now an "Empire on which the sun never sets." The time told on an iPhone is superior to a wristwatch precisely because of its digital nature.

language, which had mathematical and algebraic roots.[12] His project, then, was to figure a language to render reality. Imogen Forbes-Macphail (2016) argues that such strains of thought converge in Lovelace's work, writing: "Lovelace perceives, in the Analytical Engine, some incarnational or materializing possibilities that Coleridge and Byron yearned for in literature…The Analytical Engine incarnates mathematical operations in the movements of its physical, metallic structure… Mathematical symbols and operations are manifested in matter and movement; simultaneously, reciprocally, the material Engine becomes capable of signifying" (152–153).[13] I suggest that this incarnational poetics is evident in Lovelace's union of the symbolic

[12] Imogen Forbes-Macphail (2016) writes: "In Coleridge's system, a pun, a rhyme, or an instance of alliteration or assonance is not a mere idle likeness, but an expression of symbolic relationship between two words, which facilitates the excavation of an immense history of changing meaning, and is therefore legitimate grounds for argument, infusing these poetic techniques with a significance which goes beyond a merely aesthetic quality" (144).

[13] Betty Alexandra Toole (1991) also argues that Lovelace's mathematical understanding show "the so called "analog" or poetical skills of imagination, visualization, patterning and the use of metaphor" (61). Dorothy K. Stein (1984), however, criticizes Lovelace's mathematical understanding as "essentially intuitive and mystical" (63). Stein's project on Lovelace is an historicist one, where she concludes that Lovelace's work was "more a reflection of…the political purposes of the inventor [Babbage], and, above all, of the social and cultural context in which it was written" (34). Toole argues against Stein and says that the archive that Stein draws from "are a skewed sample: they represent what Ada did not know, not what she did know" (Toole 1991, 64). That is, the pieces that Stein uses in her article, and in her later work *Ada: A Life And A Legacy*, were used to construct a story of Lovelace that criticizes what she never aimed to do in the first place. Although historical and cultural contexts are important for understanding the philosophies that Lovelace draws from, as we have been considering in this essay, Stein's argument falls into the historicist trap of trying to fit Lovelace into what was normative of the time. When Lovelace didn't fit, Stein criticizes her for it. In fact, it is her unfittingness, her "mathematical uncertainty" (Stein 34), that allowed for the paradigm shift in mathematics towards computation. As Ellen Moll writes: "many narratives of Lovelace, both fictional and biographical, tend to portray her as refusing to stay put in her own century" (Moll n.p.).

In Lev Grossman's (1998) review of Stein's critique on Lovelace, he notes that "[a]nother of Stein's criticisms hinges on the question of whether or not the Analytical Engine could be made to perform symbolic manipulations" (67). That is, Stein argues that Lovelace did not understand the representational capabilities of the Engine. But, as researchers John Fuegi and Jo Francis point out, symbolic representation by the Engine do not relate to Lovelace's contributions to computer science. Instead, it was her conception of "symbolic substitution" (Fuegi and Francis 2003, 16) affecting the material machine that revolutionized computing. Fuegi and Francis also criticize Stein's claim that Lovelace was mathematically inept (Fuegi and Francis 26n29). What they find in the correspondences between Babbage and Lovelace "suggest[s] that [even Babbage, the inventor of the Analytical Engine] had learned something new about his own machine from Lovelace's queries and speculations" (21).

and the material in her operational methods for the Analytical Engine. This union manifests in her mathematical notation, specifically in her use of the superscript to index how many times an arm of the Engine has been used. The index ultimately moves the Engine beyond mere moment-to-moment enumeration or calculation, as might be done by an abacus or a clock. In other words, Lovelace's use of indexing inscribes the *memory* of previous activities for the Engine, thus re-conceiving the machine into a computational being.[14]

The poetical science of memory-making

If we follow Coleridge's theory that poetry attempts to use language to render reality (not just represent it), then poetry can also be said to inscribe memory. Thus, the activity of inscription denotes a double duty: something is carved into existence (that is, created or made), and because of its symbolic form, it is also representational and readable. In contemporary academic terms, Susan Stewart (2002) writes: "Poetry is both the *repetition* of an ontological moment and the *ongoing process or work* of enunciation by which that moment is recursively known and carried forward" (15). The repetition of moment refers to the representation and reading of poetry, where the reader enacts an experience by reading inscription, while the "*ongoing process*" refers to the act of inscription (or "enunciation") in the now, by which the enactment is inscribed. To explain one way poetry does this, Stewart uses the rhetorical concept of *deixis*, which refers to linguistic shifters whose meanings only emerge in a given context. Deictic terms, or indices—such as "I," "here," or "now"—are not just context-dependent for signification, but they also set the bounds for the contexts themselves, "[e]mphasizing the bringing forth of forms over notions of imitation and representation" (150). The word "now," for example, refers to the temporal moment of its inscription. As an indexical reference, however, the same "now" read in the future would both refer to a past moment (the moment of inscription) as well as the

[14] I use the term *memory* here for its significance in both the humanities and computer science. In fact, the engineering take on computer memory has quite an affective and physiological perspective, while poetry has manifested in forms, such as villanelles and ghazals, that bear resemblance to computational operation like iteration, recursion, and while-loops: both seek to replicate or create some process of *memory*.

present moment of its being read. That is, the "now" is enacted by the reader. Thus "now" functions both as a symbol of the present and as an index that refers to a past present. This double duty of the deictic presents the *index* as both inscription and reference.

For a more concrete example, let us take a serendipitous look at an excerpt from the work of Lovelace's father, the poet Lord Byron. In the third canto of *Childe Harold's Pilgrimmage*, Byron (1965) begins by addressing his daughter: "Is thy face like thy mother's, my fair child!/ Ada! sole daughter of my house and heart?" (50). In stanza 115, towards the end of the canto, he addresses her again, almost as a kind of invocation:

> My daughter! with thy name this song begun;
> My daughter! with thy name thus much shall end;
> I see thee not, I hear thee not, but none
> Can be so wrapt in thee; thou art the friend
> To whom the shadows of far years extend:
> (115.1–5)

The stanza starts by highlighting the textual distance between Byron's mentions of Lovelace in the first stanza and that in the current one. To be exact, they are 113 stanzas apart. Perhaps he does this to imply that she has been in the back of his mind through the whole text, through Rousseau and Napoleon and the Alps, and so underlies the themes of Childe Harold's philosophical wanderings. The reminder of Byron's relationship with and separation from Lovelace recalls the memory for the reader. That is, the phrase "with thy name this song begun" reminds readers of the first stanza, and by doing so, collapses the distance of 113 stanzas. This collapse is further amplified by the repetition of "My daughter!" in the next line, the first iteration being connected to "begun," the second to "end." In a sense, Lovelace serves as an *index* for the reader, indicating iterations to mark how the relationship between Byron and his daughter has changed from the beginning. It is ironic, however, that Byron does not explicitly mention his daughter's name as he did in the first stanza, even though he writes "with thy name thus much shall end." Perhaps this is one of

the changes he wants the reader to mark: that his passage through metaphysical pilgrimage has brought the weight of separation from his daughter to be too much for Byron to bear that he cannot even write her name. Instead, he calls her "the friend/ To whom the shadows of far years extend." There is much we can analyze in the syntactical and connotative play among "friend," "shadows," "far years," and "extend," but for the scope of this essay, I will use "far years" as a deictic basis.

With that particular phrase, Byron challenges our conception of space and time and leads us to a new way of thinking about their relationship. We might conventionally talk about the length of years in terms of duration, but describing it with "far" is a move that questions the distinction between space and time. Just as Byron collapses 113 stanzas of his text with iterations of "My daughter!", "far years" collapses the spatial into the temporal through the grammatical relationship between noun and adjective. The temporal noun is described with a spatial term, making time seem concrete and reachable. That is, the denotation of "far years" indexes our conventional conceptions of time and space, while its form inscribes their convolution. If we might eventually reach a far land having sailed enough distance, we might also be able to reach "far years." But how does this apply to Byron and Lovelace? For one, they were separated by physical distance, Lovelace growing up in England while Byron wandered the Continent. They were also separated by time, for it had been months since Byron last saw her. In another sense, however, the use of the temporal period "years" indicates that Byron also considers the separation between his present time and Lovelace's, which suggests his speculation that she will eventually read this line in the future, in Lovelace's own present. Byron creates an imaginative distance between father and daughter for the reader: he roams the Continent and writes to his daughter while she, perhaps years later, reads her father's poetry.[15] In other words, "far years" repeats Byron's ontological moment of longing while enacting the ongoing process of carrying the years further and further away.

[15] Jerome Christensen (1995) describes the extent of Byron's international celebrity and textual popularity, so this speculation is not too far removed from reality. As Imogen Forbes-Macphail (2016) shows in her review of Lovelace's correspondences, "[w]hile Lovelace would later write that it was impossible to "think of my Father's memory with feelings of respect or veneration," she nevertheless

The quick connection we can make to computer code, in terms of *deixis*, is the function of code as pointers. A more technical definition of this can be found in computer science, but conceptually speaking, the function of any term or variable in code is largely to point towards where a value is stored in the computer's memory. Code, then, basically represents addresses for memory storage or figurative indices. Just as "far" and "years" are addresses to conceptions of space and time in our memories, coding terms such as "while," "print," and "return" index various computational processes. Similarly, the syntactical form of "far years" convolutes our conceptions just as much as data structures and algorithms manipulate computer memories. We can see, then, that the affinity between poetics and paradigms of computation is based on the relationship between inscription and material context. I posit that this affinity is expressed in Lovelace's use of indices in her mathematics, inspired by the innovations in new weaving practices of the time.

Before we examine specific instances of how Lovelace's mathematical notation employs this poetics of writing memory, it is important to understand the machine for which these notations were written. The Analytical Engine, known as the first designed digital computer, was designed by Charles Babbage in 1837. Its main body was a "mill" which contained complex arrangements of gears and a number of mechanical columns, or arms, on which the numerical values to be calculated would be inscribed (See Appendix B for an image of Babbage's trial model of the Analytical Engine). The operator would provide instructions to the Engine by means of hole-punched cards, inspired by the Jacquard loom, which contained the binary information of the elements of mathematical expressions. The difference between the format of these expressions and that found in conventional mathematics is that the former had to directly affect the material movements of the Engine; equations had to be performed by the machine, rather than remaining in the abstract. The first attempts at computational expressions that considered the materiality of the Engine

---

maintained that he had transmitted to her the "flower of his characteristics," and "bequeathed" her a task so that she might "make a compensation to mankind for his misused genius," all of which suggests that she had familiarized herself with his work" (146).

are found in L. F. Menabrea's "Sketch of the Analytical Engine Invented," written in French and translated by Lovelace in 1843.

As Lovelace (1843) points out in her Notes to her translation, however, Menabrea's notations do not take into account the memory of the Engine. Menabrea's expression for any numerical value to be calculated by the Engine was V_x, where "V" symbolized any numerical value and the subscripted index "x" indicated the specific mechanical arm in the Engine on which "V" was inscribed. Lovelace's revision of Menabrea's expression was to add a superscript, an upper index, "to indicate any *alteration* in the value which a Variable represents" (708). (As Lovelace writes: "No notice has yet been taken of the *upper* indices which are added to the left of each V in the diagram; and addition which we have also taken the liberty of making to the V's in M. Menabrea's tables" [708].) Lovelace's revised expression came out to be ${}^{y}V_x$ (See Fig. 2 in Appendix A for concrete examples of this notation used by Lovelace's algorithm to calculate the Bernoulli numbers through the Analytical Engine). The inclusion of this upper index, the superscripted "y" as the indication of "*alteration*," is one example of where Lovelace considers the material memory of the Engine. The index both inscribes the embodied experience of the Engine (its present experience as opposed to its future one) and narrates its response to the input. In this way, Lovelace applies the union between symbol and material, poetically recognizing and writing memory for the Engine, what it has done and what it will do. Evidence of this union between symbol and material are affirmed, as Forbes-Macphail (2016) has pointed out, by Lovelace's theoretical descriptions. For example, Lovelace writes: "the engine may be described as being the material expression of any indefinite function" (691). These theories translate into computational expressions by referring to "the *shifting* meaning of many of the symbols used in mathematical notation" (693). Lovelace was not the first to think this, and it is not uncommon to consider arbitrary symbols as part of mathematics in the mid-nineteenth century.[16] But to see how "*shifting* meaning" fits

[16] In algebra, the notation "x" is used to designate a variable that can encompass any numerical value. Let's take the equation "x = y + 5" as an example. In this case, the value of "x" would be dependent

in with her inclusion of the upper index, I will first describe how this ambiguity in mathematics applies to numbers in general.

The mathematical expression "2^2," for example, shows two different uses of the number 2. The first "2" holds a numerical value of 2, which is a conventional arithmetic conception of number. The superscripted "2," however, while also bearing a numerical value, is an operational function. That is, the symbol "2" is both an object receiving an action and a subject that acts on another number. The expression "2^2" could be rewritten as "the numerical value of two multiplied by itself *twice*." Although the two instances of "2" are represented by the same symbol, their meanings shift from "two" to "multiply denoted number by itself" according to the spatial context of each. We can see this expression of *shifting relations* with and by symbols in Lovelace's use of indices. For Menabrea and Babbage, the indices were used to mark which arm of the Engine was calculating. For Lovelace, however, they would mark each time the value of an arm was altered. Marking and inscribing change shows a consideration of memory, shifting the paradigm from the mathematical to the computational.[17]

For instance, Lovelace writes:

$${}^{1}V_{21} + {}^{1}V_{31} = {}^{2}V_{21}$$

(Lovelace 714)

Recall that "V" is the arbitrary symbol for any variable number that the engine calculates, the lower indices refer to the physical location of each variable (i.e. the column of the engine), and the upper indices refer to the iteration of each alteration. That is, the upper index labels the current state, hence memory of the specified

---

on "y," so if we assigned "y = 5," and substituted this value into the equation, then "x = (5) + 5," in which case the value of "x" would be 10. Here we clearly see the arbitrariness of the symbols "x" and "y," though "=," "+," and "5" are definite.

[17] It is important to reemphasize that numbers in mathematics are conventionally taken in the arithmetic sense, as numerical values. Abstraction of these values progressed in algebraic notation, where *x* could stand for 2, 12, or 3247, but these objects were still considered to be numerically valued. The transition to computation takes the abstraction of mathematical objects even further.

mechanical arm. As Lovelace writes, the upper indices "furnish a powerful means of tracing back the derivation of any result; and of registering various circumstances concerning that *series of successive substitutions*, of which every *result* is in fact merely the final consequence" (709). The terms "tracing back" and "registering" show the connection between Lovelace's symbolic notations and the material memory of the engine. We can see from the above example that ${}^{1}V_{21}$ is the first of the substitution series, differing from ${}^{2}V_{21}$ only in terms of the upper index. Thus, we understand that ${}^{2}V_{21}$ is the second iteration of a value placed on the twenty-first column of the engine, "merely the final consequence" of this calculation.

The above equation expresses that the number in the twenty-first column of the engine is added to the number in the thirty-first column, and the result is registered onto the twenty-first column. The final result of this calculation replaces the initial numerical value of the twenty-first column, and this replacement is indicated by the upper index of ${}^{2}V_{21}$. In a sense, the engine *forgets* its first number in order to *remember* a second. This is one way in which Lovelace's work shifts the paradigm of mathematics towards computation: the formula speculates on the memory of that which it affects, and then it narrates the response. In terms of notation, an expression like "x = x + 1," ubiquitous in computer programs as counters, is not logically sound in conventional algebra, where there is no need for variables to forget or remember themselves. In algebra, "x = x + 1" would be false, because "=" indicates identity in terms of value, and it would be false to say that "x" is identical in value to one more than itself. In computer science, however, "=" indicates identity in terms of designation, that is, "x" is designated to become one more than itself. To extend our metaphorical definition of computer code as addresses, the lower index indicates the physical location while the upper index indicates a change of resident. Effectively, Lovelace's index creates a traceable history of residents, without which algorithms (which are based on linear steps and procedures) would not make sense.

Additionally, this process of inscription, or the rewriting of memory, only works when there is a medium on which memory is inscribed. That is, memory is considered in computation because computation is finite and material. On the other

hand, algebra, in its abstraction, is infinite and immaterial; therefore space, hence memory, need not be considered. As Lovelace writes, "[t]he bounds of *arithmetic* were…outstepped the moment the idea of applying the cards had occurred; and the Analytical Engine does not occupy common ground with mere 'calculating machines'" (696–697). To be sure, this "idea of applying the cards" was inspired, in both Lovelace and Babbage, by the workings of the Jacquard loom. The material context of weaving gave impetus for Lovelace to inscribe memory, thereby stepping beyond calculation into computation. Similarly, poets have their impetus in readers (be it themselves or others), making use of the deictic properties of language to weave and unweave the process of memory. For example, "My daughter!" is, for Byron, an index of alteration. The memory of the first iteration of "My daughter!" comes from how the canto "begun," while the second replaces that beginning with its "end." In this way, Byron operates on the memory of the reader as much as Lovelace does with the memory of the machine.

To see how Lovelace continues to inscribe memory for the Engine, let us consider the next iteration of its twenty-first column:

$$^{1}V_{33} \div 2 + {}^{2}V_{21} = {}^{3}V_{21}$$

(Lovelace 715)

Lovelace again alters the memory of this column. She halves the value of the thirty-third column and then adds it to the twenty-first column. The difference in superscripts between $^{2}V_{21}$ and $^{3}V_{21}$ indicates that the value on the twenty-first column has changed. Again, the column *forgets* and then *remembers* something new; the "=" here designates the latest numerical identity for the twenty-first column. In order for a new value to be inscribed, the old one must be erased. Thus, the repeated process of erasure and inscription, or "the *ongoing process or work* of enunciation by which [a] moment is recursively known and carried forward" (Stewart 2002, 15), affirms the union of symbolic inscription and material context in Lovelace's work. To be sure, this poetic paradigm does not end with numbers. As Lovelace writes in a later part of her Notes

> (t)he following would be the successive sets of operations for computing the coefficients of $n + 2$ terms:–
> (×, ×, ÷, ×), (×, ×, ×, ÷, +, +), n (×, +, ×, ÷, +).
> Or we might represent them as follows, according to the numerical order of the operations:–
> (1, 2 ... 4), (5, 6 ... 10), n (11, 12 ... 15).
> (Lovelace 1843, 716)

In this case, Lovelace uses numbers to denote "numerical order". But the objects that the numbers order are not numerical values or even alteration of values, as done by the upper indices we have just examined. The novel move Lovelace makes here is taking *operations* themselves, the performers of alterations, as objects of computation. Her computational mathematics manipulate not just numerical objects, but also the relationships that happen between the numbers. (Lovelace writes in Note A: "In studying the action of the Analytical Engine, we find that the peculiar and independent nature of the considerations which in all mathematical analysis belong to *operations*, as distinguished from *the objects operated upon* and from the *results* of the operations performed upon those objects, is very strikingly defined and separated" [692].)

As Lovelace (1843) explains: "the symbols +, ×, &c., in reality represent (as their immediate and proximate effect, when the formula is applied to the engine) that a certain prism which is a part of the mechanism (see Note C), turns a new face, and this presents a new card to act on the bundles of levers of the engine" (720). Here Lovelace explicitly connects her computational language to the material movements of the Engine, explaining that symbols "act on the bundles of levers". The operational symbols are at once instructions for the Engine to perform and, at the same time, objects on which to perform. In fact, Lovelace's transformation of relationships between objects as objects themselves to be symbolically manipulated anticipates object-oriented programming in computer science of the mid-twentieth century. We can speculate on a theory of object-oriented relationships in Lovelace by comparing it with Byron's (1965) operations in "far years" (86). Since "far" is a spatial

term and "years" is a temporal term, as previously discussed, our minds normally consider their relationships as distinct from one another. By reading "far years" in the context of the poem, our minds attempt to reconcile the differences between spatial and temporal relations by imagining time as "far" and space in "years". We perform the collapsing of distance between the objects of our imagination, altering our definitions of time and space as much as the value on the Engine's twenty-first column. In other words, our own code is rewritten. After all, the objects of our imagination are as arbitrary as symbols and numbers. They could be Byron and Lovelace, parent and child, writer and reader, etc. In this sense, the imagination generated by syntactic play produces experience in us as much as the "×" makes the Engine multiply. Our own organic gears twist and turn and wrench due to all the ways that "far years" separate. The double duty of Lovelace's notation, on the one hand as instructions and on the other as manipulable objects, indicates the inscription and transference of memory by the encoder onto the machine. Thus Lovelace's algorithms form a linguistic bridge between the binarized information on punched cards to the manipulations of mathematical objects, adopting a digital paradigm (due to the material limitations of the loom-inspired Engine) as the basis for computational language. In turn, the algorithms, being based in the digital paradigm, can only be encodable into a binary form, transforming the process of memory-making into a series of ones and zeros, or truths and falses, or presences and absences.

Conclusion

In the material context of weaving, we might consider *memory* through one of the most influential uses of the Jacquard loom, at least at the time, for Lovelace and Babbage: the automated loom-woven portrait of Jacquard himself (Lovelace refers to this portrait in her Note C and F). As noted previously, this portrait "was essentially a digitised image. Made using 24,000 punched cards, it wove an image of the inventor of the loom on which it was woven" (Essinger 2014, 132). The concept of memory and inscription in this context takes another turn: a memory of something (e.g. Jacquard's countenance) is inscribed through the process of weaving. What makes the loom-

woven portrait seem such a wonder, however, is the appearance that the Jacquard loom wove it all by itself. Other than cheaper labor costs, autonomous creativity is the illusion that automation sells. At the end of the day, the 24,000 punched cards had to be punched, the resulting design of which having been predetermined by Jacquard himself. Someone had to encode the image into binary information. Even Lovelace was careful to consider the limitations of the Engine, writing that the "Analytical Engine has no pretensions whatever to *originate* anything. It can do whatever we *know how to order it* to perform. It can *follow* analysis; but it has no power of *anticipating* any analytical relations or truths. Its province is to assist us in making *available* what we are already acquainted with" (Lovelace). In other words, the Jacquard loom was able to weave the portrait of Jacquard because the hole-puncher was already acquainted with what a Jacquard portrait would look like. The difference, of course, is in speed and efficiency of production; but the predetermined framework remains. As I have noted above, weavers who adopted the Jacquard loom and other such methods had less pretensions about the autonomy of the machine, instead focusing on their commercial profitability.

Contemporary conversations surrounding computational methods such as *machine learning* and *Bayesian networks* (using probabilistic algorithms) seem to disagree with Lovelace's contention that computers would have "no power of *anticipating* any analytical relations or truths" (Lovelace). After all, artificial intelligence in technologies ranging from search engines to Siri to self-driving cars certainly appear to replicate our abilities to interpret information within given systems to derive relations and truths on which we make decisions. Ultimately, however, methods of artificial intelligence are still based on "what we are already acquainted with" (Lovelace). Whether it be training data for *machine learning* or probability models for *Bayesian networks*, the intelligence of the machine still comes from the information we feed it. Furthermore, as we've considered through Lovelace's poetics, the intelligence of our machines also comes from the forms by which we structure information (or the process of memory). Yet, the most prevalent means of evaluating artificial intelligence tend to focus on whether or not we can tell the difference between our cognition and that which attempts to replicate it (i.e. the

Turing Test), suggesting we might justify any computational process as intelligent if its distinction from the human surpasses common sense. If a self-driving car is able to recognize a stop sign, for example, and hit the brakes, it is considered as intelligent as any human driver. The *process* that goes into recognizing a stop sign, however, bears much less weight for "intelligence" than its result. That is, "intelligence" is defined (and based on the Turing Test) by its ability to attain an outcome, rather than by a cognitive process. The poetics of this kind of artificial intelligence is tied to the pragmatic domains of science, business, and military operations, inherited from the frameworks of industrialization and commercialization that emerged at the turn of the nineteenth century. The discourse surrounding Kashmiri shawls and the superiority of British imitations show an early example of how we have conflated "intelligence" with profitable outcomes.

In the Digital Humanities, debates regarding the efficacy of computational methods in humanities research show the limits of this conflation. The dispute between Annie Swafford and Matt Jockers regarding the use of sentiment analysis for examining the shape of literary plots is one example. Jockers proposed that words contain values of sentiment which, if analyzed and computed, can be used to indicate literary qualities of a text. A text is thus split into sentences and each sentence is assigned a positive or negative number based on the sentiment of its words. On the other hand, Swafford criticized sentiment analysis for the limitations of Jockers' algorithm; for example, it cannot take into account modifiers, connotations, or multiple meanings that words might have. Schmidt takes the debate further and notes that the "Swafford and Jockers debate hinged over not just an algorithm, but a concretely defined transformation," what is known as a Fourier transform, which assumes that "plots, like sound or light, are composed of endlessly repeating signals" (Schmidt 2016). In other words, though computationally and mathematically sound, the conclusions that Jockers' program came to were, according to Swafford, based on false assumptions.

In terms of this essay: having inscribed the memory of a Fourier transform into the basis of the computer program, the computer cannot but "remember" its experiences as Fourier transforms. A poetics analysis would show, as Schmidt

effectively considers, that the material contexts of the Fourier transform relate to its symbolic inscriptions in ways that might not be conducive for literary analysis. Even the basis of sentiment analysis itself is suspect. The computational method attributes "sentiment" to certain terms based on a predefined dictionary. Andrew Piper and Richard Jean So, for example, used this method to conclude that novels written during the Victorian era are the most sentimental of literatures. They examined about 2,000 novels and "searched for indications of differing levels of sentimentality using dictionaries developed by Bing Liu, one of the more prominent researchers in the field. The more a novel contained strongly positive or negative words (*abominable*, *inept*, *obscene*, *shady*, on the one hand, *admirable*, *courageous*, *masterful*, *rapturous* on the other), the higher its score" (Piper and So 2015). That is, the conclusions that their sentiment analysis program came to were based on a *memory* that inscribed "inept" or "courageous" as sentimental words. In terms of poetics, we might ask: 1) what material contexts inscribe "inept" or "courageous" as sentimental, and 2) what algorithmic process distinguishes each apart from the other? The second question particularly asks what cultural pressures have led us to consider such a logic for such an operation, and perhaps describes more what we think of sentimentality than sentiment itself, say, for Victorians.

For the Victorian who has woven her way through this essay, the connection between mathematics and poetry in the development of algorithms is certainly not a sentimental one. Binary operations, it seems, finds its logic in the cultural pressures of commercial production, and in our century, we continue to reproduce this logic through our computer code. My contention here is that computers still do not have the power to anticipate analytical relations or truths because the intelligence on which their computational powers are based were shaped for economic outcomes. That is, "what we are acquainted with," in Lovelace's terms, are data structures that developed from commercialization. That is not to say these structures we have currently developed from computer science are useless; in fact, they have been quite useful. It is certainly useful for self-driving cars to recognize stop signs to promote traffic order and prevent accidents. For those in power, it matters less *how* their

cars recognize a stop sign than their possible risks and liabilities. This pragmatic paradigm, however, evident in the poetics of those data structures, has its limitations. Having used a multitude of data (images of stop signs) to train recognition software, self-driving cars are still unable to recognize stop signs if, say, they happened to have stickers on them. (A case in late 2017 showed that some stickers on a stop sign confused a self-driving car into thinking that it was a 45-mph speed limit sign [Gitlin 2017].) The danger, perhaps, lies not just in the margins of error of such artificial intelligence methods; instead, they are based on the belief that intelligence is getting the answer right based on previously affirmed information. What machine learning does, then, is inscribe memory as a process of inductive reasoning, predictive models that are based on the scientific process. On the other hand, as many poetic forms show, the logics of our memory are more fluid and dynamic. Most of us, as far as I know, do not learn to recognize a stop sign based on being shown tens of thousands of stop signs.

I am not suggesting that we impose conventional poetic forms on computer code, though there might be generative possibilities in comparing, say, the structure of a sonnet with that of defining a computational function. Rather, I propose that computer code is already structured in poetic forms. Computational processes, such as iteration, recursion, or while-loops, for example, are symbolic inscriptions that produce material effects; they stage a construction of meaning and memory based on their material contexts. Coding languages underlie the entirety of our digital technologies; and since most of our everyday lives intertwine with the digital, the forms its languages take produce immediate ramifications. Besides using these technologies for computational research, it would be productive for the humanities to be literate in the paradigms that underlie their encodings. Ones and zeros and specifiable algorithmic steps do not just produce new forms of information but, perhaps more importantly to the humanities, transform the ways we handle information and make meaning from it. After all, computer codes are human languages: they are woven with wefts we warp, themselves weaving and unweaving clouds of thought into the electric shawls of our screens.

Additional Files

The additional files for this article can be found as follows:

- **Appendix A.** Diagram of an algorithm for the Analytical Engine for the computation of Bernoulli numbers and close-up of portion of diagram showing the first two rows of algorithmic equations, from "Sketch of The Analytical Engine Invented by Charles Babbage" by Luigi Menabrea. DOI: https://doi.org/10.16995/dscn.355.s1
- **Appendix B.** Trial model of the Analytical Engine constructed by Charles Babbage. DOI: https://doi.org/10.16995/dscn.355.s2

Competing Interests

The author has no competing interests to declare.

References

Allen, R.C. 2011. "Why the Industrial Revolution Was British: Commerce, Induced Invention, and the Scientific Revolution." *The Economic History Review* 64(2): 357–384. DOI: https://doi.org/10.1111/j.1468-0289.2010.00532.x

Byron, George Gordon. 1965. "Childe Harold's Pilgrimage: Canto the Third." 1816. In *Byron: Selected Poetry and Letters*, edited by Edward E. Bostetter, 50–86. New York: Holt, Rinehart and Winston.

Christensen, Jerome. 1995. *Lord Byron's Strength*. Baltimore, M.D.: Johns Hopkins University Press.

Essinger, James. 2014. *Ada's Algorithm: How Lord Byron's Daughter Ada Lovelace Launched the Digital Age*. New York: Melville House Publishing.

Forbes-Macphail, Imogen. 2016. "'I Shall in Due Time Be a Poet': Ada Lovelace's Poetical Science in Its Literary Context." In *Ada's Legacy: Cultures of Computing from the Victorian to the Digital Age*, edited by Robin Hammerman and Andrew L. Russell, 143–168. New York: Association for Computing Machinery; San Rafael, California: Morgan & Claypool. DOI: https://doi.org/10.1145/2809523.2809533

Fuegi, John, and **Jo Francis.** 2003. "Lovelace & Babbage and the Creation of the 1843 'Notes.'" *IEEE Annals of the History of Computing* 25(4): 16–26. DOI: https://doi.org/10.1109/MAHC.2003.1253887

Gitlin, Jonathan M. 2017. "Hacking Street Signs with Stickers Could Confuse Self-driving Cars." *Ars Technica*. September 1. Accessed October 5, 2019. https://arstechnica.com/cars/2017/09/hacking-street-signs-with-stickers-could-confuse-self-driving-cars/.

Grossman, Lev. 1998. "What Ada Knew: Was Lord Byron's Daughter the First Computer Programmer?" *Lingua Franca: The Review of Academic Life* 8(7): 62–69.

Jockers, Matthew. 2015. "Revealing Sentiment and Plot Arcs with the Syuzhet Package." *Matthew L. Jockers*. http://www.matthewjockers.net/2015/02/02/syuzhet/.

Jones, Caroline A., and **Benjamin Aranda.** 2009. "Algorithms as Difference Engines." *Thresholds* 35: 28–33. DOI: https://doi.org/10.1162/thld_a_00205

Liu, Bing. 2015. *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge, UK: Cambridge University Press. DOI: https://doi.org/10.1017/CBO9781139084789

Lovelace, Augusta Ada, trans. 1843. "Sketch of the Analytical Engine Invented by Charles Babbage, Esq." by L. F. Menabrea. *Scientific Memoirs, Selected from the Transaction of Foreign Academies of Science and Learned Societies, and from Foreign Journals* Vol. III, edited by Richard Taylor, 666–731.

Moll, Ellen. 2014. "A Network of a Line?: Gender, Technology, and Cyberfeminist Figurations of Time." *Rhizomes* 26. http://www.rhizomes.net/issue26/moll.html.

Piper, Andrew, and **Richard Jean So.** 2015. "Quantifying the Weepy Bestseller." *The New Republic*. Accessed October, 2018. https://newrepublic.com/article/126123/quantifying-weepy-bestseller.

Schmidt, Benjamin M. 2016. "Do Digital Humanists Need to Understand Algorithms?" *Debates in the Digital Humanities*. Accessed October, 2018 Minneapolis, MN: University of Minnesota Press. https://dhdebates.gc.cuny.edu/read/untitled/section/557c453b-4abb-48ce-8c38-a77e24d3f0bd.

Stein, Dorothy K. 1984. "Lady Lovelace's Notes: Technical Text and Cultural Context." *Victorian Studies* 28(1): 33–67.

Stewart, Susan. 2002. *Poetry and the Fate of the Senses*. Chicago: University of Chicago Press.

Swafford, Annie. 2015. "Problems with the Syuzhet Package." *Anglophile in Academia: Annie Swafford's Blog: Interdisciplinary Nineteenth-Century Digital Humanities*. https://annieswafford.wordpress.com/2015/03/02/syuzhet/.

Tenen, Dennis. 2017. *Plain Text: The Poetics of Computation*. Stanford, CA: Stanford University Press.

Toole, Betty Alexandra. 1991. "Ada, an Analyst and a Metaphysician." *Ada Letters* 9(2): 60–71. DOI: https://doi.org/10.1145/122028.122031

Vee, Annette. 2017. *Coding Literacy: How Computer Programming is Changing Writing*. Cambridge, MA: The MIT Press. DOI: https://doi.org/10.7551/mitpress/10655.001.0001

White, Charles. 1840. *The Cashmere Shawl: An Eastern Fiction*. London: Henry Colburn, Publisher.

Zutshi, Chitralekha. 2009. "'Designed for Eternity': Kashmiri Shawls, Empire, and Cultures of Production and Consumption in Mid-Victorian Britain." *Journal of British Studies* 48(2): 420–440. DOI: https://doi.org/10.1086/596107

How to cite this article: Tan, Czander. 2020. "The Poetics of Computer Code: Tracing Digital Inscription in Ada Lovelace's England." *Digital Studies/Le champ numérique* 10(1): 3, pp. 1–30. DOI: https://doi.org/10.16995/dscn.355

Submitted: 12 April 2019 **Accepted:** 09 June 2019 **Published:** 03 July 2020

Digital Studies/Le champ numérique is a peer-reviewed open access journal published by Open Library of Humanities. **OPEN ACCESS**