

Reading Note G: Ada Lovelace and the Clerical Labor of Codework

Zachary Mann <zmann_at_usc_dot_edu>, University of Southern California  <https://orcid.org/0009-0003-8576-1908>

DOI: <https://doi.org/10.63744/zmgh3nty93>

Abstract

This article examines human-machine co-authorship as it represented by Ada Lovelace in her famous translation of and appendices to L. F. Menabrea's "Sketch of The Analytical Engine Invented by Charles Babbage." Lovelace's translation notes and correspondence with Charles Babbage are read alongside the Engine itself, as a platform, and the histories of calculating engines, software development, and nineteenth-century clerical labor. Inspired by critical code studies, the article performs a close reading of what is today referred to as the "first computer program," a sequence of steps that Lovelace adds to her translation's final "Note G" as an example of something the Engine can do. Ultimately, the article argues that the gendered power structures of collaborative work in Lovelace's time — and the challenges women authors faced in the nineteenth century more broadly — influenced understandings of machine programming, and that Lovelace's representation of the human-machine relationship in the first programmable calculating machine complicates the organizational structures of both clerical labor and software design.

Introduction

Histories of computing, if they recognize Ada Lovelace as a software pioneer, include some version of the following origin story: On June 17, 1833, a young Lovelace visited Charles Babbage's home for the first time and beheld the working model of his Difference Engine, the first ever fully automatic calculator. That tower of cogs — one seventh of the never-finished Engine — charmed the teenage Lovelace and tied her professional future to its inventor; the following year, Babbage conceived of the Analytical Engine, a calculating machine of greater capability and the subject of Lovelace's one publication and claim to historical significance. I bring up this story because most of those same histories neglect to mention the other mechanical curiosity that Babbage showed off to his visitors that day. The mathematician also had in his possession many automata, including one which he kept in a similar glass case as the working model of his first Engine: John Merlin's "The Silver Lady," which Babbage himself describes as a "danseuse [who] attitudinized in a most fascinating manner. Her eyes full of imagination, and irresistible" [Babbage 1864, p. 274]. Here was a machinic woman, a windup ornament judged for its grace and appearance, burnished with the same status as the first automated "thinking machine" (Lady Byron qtd. by [Toole 1998, p. 38]). In this other telling, computing history begins flanked by two symbolic female figures: One is a pre-programmed object of desire, and the other has come to be known as the "first computer programmer."^[1]

Mar Hicks reminds readers of a truism in *Programmed Inequality*, that "all history of computing is gendered history" [Hicks 2017, p. 234]. This article extends this truism to computing's prologue by examining the gendered power structures in place during the earliest formulations of general-purpose programming. Computing history maps onto a cultural history in which women were once considered best suited to perform repetitious, clerical tasks — and desired as assistants by men who would rather not perform those tasks. Sadie Plant, who invokes the juxtaposition of Lovelace and "The Silver Lady" in *Zeros + Ones*, offers a revisionist history which both centers Lovelace's role in software development and argues that software is feminine by design. Plant, citing the career of Grace Hopper (the "Ada

Lovelace of the Harvard Mark 1”), highlights theoretical resonances between female sexuality and the ability of software to fulfill command requests by general-purpose imitation [Plant 1997]. More recently, Wendy Hui Kyong Chun in *Programmed Visions* cites Alan Turing and the Women’s Royal Navy Service (a.k.a. “Wrens”) to demonstrate “programming’s clerical and arguably feminine underpinnings — both in terms of personnel and of command structure” [Chun 2008, p. 46]. Chun, building on Plant’s argument, points out that the Bletchley Park Wrens, responsible for the tedious operation of cryptanalysis machines, were called “slaves” and “servants” by Turing, establishing a command structure which would become reflected in the design of computer science thereafter. Until very recently, computer programming was considered “a service occupation” [Brown 2018, p. 266]. Here is a third symbolic figure at the heart of computing history: the clerk who, through her servile labor, becomes both the performing object *and* the programmer.

This article recontextualizes, through the figure of the clerk, Lovelace’s published work — which occurred under the conditions of clerical service and writing as translation, while nevertheless challenging authority through her own original contributions and authorship. I examine her translation, “Sketch of the Analytical Engine Invented by Charles Babbage,” and one appendix in particular, “Note G,” which includes an algorithm designed to be executed by the Engine that has come to be known as “the first computer program” [Bowden 1953]. The mathematician Luigi Federico Menabrea published “Notions sur la machine analytique” in the October 1842 issue of the French-language *Bibliothèque Universelle de Geneve*. The “Notions” provided a cursory but enthusiastic summary of the concepts upon which Babbage was basing his newest invention. A year later, to drum up financial support for the machine in England, Lovelace wrote an English translation for *Taylor’s Scientific Memoirs*. This translation was notable for her long appendices, which more than doubled the length of Menabrea’s original text.^[2] In the famous *Notes*, Lovelace describes the Engine beyond concept, as if Babbage’s ideas amounted to a real machine, and she emphasizes its use of a punch card system to accept a program as input. The algorithm in “Note G,” which Lovelace calls a “diagram of development,” is a table that reflects the sequence of operations that the machine would perform to calculate the eighth Bernoulli number. Taking the claim that Lovelace wrote “the first computer program” at face value, I read this “diagram” as if it were such a program, despite the anachronism, to demonstrate how “Note G” both serves a technical function and reveals the shifting social systems within which it operates, imagining the kinds of relationships that programmers will have with future programmable machines. I argue that “Note G” reflects and contributes to the labor dynamics of the era in which she was writing. I reconsider in my reading of Lovelace’s “program” the historical command structures — master and servant, human and machine, subject and object — embedded in the ideas behind computing at its earliest stage in development.

Specifically, I situate Lovelace’s article beside three clerical figures that predate Turing’s Wrens: (1) the literary amanuensis, an assistant who copies for or takes dictation from poets and authors in order to free their minds to think; (2) the mathematician’s clerk, a human computer who calculates longhand the drudgery of astronomical equations; and (3) the secretary, a figure that becomes important after Lovelace’s death, with the rise of shorthand notation and the typewriter. These intermediaries between the active human mind and its passive (or mindless) execution undergo a transformation during the nineteenth century. As Jay Bolter and Richard Grusin argue, “the cultural work of defining a new medium may go on [...] before the invention of the device itself” [Bolter and Grusin 2000, p. 66]; the above clerical practices, labor histories which define the movement of women into public service positions and the rise of telecommunications, provide a sort of prologue for innovations in which language or thought become automated. This article reads Lovelace’s description of the Analytical Engine against contemporary notions regarding these clerks as “perfect mediations.” More to the point, I consider how the secretarial figure, as a metaphorical fulcrum “between inspired minds and automatic hands,” embodies Victorian-era motivations to eradicate error in recording and calculation which in turn influence technological innovation [Price and Thurschwell 2005, p. 5]. The concept of service is built into the digital; it is a design that remains “deeply gendered” and shapes representations of labor in computing today [Brown 2018, p. 262]. The above labor divisions — writer and steno, orator and recorder, philosopher and clerk — are the background against which Lovelace writes the future.

There are two Lovelaces. We are most familiar with the brilliant daughter of the poetic genius, a founding mother of computer science, and an enduring symbol for women in S.T.E.M. That version often obscures the reality. As Sydney Padua writes, Lovelace’s iconic status (and relation to her father, Lord Byron) has made it impossible to “hack through

the thicket” of the many contradictory accounts [Padua 2017, p. 216]. Indeed, scholars have muddied this history further by waging a politically charged discourse war over the legitimacy of her contributions.^[3] This paper will instead privilege her actual writing over her symbolic role in computer history narratives, focusing on how Lovelace’s contributions, like many contributions by women, take on a different (but no less influential) shape than those by “great men of history” [Burton and Hicks 2020, p. 1]. Lovelace, like many women authors before her, used strategies of anonymity, translation, and metaphors of mediation to claim authorship within male-dominated arenas. She may not have been the one to calculate the Bernoulli numbers for her *Notes*, but she did invent the machinic form Babbage’s calculations would take — authoring the Engine’s design.^[4] Her *Notes* are “small experiments” that invited new critical understandings of computing’s future through its material application [Whitson 2015, p. 165]. She challenged a nineteenth-century gender dynamic in which clerical figures become either mindless technologies of transcription (onto which a male stylus might transcribe his genius) or a threat to masculine authorship (by thinking for themselves). I argue that, in both the *Notes* and her correspondence, Lovelace thus complicates Babbage’s intended schematics for the Analytical Engine — as a perfect mediator between inspired minds and automatic hands — by extending her ambiguous clerical position to her technical writing. She inaugurates an idea of the technologized mathematician’s clerk that is both tool and guidance protocol, both programmed and programmer.

Automatic Hands

During Lovelace’s life, the clerical figure was largely male. However, as Leah Price and Pamela Thurschwell write, “the iconic late-nineteenth-century female secretary is preceded by the mid-century male clerk”; over the course of the century, one becomes the other [Price and Thurschwell 2005, p. 8]. This shift occurs along two timelines: the introduction of women into the labor force as social service workers, and the adoption of shorthand notation systems. Susan Brown argues that there was a perceived “continuity” between clerical positions and the domestic service roles that women were long encultured to occupy [Brown 2018, p. 266]. Though shorthand had already been invented, it wasn’t until Isaac Pitman’s 1837 book *Stenographic Sound-Hand* that the practice, which he later called “phonography,” reached cultural relevance [Gitelman 1999, p. 24]. Ivan Kreilkamp argues that 1837 “marked a new phase in English print culture’s relationship to speech”. Before phonography, Samuel Johnson paraphrased Parliamentary speeches with editorialized flare. Charles Dickens, a close friend of Lovelace and “probably the most famous figure ever to combine the secretarial and authorial roles in one person,” used an early form of shorthand called brachygraphy to record speech at almost natural speed [Kreilkamp 2005, p. 13]. But while Dickens was able to reminisce about his life as a clerk in romantically masculine terms, Pitman’s phonography, a more literal translation of sound to text, later replaced these older practices with an automatic form of transcription. This more mechanized writing transformed the user’s body into either the writing surface or the stylus. As the clerk figure became more technologized for the sake of accuracy, the more the position was transformed into a writing machine, stripped of intellectual agency and, as many have argued, thus feminized.

A similar transformation occurs in the figure of the amanuensis, the literary assistant who either takes dictation from or copies the work of an author. While most clerical labor was still dominated by men during the middle of the century, the relationship between an author and their assistant increasingly mapped onto heterosexual power structures; over time women’s hands replaced the female muse [Price 2003, p. 213]. John Milton, going blind, dictated his poetry to his daughters. Teenage girls, including Elizabeth Pigot, Claire Clairmont, and Mary Shelley, more legibly re-penned the scribbles of Lovelace’s estranged father. In these relationships, the power dynamics are so starkly drawn the amanuensis disappears altogether. She might shape her employer’s intellectual or creative output — as Jane Stabler writes, she is often “the first editor” [Stabler 2017, p. 58] — but gendered notions of the “lone genius” make invisible her labor; these women are, so to speak, reduced to mere writing machines. Like tools, their intellectual contribution is measurable only when it “breaks” [Brown 2018, p. 268]. Shelley compares the work of an amanuensis to “wifely servitude,” appropriately confusing the female copyist or stenographer as both a machine for reproduction and an object for sexual desire. As Price and Thurschwell point out, each shares a similar strategy: the reduction of the body to its physical rather than intellectual faculties [Price and Thurschwell 2005, p. 5]. For the (male) author, the ideal amanuensis was she who, like the Silver Lady, carried out her orders with sophisticated articulation and graceful motion, but without contributing thoughts of her own.

There is a long history of subject-object confusion between women and machines; words like “typewriter,” “computer,” and “secretary” have at times referred to either person or machine (or desk). Friedrich Kittler narrates a “queen’s sacrifice,” in which the female typewriter replaces the female sex object, reduced to a mere functionary — a cog — in the discourse network [Kittler 1990]. Idealized secretaries merge conceptually with machines because their labor teeters at the edge of sentience: because their use-value intersects. When Father Busa teamed up with IBM in the 1950s to make the *Index Thomisticus*, he chose young women who did not know Latin to input the words of Thomas Aquinas onto punch cards because he believed their accuracy derived from a lack of familiarity with the material [Terras and Nyhan 2016, 61].^[5] As Turing writes of artificial intelligence, only “those problems which can be solved by human clerical labour, working to fixed rules, *and without understanding* [emphasis mine]” are “capable of solution by the machine” [Turing 1986, p. 38]. At the time, as mental and physical labor become separate categories, such a fine line carried extra significance. Beyond automata, metaphorical discussions of the master-slave binary only make sense without historical realities. The rise of clerical labor, then the rise of computing, triggered techno-social metaphors designed to protect the concept of a free thinker (the “brains”) who did not work, and the robotic, unthinking laborer (the “brawn”) who carried out his ideas. This conflation of mastery with a certain kind of intelligence — retaining the concept of mastery for only educated European men — shadows the rise of automation.^[6] The fantasy is the reduction of these servile middle-machines until there is no “difference between a command given and command completed” [Chun 2008, p. 29].

This perfection of mediation is, as theorized by Price and Thurschwell, a “fantasy of divorcing transmission from understanding” [Price and Thurschwell 2005, p. 6]. This is the myth of automatic hands; the “genius” becomes constructed as the opposite of the “typist” who mindlessly punches keys. The extreme case is psychography (or, relatedly, “automatic writing”), in which a writer acts as amanuensis for abstractions (spirits, muses, etc.) through meditative states, seances, or trances. For many theorists, this labor relationship is mapped onto extant cis- and hetero-gender myths. Jill Galvan suggests that women were believed to possess “a sympathetic excess” that made their medial work possible [Galvan 2010, p. 2]. As Kittler writes, “an omnipresent metaphor equated women with the white sheet of nature or virginity onto which a very male stylus could then inscribe the glory of its authorship” [Kittler 1999, p. 186]. Plant, paraphrasing, adds: “Like all ideal women and machines, secretaries and shorthand typists were only supposed to be processing information which had been produced and organized elsewhere” [Plant 1997, p. 121]. In other words, the ideal secretarial figure only “processes” information but does not contribute to its shape. The Bletchley Park Wrens, in this ungenerous view, cracked codes but had no knowledge of the messages; they only switched the wires at the back of the machine, set the drums to the front, and punched cards. Plant, describing this labor, writes: “She hears, but she isn’t listening. She sees, but she does not watch. Pattern recognition without consciousness” [Plant 1997, p. 125]. Extending this to mathematics, any ideal calculating machine would not be, as Lady Byron suggested of the Difference Engine, a “thinking machine.”

The Unthinking Machine

The mathematician’s clerk belongs to this history, too. While not quite the same use of automatic hands — performing calculations rather than recording or copying — all such activities become optimized in the nineteenth century. Gaspard de Prony’s logarithmic tables allowed young clerks, with little education, to complete in days what mathematicians once did longhand over months [Daston 2018, p. 13–16]. Though this was rote arithmetic, like Johnson at Parliament, accuracy was a problem. Before equations were solved, one small human error might force the computer to start over. (An extreme example is William Shanks, who spent his life calculating the digits of *Pi* and, after one mistake, spent the last 20 years doing it incorrectly.) Prony’s tables allowed these clerks, like recorders using phonography, to calculate without understanding the higher mathematics — but they did not eliminate the need for sharp, focused minds. Babbage’s engines, in theory, did. Calculation is not, like recording, just the remediation of information. But it can be expressed as an equation: the conversion of a logical inquiry into a notational language which can then, according to set rules, be processed into numerical expression. If Babbage’s engines worked, the mental labor required to perform logarithmic arithmetic could be scripted and thus reliably accurate. Like the stenographer, the platonic ideal of the mathematician’s clerk speeds up transcription and computes without error the desires of the philosopher; this dream machine would perfectly mediate a given equation or logic problem into its solution.

The Analytical Engine was born of this dream. Babbage premised his earlier invention on the idea that the labored monotony of arithmetical calculations required a machine to “become a substitute for one of the lowest operations of human intellect” (Letter to Humphry Davy, 1822). This substitution was, for Babbage and in accordance with the clerical ideal, to reduce errors and save time. But his first attempt at the steam-powered clerk was not good enough. As Menabrea puts it in Lovelace’s translation, the “chief drawback” of earlier calculating engines — the need for continual human intervention — provided “a source of errors.” The new Engine corrected this flaw with a punch card system, inspired by the jacquard loom, which allowed the Analytical Engine to “spare intellectual labour,” insure “rigid accuracy,” and maintain an “economy of time” [Lovelace 1843]. Lovelace emphasizes “the object of the engine” in her *Notes*: to achieve the “*utmost practical efficiency*” (“Note E”); “reduce to a minimum the *time* necessary for completing the calculation” (“Note D”); and finally annihilate the “necessity for the intervention of human intelligence during the performance of its calculations” such that “there is much less chance of error, and likewise far less expenditure of time and labour” (“Note A”). The Engine thus embodies the desires of its era to perfectly mediate one expression (“inspired minds”) into an output via a clerical function (“automatic hands”), demonstrating how social power dynamics influence and become reflected in technological design.

I argue also that the more technical details within the Engine’s design are influenced by the clerical history above. Regarding how the engine is programmed, for instance, some of the punch cards required are not so different than shorthand; the *Operation Cards* are, Menabrea explains, “merely a translation” of the four algebraical notations (+, −, ×, and ÷) — linguistic symbols turned binary via punched holes. As Dionysus Lardner writes, once the question is “translated into algebraical signs” and “reduced to an equation,” the algebraicist “is relieved from the consideration of the complicated relations of the quantities” [Lardner 1834]. Human intellectual labor stops at the communication of the equation in natural language; the cards then “consign the execution” to the machine [Lardner 1834, p. 272]. This input system is more complex than that, as I will discuss later, but it relies on a key premise from “Note G”:

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.

This passage, which Turing refers to as “Lady Lovelace’s Objection,” is often read in the context of artificial intelligence [Turing 2009, p. 450]. Lovelace reaffirms Menabrea’s claim that “the machine is not a thinking being, but simply an automaton which acts according to the laws imposed upon it.” Like the Silver Lady, the machine servilely follows a script. Once instructed to begin, it will continue to behave in a singular manner until instructed to stop. As Lovelace explains, the punch cards activate “a series of different *states* [...] the *adding state*, or the *multiplying state*, &c.” (“Note B”). Once the Engine is placed into a “state,” it will remain in that state until instructed otherwise. This demonstrates the machine’s lack of sentience, that it cannot “originate,” but also describes its activity as automatic and even trance-like.

In that often-unquoted final sentence, the expressed purpose of the engine — “to assist us” in merely “making available” the already known — skips over the actual labor of the machine, erasing the difference between commands given and tasks completed. Similar to how the LISP function “Hello, World” delivers input to output without ever “reading” what’s bracketed within quotation marks, according to Lovelace here, the Engine does not ever interpret its input data: It is mere pattern recognition without consciousness. It is an amanuensis exactly remediating the intellectual product of its author. It is no different than the technology-assisted knowledge work in digital humanities that is often relegated to “support” rather than “scholarship” by the tradition of academia [Brown 2018, p. 264]. In this instance, Lovelace seems to agree with Babbage and Menabrea that the Analytical Engine has reduced mathematics to a kind of frictionless, algebraic stenography. The Analytical Engine is the “brawn” which serves as the seamless extension of the mathematician’s “master” mind, so limited in its capacity to shape the process along the way that it can only “make available” that which a human already dictated.

The Ghost Writer

Lovelace often positioned herself, at least on the surface, as a servile clerk. Her correspondence with Babbage between

1835 and 1852 paints a deferential role to the mathematician. When she first offers him her services with the Analytical Engine on January 12, 1841, Lovelace hopes that “my head may be made by you subservient to some of your purposes & plans,” framing herself as an object, a tool — a brain just like the Engine is a brain. Upon completing her *Notes*, she writes again to say: “I give to you the first choice & offer of my services & my intellect” (Letter to Babbage, August 15, 1843). A letter of unknown date reads: “I am willing to give myself *wholly to your interests*.” These words often undermine her contributions to his project. Plant writes that Lovelace is “remembered as Charles Babbage’s voice, expressing his ideas with levels of clarity, efficiency and accuracy he could never have mustered himself” [Plant 1995, p. 47]. She was the custodian of his work, but in Lovelace’s correspondence it was usually *his* work, his ideas, written in her hand. She perhaps felt as Shelley did while working for Byron: “under filial and philosophical obligation to subordinate her personal pain — and her own talent — to assisting the man of genius” [Stabler 2017, p. 56]. When writing the *Notes*, Lovelace read them aloud to Babbage at his home so that he may doublecheck that they reflected his ideas — to perfect their mediation (Lovelace, Letter to Babbage, July 10, 1843). But the partnership between Lovelace and Babbage was also, beneath the surface, a complicated labor relationship. This section will show that while her intentions as an ideal clerical assistant might be true, Lovelace troubles the distinctions between knowledge and its imagined application.

Perfect mediation was also the goal of her translation: to get across Menabrea’s words as accurately and with as little editorial paraphrasing as possible. Lovelace’s proficiency in French was such that she was able to do just that — at least, by the standards of translation. Like a “ghostwriter” — that phenomenon which arises with the figure of the secretary — Lovelace at first seemed content to write for but give the full credit of authorship to Menabrea in language and Babbage in concept. As Babbage explains later in his autobiography, Lovelace never intended to do more than play their unseen infrastructure; after she finished the translated portion of the “Sketch,” Babbage “asked why she had not herself written an original paper on a subject with which she was so intimately acquainted? To this Lady Lovelace replied that the thought had not occurred to her” [Babbage 2019]. She had erased herself from the publication of these men’s ideas. Partly due the social limitations conditioned into her gender — even at her station, she was not allowed to attend college and was criticized for having opinions about mathematics — and partly due to the opinion amongst her class that putting one’s name on literary work was overly commercial, she hadn’t considered writing her own paper or appendices. She only decided to add her initials to the published document after her husband’s encouragement (Lovelace to Babbage, July 4, 1843). Outside of “A.A.L.,” the only evidence of her presence in the *Notes* is the use of the pronoun “we”: “We have dwelt considerably [...] We are desirous [...] We refer the reader [...]” (“Note A”). For the most part, she is a ghostly figure which haunts the text in effort to better usher forth its message.

It is no coincidence that interest in the paranormal increases as technologies turn cultural transmission invisible. Galvan identifies the 1830s as the moment when these interests spike; with the telegraph and photography also came the rise, due to their “sympathetic excess” and capacity for automatism, of “mediating women” both occult and occupational [Galvan 2010, p. 17]. Consider Lovelace’s description of herself as a “Prophetess.” In a letter to her mother toward the end of her life, as her health deteriorated and medication increased, Lovelace annihilates her own presence in her work (November 11, 1844):

You know I am not a bit my own agent as to my scientific progress & objects. I am simply the *instrument* for the divine purposes to act *on & thro’*; [...] Like the Prophets of old, I shall but speak the voice I am inspired with. [...] The only *merit* that can ever be due to *me*, is that of putting myself, & maintaining myself, in *such* a state (physically & mentally) that God & His agents, *can* use me as their *vocal* organ for the ears of mortals.

As in psychography, Lovelace here imagines herself as both writing surface and stylus (“*on & thro’*”). She is the blank sheet of female nature and the transcribing stylus that only communicates the “voice” dictated to her, a phonetic functionary for the discourse of “God & His agents” (carried out, like the Wrens, without her knowing the message). She is only the technology, the “vocal organ,” for these ghostly geniuses. Most stunning is the reduction of her own abilities — her energetic writing style, her mathematical knowhow, and her reportedly flawless French — to a knack for entering and “maintaining” a series of automated “states,” a description eerily like the Analytical Engine by her own writing. Her script is her inspiration, and her “merit” is her ability to unwaveringly follow its instructions.

Not only does this language reduce Lovelace's head to its cybernetic utility, due its similarity to the object of her *Notes*, Lovelace's figurative ghostwriting suggests a cultural exchange between fantasies around gendered secretarial labor, calculating engines, and even Romantic literature. While Lord Byron used amanuenses, other poets preferred a different kind of remediation. Imogen Forbes-Macphail has written on the way constructions of imagination in Romantic poetry share affinities with Lovelace's approach to mechanism. [Forbes-Macphail 2016] points out, for instance, how Samuel Taylor Coleridge's "poetic sensibility" has at times been defined as a kind of automation in which the poet has "nothing to do with [the composition]"; the poet "only looked on, while the blind causes, the only true artists, were unfolding themselves." Forbes-Macphail suggests further that this poetic sensibility "lies not in the active will of the poet or thinker, but rather in the passive qualities of their constitution which allow them to be more receptive" and thus more perfect a mediation of that poetic input. Lovelace's statement that the Engine "cannot originate" is thus inseparable from notions of a "blind" poetic receptibility [Forbes-Macphail 2016, p. 153]. Here the intelligent design of the poet, the technical design of the Engine, and the capacity of medial women or clerical figures for "sympathetic excess" converge as a kind of pure infrastructure. In any "science of operations," whether poetry or engineering, data is processed and then delivered as output. The minimization of that process — when the output nearly matches the input (e.g., "Hello, World") — is for these poets, for the Engine, and for Lovelace in the above self-portrait, a mode of virtuous authorship.

19

The Discoverer

But Lovelace's labor also rejects clerical ideals. Her mediation is, notably, imperfect. Before starting the *Notes*, she sent a translation she'd written to Lord Lovelace (of a German ballad by Friedrich Schiller), adding the following disclaimer: "I cannot call it either a translation; for (except the first stanza, which is rather close), it is entirely different from the original" (April 1842). Like Shelley's tortured manuscripts of Byron's scribbles, Lovelace's translations "emerge as creative acts in their own right" [Stabler 2017, p. 58]. The same writerly intervention can be said of Menabrea's memoir; the translation of Menabrea's original text is around 8,000 words, but the *Notes* are around 20,000. Plant suggests that Lovelace's "footnotes" retain the "hierarchal divisions between centers and margins, authors and scribes," but also that paratextual spaces have historically allowed female typists to deviate from the communication of men's thoughts [Plant 1997, p. 9–10]. In those *Notes*, Lovelace excels the imagination of Babbage and Menabrea. She crosses the threshold between the machine's capacity for only calculation and, due the symbolic substitution of any kind of information in place of numbers, the possibility of general-purpose computing — what is called "Lovelace's Leap." [7] A reconsideration of Lovelace's positionality complicates whether the "interpretest" can call her *Notes* a "translation." She gives her "service" and "intellect" to Babbage's "interests," "purposes & plans," but the master she serves, if we look closely at her choice of words, is more the Engine than the man; she hopes to help unfold the "blind causes" of automated calculation itself.

20

It is worth noting that Babbage did not ask Lovelace to produce a translation. Like Mephistopheles offering to become Faust's assistant, Lovelace did not inform Babbage until a draft was finished. When I refer to Lovelace as a secretarial figure, I am invoking not the ideal clerical function that Babbage and Menabrea imagined the Engine to embody but the more ambiguous figure — the one who is imagined as an ignorant operator but instead threatens the authority of the "genius" by virtue of her own contributions. Lovelace, too, is an author. She is an author in the tradition of women who translated works by men to enter masculine spaces.^[8] She was similar in position to her close friend, Mary Somerville, who described her translation work not unlike Lovelace's "instrument" — as pure mediation. Such rhetoric has traditionally served as a strategy for women to assume authority over a subject while calling their authorship by another name. Women authors had to choose between a limited readership, mediumship, or other forms of anonymity — as another of Lovelace's friends, Joanna Baillie, knew well — each allowing self-expression within their limited literary avenues. Lovelace wrote poems, hoped to write books on flying machines, and pitched more articles after her *Notes*. She referred to her own writing style, likely with her father in mind, as "pithy & vigorous" with a "half-satirical & humorous dryness" (Lovelace, Letter to Babbage, undated). She wrote to her mother that she will "in due time be a Poet" (January 11, 1841). Likely she feigned sympathetic excess or humility as cover for her desire to write what she thought.

21

Her contribution was more than her Leap. It was Lovelace that kept a contract with the printer, who had the final call on what would be included in *Taylor's Scientific Memoirs* (and exercised that power to exclude a note Babbage wanted to

22

add). She was nobility to his nouveau riche and, frankly, Babbage was an impulsive genius who needed reining in. She even fixed his errors.^[9] She performed for Babbage the many “soft” skills — marketing, debugging, updating, even emotional labor, etc. — that are often assigned to women and “erased from the history of technology” [Brown 2018, p. 267–8]. She was both writer and editor and, as such, referred to the *Notes* as her own, including calling them a “child of mine” (Letter to Babbage, July 27, 1843). Lovelace’s Leap, and the vantage of her positionality, further complicate any attempt to boil down the invention of the Engine to any single “great man.” She, too, drafted Babbage’s Engine beyond his instructions; where Babbage’s draftsman Joseph Clement did so with measurements and materials, Lovelace designed a user experience with analytical and imaginative faculties. She is, with Babbage, Clement, and Menabrea, an author of the virtual machine.^[10] Despite her gender, Lovelace’s class and fame afforded her power. She had her own copyist. She designed the diagrams in pencil and, as she wrote to Babbage, “They have been made out with extreme care and all the indices most minutely and scrupulously attended to. Lord L is at this moment kindly inking it all over for me” (July 2, 1843). In this moment (and presumably others), hers is the intellectual work, and her husband becomes the automatic hands. When she refers to herself, in a September 15, 1843 letter to her mother, as the “High-Priestess of Babbage’s Engine,” she assumes the role of the inspired mind. When Babbage signs a letter to her, “Your faithful slave,” he places her on a different kind of pedestal than his Silver Lady (September 12, 1843). The translator excels her translated, and the power binary which later is used to define Turing and his Wrens is reversed.

In an essay she wrote on January 5, 1841, Lovelace asks, “What is Imagination?” In the essay, she defines imagination as “that which penetrates into the unseen worlds around us, the world of science,” and she argues that scientists are “those who have learned to walk on the threshold of the unknown worlds.” The sentiment echoes the moment Babbage first announced his idea for the Analytical Engine; in her journal, Lady Byron described the inventor as if he stood on a sublime precipice and quoted him comparing his new Engine to “throwing a bridge from the known to the unknown world” [Essinger 2014, p. 113]. For Lovelace, the machine itself served as a such a bridge, a mechanism of automation harnessed as a tool for imagination. To her mother, Lovelace wrote that she believed herself “to possess a most singular combination of qualities exactly fitted to make me *pre-eminently* a discoverer of the *hidden realities* of nature” (July 30, 1843). When she ghostwrote for the “true artists” (providing those mechanisms of automation on her own), her goal was to reveal or express what was hidden or inexpressible. She believed science, metaphysics, and poetry were synonymous practices, and to that end, she considered herself as imaginative as her father. Lovelace famously asks in “Note A”: what if “the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of [the punch card system’s] expression and adaptations”? This digression, one of her more powerful “leaps,” counters the idea that the Engine will only “assist us in making *available* what we are already acquainted with.”

This creative streak is not separate from mathematical ability. Though some critics have accused Lovelace of amateurism, she had, according to Babbage (via poet Henry Reed), a “peculiar capability, higher, he said, than that of anyone he knew, to prepare the descriptions connected to his calculating machines” [Padua 2017, p. 217]. To describe the Engine in her *Notes*, she had to guess at Babbage’s intent, drawing on their conversations, public lectures, and his contradictory and incomplete blueprints. When she made errors, they were missed guesses at Babbage’s mind, not bad math; much of what she was surmising in the *Notes* was “impossible [...] to know by intuition” (Babbage to Lovelace, July 2, 1843). Like programmers executing scripts in a test environment, Lovelace tested her ideas against the inventor’s whims. She designed without a script the innerworkings of the virtual machine, an exercise John Fuegi and Jo Francis describe as being “of almost inconceivable difficulty,” and she did so well enough that Babbage learned something new about his invention [Fuegi and Francis 2003, p. 19–21]. Lovelace possessed a clarity of technological vision, a (dare I say “poetic”) talent for translating imagination into clearly articulated technical documents. She was an expert in what we now call “critical making,” the combination of an individual’s ideas with their collaborative application [Whitson 2015, p. 165], and that is perhaps her greatest Leap. By virtue of having written the *Notes* with such “peculiar capability,” Lovelace resists becoming the clerical function she describes herself as serving. Her oscillations between clerk and inventor parallel debates around Lovelace’s contributions to computer science; it is a positionality which destabilizes her historical memory and is embodied in her “science of operations.”

The Meaning of “Note G”

23

24

A	B	Variables acted upon	Variables receiving results	...	Data			Working Variables				Result Var.	
					1V_1	1V_2	1V_3	0V_4	0V_5	0V_6	...	${}^1V_{21}$	...
1	×	${}^1V_2 \times {}^1V_3$	${}^1V_4, {}^1V_5, {}^1V_6$			2	n	$2n$	$2n$	$2n$			
...													

Figure 1. A = “Number of Operation”; B = “Nature of Operation”; source: the “Diagram of Operations.”

For full diagram: https://www.fourmilab.ch/babbage/figures/menat6_2k.png.

26

“Note G” defines a virtual engine with a series of mathematical operations: “the steps through which the engine could compute the Numbers of Bernoulli” and specifically “the value (either numerical or algebraical) of any n th Number of Bernoulli B_{2n-1} , *in terms of all the preceding ones*, if we but know the values of $B_1, B_3...B_{2n-3}$.” Those steps are visualized by a large table with columns that represent the input (“Data”), process (“Nature of Operation”), intermediate results (“Working Variables”) and output (“Result Variables”) that would occur during the calculation of B_7 (according to a hypothetical scenario in which the Engine had just completed the calculations of B_5). The table’s rows, except for certain repetitions (as discussed below), represent every step, or change of state. The variables (V) each represent a location in the machine’s memory where a value is pulled or inscribed; in Figure 1, for instance, after 1V_2 and 1V_3 are multiplied, the result is designated to be inscribed onto three separate locations (${}^1V_4, {}^1V_5, {}^1V_6$) to be used again in a future step. At first glance, what this program *does* is culturally uninteresting. As an expression, the eighth Bernoulli number ($-1/30$) does not mean much. But, because Lovelace writes the “diagram” for human readers, even at its most technical “Note G” remains a text with embedded extra meanings. Unsurprisingly, the program has been studied extensively for its affinities to today’s software. Curious programmers, for instance, have translated the diagram into modern programming languages (including C, Python, and FORTRAN).^[11] But while these studies venture into human contexts, imagining a social history of computer programming that begins before the twentieth century, none consider the gendered labor structures built into that “code” and inherited from nineteenth-century clerical figures.

27

To start, it’s worth considering why Lovelace chose this equation over others. She wrote to Babbage: “I want to put in something about Bernoulli’s Numbers in one of my Notes as an example of how an implicit function may be worked out by the engine without having been worked out by human head or hands first” (July 10, 1843). She was familiar with the equation because she had studied it during her lessons with Augustus de Morgan, and she knew the Engine could “handle” the calculation because Babbage boasted as much (Letter to Alexander von Humboldt, January 1842). The aim was to demonstrate, beyond mere “facility of computation,” how a many-step calculation would look without the possibility of human error. In some ways this “example” is yet another expression of the Engine as a clerical substitute; but it is an extreme case, chosen because the faculties required to quickly calculate this equation exceeded “human head or hands.” The Bernoulli numbers jumpstarted the clerical function beyond stenography and arithmetic to the start of something new, a precipice of true automatic computing. As Lorrain Daston writes, “The analytical intelligence applied to making human machine cooperation in calculation work was a rehearsal for an activity that would become known first as Operations Research and later computer programming” [Daston 2018, p. 28]. As Chun writes about a different moment in time, “One could say that programming became programming and software when the command structure shifted from commanding a ‘girl’ to commanding a machine” [Chun 2008, p. 29]. But this conclusion, in which software is defined as feminine and clerical, does not seem to apply to Lovelace as unambiguously as Turing described the Wrens. It assumes divisions between knowing and making that Lovelace erodes in her *Notes*. This section asks: What can be said differently about the command structures built into code’s origins if those origins can be traced back to the diagram

28

There is something of Lovelace’s imagination in “Note G,” too. Even the selection of the Bernoulli numbers seems to bear on her life and foresight. Lovelace would have known the equation’s application for predicting random probability, herself having been quite interested in the potential for mathematics to solve games of chance. Today, the equation is used to calculate the flow of air or liquid across space, specifically for determining the value of points along the path of that motion. Since the equation’s discovery at the end of the seventeenth century, its application had only grown in importance, from building dams to estimating the structures of stars and galaxies — as Lovelace defined the scientific imagination, “penetrat[ing] into the unseen worlds around us.” Pertinent to Lovelace, who once wanted to write a book on “flyology,” the equation would become integral for determining the airflow across the wing of an airplane and designing the exhaust of the Apollo rockets. “Note G” forecasts the ability of machines to make calculations at scales large enough to accomplish such innovations. It is a demonstration of the Engine’s potential for universality and scalability. Lovelace defines an “operation” as “any process which alters the mutual relation of two or more things,” a definition that includes “all subjects in the universe” (“Note A”). The expression of an equation as real-world applicable as Bernoulli’s via mechanical “operation” opens the utility of the Engine to those unfathomable futures. It is the taming of “all subjects in the universe” into malleable “things,” the sublime movements of sound and stars into punched cards.

But is it code, really? The question is definitional. Herman H. Goldstine and John von Neumann argue that code is not simply a translation from the language of mathematics — “the numerical procedure by which [the programmer] has decided to solve problem” — into machine instruction [Goldstine and Van Neumann 1947, p. 2]. Others have defined code as a symbolic set of signals given meaning only once compiled or executed.^[12] Thomas J. Misa, to avoid these semantics, describes the Engine as “reconfigurable” [Misa 2016, p. 22]. Scholars, citing the text of “Note G,” refer to the diagram as an “execution trace,” a log of the sequence of operations performed by the Engine [Campbell-Kelly and Aspray 1986]; indeed, Lovelace described it as “a complete simultaneous view of all the successive changes [necessary] to perform the computation.” Others, attending more to the program’s intent, compare the table to “opcode,” or the parts of assembly language which pertain to machine instruction [Target 2018]. Each of these positions is defensible — and the anachronism prevents any conclusion. But at the very least, the program shares both purpose and behaviors with computer programming. The latter is best illustrated by its famous error:

A	B	Variables acted upon	Variables receiving results	Indication of change in the value on any Variable	Statement of Results	...	Working Variables				
							⁰ V ₄	⁰ V ₅	...	⁰ V ₁₁	...
...											
2	−	¹ V ₄ − ¹ V ₁	² V ₄	$\{\overset{1}{V}_4 = \overset{2}{V}_4\}$ $\{\overset{1}{V}_1 = \overset{1}{V}_1\}$	$= (2n - 1)$		$2n - 1$				
3	+	¹ V ₅ + ¹ V ₁	² V ₅	$\{\overset{1}{V}_5 = \overset{2}{V}_5\}$ $\{\overset{1}{V}_1 = \overset{1}{V}_1\}$	$= (2n + 1)$			$2n + 1$			
4	÷	² V ₅ ÷ ² V ₄	¹ V ₁₁	$\{\overset{2}{V}_5 = \overset{0}{V}_5\}$ $\{\overset{2}{V}_4 = \overset{0}{V}_4\}$	$= \frac{(2n - 1)}{(2n + 1)}$		0	0		$\frac{(2n - 1)}{(2n + 1)}$	

Figure 2. A = “Number of Operation” ; B = “Nature of Operation.”

For the fourth operation, as recorded in Figure 2, Lovelace lists the “Variables acted upon” as ${}^2V_5 \div {}^2V_4$, or the resulting value of the third operation (as saved in the second position of the fifth *Variable* column) divided by the result of the second operation (saved in the second position of the fourth column). As many mathematicians have pointed out, this equation should be written the other way around: ${}^2V_4 \div {}^2V_5$. The error is likely one of typesetting — maybe even a copying mistake by Lord Lovelace — because the other columns, both the “Statement of Results” and the “Working Variable” (${}^0V_{11}$), reflect the correct order: $(2n-1) \div (2n+1)$. Perhaps for that reason, the arbitrarily misplaced letter or number, rather than a mistake in mathematics, we might consider this typo in “Note G” the “oldest bug in computing” [Target 2018]. It is an error which reveals the contributions of otherwise transparent support infrastructure.

The error highlights similarities between the subjectivity of Lovelace and future coders. In interviews, multiple programmers have shared that “Note G” resembles “the experience of programming”; Sinclair Target, for instance, told me the exercise reminded him of introductory programming classes (just with different variables), that even today many early lessons in computer programming ask students to write code for the calculation of a sequence, usually Fibonacci [Target 2020]. Code’s abstraction from hardware means there is always some level of ignorance in programming; the coder simultaneously occupies the subject position of both commander and guesser. Arguably, the programmer’s “ignorance” of “the path of decision-making within [her] own program” later becomes definitive to how software works (Joseph Weizenbaum, qtd. by [Hayles 2010, p. 28]). It mixes “pattern recognition without consciousness” with the “inspired mind,” not unlike the way Lovelace describes her own work. Babbage, in accordance with Enlightenment notions of experiential knowledge, invites audiences to watch his Difference Engine grind and gnaw from the outside. In “Note G,” Lovelace flips the reader’s position such that they share her own, like the Romantic poet, thinking the machine from within. We join with Lovelace as she remains servile to Babbage’s designs yet contributes creatively to and writes commands for its virtual application. We can see that there is room for her critical imagination, that any programmer creates their own unknown worlds even as they are burdened by the designed worlds of others. Lovelace thus inaugurates in “Note G” a strange, new labor position, a relationship between humans and machines that threads the needle between multiple levels of authorship and clerical functions.

32

LOOP

The actual code implied by “Note G” — the decks of punch cards which input data and inform the Engine to change state — was never produced. Lovelace writes a program without a programming language, but one which anticipates its translation; her *Notes* describe in detail the interplay of *Operation Cards* with *Variable Cards*, *Number Cards*, et al.^[13] At its most elementary level, the *Operation Cards* determine the state — e.g., “divide” or “add” — while *Variable Cards* identify the “Variables acted upon” and the “Variables receiving results.” For instance, in Figure 2, the fourth operation required an *Operation Card* to activate the “divide” state, two *Variable Cards* to pull the values 2V_5 and 2V_4 from memory, another *Variable Card* to deliver the operation result to ${}^0V_{11}$, and two more *Variable Cards* to instruct the Engine to reset V_4 and V_5 once the operation is complete (by returning their values to zero). This system makes possible the separation of the Engine’s memory from its operations. But this was far from the most advanced capability of Babbage’s design. The reason the diagram is considered a forerunner to modern programming is the Engine’s capacity (using a fourth kind of punch card, *Combinatorial Cards*) to designate sequences of punch cards to *repeat* on command: to cause “the prism over which the train of pattern-cards is suspended to revolve *backwards* instead of *forwards*” in order to “bring the card or set of cards in question into play a second time” and thus re-perform a sequence of operations (“Note C”). These cycles, like the aforementioned “states,” are executed automatically, reducing the number of total cards required and allowing “more symmetry and simplicity in the arrangements” (“Note G”). Babbage called this “eating its own tail.”^[14] All used input cards function as an accessible memory bank, a cache or library of performable operations that can be re-accessed again without human intervention.

33

34

Number of Operation			Nature of Operation	Variables acted upon	Variables receiving results
...					
13	{	{	—	${}^1V_6 - {}^1V_1$	2V_6
14			+	${}^1V_1 + {}^1V_7$	2V_7
15			÷	${}^2V_6 \div {}^2V_7$	1V_8
16			×	${}^1V_8 \times {}^3V_{11}$	${}^4V_{11}$
17		{	—	${}^2V_6 - {}^1V_1$	3V_6
18			+	${}^1V_1 + {}^2V_7$	3V_7
19			÷	${}^3V_6 \div {}^3V_7$	1V_9
20			×	${}^1V_9 \times {}^4V_{11}$	${}^5V_{11}$
21	×		${}^1V_{22} \times {}^5V_{11}$	${}^0V_{12}$	
22		+	${}^2V_{12} + {}^2V_{13}$	${}^3V_{13}$	
23		—	${}^2V_{10} - {}^1V_1$	${}^3V_{10}$	
Here follows a repetition of Operations thirteen to twenty-three					

Figure 3. The opening brace ({} identifies which sequences of operations will repeat.

It is this mnemonic structure that makes possible the Engine's capacity, to use a modern parlance, to perform a "while" LOOP. When programmers praise Lovelace's diagram, they refer to the GO-TO statements and IF-THEN branching in its logic, but especially the capacity of the diagram to repeat a pattern based on external input (iteration) and internal logic (recursion) for predetermined durations.^[15] Lovelace's diagram lists 25 operations but, as indicated by the three braces ({} in Figure 3, some of those operations (13-23) are designated to be performed twice, with operations 13-16 and 17-20 each repeating within the parent LOOP. The diagram implies but does not explicate the use of *Combinatorial Cards*, added to operations 16, 20, and 23, which would activate the Engine's "reverse" state for a predetermined number of rotations (and under certain conditions). The necessary repetitions for computing numbers of Bernoulli required Lovelace, as she intended, to show just how flexible the punch card system could be. Dana Angluin suggests that the equation's complexity "led to the development of some of the more subtle and powerful concepts, e.g., looping and indexing" [Angluin 1976, p. 7]. While the Engine's capacity for recursion might not resemble software today, the flexibility of the punch card system — the programmer's ability to set the variables determining the parameters of the LOOP: when to start and stop rewinding or skipping — allows for infinite free play between the Engine's library and its operations. The idea that a machine could retain information but also *alter* that information — reorder or partially repeat — was a century ahead of its time; this was both stylus and a writing pad, which self-erased and self-inscribed. Lovelace's approach to the challenge of the equation illustrated and developed the Engine's potential as a general-purpose machine.^[16] It is no windup danseuse; once preset, it could be let loose.

Of course, this is all theoretical — or, as Lovelace terms it, "*experimental*." When asked whether the Engine is "*really* even able to *follow*" such a program, she responds that only seeing is believing ("Note G"). Per Goldstine and Neumann, it is impossible to "foresee in advance and completely the actual course" of a program; coding is "the

technique of providing a dynamic background to control [only] the automatic evolution of a meaning” [Goldstine and Van Neumann 1947, p. 2]. To put it another way: The Engine’s pathways are rigid, but full of so many twists and turns that it becomes abstracted, that it resembles thought enough to give the impression of taking control, of becoming its own guidance system. The artist Wilfried Hou Je Bek describes a LOOP as that “gargoyle of cyclical imagination in computation” [Hou Je Bek 2008, p. 179]. Like Lovelace, he sees new possibilities in this corner of machine programming, an escape from the prison of mathematics for the freedom afforded by poetry; the capacity of a program to LOOP infinitely functions as analog to the human ability to think outside the bounds of known reality. Between the equations of “Note G” are the “hidden realities” that a scientist-poet seeks to discover. Just as Lovelace’s virtual design taught Babbage something new about his invention, the machine can “discover” new understandings of mathematics: “the relations and the nature of many subjects in that science are necessarily thrown into new lights, and more profoundly investigated,” leading to “*collateral* influences” (“Note G”). This is a far cry from merely “making available” what is already known. The performing object guides the inspired mind; as Roger Whitson suggests, Lovelace imparts “an agency to mathematics itself” [Whitson 2015, p. 168]; she welcomes the machine as a fellow collaborator in their discovery.

While the Engine was designed for a clerical function, sometimes when it is personified by Lovelace and Babbage, it comes off as more. Lovelace, for instance, writes to Babbage on March 24, 1839, “Surely the machine allows you a holiday sometimes.” The *Notes* may have been her “child,” but the machine takes on a life of its own, and some of that imagined subjectivity is due the Engine’s capacity for remembering past processes and reacting to the new variables it produces. Lovelace in a footnote explains that the Engine anticipates what it needs beyond instruction: “The engine is capable, under certain circumstances, of *feeling* about to discover which of two or more possible contingencies has occurred, and of then shaping its future course accordingly.” Babbage himself imagined that he was “teaching the Engine to foresee and then act upon that foresight” [Babbage 1961, p. 53]. It is not the programmer but the *Variable Cards* who “furnish the mill with its proper food” (“Note B”), not the mathematician but the *Combinatorial Cards* who reshape “its future course.” It is a design that echoes the way Turing imagines the possibility of real machine intelligence: When computers are not programmed with explicit instructions on how to behave in given circumstances, but instead are *taught* in some fashion [Abramson 2008, p. 156]. This is not unlike how we think about the automated tools available today in the digital humanities, as that which we teach to become guides that we trust too readily, and credit too rarely. At the very least, the Engine, like Lovelace, contributes beyond its clerical position. Though “Lovelace’s Objection” looms large in computer history, her *Notes* trouble its message again and again; the Engine does what it is ordered to perform, but it also “feels” and “discovers” the “possible.”

37

Conclusion

More than a century later, Turing stood on a similar precipice as Babbage. His improvement on the design Lovelace left for history was the notion that subprograms could be added to rewrite others and themselves — that programs could be programmers, too. His view was a future where it is the “masters who are liable to get replaced” rather than the “servants,” and that the future of software would be the eventual transfer of authority from the human to a program stored in the machine. These realizations coincide with the invention of the ENIAC, the first electronic computer and “a computer of the Babbage type” (Neumann, qtd. by [Fuegi and Francis 2003, p. 18]). In 1945, an ENIAC subprogram called the “master programmer” — a different kind of “thinking” guidance system — sent change orders to the human computers. This new labor structure persisted with the digital turn; as Chun explains, there remains a “sense that we are slaves, rather than masters, clerks rather than managers — that, because ‘code is law,’ the code, rather than the programmer, rules” [Chun 2008]. For Chun, “code as law” flips the command structures of human and machine; it “establishes a perpetual oscillation between the two positions” [Chun 2008, p. 20]. Anne Balsamo expands this role reversal more broadly, to “the doubled nature of technology: as determining and determined, as both autonomous and subservient to human goals” [Balsamo 2011, p. 31]. And Hayles argues that the human and the machine are neither mutually exclusive nor overlapping subject positions; instead, subjectivity is generated by a feedback loop between the two states [Hayles 2010, p. 20]. I argue that this feedback loop was there from the beginning, embedded within Lovelace’s 1843 *Notes*. Though born of a desire for an ideal clerk, the “thinking machine” shares with its fellow “peculiar” support figures a tendency to think beyond (with and for) the inspired minds who imagine them as servile.

38

J. Chuan Chu, one of the hardware engineers for the ENIAC, called “software [...] the daughter of Frankenstein” — hardware’s feminine counterpart ([#Chuan Chu 2008, p. 33]). If hardware was Frankenstein’s creature, software is an Eve born of Adam’s hardware rib. The metaphor mirrors the relationship between Babbage and Lovelace, too; he, the man with a machine in his mind, and she who came later, the only person who communicated with that machine directly. Shelley, who in the novel *Frankenstein* reduced the Enlightenment-educated man to the servant of his hardware creation, had Dr. Frankenstein destroy his daughter. But computing history tells a different story. Software replaces the man. Though the modern computer was not influenced by Lovelace’s writing, it was “a striking case of design convergence” [Priestley 2011, p. 49].^[17] Software eventually inherited the same structures of co-authorship which Lovelace embodied and described. Building on the secretarial workforce at the end of the nineteenth century, itself built on the clerical labor of the previous century, women would make up the bulk of computing labor. Lovelace begot the Wrens and the so-called “ENIAC girls,” the “brawn” to the corporate “brain” of twentieth-century computing innovation. Even after WWII, this idea of female clerical labor was so fundamental to the structure of the industry that IBM measured production by “girl hours” not “man hours” [Hicks 2017, p. 21].^[18] Goldstine and Neumann, whose work on the ENIAC essentially coined the verb “to program,” defined computing as a relationship between scientists, who posit algorithms, and female programmers, who inscribe them. Software replaced this structure with one just like it: subroutines that are both automatic and authoritative, both mindless and full of high-level abstraction, as it was embodied in the first computer programmer and her virtual object of desire.

Notes

[1] Lovelace’s modern reputation can be traced back to Alan Turing’s 1950 article, “Computing Machinery and Intelligence.” Soon after, her reputation was solidified by B.V. Bowden in *Faster Than Thought* (1953) and later D. L. Moore, who claimed that Lovelace was “the first computer programmer.” Subsequently, the Department of Defense first began developing the Ada programming language in her memory (1977).

[2] The “Sketch,” published on August 24, 1843, was considered a success by its small audience. Babbage would later write to Lovelace’s son, Byron Noel, that his mother had written “the only comprehensive view of the powers of the Analytical Engine which the mathematicians of the world have yet expressed” (1857). However, the article was not widely read and not republished until 1889. This paper uses the version online at: www.fourmilab.ch/babbage/sketch.html

[3] The debate around Lovelace has a long critical history. D. Stein, in *Ada: A Life and a Legacy* (1985), wrote off Lovelace as “a figure whose achievement turns out not to deserve the recognition accorded it” [Stein 1985]. A. Bromley has refuted Lovelace’s reputation as the first programmer [Bromley_1990] [Bromley_1990]. M. Campbell-Kelly echoed Stein when he wrote that “the extent of Lovelace’s intellectual contribution to the Sketch has been much exaggerated” (1996), and many other works on Lovelace have overly focused on her biography and temperament. However, a counter narrative simultaneously emerged. J. Baum’s *The Calculating Passion of Ada Byron* (1986) treats her writing seriously, and B. A. Toole’s *Ada, The Enchantress of Numbers* (1992) has inspired many works, some cited in this paper, to treat Lovelace as an important innovator in computer history. This article follows Roger Whitson’s claim that Lovelace’s contribution becomes salient when we make room for “critical making” in intellectual genealogies [Whitson 2015, p. 166].

[4] The *Notes* were a collaborative effort: Babbage and Lovelace “discussed together the various illustrations that might be introduced.” He remembers in his autobiography that “I suggested several, but the selection was entirely her own. So, also, was the algebraic working out of the different problems, except, indeed, that relating to the numbers of Bernoulli, which I had offered to do to save Lady Lovelace the trouble” [Babbage 2019]. But even if Babbage had provided her with the original equations, it was Lovelace who translated the algebra into a step-by-step algorithm of her own design — an achievement which, as Thomas J. Misa outlines extensively, is “reasonably clear” and “clearly indicated” in her letters [Misa 2016, p. 14, 26].

[5] Melissa Terras and Julianne Nyhan here are quoting Marco Carlo Passarotti of the CIRCSE Research Centre. Terras and Nyhan also point out that “the names of the women have not been preserved in the historical record”; maybe because the *Index Thomisticus* was one of the first digital humanities projects, DH projects today often unwittingly reproduce a similar labor structure [Terras and Nyhan 2016, p. 61–63].

[6] The use of loaded jargon (like “slave”) has recently been reconsidered in science and technical writing, related to ants and programming languages respectively. Though useful in a vacuum, such terms imply analogies between human and nonhuman behaviors. Ron Eglash makes the historical connection between late nineteenth-century uses of “slave” and “servant” in engineering (e.g., servomotors and slave clocks) to Charles Babbage, crediting his 1832 book *On the Economy of Machinery and Manufactures* for popularizing the division of high-level and low-level thinking in labor (“Broken Metaphor: The Master-Slave Analogy in Technical Literature,” in *Technology and Culture* Vol. 48 No. 2 (2007), 367). It should also be noted that Black slaves invented many new technologies but did not own their inventions. See: Rayvon Fouché, *Black Inventors in the Age of Segregation: Granville T. Woods, Lewis H. Latimer, and Shelby J. Davidson* (JHU Press, 2005).

[7] In response to claims of her emotional stability and mathematical amateurism, scholars who defend Lovelace’s historical significance refer to the creative thinking which she provided and which Babbage seemed to lack. “Lovelace’s Leap” usually describes her impulse to ask *what if*. Even Doron Swade, who has marginalized Lovelace’s role in the past, argued in an interview for the documentary *To Dream Tomorrow* that “What Lovelace saw [...] was that numbers could represent entities other than quantity” [Fuegi and Francis 2003, p. 24]. She suggests that because the Engine might act upon non-number variables, it might produce things besides numbers if those original variables (notes, colors, etc.) shared a relationship expressible by algebraical operation.

[8] I am referring to women, like Elizabeth Cary, who translated Italian plays in Early Modern England, and works of science from one European language to another; famously, Emilie du Chatelet translated Isaac Newton, Clemence Royer translated Charles Darwin, Anna Helmholtz translated Tyndall, and Marie-Anne Paulze Lavoisier translated her husband. Mary Somerville’s entry into the elite scientific societies of London was based on her translation of Pierre-Simon Laplace’s *Traité de mécanique celeste* (or *The Mechanism of the Heavens*, 1831), only after which she was able to publish her own work, starting with the very successful *On the Connexion of the Physical Sciences* (1834).

[9] Babbage remembered that Lovelace sent back the Bernoulli calculations he had done for her, “having detected a grave mistake which I had made in the process” [Babbage 2019, p. 136]. The nature of this mistake is not known, but this anecdote reaffirms why the machine was invented. She had also once criticized the published work of her tutor, Augustus de Morgan, and was proven correct years later.

[10] The Analytical Engine was never built, and unlike the Difference Engine, it was never drafted with consistent enough technical detail to provide functional blueprints. It existed only in the minds of Babbage, his principal draftsman Joseph Clement, and Lovelace. It exists today only as an idea, scattered across Babbage’s journals, and most definitively in Lovelace’s words. Indeed, the “Sketch” serves as the blueprints Babbage never published; Allen Bromley writes that, “aside from the Bernoulli numbers program prepared for Ada Lovelace’s notes, there is no evidence that Babbage prepared any user programs for the Analytical Engine after his 1840 trip to Turin.” (“Allan Bromley Explores Babbage’s Analytical Engine Plans 28 and 28a,” IEEE Annals of the History of Computing, 2000, 11). By 1837 Babbage had produced many stereotype plates. He distributed them but never published them, and they are far from complete. All this said, like Maxwell’s demon, Schrodinger’s cat, and Sadi Carnot’s engine, the Engine is a thought exercise that has launched its own discourse.

[11] In a 1979 Russian study titled “The Babbage Machine and the Origins of Programming,” A. K. Petrenko and O. L. Petrenko translate the algorithm into a 65-line FORTRAN program. More recently, Jim Randell translated the diagram into Python, which he describes in a blog post titled “Ada Lovelace, Charles Babbage and the First Computer Program” (December 8, 2015): <http://jimpulse.blogspot.com/2015/12/ada-lovelace-charles-babbage-and-first.html>. Sinclair Target translated the program into C for his blog *Two Bit History*, in a post titled “What did Ada Lovelace’s Program Actually Do?” (August 18, 2018): <https://twobithistory.org/2018/08/18/ada-lovelace-note-g.html>. Following Lovelace’s comment on music, D. De Roure and P. Willcox built a web-based app for the Engine to produce music: <http://numbersintonotes.net/>. And John Walker for *Fourmilab Switzerland* developed a web emulator for the Engine in Java: <https://www.fourmilab.ch/babbage/emulator.html>.

[12] These are the two main arguments against the diagram’s definition as “code,” and they are both contested positions. For one, the difference between machine and higher-level code — the difference between an algorithm to solve an equation and an abstracted, linguistic command — is mostly about proximity to hardware. The idea that code is only code if it is executable may make some theoretical sense; a programmer “is produced through the act of programming” and a “source code only becomes a source after the fact” [Chun 2008, p. 24]. But in practice, code is often not executed and sometimes designed as such (e.g., code poetry).

[13] The following endnotes provide an explanation of the punch card system, paraphrasing Lovelace’s translation, both her and Menabrea’s words (with help from other works cited in this paper). While Babbage only describes two types of punch cards in his journals, by the time the *Notes* are written, there are at least three. *Operation Cards* determine the algebraic state (add, multiply, etc.). *Variable Cards* inform the machine which columns (V’s) in the “Store” to fetch values from and deliver intermediate results to. These are alternately expressed by Lovelace as either *Supplying Cards* or *Receiving Cards*, of which there are usually three per operation. For instance, if the machine needs to add together variables A + B to make C, an *Operation Card* will activate the “addition” state, two *Variable Cards* will “supply” A and B by designating the two columns where they are stored, and a third *Variable Card* will identify a third column to “receive” the calculated sum C (at which point the calculation will end, or C can be further used as a new variable in a future operation). These *Variable Cards* contain additional information, such as whether the column should be reset to zero after the calculation is complete, whether the number should be toggled between positive and negative, and whether the column should be treated as a quantity or, in the case of indices of power, types of “operations” (in which case *Variable Cards* are called upon to act as *Operation Cards*). Rather than pointing to columns, *Number Cards*, Menabrea explains, specify an actual number (or algebraic expression); these cards are used for timesaving: For example, by having a complex value such as *Pi* already expressed by punch card, the Engine is saved from calculating *Pi* every time it is needed to determine a circumference. This would allow the Engine to, for instance, calculate the next Bernoulli number without having to calculate the preceding one.

[14] A possible fourth type of punch card, called *Combinatorial Cards*, are the least explained in published materials but probably the closest approximate to code. These cards are instructions to the Engine for manipulating the other processes. If all punch cards used in a calculation enter a “library” — a sort of holding area, or cache — *Combinatorial Cards* can manipulate the prism to access those already used cards and/or skip others. As Lovelace explains in the *Notes*, because punch cards are entered into the Engine in a particular order, the *Combinatorial Cards* can do two things: (1) enter the prism into a “reverse” or “forward” state, and (2) set an end variable so the machine knows when to return to normal function. As Plant points out, “The cards were selected by the machine as it needed them and effectively functioned as a filing system, allowing the machine to store and draw on its own information” [Plant 1995, p. 52]. This greatly reduced the number of cards required; the calculation of the eighth Bernoulli number required under 100.

[15] As Wilfried Hou Je Bek points out, the term LOOP denotes a vast chain of beings (iterators, GO TO statements with passing arguments, count-controlled loops, condition-controlled loops, collection-controlled loops, tail-end recursion, enumerators, continuations, generators, Lambda forms, et al.) [Hou Je Bek 2008, p. 182]. It includes two parts that are no different in function but different in cause and feel: iteration and incursion. The PERL glossary defines “iteration” as “doing something repeatedly.” The entry for “recursion” begins: “The art of defining something in terms of itself,” and ends: “[Recursion] often works out okay in computer programs if you’re careful not to recurse forever, which is like an infinite loop with more spectacular failure modes.”

[16] Babbage is credited with inventing the general-purpose computer for the same reasons that the Analytical Engine is different than the Difference Engine and the Silver Lady. Those automatons each have single states: add or dance. But the Analytical Engine allows for both variables and different states. While a washing machine allows for settings (hot and cold) and uses multiple states (rinse and spin), the Engine, due the programmer’s ability to set values, branch, and loop, allows for an infinite amount of “free play” between variables and states. It is that quality which, in addition to her Leap, Lovelace expresses so beautifully in the *Notes* — an illustration of the Engine as “the material expression of any indefinite function of any degree of generality and complexity” (“Note A”). As Angluin put it, “There in [the 1843 paper], a century before its time, is the concept of a general-purpose digital computer, developed to an amazing degree of sophistication” [Angluin 1976, p. 6–7].

[17] According to Fuegi and Francis, Howard Aiken referenced Babbage’s designs in 1937 while working on the electric calculator for IBM, and Konrad Zuse encountered Babbage’s designs the same year [Fuegi and Francis 2003, p. 18]. Actual software would not be theorized until Turing or fully realized until the Manchester Baby in 1948.

[18] The labor organization that Chun describes at Bletchley Park persisted until the invention of automatic computer programming allowed coders to communicate with these new subprograms instead of the machine directly. Ironically, this sparked fear that the “manly” practice of coding was becoming de-skilled. Instead of suffering through machine language like a “real man,” shortcuts allowed programmers to write without knowing how the subroutines worked [Chun 2008, p. 43]. The development of such languages ended up displacing mostly female computers and ushering in a male-dominated era of computer science. Chun points out that the very anxieties around feminized software have, more recently, “paradoxically led to the romanticization and recuperation of early female operators of the 1946 Electronic Numerical Integrator and Computer (ENIAC) as the first programmers, for they, unlike us, had intimate contact with and knowledge of the machine” [Chun 2008, p. 19].

Works Cited

- #Chuan Chu 2008** Chu, J.C. (1980). "Computer development at Argonne National Laboratory", in N. Metropolis, J. Howlett and G. Rota (eds.) *A History of Computing in the Twentieth Century*. Academic Press, pp. 345-346.
- Abramson 2008** Abramson, D. (2008) "Turing's responses to two objections", *Minds & Machines*, 18.
- Angluin 1976** Angluin, D. (1976) "Lady Lovelace and the analytical engine", *Newsletter of the Association for Women in Mathematics*, 6, pp. 6-8.
- Babbage 1864** Babbage, C. (1994). *Passages from the life of a philosopher*. (M. Campbell-Kelley, Ed.). Piscataway.
- Babbage 1961** Babbage, C. (1961) "Of the Analytical Engine", in P. Morrison and E. Morrison (eds.) *Charles Babbage and His Calculating Engines: Selected Writings*. Dover Publications.
- Babbage 2019** Babbage, C. (2019) *Passages from the life of a philosopher*. Good Press.
- Balsamo 2011** Balsamo, A.M. (2011) *Designing culture: The technological imagination at work*. Duke University Press.
- Bolter and Grusin 2000** Bolter, J.D., and Grusin, R.A. (2000) *Remediation: Understanding new media*. MIT Press.
- Bowden 1953** Bowden, B.V. (1953) *Faster Than Thought*. Pitman Publishing Corporation.
- Bromley_1990** Bromley, A. G. (1990). "Difference and analytical engines", in W. Aspray (eds.) *Computing before computers*. Iowa State University Press, pp. 59-98.
- Brown 2018** Brown, S. (2018) "Delivery service: Gender and the political unconscious of digital humanities", in *Bodies of Information: Intersectional Feminism and Digital Humanities*, pp. 261-286. Minnesota University Press.
- Burton and Hicks 2020** Burton, E., and Hicks, M. (2020) "A history of women in British telecommunications: Introducing a special issue", *Information & Culture*, 55(1), pp. 1-9.
- Campbell-Kelly and Aspray 1986** Campbell-Kelly, M., and Aspray, W. (1986) *Computer: A history of the information machine*. HarperCollins.
- Chun 2008** Chun, W.H.K. (2008) "On sourcery and source codes", in *Programmed Visions: Software and Memory*, pp. 19-54.
- Daston 2018** Daston, L. (2018) "Calculation and the division of labor, 1750-1950", *Bulletin of the German Historical Institute*. pure.mpg.de.
- Eck 2000** Eck, D. (2000) *The most complex machine: A survey of Computers and Computing*. A.K. Peters.
- Ensmenger 2010** Ensmenger, N. (2010) *The computer boys take over: Computers, programmers, and the politics of technical expertise*. MIT Press.
- Essinger 2014** Essinger, J. (2014) *Ada's algorithm: How Lord Byron's daughter Ada Lovelace launched the digital age*. Melville House.
- Forbes-Macphail 2016** Forbes-Macphail, I. (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*, pp. 143-168.
- Fuegi and Francis 2003** Fuegi, J., and Francis, J. (2003) "Lovelace & Babbage and the creation of the 1843 'Notes'", *IEEE Annals of the History of Computing*, 25(4), pp. 16-26. Available at: <https://doi.org/10.1109/MAHC.2003.1253887>
- Galvan 2010** Galvan, J. (2010) *The sympathetic medium: Feminine channeling, the occult, and communication technologies, 1859-1919*. Cornell University Press.
- Gitelman 1999** Gitelman, L. (1999) *Scripts, grooves, and writing machines: Representing technology in the Edison era*. Stanford University Press.
- Goldstine and Van Neumann 1947** Goldstine, H.H., and Von Neumann, J. (1947) "Planning and coding problems for an electronic computing instrument", *Report on the Mathematical and Logical Aspects of an Electronic Computing Instrument Part II*, Vol. 1. Institute for Advanced Study.
- Hayles 2010** Hayles, N.K. (2010) *My mother was a computer: Digital subjects and literary texts*. University of Chicago Press.
- Hicks 2017** Hicks, M. (2017) *Programmed inequality: How Britain discarded women technologists and lost its edge in computing*. MIT Press.

- Hollingsm Martin, and Rice 2018** Hollings, C., Martin, U., and Rice, A. (2018) *Ada Lovelace: The Making of a Computer Scientist*. Bodleian Library.
- Hou Je Bek 2008** Hou Je Bek, W. (2008) "Loop", in M. Fuller (ed.) *Software Studies: A Lexicon*. MIT Press.
- Kittler 1990** Kittler, F.A. (1990) "Queen's sacrifice", in *Discourse Networks 1800/1900*. Translated by M. Metteer. Stanford University Press.
- Kittler 1999** Kittler, F.A. (1999) *Gramophone, film, typewriter*. Translated by G. Winthrop-Young and M. White. Stanford University Press.
- Kreilkamp 2005** Kreilkamp, I. (2005) "Speech on paper: Charles Dickens, Victorian phonography, and the reform of writing", in *Literary Secretaries/Secretarial Culture*, pp. 13-31. Routledge.
- Lardner 1834** Lardner, D. (1834) "Babbage's calculating engine", *Edinburgh Review*, 59, pp. 263-327.
- Lovelace 1843** Menabrea, L.F., and Lovelace, A. (1843) "Sketch of the analytical engine invented by Charles Babbage", *Taylor's Scientific Memoirs*, pp. 666-731.
- Misa 2016** Misa, T.J. (2016) "Charles Babbage, Ada Lovelace, and the Bernoulli numbers", in *Ada's Legacy: Cultures of Computing from the Victorian to the Digital Age*, pp. 11-31. Association for Computing Machinery.
- Padua 2017** Padua, S. (2017) "Picturing Lovelace, Babbage, and the analytical engine: A cartoonist in mathematical biography", *BSHM Bulletin: Journal of the British Society for the History of Mathematics*, 32(3), pp. 214-220.
- Plant 1995** Plant, S. (1995) "The future looms: Weaving women and cybernetics", *Body & Society*, 1(3-4), pp. 45-64.
- Plant 1997** Plant, S. (1997) *Zeros + Ones: Digital Women and the New Technoculture*. Fourth Estate.
- Price 2003** Price, L. (2003) "From ghostwriter to typewriter: Delegating authority at fin de siècle", in R.J. Griffin (ed.) *The Faces of Anonymity: Anonymous and Pseudonymous Publication from the Sixteenth to the Twentieth Century*. Palgrave Macmillan.
- Price and Thurschwell 2005** Price, L., and Thurschwell, P. (2005) "Invisible hands", in *Literary Secretaries/Secretarial Culture*, pp. 1-12. Routledge.
- Priestley 2011** Priestley, M. (2011) "Babbage's engine", in *A Science of Operations*. Springer.
- Stabler 2017** Stabler, J. (2017) "1816 in the manuscripts of Lord Byron and Mary Shelley", *Keats-Shelley Journal*, 66, pp. 55-76.
- Stein 1985** Stein, D. (1985) *Ada: A life and a legacy*. The MIT Press.
- Target 2018** Target, S. (2018) "What did Ada Lovelace's program actually do?", *Two Bit History*. August 18. Available at: <https://twobithistory.org/2018/08/18/ada-lovelace-note-g.html>.
- Target 2020** Target, S. (2020) Interview by Z.M. Mann. Video call.
- Terras and Nyhan 2016** Terras, M., and Nyhan, J. (2016) "Father Busa's female punch card operatives", in *Debates in the Digital Humanities*, pp. 60-65. University of Minnesota Press.
- Thurschwell 2001** Thurschwell, P. (2001) *Literature, Technology and Magical Thinking, 1880-1920*. Cambridge University Press.
- Toole 1998** Toole, B.A. (1998) *Ada, the enchantress of numbers: Prophet of the computer age, a pathway to the 21st Century*. Critical Connection.
- Turing 1986** Turing, A.M. (1986) "Proposal for development in the mathematics division of an automatic computing engine (ACE)", in B.E. Carpenter and R.W. Doran (eds.) *A. M. Turing's ACE Report of 1946 and Other Papers*, pp. 38-39. MIT Press.
- Turing 1995** Turing, A.M. (1995) "Lecture to the London Mathematical Society on 20 February 1947", *MD COMPUTING*, 12, pp. 390-390.
- Turing 2009** Turing, A.M. (2009) "Computing machinery and intelligence", in *Parsing the Turing Test*, pp. 23-65. Springer.
- Whitson 2015** Whitson, R. (2015) "Critical making in the digital humanities", in *Introducing Criticism in the 21st Century*, pp. 157-176. Edinburgh University Press.

Recommendations

DHQ is testing out three new article recommendation methods! Please explore the links below to find articles that are related in different ways to the one you just read. We are interested in how these methods work for readers—if you would like to share feedback with us, please complete our short evaluation survey. You can also visit our documentation for these recommendation methods to learn more.

SPECTER Recommendations

Below are article recommendations generated by the SPECTER model:

1. Introduction: Reading Code in the Age of AI, 2026, Jeremy Douglass, University of California, Santa Barbara; Mark C. Marino, University of Southern California
2. Reverse Engineering the Gendered Design of Amazon's Alexa: Methods in Testing Closed-Source Code in Grey and Black Box Systems, 2023, Lai-Tze Fan, University of Waterloo
3. A review of Nathan Ensmenger, *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise* (Cambridge, MA, and London: MIT Press, 2010), 2013, Trisha Campbell, University of Pittsburgh
4. Minimal Computing from the Labor Perspective, 2022, Tiffany Chan, University of Victoria Libraries; Jentery Sayers, University of Victoria
5. Defactoring Pace of Change, 2026, Matt Burton, University of Pittsburgh; Joris van Zundert, Huygens Institute of the Royal Netherlands Academy of Arts and Sciences

DHQ Keyword Recommendations

Below are article recommendations generated by DHQ Keywords:

1. Reverse Engineering the Gendered Design of Amazon's Alexa: Methods in Testing Closed-Source Code in Grey and Black Box Systems, 2023, Lai-Tze Fan, University of Waterloo
2. A review of Nathan Ensmenger, *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise* (Cambridge, MA, and London: MIT Press, 2010), 2013, Trisha Campbell, University of Pittsburgh
3. Orientation: "Man and His Tool, Again?", 2015, Nicole Starosielski, New York University
4. Books Aren't Dead: Resurrecting Audio Technology and Feminist Digital Humanities Approaches to Publication and Authorship, 2021, Emily Edwards, Bowling Green State University; Robin Hershkowitz, Bowling Green State University
5. The Lives of Mistresses and Maids: Editing Victorian Correspondence with Genealogy, Prosopography, and the TEI, 2022, Kailey Fukushima, University of British Columbia; Karen Bourrier, University of Calgary; Janice Parker, Calgary Public Library

TF-IDF Recommendations

Below are article recommendations generated by the TF-IDF Model:

1. Tremendous Mechanical Labor: Father Busa's Algorithm, 2020, Geoffrey Rockwell, University of Alberta; Stéfan Sinclair, McGill University
2. BASIC FTBALL and Computer Programming for All, 2023, Annette Vee, University of Pittsburgh
3. Introduction: Reading Code in the Age of AI, 2026, Jeremy Douglass, University of California, Santa Barbara; Mark C. Marino, University of Southern California
4. What can the digital humanities learn from feminist game studies?, 2015, Elizabeth Losh, University of California, San Diego
5. Enlisting "Vertues Noble & Excelent": Behavior, Credit, and Knowledge Organization in the Social Edition, 2015, Constance Crompton, University of British Columbia, Okanagan; Raymond Siemens, University of Victoria; Alyssa Arbuckle, University of Victoria; Implementing New Knowledge Environment (INKE)



This work is licensed under a Creative Commons Attribution-NoDerivatives 4.0 International License.