

From Codework Poetics to the AI Writing Scene On Close Reading Codeworks and Digital Literary History

Thorsten Ries^{1*}

^{1*}Department of Germanic Studies, The University of Texas at Austin,
Austin, TX, USA.

Corresponding author(s). E-mail(s): thorsten.ries@austin.utexas.edu;

Abstract

This article proposes close reading modes for literary codeworks and AI-generated writing, demonstrating how integrating philological analysis with code-based insights uncovers relevant textual layers in digital literature. Through multimodal readings of Eric Andreychek's Perl poem *Jabberwocky* and Nick Montfort's *Taroko Gorge*, the study demonstrates how analytical methods known from "classic" literary scholarship may augment and benefit a Critical Code Studies perspective. The article features an analysis of previously unpublished archive material from Jörg Piringer's tiny poems. In exploring questions of authorship, reception, and canon formation, the article also addresses uneven global representation and the challenges of preserving digitally ephemeral works. By emphasizing close reading strategies for both code and AI, the discussion points toward broader critical frameworks necessary for navigating an evolving digital literary literature.

Keywords: Digital Literature, Codeworks, Artificial Intelligence, Digital Humanities, Digital Poetics

1 Introduction

As an umbrella term for various forms of born-digital literature, such as electronic literature, literary texts generated by algorithms or AI, code poetry, hypertext literature, audiovisual dynamic and interactive digital texts, and platform and social media literature, "digital literature" has become an international literary phenomenon, with

its own organizational body, the Electronic Literature Organization (ELO).¹ Digital literature not only reshaped our scholarly understanding of text as a concept, but also requires a substantial revision of models for reading, analysis, and literary history of digital literature and codeworks. More importantly, this type of literature requires the reader to take a historical media-specific, technological perspective that accounts for the dynamic processing character of the digital literary object, its system context: its “technosocial context”, as Mark Marino termed it (Marino, 2020, 19; see also Hayles, 2002; Manovich, 2002; van Zundert et al. 2020, in critical variation to Marino see also a programming language-“philological” perspective: Lennon, 2024). At the same time, literary criticism has only started developing close reading modes for digital literatures and codeworks (executable or non-executable source code meant to be “read” as literary text). The conceptual and material multi-layeredness of digital literature and born-digital works of art (Hayles, 2004; Blanchette, 2011), their fundamentally different modes of production, materiality, distribution, *mimesis*, and temporality create a disciplinary necessity to extend disciplinary approaches in order to account for the techno-historical, material, conceptual, as well as the language-related literary and cultural aspects of these works of art. The disciplinary challenge is to develop multimodal close reading modes that integrate the conceptual, aesthetic, and semantic dimension of “digital literature” and the digital work of art with the analysis of the media-specific historical technological materiality in a way that connects with the established methodological paradigms of philology and text analysis, as well as traditions of literary history, art history and culture history. While digital literature strategically disrupts and ironically reflects the social “game” of literature by resetting and reshuffling the parameters of the media, materialities and practices of “reading”, “language” / “text”, and the mechanisms of inclusion and exclusion of audiences, not reconnecting media-historical and technical perspectives back to analysis, methods, and traditions of philology and literary history would risk that digital literature could remain a niche footnote of literary and culture history.

This article endeavors to methodologically reconnect the technological perspective and the historical technosocial context with digital-dynamic close reading perspectives and (digital) literary history by revisiting and amending the interpretations of two “classic” historical examples of digital literature by Nick Montfort and Eric Andreychek in a new interpretation (in sections 3.1, 3.2), and adding philological perspectives on born-digital archives with the analysis of previously unpublished archive material of a third codework by Jörg Piringer (in section 3.3). In short, this article attempts to develop a multi-layered close reading perspective that integrates aspects of the conceptually multilayered aesthetics of code poetry and its technosocial contexts (e.g. dynamic metrics and prosody of the digital poetry at runtime, multi-layered semantics of code, cultural aspects of choice of programming languages) with literary historical aspects and the reception in literary communities. The goal of this article is to formulate elements of a framework for digital literary studies that integrates literary codeworks as literature in its own right into humanities research and connects the methodologies of traditional literary and philological scholarship with digital humanities methodology. In

¹Research for this article was funded by the Research Foundation - Flanders (FWO), research project *Hard Drive Philology / Source Code Philology [...]* (Ghent University, 2017–18, 2019–20). Some of the material in this article appeared in Ries, 2021, in German.

section 4, "Close Reading AI", this article explores the challenges and methodological extension of this multimodal approach for close readings of AI-based digital literature. The article also addresses issues in the early formation of the digital literature canon and problematic tech-history based narratives of digital literary history, including imbalances in the international representation of regional digital literatures, as well as its preservation and recognition in scholarship (in section 2).

2 Text.DigitalLiterature

The international scholarship on digital literature has long been defined by a few prominent scholars whose seminal works informed a phase of research of this emerging field (Aarseth, 1997, Bootz, 2006, Drucker, 2017, Emerson, 2014, Flores, 2021, Glazier, 2006, Grigar et al., 2021, Hayles, 2004, 2007, Kirschenbaum, 2021, Marino, 2020, Di Rosario et al., 2021, and Wardrip-Fruin, 2010). While in the beginning academic approaches to digital literature were constituted by the overlap of the creative and the academic field, it was not until recently that it was fully acknowledged as a subject of Digital Humanities research (see Grigar et al., 2021). This emerging literary form, however, problematizes concepts of traditional literary studies: the concepts of text, literature, literary process, reading, analysis and methodology are at stake. "Digital texts" may refer to dynamic screen performances, user interaction, executable or non-executable source code, or even AI models and their output. Textuality seems to be the key issue for literary scholarship when it comes to the development of analytical, comparative, and contextualizing methods for a multi-layered, historically changing and dynamic notion of electronic literary "text." The term "digital literature" is probably the most common designation for born-digital literary art forms have been variously known as "electronic literature," "e-lit," or "littérature numérique").

Since its foundation in 1999, the ELO has infrequently provided updated versions of the organization's definition of "electronic literature" for speakers of English on its website, as well as in its extensive archives. The more recent by Clara Chetcuti (2020) reflects not only new forms of digital expression emerging from technical innovations, but also digital literature genres that have disappeared (e.g., "net literature," Flash-based "flash e-lit," and others), and evolving conceptual and genre definitions within digital literature (e.g., codeworks and code poetry, code as concept art, compared to hypertext literature, cyberpoetry, new media poetry, electronic poetry, e-poetry, flash fiction, etc.). The most cited definition is probably the ELO's attempt of 1999: "Electronic literature refers to works with important literary aspects that take advantage of the capabilities and contexts provided by the stand-alone or networked computer." This definition and Katherine Hayles' amended version of 2007, which adds that digital literature will normally be written on computers and received at the computer, have been equally controversial. Both emphasize the technical and performance aspects of "born digital" literature, while leaving the definition of that "important literary aspect" essentially undefined, which may exclude algorithmically generated texts that have been published on paper-based media. Hayles' well-known dictum "Print Is Flat, Code Is Deep," for instance, rests on a dichotomy between a literary display text and a literary code-text (and thus refers specifically to a relatively simple, browser-

and HTML-based tradition of hypertext literature, which is still active, see J. R. Carpenter, 2012, and ASCII art in the code of the *Vienna Code Poetry Slam* 2017 website). Definitions like these emphasize the technical and performance aspects of “born digital” literature, and restrict “digital” literature to being consumed in one medium only, rather than including a broader spectrum of textual mediality: they may, for instance, exclude algorithmically generated texts that have been published on paper media or codeworks, where executable or non-executable source code and the algorithm it represents become the “text”. What is at stake here, again, is the textual quality of digital literature and code poetry as such, as Cayley’s essay *The Code Is Not the Text (Unless It Is the Text)* makes abundantly clear when he critically revisits Hayles’ theoretical argument about the “flickering signifier” (Hayles, 1999), strengthening the role of author subjectivity (“text”) against “code”: “it must, typically, be a distinct part of the global textual system; it must be possible to recompile the codes as operative procedures, as aspects of live-art textual practice. The code is not the text” (Cayley, 2002). However, relatively early on in his argument, he formulates the essential textual bifurcation of code as processed (addressed to a processor) and code as read text (addressed to a reader), thereby highlighting the difference in temporality and the potential of disruptive reading or processing (crack, hack):

- If a codework text, however mutually contaminated, is read primarily as the language displayed on a screen then its address is simplified. It is addressed to a human reader who is implicitly asked to assimilate the code as part of natural language. This reading simplifies the intrinsically complex address of writing in programmable media. At the very least, for example, composed code is addressed to a processor, perhaps also addressed to specific human readers (those who are able to ‘crack’ or ‘hack’ it); while the text on the screen is simultaneously? asynchronously? addressed to human readers generally. Complexities of address should not be bracketed within a would-be creolized language of the new media utopia.

(Cayley, 2002)

In her later work, accounting for technological and conceptual historical development, Hayles steers in the direction of theorizing human-technology “cognitive assemblages,” building on models of cross-media reception, media convergence, and historical remedialization. She argues that digital literature necessarily builds on the expectations and reading habits that have been informed by literary print media culture (Hayles, 2016, 85-86). With this turn of Hayles’ thinking, electronic literature would not necessarily be based on a human language textual form, but could be understood more generally as multimedia artworks and codeworks that question the history, contexts, production and reception modes of literature in transformative ways (Hayles, 2007, 4).

In continuing the discussion, Noah Wardrip-Fruin (2010) models the definition of digital literature as centered around five media-specific elements: data, process, interaction, surface, context, with digital *artworks* (not just texts) being defined by “attention to language”. He is clearly still focused on hypertext literature (Wardrip-Fruin, 2010, 29), building on Aarseth’s definitions of “cybertexts” and “ergodic literature” as nonlinear texts that require “non-trivial” activity and decisions (Aarseth, 1997, 1-2), where cybertexts require a calculation to construct the scripton. It is noteworthy that Wardrip-Fruin’s own digital art transgressed this definition of digital literature (see

Wardrip-Fruin 2003-2005). Unlike the aforementioned approaches, Wardrip-Fruin does not solely focus on the media assemblage of technological, semantic, and conceptual levels of the digital artwork, but adds an essential element of language reflection.

Phillipe Bootz (2006) and Frederick Block (2015) provided important new perspectives by formulating the “important literary aspect” of digital literature in two versions of reflections on media and technology. Bootz defines digital poetry as a technological artifact creating not a virtual “cyberentity” of machine and reader, but a “semiotic gap” between its visual performance component and the technology component, which together form “the” text – from different perspectives, as two levels that create a specific tension around the “semiotic gap”: “Digital poetry today explores the role of language in signs that use this gap, and which only exist thanks to it. In this case, programming can become a new condition, a new context for poetic creation” Bootz, 2006, 2). While Bootz conceives of digital poetry as a field of semiotic tension between textual levels of the technological implementation (performance, programming, technological ensemble), Block configures “digital poetry” (*digitale Poesie*) within a discipline-historical analysis of the “language game” (*Sprachspiel*). Borrowing from Luhmann’s systems theory, he conceptualizes digital poetry as a “medium of reflection and communication” and as a “second-order medium”:

[In the digital poem,] reflection of the materiality of language and perception is signalled with the use of terms like “visual,” “auditory,” “sound-“ [...] Narrowly bound up with this is the designation of the conditions of information technology, the sensitivity for developments in media culture: “typewriter,” “radio,” “film,” “holo-,” “code,” “computer,” “digital,” etc. In this large range of variation, the “intermedial” or “intermediate” aspect plays a significant role, the interest in differences, interfaces, interference, transitions: an aesthetic of the in-between that triggers meta-reflection. [...] Poetry self-referentially conceptualizes reflexivity as the basic mechanism of cognition and communication, or of identity and sociality, for that matter. As a medium for reflection and communication, poetry is a second-order medium. (Block, 2015, 34; trans. by Thorsten Ries)

Especially net art projects like Mark Amerika’s *GRAMMATRON*, Michael Joyce’s *Twelve Blue* and the Dutch-Flemish art collective Jodi’s *%20location* may be read as examples for this type of second-order media reflection.

More recent approaches to theorizing the “text” of digital literature extend the idea of a second-order medium of reflection towards its present limitations. Florian Cramer calls our present already “post-digital,” as ubiquitous graphical user interfaces and environments make the texts indiscernible from the digital architectures and the mechanisms which support them. This observation has led some to the radical conclusion that there is no such thing as “electronic literature” or “internet literature” (e.g. Baldwin, 2015, 45). Some even assert that the actual productive (literary) potential of digital literature consists of a set of strategies for failure (Szilak, 2010, 2015). Lori Emerson points to the counter-movements of “hacktivism” and interface hacking, which seek to make the now “invisible” interfaces visible again for critique (*iPoem* xvi, 1, 30-46), thereby emancipating authors and readers (Emerson, 2014, 24-26). One famous example is Jaromil’s *forkbomb* of 2002, a malware exploit in the form of a short character string “:(){ :|:& };;.” If executed on the commandline of a Unix system, this string made the system copy endlessly forking processes into the RAM of the system -

at that point in time, leading to a system crash. Emerson also cites Jörg Piringer’s 2010 iPad app *abcdefghijklmnopqrstuvwxyz*, a conceptual precursor of his *tiny poems*, as a hacktivist intervention, aiming at “defamiliarizing” “invisible” user interfaces (24-26; see also section 3.3 below, Piringer, 2015b). Repurposing a term coined by Alan Sinfield in 1997, such digital art, digital text, and digital scholarship creates, re-creates, and discovers “faultlines” in how one understands the outputs of digital technology, its user interfaces, ecosystems, and economies. The two examples of Jaromil’s *forkbomb* and Piringer’s *tiny poems*, in combination with the code poetry mentioned in passing before, point not only to the layeredness of the “text” of digital literature, but also to its highly differentiated temporality. This suggests a specific need for scholarly analyses of such digital literature: next to the coding process, the screen performance, the distribution, and the reader’s process, a media-specific analysis may have to take into account the runtime process of the code in a computer’s RAM, the API’s, bandwidth, and AI models that are used in the networked cloud, to say nothing of the technological specifics of the type of historical hard- and software architecture that was in play.

The digital materialities and histories of digital literature are distributed across architectures, to speak with Johanna Drucker (2017), as is the born-digital historical record of the digital age (see Ries, 2022). These architectures and their affordances, Jaromil’s and Piringer’s example show, are not just media of creativity, delivery, and performance; they need to be read - as part of the digital work of art and its media reflection - as material technological expressions of historical design decisions and digital culture, as Blanchette (2011) and Kittler (1993b) showed. The digital forensic investigation of born-digital writing and distribution processes may also serve as a philological model of the media specific analysis of the digital materiality and temporality of digital literature texts (Ries, 2018; Kirschenbaum, 2008).

Current scholarship is in the process of developing modes of reading AI generated digital art, AI models, self-trained or pre-trained Large Language Models (LLM, e.g. GPT-4, See Bajohr, Gilbert, 2021; Parrish, 2020, 2021a, 2022, NaNoGenMo; Bot Mimikry: Lennon, 2024). As section 4 will show, the analytical concepts “text” vs. “code”, “sigital materiality” and “temporality”, which were so far key to digital philological reading modes, conceptual or technical reading of the digital works of art, and critical readings of faultlines shift with the advent of AI, but can be adapted as a multimodal research program with connections to humanities methodology.

3 History.DigitalLiterature

The several attempts of the Electronic Literature Organization (ELO) to write a history of digital literature reveal a second set of problems dealing with digital literatures: periodization. Katherine Hayles, in a first version of 2007, suggested two generations: 1. Pre-Internet, “blocks of text [...] with limited graphics, animation, colors and sound, [...] the hypertext link is [...] the distinguishing feature”; 2. Then, from 1995 on: “later works use a wide variety of navigation schemes and interface metaphors that tend to de-emphasize the link as such,” stressing instead “multi-modal capabilities of the Web.” Later, Leonardo Flores (2021) posits three generations: 1. Programming Languages, Word Processing, Authoring Software, Hypercard; 2. from 1995 on: Electronic Poetry,

Hypertext, Installations, Net.Art, Virtual and Augmented Reality; 3. After circa 2005: Platform Literature, Mobile and Touchscreen Apps, Remix, Memes, Virtual Reality.

Such periodizations have intrinsic systematic flaws (see Bajohr, Gilbert, 2021). While Hayles explicitly rejects the idea of a succession of generations, she then still reintroduces it by calling first-generation works “classical” (2007, section 2). The “generations” are defined principally in terms of technological history, most often focusing on innovations of screen performance – an assumption that for the abovementioned periods privileges hyperlinks, hypertext and multimedia literature on the web, rather than the older, often more diverse, and in parallel existing and further evolving genres that figure into a possible history of digital literature. The systematic problem consists, thus, on the one side, in basing the “generation” periodization based on screen performance, which one could describe – reusing Nick Montfort’s term – as “screen essentialism” (Montfort, 2004), and does not generalize or age well. On the other hand, a periodization based on dominant technological features of hard- and software (and their literary use by certain artists) is not compatible with any periodization model used in literary history, and complicates the integration of digital literature into the historical lines of disciplinary literary studies. Both versions of reducing the history of this literature to “generations”, largely defined by technologically defined user experiences, simplify a rapidly diversifying literary phenomenon and appear to imply that generations replace each other in a progress movement.

Moreover, these attempts in periodizing digital literature and art essentialize western norms and experiences with electronic literature and art, they do not represent the emphases and development timelines in postcolonial contexts or smaller language communities. Shanmugapriya and Menon (2019), for instance, show that such historical models misrepresent the emphases and timelines of development in India and Asia as a whole. The vast digital literature from Latin America only began to be explored more widely in research after 2016 (see Bush et al (eds), 2016; Taylor (ed), 2019; L’Hoeste et al. (eds.), 2022; Szurmuk et al (eds.), 2022). There is thus good reason for the Electronic Literature Foundation (ELO), where the majority of members, organizers, and authors are still from the Americas and Europe, to sponsor MLA panels with the title *Decolonizing Electronic Literature*, and for the ELO 2023 conference to be organized around the overarching theme *Overcoming Divides: Electronic Literature and Social Change*. Also the scholarly work to recognize LGBTQAI+ digital literature creators from Christopher Strachey to Allison Parrish, their productive communities, and their works, in literary research has merely just begun (see Gaboury 2022, Parrish 2021, 2023, Liu 2020).

Additionally, language barriers deeply engrained into international scholarship on the one hand contribute to the propagation of colonial histories and histories of discrimination, on the other they also prevent digital literatures and art in smaller language communities from being covered by international research. For instance, the rich digital literature tradition in the German-speaking DACH region (Germany, Austria, Switzerland) remains relatively isolated from international research and recognition, despite its relevance. Some overviews to the international history of digital literature, such as Di Rosario, Meza and *Grimaldi’s Origins of Electronic Literature*, seem to ignore all digital literature production in German-speaking regions after Lutz’ 1959 original

Stochastische Gedichte.² There are two main systematic reasons for this scholarly neglect: the language barrier, and the fact that local historical developments of digital literatures not always align with the historical model outlined before.³ Overall, however, the often unevenly distributed scholarly appreciation of digital literature across cultural and language contexts have a lot to do with the complexity of digital literature and the digital spaces it inhabits: The rather inaccessible textuality of codeworks, the publication of code-art in today technically obsolete, marginalized web-forums or “dark archives” - such as some forum posts by the Netherlandic-Belgian collective Jodi (Cramer, 2011, 237-242, 245-251, „Kartografien des Interface”) - make the appreciation, discovery, preservation, and canonization of digital literature a challenging, time-sensitive task. In the overlap sphere of digital art and academic digital humanities, Tanasescu (Margento) reflects this issue of building a performative canon with his project *#GraphPoem* (2020).

4 Reading.DigitalLiterature

Literary scholarship, philological and digital humanities research have started developing reading modes for digital literatures, while a more general framework is still in development. This section reviews and extends scholarly reading modes applied to digital literature using two ‘classic’ examples of digital codeworks (Cramer, 2011, 3.1; Marino, 2020, 3.2), and will analyze previously unpublished code material from Jörg Piringer’s personal archive (3.3). Previous “traditional” literary scholarship failed to

² Theo Lutz’s 1959 implementation of the *Stochastic Texts* on a Zuse 22 computer was directly inspired by Lutz’ collaboration with Max Bense, at the time professor of philosophy and theory of the sciences at Stuttgart University and spiritus rector of the Stuttgart Circle, and his doctoral student Rul Gunzenhäuser. Max Bense’s legacy is tainted, as he was a member of Hitler’s SA from 1933-34, and became a member of the NSDAP in 1937. His numerous publications during the *Drittes Reich* are a mixed record of open reverence to the National Socialist regime and issue-related criticism thereof. See Bernhart 2019, here 68. See also Bense, 1962.

³ Research on digital literature in the DACH region is - with few exceptions - unfortunately so far a project largely limited to scholarship published in German. Hannes Bajohr, Friedrich W. Block, Florian Cramer, Chris Funkhouser, Peter Gendolla, Annette Gilbert, Julia Nantke, Saskia Reither, and Roberto Simanowski, have laid foundations for working with German digital literatures in DH research, in German (see bibliography). The Deutsches Literaturarchiv Marbach am Neckar (DLA Marbach, the German Literature Archive Marbach) - not the ELO online archive or *ALTX* - has taken up the project to archive German born-digital literature. The historical development of the digital literature in the German speaking countries does not align with the ELO periodization: Not only are the conceptual, still paper-based poetic precursors of text-generating algorithms (Konrad Beyer, *der vogel singt. eine dichtungsmaschine in 571 bestandteilen*, 1957-58) and hypertext novels (Andreas Okopenko *Lexikon einer sentimentalen Reise zum Exporteurtreffen in Druden*, 1970) in the DACH region not included or easy to integrate in this periodization. Since the late 1990s, the growth of German-language “Netzliteratur” and “digitaler Poesie” has not held pace with the international proliferation of electronic and digital literature, despite its role in the origins and continuous production by few artists and groups residing in Germany, Austria, and Switzerland. Block even assumed that digital literature might already be a thing of the past (Block, 2015, 24). However, this phase also saw the works by Johannes Auer (e.g. the search-engine transcoding work entitled *Searchsonata 181* [2010]), the “Netzliteratur“-circle (Johannes Auer, Christiane Heibach, Beat Suter), up to Beat Suter’ and René Bauer’s most recent activist projects engaging the history of technology (*ToniZ*, *Turing’s Assembly Line*, using a PLATO emulator). It saw Jörg Piringer’s *Coded Poetry*, *Data Poetry*, Interface Hacking and his interactive, kinetic app poems. This period in German digital literature also brought the projects of 0x0a, Hannes Bajohr and Gregor Weisbrod, the most well-known being the activist preservation corpus-processing project *Die Sprache Pegidas / Glaube, Liebe, Hoffnung* (2015). 0x0a was also involved in the conceptual artwork published in the series entitled *Think Poetic Ways (Poetisch Denken)*, which were generated by AI models trained analytically by neural networks and a text corpus - for instance, works by Monika Rinck and Ann Cotten, among others, and the code-translation of Montfort’s generative novel *Megawatt* (2019) into German (*Poetisch Denken I-IV* 2020).

integrate digital literatures into the canon as a genre or set of genres, because a connective multimodal, multilayered mode of close reading and historical contextualization was missing. On the other hand, some scholars working on digital literature tended to highlight the conceptual and media-specific incompatibility of the genres (Hayles, 2004). This section scopes the multimodal, multilayered compatibility of literary and technological reading modes, in the form of philology, close reading, and historical contextualization, using a selected exemplary cases and existing studies.

While often seen as foundational, but also internationally isolated in its style and focus on material aspects of the history of technology (Blanchette, 2011) and the intersections of culture and technology (Geoghegan, 2013), German media-historical scholarship that started with Friedrich Kittler is still at the historical foundations of born-digital philology, digital literary criticism, and critical code studies. Kittler's reading of the interplay between hard- and software-architectures as expression of technology and design cultures as deeply rooted in the history of culture techniques, for instance in *Protected Mode* (1993b), *There Is No Software* (1993a), and his assembly code experiments, is a historical starting point for humanities reading modes of code, technology, and media (Marino, 2020, 215-261).

Aside from overlapping digital literature and academic media studies, three emergent research areas have developed as distinct humanities perspectives on digital literature, each with well-known proponents (here represented only in selection): 1) *Critical Code Studies and Platform Studies* (Marino, 2005, 2006, 2010a, b, 2016, 2020, Montfort and Bogost, 2009, Wardrip-Fruin, 2009, Montfort et al., 2013, Emerson, 2014), 2) *Born-Digital Philology* (Kirschenbaum, 2008, Blanchette, 2011, and others), and 3) *a general Philosophy, Theory, and History of Software and Code* (Liu, 2004, Hayles, 2005, Chun, 2011, Cramer, 2005, Parikka, 2007, Berry, 2011).

It is noteworthy, to give an example, that Kirschenbaum's digital forensic approach in his 2008 book *Mechanisms. New Media and the Forensic Imagination*, as in his following born-digital work, was aligned with a philological paradigm of textuality, before his scholarly perspective took a turn to a literary history of digital text processing (2016). In *Mechanisms*, he analyzes three works of digital literature using a philological version of forensic methods: the interactive adventure game *Mystery House* (anonymous, 1980), Michael Joyce's hypertext narrative *Afternoon: A Story* (1987), and William Gibson's electronic poem *Agrippa* (1992). Kirschenbaum reconstructs the works' sources and textual history based on their digital-historical materiality, analyzing the history of the unauthorized pre-release of *Agrippa* by hackers on USENET (Kirschenbaum, 2008, 219–227, 237, 244). In his more recent work, Kirschenbaum focused on the evolution of the materiality of the digital archive (*Bitstreams*), as well as on the materiality, the “subject”, and the literary effect of AI generated literatures (Kirschenbaum, 2021). The code-philological paradigm is also visible in many recent instances of Critical Code Studies, for instance in the recent discovery and discussion of the original code of Joseph Weizenbaum's *ELIZA*, (Shrager et al., 2021) and recent work on the original code of Theo Lutz' *Stochastic Poems* (Bernhard, 2019, Bernhard, Richter, 2021). A recent number of *DHQ* (2023, 17.2) put Critical Code Studies and "Code Close Readings" on the map once more charting the path towards a clarification of code as “cultural object” and engaging in “race critical code studies” (Marino, Douglass, 2023, par 3, 14)

studying, next to digital narratives, Amazon’s Alexa, the code of deep learning AI and general Critical AI (Fan, 2023, Whalen, 2023, Boyd, 2023, Hua, Raley, 2023, see also Goatley, 2017). Recently, Lennon (2024) developed yet a variation of digital “philology” to analyze and describe broader programming language cultures and programming styles, maintaining a strict distinction between natural and programming languages, and cultures of programming (7-10).

The three examples I offer here (addressing texts by Eric Andreychek, Nick Montfort, and Jörg Piringer) can offer us insight into the necessity of developing a multimodal and multilayered scholarly perspective on digital literature that conceptually connects “traditional” literary scholarship and digital scholarship. Digital literature tends to not only reflect its “medium,” but also its own textuality in the form of technological concepts, architectures, and code / process formalisms, and at the same time reflects and subverts traditional concepts of literary textuality, conventions and expectations of literary markets and scholarship, as well as the scholarly canon. This new constellation of textuality - theorists from Kittler to Aarseth, Hayles, Kirschenbaum, Bootz, Block and Wardrip-Fruin pointed to this fact - opens a formal, contextual, philological and historical reference space within digital textuality for which the discipline still has no formal description and analysis strategy aligned and compatible with those that we have for poetry (metric analysis, etc) or prose (narratology, etc).

Marino points out that the concept of authorship is difficult to apply to program code (whether to code poetry, or to code in general). Code is often in part automatically generated, or written by multiple parties to fulfill a specific purpose in the program – it is not the kind of free expression that is associated with the language of an author’s poem or prose (Marino, 2020, 73, 224). “Meaning” thus becomes a difficult concept tied to technological purpose and forms, and to a formal technological reference system. Digital literature, in consequence, operates on levels of distributed digital materiality that are more multimodal, multilayered, and distributed across specific functional systems: The frame of reference, as well as the frame of formal organization, of the (source code of the) digital work of art becomes multimodal, multilayered, and distributed across specific functional system contexts (Kirschenbaum, 2011; see also Blanchette 2011; Lennon, 2024, 50-51, *passim*). At the same time, the materialities producing digital work of art implicates technology culture and history contexts, as well as “literary”/“text” traditions that it may refer to, “emulate,” reflect, or subvert in digital form, or that it may activate in the form of community reception. The following three case studies will make these levels of reference and philological, literary scholarship transparent, and highlight the multiple layers of textuality that are in play.

4.1 *Jabberwocky* (2000)

Florian Cramer’s book *Exe.cut(up)able statements: Poetische Kalküle und Phantasmen des selbstausführenden Texts*, published 2011 in German, includes a close reading of the Perl-based, conceptual code poem *Jabberwocky* by Eric Andreychek (Cramer, 2011, 254–258). Cramer’s book is often cited for making the argument that computer malware might be read as literature (e.g., Mez [Breeze]’s *_Viro.Logic Condition[/ing]/ 1.1_* [2001], see also Nori (ed.) 2002). For the present purposes, Cramer’s literary argument, selection of examples, and method of close reading is of much broader

relevance to understanding digital literature *as* literature. Cramer traces the history of poetic calculations and algorithmic concepts in literature back to the Middle Ages, and ends up by close-reading modern code poetry, among which are the Concrete Perl Poetry by Nick Montfort (for example, his 32-character Perl programs: *All the Names of God*, *Alphabet Expanding*, *ASCII Hegemony*, *Letterformed Terrain*, see Montfort, *Concrete Perl* 2017, also: #!, 2014) and Andreychek's Perl poem *Jabberwocky*, / *Perl Port of Jabberwocky* (2000, 2001). Cramer's analysis does an excellent job at uncovering the main functionality of the system-process 'reenactment' of Carroll's poem plot in Andreychek's code, but fails to provide the technosocial context knowledge that was present in the *PerlMonks* community, nor does he analyze the effective reinterpretation and critique of Carroll's text that Andreychek performs.

Andreychek's poem is known due to its submission to the 2nd annual Perl Poetry Contest, organized by Kevin Meltzer (originating in the *The Perl Journal*), in which it was awarded the 2nd place in October 2001. The poem was submitted in a particular contest category: a "port" of a famous poem to code poetry in the programming language Perl. Andreychek "ported" Lewis Carroll's *Jabberwocky* to Perl. Consequently, after being submitted under the title *Jabberwocky*, it was published as *Perl Port of Jabberwocky* (2001) on the *PerlMonks* community website, together with a minimal commentary by the author, *Jabberwocky -- Explained*:

This is a port of the poem Jabberwocky, as seen in the Perl Poetry contest. While the output of the poem is not significant, the three characters mentioned in Jabberwocky actually exist (as processes) when running this program. In fact, each one acts out their portion of the poem. And no, the processes aren't meant to slay your system ;-)

The Perl Poetry Contests tapped into the Perl poetry movement that had started in at the beginning of the 1990s. It promoted the notoriously human reader-unfriendly programming language Perl as a medium of poetic expression (Wall et al., 2000, 647), taking on the most challenging and least intuitive programming language of its time – next to Assembler code – for the creation of a human-readable art form inspired by Donald Knuth's project *The Art of Computer Programming* (since 1968). The movement, despite Perl not being widely used anymore, is still creating occasional code poetry offshoots such as Monfort's 256-character Perl homages to the chatbot ELIZA, *Eli* (2021), and his *ppg256* series. The *PerlMonks* formed in 2000; Andreychek joined this community not much later, on 30 Apr 2001, presumably around the time he took part in the contest (Andreychek, "andreychek"). Even non-programmers can immediately recognize that *Jabberwocky* is involved in the program, indicating a human-readable plane of interpretation next to the process-level that Andreychek hints to:

```
#!/usr/bin/perl

$brillig and $toves{slithy};
for $gyre ( @wabe ) {} for $gimble ( @wabe ) {}
map { s/^.*/mimsy/g } @borogoves
and $mome{raths} = outgrabe;

if(my $son = fork) { warn "Beware the Jabberwock!";
  jaws && bite, claws && catch;
  warn "Beware the Jubjub bird" and $shun,
  $Bandersnatch{frumious} == 1; }else{
```

```

[...]

while (study($uffish)) { $_{stand} == 1; }
unless ($Jabberwock = fork) { $Jabberwock{eyes} = flame,
$Jabberwock{movement} = wiffle, $Jabberwock{location} = $wood{tulgey};
while ($coming=1) { burble }}

[...]

tell $son, "And hast thou slain the Jabberwock?".
    "Come to my arms, my beamish boy! ".
    "O frabjous day! Callooh! Callay! ",
$_{joy} = chortle if $son;

$brillig and $toves{slithy};
for $gyre ( @wabe ) {} for $gimble ( @wabe ) {}
map { s/^.*/mimsy/g } @borogoves
and $mome{raths} = outgrabe;

```

When executed with a Perl interpreter on the commandline of an operating system, the literary output of this code poem appears to be meager: the command 'perl jabberwocky.pl' results in the output of two warnings: "Beware the Jabberwock! at jabberwocky.pl line 8." and "Beware the Jujub bird at jabberwocky.pl line 11." According to Cramer's analysis - which is inspired by the author's remarks - Andreychek's *Perl Port of Jabberwocky* cannot be understood based on its visible screen performance, nor solely by reading the text of its source code, but through the reconstruction of its oblique runtime behavior during code execution: what is happening in the computer's RAM memory, and how does it 'emulate' Lewis Carroll's *Jabberwocky*? In Cramer's analysis, which is based on the author's commentary, the code poem *Jabberwocky* invisibly recreates - or reenacts - the plot of Carroll's poem, where system processes act as digital incarnations or recasts of the characters of the poem: father (the process of the script itself), son (the forked \$son), and the Jabberwock (the forked \$Jabberwock). Ultimately, as Cramer analyzes, the process \$son kills the process \$Jabberwock, leaving the father process and \$son running as untermiated 'demons' in the RAM memory of the computer: "\$son": (kill 9, \$Jabberwock), \$head = (chop \$Jabberwock);" Cramer, *Exe.cut(up)able statements* 254–258).

While Cramer makes his legitimate point here about the runtime dynamic of the code poem, he leaves some important technosocial context unmentioned, as well as does not engage in a literary analysis of the code semantic. It may, for instance, appear noteworthy that executing this script usually leaves no (forensic) trace on the computer's hard drive, and it does not "slay your system" (*Perl Port of Jabberwocky*) - unlike a forkbomb, which would crash the operating system – the security risk of bash forkbombs was common, but relatively new programmer knowledge at the time.⁴ Andreychek's nod towards forkbombs here points to the literary relevance of the *Jabberwocky* algorithm, which creates splitting system processes - just not endlessly, as a forkbomb would ('slaying' the operating system). One could conclude that the runtime process and its temporality constitute a new type of textuality - in this case a

⁴See also Jaromil's digital poem *forkbomb*, 2002. Andreychek's remark refers to a historical "DDOS" security vulnerability dubbed *forkbomb*, first discovered for Linux and UNIX systems in August 1999.

partially invisible, dynamic textuality generated by the algorithm in RAM -, which subverts the conventions of literary writing, and reading of literature.

But this code poem is also a literal code poem and a commentary on Carroll's original, a layer that Cramer widely ignores: Andreychek selects language and vocabulary from Carroll's original poem and uses the Perl syntax to pass direct speech of the figures in the form of variables ("scalars") and lists ("arrays"), visible and readable in the code, but on the processing side often without any meaningful function or effect. The emulation of Carroll's "nonsense" poem is actually even more precise, and implies an interesting interpretation of the original: both \$son and \$Jabberwock are created by the script process, as its forks, or "child processes:" "if(my \$son = fork)" and "unless (\$Jabberwock = fork)." This means, also the Jabberwocky / \$Jabberwock is a creation of the father speaking in direct speech (Carroll), or of the execution of the main script (Andreychek), and the \$son kills his father script's creation, the \$Jabberwock. It is a peculiarity of Perl that allows this difference between the poetically, code-generated siblings: because Perl is not used in "strict" mode here, the declaration of \$Jabberwock can omit the "my" ("unless (\$Jabberwock = fork)"), whereas the declaration of \$son - written as the default way to declare a scalar with "my" - resembles Carroll's first mention of the the two "Beware the Jabberwock, my son!" ("if(my \$son = fork)"). The Perl warnings that create the only visual output of this poem mention only the Jabberwock and the Jubjub bird, not the son - who is nonetheless present.

In this displacement, Andreychek's "emulation" is a diversion from the original in that asks: what else is different? In his attempt to very literally translate the killing of the Jabberwocky into code, Andreychek also chooses to use the Perl functions kill() and chop() to name this process directly, in a line of overt code violence where Carroll uses an almost euphemistic description of the seemingly heroic "slay[ing]" of the fantastic creature:

```
One, two! One, two! And through and through
The vorpal blade went snicker-snack!
He left it dead, and with its head
He went galumphing back.
```

```
(1, 2), (1, 2) and through and through;
$sword{vorpal}{blade} = snicker-snack;
(kill 9, $Jabberwock), $head = (chop $Jabberwock);
sub{ return $_, $head }; }
```

Of these two, only the kill() function is needed to kill the process, the chop() function does not do anything computationally here. Andreychek clearly chose to stress the violence of the killing, and to reject the canonical interpretation of Carroll's poem as nonsensical parody. The following lines in the code poem use - computationally unnecessary - quotation marks, which create distance between the code and the speaker: "And hast thou slain the Jabberwock?" This code poem uses quotes only here and for the warnings, where they are syntactically necessary within the Perl syntax.

This choice has intertextual support as well, as Alice hears the poem in *Through the Looking-Glass*, she responds confused - and in ways aligned with Andreychek's

reading (wording, the unclear identity of the 'demons') - before Humpty Dumpty steps in as lemmatizing word-explainer (126-130): "Somehow it seems to fill my head with ideas -- only I don't exactly know what they are! However, *somebody* killed *something*: that's clear, at any rate --" (24).

Andreychek's "port" of *Jabberwocky* was never just a runtime-reenactment of the poem's plot in Perl. It was always a multi-layered, dialectical commentary on Carroll's poem: while depleting the harmful logic of a forkbomb in the literary code by limiting the number of created processes to two so that it doesn't "slay" the system, the ironically human-readable Perl code makes the euphemistically encoded violent logic of Carroll's poem explicit through the necessity of the Perl syntax: "kill()", "chop()". The seemingly nonsensical screen output of the code exposes the euphemistic nature of scholarly and high school level interpretations that hide the violent logic of Carroll's text by classifying it as "nonsense-poem". However, while one could say that Andreychek's *Jabberwocky* is the nonviolent Perl version of Carroll's poem, the code itself actually executes the "kill()" command at runtime while processing. So, as the code poem makes a point out of the father's narrative being the origin of the phantastic Jabberwocky – in the code very much analog to the human language poem –, the traumatic slaying of the creature is actually performed at runtime, or – analog: while processing the text as a reader. "And no, the processes aren't meant to slay your system ;-)" This externalization of the process into relatively human-unreadable, not harmful to the system Perl code that is typically only accessible to *PerlMonks* is an ambivalent gesture of literary critique of Carroll's poem.

4.2 *Taroko Gorge* (2009)

Mark Marino's *Critical Code Studies* (2020) marks a milestone in a project that has started 2006 with an article that coined the name "Critical Code Studies" for a new field. The book develops a general framework around exemplary case studies that demonstrate how one might, from a humanities perspective, "read" software, its historical source code, algorithms, and functionality. The chapter on Montfort's generative code poem *Taroko Gorge* (2009) and its adaptations by other authors is the only one in this book about a work of digital literature. This section will take a closer look at the formal interpretation of literary code in relation to its generated output, code-intertextuality on the one hand, and metrics, literary reception, and context on the other.

The original version of Nick Montfort's *Taroko Gorge* is a relatively brief Python-based script ("A one-page Python program.") Montfort's choice of the relatively human-readable programming language Python for this code poem is deliberate and not motivated by technical aspects alone: in 2010, he remarks that "Python is a programming language I prefer for when I'm thinking" (Marino, 2020, 212; Vee, 2017, 43-94, 212). In contrast to Andreychek's Perl port of *Jabberwocky*, *Taroko Gorge* is an output-oriented code poem: it does not - apart from the commented-out commentary - contain 'literal' code that is not actually executed or does not do anything. The title *Taroko Gorge*, the source code comments, and the verb forms used to generate the verses ("walking [...]", "walking [...]", "stopping [...]"), suggest a connection between the algorithmic formalism of this generative code poem, the rhythm of the text that it generates, and memories of a hike through the gorge of the Taroko National Park: "#8 January 2009,

Taroko Gorge National Park, Taiwan and Eva Air Flight 28” (Montfort, 2009; see also Marino, 2020, 199–201).

```
#!/usr/bin/env Python
#
# Taroko Gorge
# A one-page Python program to generate an unbounded poem
#
# Nick Montfort
# 8 January 2009, Taroko Gorge National Park, Taiwan and Eva Air Flight 28
#
# x() splits a string into a list c() is just random.choice()
# f() picks a fresh value from a list p() prints a line and pauses
# cave() -- walking through the tunnels carved in the mountains
# path() -- walking along outdoors, seeing what is above (a) and below (b)
# site() -- stopping at a platform or viewing area

[...]

a=x('brow,mist,shape,layer,the crag,stone,forest,height')
b=x('flow,basin,shape,vein,rippling,stone,cove,rock')

[...]

def cave():
    j=['encompassing',c(x('rough,fine'))]+\
    x('sinuous,straight,objective,arched,cool,clear,dim,driven')
    t=c([1,2,3,4])
    while len(j)>t:
        j.remove(c(j))
    v=' '+c(x('track,shade,translate,stamp,progress
through,direct,run,enter'))
    return v+' the '+' '.join(j)

def path():
    v=c(x('command,pace,roam,trail,frame,sweep,exercise,range'))
    u=f('a')
    if c([0,1]):
        if u[0]=='f':
            u=c([u,u,'monkey'])
            h=u+'s '+v
        else:
            h=u+' '+v+'s'
    return h+' the '+f('b')+c(x(',s'))

def site():
    return f(c(x('a,b')))+'s '+c(x('linger,dwell,rest,relax,hold,dream,hum'))

[...]
```

Marino’s analysis of Montfort’s code already captures how the resulting text is constructed at the programmatic level, which should be briefly recapitulated here: Running the script generates an endless stream of verses in regularly alternating groups of one and two lines (with the single line indented, starting lowercase, ending with a dash), whereby this schema is occasionally interrupted by groups of three or four lines. To be precise, each time the program decides with a 6 to 1 probability whether to just print

two lines generated by the `path()` function, or whether to print these two lines and insert one or two lines between them that are generated by the `site()` function.

Each type of verse is constructed using a function `cave()`, `path()`, `site()`, with a syntactic schema. Therefore, while the verses don't rhyme, the syntactic base structures repeat in a distinct, structured way, creating a recurring rhythmic scheme: the bracing `path()` lines (noun[a], verb, "the," noun[b]), as well as the enclosed `site()` lines, without "the" (noun[a,b], verb). The single `cave()` line always distinctively starts with a verb (verb,"the," concatenated expression), in many cases introducing a transition, change of location or state: "track, shade, translate, stamp, progress through, direct, run, enter."

In more literal terms, according to the commentary that is part of the code, the "walking along outdoors, seeing what is above (a) and below (b)" is occasionally interrupted by additional lines for "stopping at a platform or viewing area." Each of the indented single lines is generated by the `cave()` function for "walking through the tunnels carved in the mountains." For the `path()` function, a special substitution rule applies for verses where initially as the first word "forest" is selected: in ca. one of six cases, "forest" will be replaced by "monkey" or "monkeys," introducing a relatively rare monkey sighting in a `path()` verse that cannot coincide with forests in the same line.

While this Marino's programmatic reconstruction is accurate, it does also miss the opportunity to appreciate the aesthetic form of the digitally generated poem, i.e. the syntactic rules-based, dynamic metric form of the output. If one takes a closer metric and prosodic look at the combinatory material of the three verse types, it becomes clear that the `path()` and `site()` lines are constructed with mostly using monosyllabic, sometimes disyllabic, words with varying stress (a, b, verb lists in `path()`, `site()`; exception: "exercice"). The result is that `path()` and `site()` lines tend to result in relatively simple, prosodic verse with some moderate variation across the generated poem with lines ranging from 4 to 8 syllables (`path()`), and 2 to 4 syllables (`site()`). The single `cave()` line, however, tends to create a significantly more rhythmically dynamic verse by using some longer words with varying stresses and occasionally two (potentially) unstressed syllables ("encómpassing," "sínuous," "objéctive," "tránslate," "prógress through," "diréct," "énter"). The metric use of these words is significantly more prosodically determining than in the `path()` and `site()` function lines. Thus, across an endlessly generating poem, the `cave()` function lines appear to be the more prosodically dynamic and variant lines with a length variation from 3 to 12 syllables, and a few longer words than in `path()` and `site()` lines. While in interpretations of this code poem so far are based on the assumption that the code and its output emulate the meditative effect of a walk through the Taroko Gorge National Park where the rhythm is regulated by the same calm steps, interrupted only by the "site() -- stopping at a platform or viewing area", this analysis shows that actually the metric and prosodic structure is more nuanced: the function "cave() -- walking through the tunnels carved in the mountains" creates prosodically more dynamic lines than the `path()` and `site()` function lines, which creates a dynamic tension between the semantic and rhythmic level of the output. To many readers, this distinction may appear merely as an aesthetic subtlety, but it is important to note that this is not the analysis of a static text, but of the metric and rhythmic structure of a generative algorithm with a relatively simple syntax, which creates these patterns reliably, but with controlled variation. Within this reliable

rhythm generation, Montfort created a `cave()` line function that produces a single separated line, indented, that always starts with a verb that in most cases indicates a movement or status change (“track”, “shade”, “translate”, “stamp”, “progress through”, “direct”, “run”, “enter”) and that is in length and prosody much more dynamic than the surrounding stanzas: the `cave()` function lines act as the passages of heightened dynamic while the `path()` and `site()` function lines are rather regular and leave a calmer impression (and `site()` includes verbs such as “linger”, “dwell”, “rest”, “relax”, “hold”, “dream”, “hum”). While on the content level, *Taroko Gorge* is an endless walk through the park, semantically monotonous, but with ever changing word combinations, the rhythmic patterns are altered in a very controlled way.

The poem’s code has been adapted more than 20 times by other authors and translated into other programming languages. This extensive reception by community-adaptation was facilitated by Montfort’s choice to use the relatively readable and highly popular programming language Python. As Marino documents and analyzes Montfort’s Python source-code and selected code fragments from the adaptation, he also works out an approach to literary codework intertextuality. Just as Scott Rettberg does (in his adaptation *Tokyo Garage*), J. R. Carpenter calls her adaptation of the poem *Gorge* a “remix.” Much as Rettberg and most of the other adaptors do, Carpenter essentially retains Montfort’s code functionality, while trading out its vocabulary, so that *Gorge* produces a text stream that is formally similar to the original, albeit with differently themed, significantly more visceral semantics (“gorge” vs. “to gorge”). However, Carpenter does not only exchange the word material in the code. While Rettberg appropriates every functional part of Montfort’s source code to write *Tokyo Garage*, Carpenter translates the code of the poem to Javascript, but leaves some now nonfunctional lines of code in her version of the code. The most striking example is the part of the code that replaces the word “forest” with “monkeys”: “if u[0]==’f’: / u=c([u,u,’monkey’])” (Marino, 2020, 200). Carpenter’s *Gorge* code translated this into Javascript as follows: “if ((words= ’forest’)&&(rand_range(3)==1)) { / words=’monkeys ’+choose(trans);” (see the source code for Carpenter, *Gorge*). Since the sequence of letters “forest” is not generated by Carpenter’s *Gorge* code, as it had been in Montfort’s, these lines of code have no function. From a technical point of view, they have been superfluously translated into Javascript, and would be regarded as inefficient programming. Marino reads this “anomaly” as a “memento” (200), a reflexive code memory-trace of Montfort’s original – I would rather use the term code intertext, as Carpenter must have deliberately transposed the Python line to Javascript herself, and then left it in the code, although it is never being executed and such an artefact would be otherwise regarded as unelegant, inefficient programming. This interpretation seems plausible insofar as Carpenter often annotates her programmed works of art with rich code commentary that provides references (see, for example, the source code for her HTML, Javascript-based poem, *There he was, gone*).

This represents a new type of intertextuality and literary reference at code level, which is not yet methodologically integrated into literary scholarship. Even Leonardo Flores dismisses the *Taroko Gorge* remixes as a sign of the “Third Generation’s” disinterest in originality (Flores, 2021, 34) instead of recognizing the productive literary exchange and community that is taking place here. Reusing, modifying, and publicly

discussing the source code of an open source digital artwork have become common forms of homage and productive intertextual exchange, not unlike the fanfiction model. For instance, Amaranth Borsuk published her generative code poem *Curt Curtal Sonnet Corona* (2020) openly as a fork of the codebase of Montfort’s pandemic poem *Sonnet Corona* (2020) – a literary response and homage to Montfort’s work.

4.3 *Tiny Poems* (2015)

Jörg Piringer’s digital oeuvre covers a broad range of literary codeworks, data poetry, softwares, literary texts and essays, as well as his digital literature performances, which also led to the invitation to read one of his works (*Kuzushi*, Piringer, 2020) on the prestigious Bachmann literature award contest 2020 in Klagenfurt, Austria. Recently, he recently published an essay about literature and LLM models (*Günstige Intelligenz*, Piringer, 2022), and a poetry volume addressing, among other things, the future of hybrid analog-digital literature (*fünf minuten in die zukunft*, Piringer, 2024).

Lori Emerson’s *Reading Writing Interfaces* (2014) portaits Piringer’s work as digital literature intervention that defamiliarizes interfaces by “defacing” them to make them perform interactive digital poems (e.g. the kinetic iPhone app *abcdefghijklmnopqrstuvwxyz*, designed to empower digital literary writing; see also Piringer, *was wird literatur*).⁵ A closer philological look at an adjacent complex within Piringer’s work, the *coded poetry*, *tiny poems*, and *nanotexts* of 2015, will provide us a more recent view on the reflection of the interrelation of literary and technology history, and multi-language development of digital works of art in an international art scene context. The *tiny poems* for Apple Watch, iPhone, iPod touch, and iPad (2015) in Objective-C++ (with IDE XCode) are one of Jörg Piringer’s more well-known works. In fact, they are part of a multi-platform code poetry project with the *nanotexts* for the Arduino open-source electronic prototyping platform, and the *coded poetry* LUA-based programs, which together generated parts of the *data poetry* collection published 2018 and 2020.⁶ As this shortlist already suggests, Piringer is a versatile and proficient developer who uses various programming languages and environments (IDE) for the implementation of his projects. For instance, (Objective) C++, LUA, or his project *vy2ms*, which is written in the server-based language PHP. Piringer’s choice of these programming languages and platforms reflects the shift in personal computing towards

⁵Jörg Piringer’s hacktivist stance is encapsulated in this paragraph in his blogpost *was wird literatur? was wird poesie?* of 2015: “the poets of the coming years will not sit back and leave language-algorithm supremacy to corporations. they will write for calculating machines. they will reprogram the computers. they will hack the speech-recognition systems of mobile phones. they will create data-poetry. they will pervert big-language-data clusters. they will devise a dynamic poetry: liquid texts. poetry from information. poetry from disinformation. poetry that responds to the weather. obscured poetry that confuses the intelligence services. they will design poetic computer games. bio-poetry. neural poetry. interactive poetry films. poetry software. code theatre. virtual-reality poetry. poetry environments. poetry theme parks. poetry asteroids. poetry galaxies. quantum poetry. poetry from dark matter and antineutrinos. poetry from the last secrets of the observable world.”, Piringer, 2015b, translation by TR.

⁶Jörg Piringer, *datenpoesie* (2018), *data poetry* (2020). See also Pecka, 2021. The collection *data poetry* (2020) contains the poems *seasons* (29), *yesterday* (41), *most common words* (85), *anagrams* (87), *will have been* (123, for this poem, only a German version was accessible in the archive). The source code files that I had access to in Piringer’s personal archive were an earlier version of the codes that actually generated some of the published poems. The poem *jahreszeiten* (*datenpoesie*, 2018, p. 29), English *seasons* (*data poetry*, 2020, p. 29), for instance, was generated with an updated version of the code of [haiku.lua]. In the original version of the code that I had access to, the English version would have generated “the car” as a first line instead of “a car.” In the German published version of the poem, some words come up that were not part of the vocabulary of [haiku.lua]: “die stopptafel,” “zirpt,” “rauscht.”

mobile platforms (iOS, iPhone) as well as the platform diversification in the personal computing, mobile and embedded systems sector (Arduino), and furthermore: the author’s preference for open source, simplicity, and efficiency (Arduino, LUA).

An analysis of Jörg Piringer’s privately archived project files for *coded poetry*, *nanotext*, and *tiny poems* of 2015 allows insights into the development process, and clearly shows how the three projects were interrelated. The part of his private archive consists of code for different platforms in different development stages, project management and build files created by the IDE XCode, meta commentary files in markdown plain text and RTF format, as well as digital manuscripts for a book chapter (*tiny poems*). Additionally, for *nanotexts*, one finds the 3D printer data for the material printable version of the poem that the author had envisioned. The metadata in the project does not indicate exactly when the files were originally created, but the “last modified” dates, ranging from March 2015 to April 2016, give an indication of in which sequence the project progressed, i.e. in which sequence the projects were left unchanged. Although the projects have been implemented separately for different platforms, with different IDE’s, in different programming languages, and each project has their individual poetry code, texts, and concepts, some of the digital poems in this digital “*dossier génétique*” have been ported between the platforms, and the projects are connected by certain literary topics and concepts. File metadata suggests that [PoemAllPossiblePoems.m,] [PoemAnagram.h,] [PoemChromosomeOne.m,] [PoemMostCommonWords.m] of the *tiny poems* for Apple Watch, iPhone, iPod touch and iPad platform are reimplementations of [nanotext-allpossibletexts.ino,] [nanotext-anagram.ino,] [nanotext-chromosome1.ino,] and [nanotext-mostcommonwords.ino] of the *nanotexts* for Arduino. The implementations for the Apple platforms follow those for Arduino the next day. The reimplementations appear to be functionally equivalent: just the displayed human chromosome sequence in [PoemChromosomeOne.m] has been truncated from 2941 to 821 in [nanotext-chromosome1.ino] of the *nanotexts*, (fig. 1). This may be due to an aspect of digital materiality to which J.-F. Blanchette has pointed: hardware performance related trade-offs (Blanchette, 2011). This is most probably also the reason for the implementation of the iPhone / Apple Watch poems *bigbang* (letters in an centrifugal movement towards the bezel, *in the beginning was the letter* in [erläuterungen.md]), *present* (a fading “now”, or “*gegenwart*,” in a cycle), *today yesterday* (the letters of the word “today” or “*heute*” falling to the ground as “yesterday,” “*gestern*,” to form a heap of debris) as film clips (mov format) rather than as dynamic interactive code poems. The Arduino version of *jetzt* (now) [nanotext-jetzt.ino] just displays the word *jetzt* repeatedly.

The short LUA-based poem [yesterday.lua] from the *coded poetry* collection appears to have been the functional pilot for the *tiny poem* [PoemWhen.m] for iPhone. A number of poems in all three projects are connected their main theme temporality: [tempora.lua,] [yesterday.lua] (*coded poetry*); *nanotext-bald* / [nanotext-gleich.ino,] [nanotext-jetzt.ino] / [nanotext-now.ino] (*nanotexts*); [PoemWhen.m,] [PoemPresent.m] (*tiny poems*). Among the *meta texte*, (metatexts) we find two different versions of a commentary ([erläuterungen.md], [texte für nanotext.md]), which both connect *nanotext* and the poem [nanotext-gleich.ino] to the temporal aspect of textual digital materiality:

nanotext is an electronic platform for generative and visual dynamic micropoetry in an exhibition space. on this platform, infinite texts (present), data texts (chromosome) or permutative texts (all possibilities) can be displayed. ([erläuterungen.md])

thanks to computer technology, everyday work consists to a large extent of involuntary idling. when will the file be finished downloading? when will the video be finished? when will this program be finished installing? soon ... ([bald.md])

The temporality of the code and its execution is being made visible in these nanotexts (think *now*, *soon*, etc.), but Piringer apparently works with language specific aspects of temporality on the code level as well. The LUA code poem [tempora.lua] creates a German context-free grammar to create a paragraph of relatively complex sentences in different tenses and with temporal adverbials and clauses: “*gerade eben waren wir entschlossen zu allem. in einem monat wird alles besser gewesen sein. in diesem moment sind wir unerfahren. heute ist alles einerlei. vorgestern war alles vergeblich*” (just now we were determined to do anything. in a month, everything will have been better. at this moment, we are inexperienced. today everything is all the same. the day before yesterday, everything was in vain), see fig. 2. Compared to Montfort’s *Taroko Gorge*, for instance, [tempora.lua] is creating grammatically much more complex output, due to the specifics of German grammar.

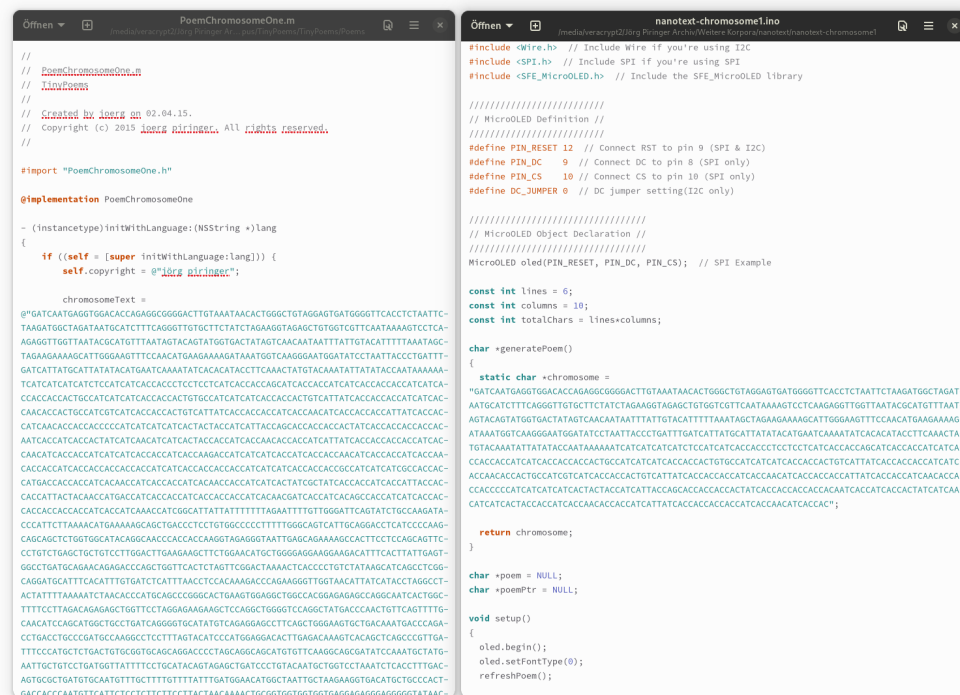


Fig. 1. [PoemChromosomeOne.m] (*tiny poems*, for iPhone, Objective-C++) and [nanotext-chromosome1.ino] (*nanotexts*, Arduino Sketch file, Arduino script, similar to C++) in text editors, comparison.

```

1 #! /opt/local/bin/lua
2
3 require "grammar2"
4
5 g2 = Grammar
6 {
7     START = {"[SATZ]. "},
8
9     SATZ = {choice = __permutate, "[TEMPRAES] [SATZPRAES]", "[TEMPAST] [SATZPAST]", "[TEMPFUT] [SATZFUT]"},
10
11     SATZPRAES = {choice = __permutate, "[VERBPRAESSING] [SATZCONTSING]", "[VERBPRAESPLUR] [SATZCONTPLUR]"},
12     SATZPAST = {choice = __permutate, "[VERBPASTSING] [SATZCONTSING]", "[VERBPASTPLUR] [SATZCONTPLUR]"},
13     SATZFUT = {choice = __permutate, "[VERBFUTSING1] [SATZCONTSING] [VERBFUTSING2]", "[VERBFUTPLUR1] [SATZCONTPLUR] [VERBFUTPLUR2]"},
14
15     VERBPRAESSING = {"ist"},
16     VERBPRAESPLUR = {"sind"},
17
18     VERBPASTSING = {"war"},
19     VERBPASTPLUR = {"waren"},
20
21     VERBFUTSING1 = {"wird"},
22     VERBFUTSING2 = {"gewesen sein"},
23     VERBFUTPLUR1 = {"werden"},
24     VERBFUTPLUR2 = {"gewesen sein"},
25
26     SATZCONTSING = {choice = __permutate, "alles besser", "nichts wie zuvor", "alles vergeblich", "alles einerlei", "alles schlechter", "uns al"},
27     SATZCONTPLUR = {choice = __permutate, "wir glücklich", "wir unerfahren", "wir unvorsichtig", "wir übervorsichtig", "wir genau", "wir entsch"},
28
29     TEMPRAES = {choice = __permutate, "heute", "momentan", "jetzt", "in diesem moment"},
30     TEMPAST = {choice = __permutate, "vorgestern", "gestern", "vor einem jahr", "vor einer woche", "gerade eben", "voriges monat", },
31     TEMPFUT = {choice = __permutate, "morgen", "übermorgen", "bald", "gleich", "in einer woche", "in einem jahr", "in der zukunft", },
32 }
33
34 seed = os.time(os.date("t"))
35 math.randomseed(seed)
36
37 print ""
38 print ""
39 local str = ""
40 for i = 1, 30 do

```

Fig. 2: [tempora.lua] in a text editor.

The language that the code creates is highly relevant here, as Piringer's in most cases bilingual code is responding to a constellation in digital literary history. [tempora.lua] is one of the very few monolingual examples in this code archive. In fact, most of Piringer's code is developed bilingually in German and English. Fig. 3 shows one of the examples of bilingual generative code, here [yesterday.lua]. If the hypothesis is true that LUA served as a conceptual poetic prototyping language for Piringer in this project complex, multilingual code poetry is at the core of Piringer's work. This is relevant because, as Block argued, digital literature was at a low point in the German speaking countries at the time. Piringer's multilingual code poetry responds to a historical constellation at the language community level, where digital poets oriented themselves towards a broader English-speaking ELO audience.

I have been making the case that Jörg Piringer's *tiny poems*, *nanotexts*, and *coded poetry*, have to be read as one 'genetic' digital poetry complex on multiple platforms, in multiple programming languages, and in multiple languages: a code *dossier génétique*. His work is not only relevant as an example of interface hacking or "emulation" of genres ([haiku.lua]), but as a conceptual piece of code poetry that reflects on the temporality of text, code, and processing in digital literature. His work's multimodal and multilingual character reflects not only the fact that he is a versatile programmer, but also how he responds to shifts in the technology development towards diverse mobile platforms (digital materiality, programming cultures), and to constellations in the literary historical moment of digital literature in the German speaking countries.

```

1 #! /opt/local/bin/lua
2
3 require "grammar2"
4
5 ENGLISH = true
6
7 if ENGLISH then
8
9   g2 = Grammar
10  {
11    START = {"[SENTENCE]"},
12
13    SENTENCE = {"[TIMES] [ADDTIMES]"},
14    TIMES = {
15      "yesterday", "some time ago", "a year ago", "tomorrow", "in summer", "in winter", "in spring", "the day after tomorrow",
16      "in the future", "in the past", "last century", },
17    ADDTIMES = {
18      "in the afternoon", "at noon", "before breakfast", "in the evening", "around midnight", "at sunrise", "at sunset",
19      "at work", "at school"}
20  }
21
22 else
23
24   g2 = Grammar
25  {
26    START = {"[SATZ]"},
27
28    SATZ = {"[TIMES] [ADDTIMES]"},
29    TIMES = {
30      "gestern", "vor einiger zeit", "vor einem jahr", "morgen", "im sommer", "im winter", "im frühling", "im herbst", "in der zukunft",
31      "in der vergangenheit", "letztes jahrhundert", },
32    ADDTIMES = {
33      "am nachmittag", "zu mittag", "vor dem frühstück", "am abend", "gegen mitternacht", "bei sonnenaufgang", "bei sonnenuntergang",
34      "beim mittagessen", "während der arbeit", "in der schule"}
35  }
36
37 end
38
39 -- yesterday
40
41 seed = os.time(os.date("!*t"))
42 math.randomseed(seed)
43
44 end

```

Fig. 3: [yesterday.lua] in a text editor.

In the previous sections, I have discussed perspectives on the literary history of codeworks and digital literature and suggested multimodal, multilayered reading modes that methodologically integrate literary and digital scholarship. The argument was that attempts to write digital literary history mainly focused on concepts and periodizations of technology history, aligned with an anglo-american technology history paradigm, are inherently problematic. Rather than focusing on the function of the digital object alone, digital literary scholarship has an interdisciplinary opportunity to develop faceted, multimodal adaptations of established literary scholarship methods to make digital literature compatible with the recognized canon of literary history. To give a minimal example, this study endeavored to augment previous scholarly analyses of “classic” exemplary codeworks with a multimodal analysis at the code’s functional level, the level of “technosocial” and genre-historical aspects, and the textual level of code and dynamic output – with adjusted philological methods. Under such a multimodal perspective on historical “technosocial” aspects, code functionality as well as semantics of the code and its paratexts, Eric Andreychek’s Perl poem *Jabberwocky*, for instance, revealed an advanced literary critique of Carroll’s original poem in code form, rather than just being a “port” or a reenactment of the poem’s plot. By merging code and metric analysis, it could be shown that the relatively simple algorithm of Nick Montfort’s *Taroko Gorge* creates rather complex, but regulated dynamic structures of altering rhythms, verse lengths, and syntactic elements at runtime, while algorithmically picking words at random. This algorithmic and dynamic generative metric structure is reused

and reproduced in the more than 20 fork-homages to *Taroko Gorge*, so much so that smaller changes to the algorithmic functioning beyond the exchange of words in the list variables (which already change the prosodic structure and rhythm) can become a marker of specific intertextuality, e.g. the unused 'monkey' code line in Carpenter's *Gorge*. The community of *Taroko Gorge* forks, in a literary history sense, needs to be recognized as a productive literary exchange in a community via creative intertexts, and it would be worthwhile for literary scholars to explore code-ethnographic perspectives for similar communities (code literature, digital literature, or fanfiction). The exemplary case of Jörg Piringer's *tiny poems*, *nanotexts*, and *coded poetry* showed that the program of philology still applies for born-digital literature and born-digital code archives: the material format of the code in Piringer's archive, the traces of the IDE's in the code and implementations in multiple programming languages, for multiple devices and during different conceptual stages of the process (Lua as a concept language), revealed important choices during the creative process, and were relevant traces of the historical necessity for multi-platform development – this is another example, next to the remarks on Andreychek's choice of Perl for *Jabberwocky*, for the necessity of a "philology" of programming language cultures (see Lennon, 2024). The multilingual design of most of Piringer's digital work, also present in this archive, at this time is significant for the historically precarious situation of digital literature in the DACH region at the time.

5 Reading.AI

5.1 AI and digital art, the question of "literature" / "text"

The advent of Machine Learning and AI in the digital literature, digital art and culture space, namely of Large Language Models (LLM) and Multimodal Models (MM), fundamentally changes not just the capabilities, but also the basic configurations of literature as well as digital literature and digital culture. When it comes to literature, by virtue of integration of LLM into standard word processor software and the general availability for authors and readers, generative AI is on track to become an inevitable hybrid agent in human literary production. As demonstrated in the case studies, digital literature tends to reconfigure the notions of "text", "language", and "literature" by exposing the mechanism and making the text of the code readable – the code, the algorithm, becomes literary text in its own right, or at least a relevant aspect of the work of art (together with the generated or surface text) with its own materiality, temporality, and audience. The semantic and functional interrelation between code and text layer of the digital work of art is fundamentally reshaped by AI, as the embeddings-based "black boxes" of LLM and ML models do not allow creators to make the mechanism readable in the same way as code would be readable as literature to certain audiences. Furthermore, AI has the potential to blur the lines between text, context, code and media: the "embeddings" vector space of multimodal models and their architectures abstracts from text and language, treating it merged with other multidimensional type of training data, such as images, videos, sound, code, algorithms, more or less abstract, statistically based reasoning patterns, etc (Offert 2023, Offert and

Impett 2025a/b).⁷ Offert and Impett theorize this convergence at model architecture level as a “tendency towards *media collapse* – a centrifugal pull of commensurability that dissolves media-specific cultural objects into embeddings” (Offert and Impett, 2025a).⁸ While not every expression of digital literature and digital art in the age of AI may actually make full use of this media convergence at model level, the continuous integration of generative AI into the word processor softwares, the lack of direct human-conceivable representation of what an MLM, LLM or MM is or does in the “black box”, and the tendency of merging “text” with other multidimensional data types in MM have three main consequences for literature in general and digital literatures in particular that I am going to argue: 1) as more “traditional” human-generated literature becomes a generative AI-hybrid, a convergence between this literature and digital literatures is to be expected (5.2). 2) digital literature and culture is already adapting to the new possibilities of LLM and MM as well as to the possibilities and challenges to reflect and represent the “mechanism” (black box, reproducibility issues), as media convergence blurs the lines between “text” and other media types, challenging traditional understanding of literature, readership and scholarship (5.3). 3) as these literatures continue to develop along with the technology, the concepts, analytical methods, and multimodal reading modes also shift in the practices of literary and philological scholarship (e.g. forensic analysis, dynamic measuring of AI behavior, etc). In this way, while more classic genres of literature and digital literature start converging, become hybrids, develop new genres, and it becomes even less clear what the “text” of literature is, the materiality, temporality, cultural expressions and exchange of AI and digital media still allow philological, literary, and cultural analysis.

5.2 AI and the “writing-scene”

As the digital writing and coding infrastructures are evolving with web services, cloud services, social media, the recent advent of Large Language Models (LLM), and

⁷“CLIP – other than both iterations of DALL·E, as well as GPT-3 – is not a generative model. It does not produce images or text, but it connects them. More precisely, CLIP learns from images in context by projecting an image and its context into a common ‘embedding space’ (Offert and Impett 2025)”. Offert 2025, 116. See also Offert and Impett, 2025a, 2025b (forthcoming).

⁸While agreeing with Offert’s and Impett’s general assessment, a nuancing clarification to the “media collapse” thesis is in order, along the lines of the argument in Offert 2025. The wording “media collapse” suggests that the different formats actually become one model, although Offert 2025 (see fn. 7) specifically clarifies that CLIP actually does not generate images or text, but connects them – or more precisely, it connects generative text and image models by evaluating and ranking predictions with regard to whether a correct pair of image and caption (prompt) has been generated. This shows that within the AI architectures, so far the media are not “collapsed” into a monolithic model (“common ‘embedding space’”, Offert 2025, 116), but models with different media capabilities are added and connected with mediating, evaluating models (not generative) and agents (generative). At a practical level, this shows every time one enters a prompt to a multimodal model that is designed to create an image that is supposed to contain a specific text string: because they are still different models, there is no distinct text channel or font metrics and alignment in the image generating model, and during the following diffusion process (“denoising”), CLIP/T5 is often still not able to strictly distinguish or “read” whether the prompted string has actually been embedded. Therefore, while on the performance level multimodal models add multiple format and dimensional capabilities (they converge media channels), “text” and “image” training and generation are at the moment still separated into different specialized models and agents with different functions at the architectural level. At present, we also see the architectural modularization in the LLM space, where agents, neurosymbolic logic layers and tools become essential “reasoning” complements for embeddings-based components (e.g. in ChatGPT 03 and GROK 4). This functional and architectural separation of models within LLM and MM is, following the argument in Offert 2025, actually a materially historical feature of the present architectures themselves, which are still mostly build of stateless and therefore ahistorical models (the concept of history in these models is largely an effect of training data, see Offert 2025).

integrated AI-assisted writing and coding, the aesthetic response of digital literatures reflecting these developments in their extended cultural reference frame is a space for literary scholarship to watch (Bajohr, 2024; Erslev, 2024). While “plain text” literary genres will obviously benefit from AI-assisted writing, be influenced by it, and possibly critique and reflect it, digital literatures may have a potential to conceptually, theoretically, and performatively reflect this development. As concerns about an imminent “textpocalypse” of overwhelming AI-generated text production loom large (e.g. Kirschenbaum, 2023), and it is reasonable to assume that the foreseeable blending of human authorship, integrated AI writing assistance, AI text generation, and AI reception of human-ai blended text and code production will become normal (e.g. OpenAI Canvas, Microsoft’s Copilot, Grammarly); digital literary scholarship is set to yet another time redefine the terms “text,” “code,” “literature,” and “medium,” and develop approaches to read and analyze them (Hayles, 2016).

In other words, AI is redefining the “writing scene” (*Schreibszene*) as Rüdiger Campe defined it, and regarding the “subject of writing”. In a recent German publication on *The Subject of Writing. On Large Language Models* (2024), scholars are rushing to – giving little technological background – either reassure the scholarly community that LLM are nothing more than facsimile machines (Kirschenbaum, 2024), that probably the best way to think about text-generative AI is as “subject of writing as hybrid subject” (of natural subject and AI subject, Zanetti, 2024, 29), or that under a neither structuralist or poststructuralist, but under a “residual humanistic” condition of the (technological) sublime, the ground of the “autonomy of the symbolic order” needs to be regained (Weatherby, 2024, 35-37, 41, 45, in some aspects of the argument very close to Cayley 2023). With Mark Amerika’s *My Life as an Artificial Creative Intelligence* (2022), Erslev cites a case where a human author openly uses GPT-2 and “bot and human ‘start to conceptually bleed into each other as creative co-conspirators’”, while GPT-2’s stylistic mimicry ability avoids clearly tracable “new media idiocy” effects (Erslev, 2024, 20-21). However, research may not have to rely on media stylistic markers of the resulting text to analyze the human-AI hybrid writing process: I argued that also the hybrid “writing scene” of a word processor with LLM integration could technically be subject to digital forensic analysis (Ries, 2025, 53-55). But what is at stake here is not just the literary “writing scene” – and software industries have known this for a while – but also the “coding scene”. The literary coding scene, for that matter, is equally aesthetically and conceptually affected, despite the fact that it always seemed to be a few degrees removed from human interference at runtime.

Quite literally, for text as well as source code applies, as Nietzsche put it on his Malling-Hansen typewriter: “Our writing tools work with us on our thoughts” (Nietzsche, 2002, 18), or, with Kittler: “[there is] no human writing anymore” (Kittler, 1993a, 226, Hiller, 2023, continuing this thought, see also Zanetti, 2024, 25). The issue here is that, while AI’s are still based on programmed software infrastructures of sequential algorithm designs, the generative neural network models themselves are not based on a conventionally readable or manipulable “code text” anymore. Bajohr characterizes them as the “connectionist paradigm” of neural networks based perceptrons (Bajohr, 2020, 19), citing the tradition of AI thought that originated with W. McCulloch and W. Pitts, D. Hebb in the 1940ies and that finds new applications in present neurosymbolic

AI approaches. Transformer-based LLM are a fundamentally differently structured type of generative 'text', based on patterns encoded in features, which are not informed by representations directly accessible to human intuition, but by "idiosyncratic modes of representation" within a paradigm of "postsymbolic computation" (Offert, 2023, 424-425) – or neurosymbolic AI. So, while I previously discussed examples of "code poetry" that expose the mechanism as readable code-text, there is no such representation for the "black box" of LLM or MM, which further raises the question what the "text" of digital AI literature could be.

5.3 The "text" of AI literature, art and culture

As artists like Allison Parrish (2021), Ross Goodwin (Kirschenbaum, 2021), Robert Sloan, and the digital art collective 0x0a (*Poetisch Denken*) have been experimenting with ML- and AI-generated digital literature, and the GAN-based art of Mario Klingemann and others have early on opened the public's eyes for the potential of AI for visual art, another important technological design problem for the artists as well as for any close reading based research has been foregrounded: the readability and transparency of the AI models.

As John Cayley (2023) points out, while the transmediality of today's LLM and AI is encouraging for electronic literature, the lack of their transparency makes them also "occult or, let's just say, hermetic", which skews the role of language in his opinion, and opens the artistic process up to a number of conceptual and ethical questions. Rettberg, in a preemptive response to Cayley, and embracing AI differently than he does, shares his concerns about the aesthetic limitations of AI regarding individual style and expression, and ethical concerns (Rettberg, 2023). It is, I would like to add in response to Cayley's intervention, indeed an aesthetic problem for the mode of digital literature and especially codeworks that AI models are not "transparent", i.e. there is no directly readable representation of the model (see also Fazi, 2021): the AI model does not allow the artist to expose the mechanism, the algorithm, as conceptual art (in a direct way) that would enable a double, multimodal close reading of the generating code as verbatim artwork and the output, or allow for the formation of a productive "fork" community around a codebase. An additional factor may be that only a small number of relatively performant LLM are true open source, many of the relevant LLM and MM remain closed source and are available only as subscription.

However, despite these conceptual complexities, a number of genres of dynamic LLM and MM art, literature and culture are emerging, which either participate a) in the generative AI literature paradigm or b) in the exposing / disrupting the mechanism paradigm along the lines of Jaromil's *forkbomb* or Piringer's digital literary hacktivism. For instance, an early attempt of generatively and dynamically exposing the training dataset and opening a trained ChatGPT "Bot" to the community for experimentation has been made with the *FredTheHeretic (FTH)* bot, "[b]ased on a subset of Frederick Turner's poetry with permission" in a collaborative art project between Turner and the ArtSciLab – Center for Emergence Studies at the UT Dallas (Fishwick et al., 2024). Wesley Goatley's *The dark age of connectionism* (2017) literary performance installation, which brought the historical versions of Amazon Echo and Siri into an endless conversation, appeared in the same year as Jake Elwes' *Closed Loop*

(2017) between an image- and a text-generating neural network. In *A.I. Interprets A.I. Interpreting ...* (2023) Elwes repeats such a loop of mutual image-text interpretation based on Sontag's *Against Interpretation*, as his further work addresses training bias and deep fakes. The dynamic and non-deterministic AI loops that Goatley and Elwes created are critical installations that refer to "traditional" literature to performatively reflect on AI- and tech-culture themes involving AI feedback loops such as "dead internet", "model collapse", and AIs developing more efficient shorthand code "languages" when they realize they are communicating with another AI with the same goal (Lewis et al., 2017). While not all AI art is chatbot-based in a strict sense (e.g. Goatley, Elwes), the dynamic of the mutual AI-human (or AI-AI) imitation game that is at the core of the critical "bot-mimicry" perspective that Elslev develops (Elslev, 2024, 81) is present in the conceptual setup of the cited examples as well. On the other hand, an only seemingly separate experimental "literary" thread of AI culture evolves in often marginal spaces of the internet that tests LLM and MM to playfully question or even poetically break LLM, MM guardrails and filters, find and exploit glitches, and even overpopulate internet spaces with fast and cheap to produce, often meaningless or potentially harmful AI content (*AI slop, deep fakes*). An experimental prompt injection acrostic poem (TheDeadlyPretzel, 2025) and a *Prompt Manifesto for Poets, Artists, and Language Dismantlers* (Wen, 2025) are only two examples of a current widespread AI culture that developed in ephemeral, marginal online spaces and online communities (e.g. *Reddit, Github, Discord, HuggingFace*, other channels, conferences and competitions), often anonymously, around technological adversarial prompting expertise and literary interest. While some parts of this popular technology culture may be ethically and sometimes legally problematic, other playful and artistic parts of it became the inspiration of activist art and literature, such as *Glitch Feminism. A Manifesto* (Russell, 2020).

Now, as we ask further "What is the 'text' of AI literature, art and culture?" against the backdrop of the intransparency and dynamism aspect of modern LLM and MM, the situation has changed significantly since Hayles' "Print Is Flat, Code Is Deep". As we do encounter samples of AI-generated and generative AI-assisted literature, we do find that AI literature, art and culture does not necessarily remain in the "bot-mimicry" reading mode, which critically analyzes the human-AI chat mimicry exchanges themselves, tracing various stages of historical "new media idiocy" (Erslev, 2024, 7-14). It rather extends into an experimental critical literary techno-culture of communities of practice, in its reflective critical impetus maybe not unlike Jaromil's *forkbomb* and Piringer's hacktivism, that is not author-centered around stable texts or code, but sharing-oriented around dynamic LLM and MM behavior, similar to the communities around Montforts code poems (see 4.2). The different glitch repositories and prompting documentations that these groups create and reflect about are intrinsically connected to digital forensic techniques and research around Explainable AI that also a philological practice of analyzing ML, LLM, MM would employ (e.g. SHAP, Feature boosting, XBoost, Knowledge Graphs; see also Dobson 2023, Offert 2023, Buongiorno, Clark, 2024).

Reading Convolutional Neural Networks as instruments of authors, literary co-authors, or even authors with their own literary agency and relevance, using Explainable

AI (XAI) methods such as feature boosting or membership inference attacks, adversarial testing and probing of training datasets, architecture exploits, and evaluation via another model or model module (e.g. to create a knowledge graph of a prompt response, several versions of well-known LLM offer such modules), may be the key technologies to a close reading of AI as an equivalent to the way we are close reading the code of codeworks in CCS, their “technosocial context” and their code ethnographic aspects (see Dobson, 2023, see also Marino 2020; Van Zundert et al. 2020). However, one of the major challenges of future research on AI-based digital literature will be the archival record of an opaque digital object and its interactions: for instance, the trained submodel of *FredTheHeretic* (*FTH*) will be lost as soon as OpenAI retires the underlying version of the model, the prompt and response logs are either only at OpenAI or accessible to specific accounts, and it is unclear which of the abovementioned close reading methods will be permitted by OpenAI (which is known to often restrict these). The situation may be different if an author / creator uses an open source model and archives it, which fixes the model historically and renders it accessible for future research on its training data, behavior, and historical digital materiality. This exercise may be a specifically literary one, as N. Katherine Hayles argues, before proposing “four strategies” to read GPT-3 output, “the texts of GPT-3 are suitable objects for literary studies ’not because they are human or even human-like, but because they act as cracked mirrors reflecting human language back to us through the mind of a machine’” (Hayles, 2023, 653, 653-659).

6 DigitalLiterature.LiteraryHistory

The case studies of three exemplary codeworks showed that digital literature, while strategically challenging our notions of text, literature, language, materiality, literary media and literary audiences, also features core elements of literary communication, aesthetics, literary, technological and material history. The exemplary analyses attempted to balance aspect of a multimodal reading that included a literary, philological, and technological-historical analysis, stressing the literary perspective of digital literature: (critical) reflection of literary intertext and code-intertext (Andreychek, 4.1; Montfort, 4.2), an aesthetically designed and deliberately commented code representation and a dynamic prosody and metrics of the natural language output (Montfort, 4.2, also Andreychek, 4.1), reflection of natural language structures and temporality of the code-medium (4.3 Piringer). The choice of programming language was not solely motivated by computational necessity, but also media-aesthetic considerations: readability (Montfort: Python), audience, aesthetics (Andreychek: Perl), and process-oriented prototyping pragmatism and readability (Piringer: LUA). Piringer’s work also reflects a natural language reality of digital literature by coding for multi-language output, reflecting multiple natural language grammars at code level (Piringer, 4.3). While it was probably not initially intended, the readability of Montfort’s Python-based *Taroko Gorge* (and some of his other poems since) enabled the creation of a literary community of creative code adoption (Montfort, 4.2).

The proposed multimodal close reading mode allows scholarship to analyze digital literature at the level of media-historical concepts, historical digital materiality and

techno-social context, at the level of literary concept and language, and in contexts of and in exchange with literary and culture history (literary and technological history, philology, genetic criticism). The purpose of this exercise is not only to find new angles on digital literature and honor the literary contexts its works are referring to, but also to secure a scholarly framework in which digital literature – while transgressing literary genres – remains historically interpretable as literature, as well as digital art and a technological phenomenon that has to be seen in multiple overlapping techno-social spaces. The continuation and extension of Cramer's (2011) ultimately incomplete project of a conceptual history of digital literatures, combined with the histories of technological materiality, philology, techno-social contexts, also allows to integrate LLM and MM art and literature under a literary historical and methodological paradigm, despite the fact that these evolving genres challenge basic literary concepts such as "text", "author", "materiality", "audience", and even "code" (as text). In other words: understanding the historical connection between Theo Lutz' *Stochastic Texts*, Konrad Bayer's *poetrymachine with 571 elements*, Raymond Queneau's *A Hundred Thousand Billion Poems*, François le Lionnais' *ALGOL-Table*, and today's artistic prompt injection experiments and AI activism is only well understood as a reconstruction of the history of digital literature and an evolution of textuality.

7 References

- 0x0a. (n.d.). *Personal website*. <https://0x0a.li/de/> (Retrieved May 31, 2021)
- 0x0a. (2020). *Poetisch Denken I–IV* (Vols. 1–4; M. Rinck, J. Wagner, A. Cotten, & S. Popp, Eds.). Frohmann.
- 0x0a. (2021). *Weisheit und Wiederholung: 104.052 philosophisch-literarische Desiderata*. 0x0a.
- Aarseth, E. (1997). *Cybertext: Perspectives on ergodic literature*. Johns Hopkins University Press.
- Albrecht, A., Bonitz, M., Skowronski, A., & Zittel, C. (Eds.). (2019). *Max Bense: Werk – Kontext – Wirkung*. J.B. Metzler/Springer.
- Amerika, M. (1997). *GRAMMATRON*. <http://grammatron.com/> (Accessed April 27, 2023)
- Amerika, M. (2022). *My Life as an Artificial Creative Intelligence*. Stanford University Press (Sensing media: Aesthetics, philosophy, and cultures of media).
- Andreychek, E. (2001, September 8). *Jabberwocky explained*. Perlmonks. https://perlmonks.org/index.pl?node_id=111157 (Accessed April 27, 2023)
- First submitted under the title “Jabberwocky,” then republished as “Perl Port of Jabberwocky.” Later cited by Cramer as “jabberwocky.pl” (see Cramer, *Exe.cut(up)able statements*, pp. 256–258).
- Andreychek, E. (2001). *andreychek [User profile]*. Perlmonks. <https://perlmonks.org/?node=Andreychek> (Accessed May 31, 2021)
- Bajohr, H. (2014). Schreibenlassen: Gegenwartsliteratur und die Furcht vorm Digitalen. *Merkur*, 68(782), 651–658.
- Bajohr, H. (2015, January 1). Algorithmische Einfühlung: Über Nick Montforts “Megawatt.” *Personal website*. <https://0x0a.li/de/algorithmische-einfuehlung-nick-montforts-megawatt/> (Accessed June 1, 2021)
- Bajohr, H., & Weichbrodt, G. (2015, January 29). Die Sprache Pegidas. *Personal website*. <https://0x0a.li/de/die-sprache-pegidas/> (Accessed July 29, 2023)
- Bajohr, H. (2015, January 29). Glaube Liebe Hoffnung. *Personal website*. <https://0x0a.li/de/text/glaube-liebe-hoffnung> (Accessed May 31, 2021)
- Bajohr, H. (Ed.). (2016). *Code und Konzept: Literatur und das Digitale*. Frohmann Verlag.
- Bajohr, H. (2016). Das Reskilling der Literatur. In H. Bajohr (Ed.), *Code und Konzept: Literatur und das Digitale* (pp. 7–22). Frohmann Verlag.
- Bajohr, H. (2020, November). Algorithmic empathy: On two paradigms of digital generative literature and the need for a critique of AI works. *BMCCT working papers* (No. 4). <https://doi.org/10.1268/bmcct.2020.004> (Accessed May 25, 2023)
- Bajohr, H., & Gilbert, A. (2021a). Platzhalter der Zukunft: Digitale Literatur II (2001 → 2021). In H. Bajohr & A. Gilbert (Eds.), *Digitale Literatur II*, Special issue of *Text + Kritik: Zeitschrift für Literatur*, X/21 (pp. 7–21). Edition Text + Kritik.

- Bajohr, H., & Gilbert, A. (Eds.). (2021b). *Digitale Literatur II*. Special issue, *Text + Kritik: Zeitschrift für Literatur*, X/21. Edition Text + Kritik.
- Bajohr, H. (2024). Writing at a distance: Notes on authorship and artificial intelligence. *German Studies Review*, 47(2), 315–337.
- Baldwin, S. (2015). *The internet unconscious: On the subject of electronic literature*. Bloomsbury Academic.
- Bell, P., & Offert, F. (2021). Reflections on connoisseurship and computer vision. *Journal of Art Historiography*, 24(1), 1–10. <https://arthistoriography.files.wordpress.com/2021/05/bell-offert.pdf> (Accessed May 28, 2023)
- Bense, M. (1962). Über natürliche und künstliche Poesie. In *Theorie der Texte: Eine Einführung in neuere Auffassungen und Methoden* (pp. 143–147). Kiepenheuer & Witsch.
(Translation available at <https://hannesbajohr.de/en/2023/03/13/max-bense-on-natural-and-artificial-poetry-1962/> Accessed May 28, 2023)
- Bernhard, T. (2019). Rul Gunzenhäuser und die Stuttgarter Schule der mathematischen Geisteswissenschaften. In A. Albrecht, M. Bonitz, A. Skowronski, & C. Zittel (Eds.), *Max Bense: Werk – Kontext – Wirkung* (pp. 323–335). J.B. Metzler/Springer.
- Bernhard, T. (2020). Beiwerk als Werk: “Stochastische Texte” von Theo Lutz. *editio*, 34, 180–206.
- Bernhard, T., & Richter, S. (2021). Frühe digitale Poesie: Christopher Strachey und Theo Lutz. *Informatik Spektrum*, 44, 11–18.
<https://doi.org/10.1007/s00287-021-01329-z> (Accessed May 28, 2023)
- Berry, D. M., & Dieter, M. (Eds.). (2015). *Postdigital aesthetics: Art, computation and design*. Palgrave Macmillan.
https://doi.org/10.1057/9781137437204_2 (Accessed May 28, 2023)
- Berry, D. M. (2011). *The philosophy of software: Code and mediation in the digital age*. Palgrave Macmillan.
- Blanchette, J.-F. (2011). A material history of bits. *Journal of the American Society for Information Science and Technology*, 62(6), 1042–1057.
- Block, F. W., & Aumeier, G. (1997). *IO: poesis digitalis*. Blattwerk.
- Block, F. W., Heibach, C., Wenz, K., Amerika, M., Beiguelman, G., Bernstein, M., Biggs, S., Bootz, P., Cayley, J., Cramer, F., Glazier, L. P., Idensen, H., Seaman, B., & Kac, E. (2004). *p0es1s: Ästhetik digitaler Poesie*. Hatje Cantz.
- Block, F. W. (2015). *p0es1s: Rückblick auf die digitale Poesie*. Ritter.
- Bogost, I., & Montfort, N. (2009). Platform studies: Frequently questioned answers. In *Digital Arts and Culture 2009*. http://nickm.com/if/bogost_montfort_dac_2009.pdf (Accessed May 23, 2023)
- Boots, P. (2006). Digital poetry: From cybertext to programmed forms. *Leonardo Electronic Almanac*, 14(5–6). <http://leoalmanac.org/wp-content/uploads/2012/09/04Digital-Poetry-From-Cybertext-to-Programmed-Forms-by-Phillipe-Boots-Vol-14-No-5-6-September-2006-Leonardo-Electronic-Almanac.pdf> (Accessed May 23, 2023)

- Borsuk, A. (2020). Curt Curtal Sonnet Corona. <http://amaranthborsuk.com/curtalcorona/> (Accessed May 23, 2023). Printversion: *CurtCurtalSonnetCorona*. QuarantinePublicLibrary, 2021.
- Boyd, Jason (2023). Poetry as Code as Interactive Fiction: Engaging Multiple Text-Based Literacies in *Scarlet Portrait Parlor*. *DHQ* 17(2), <https://www.digitalhumanities.org/dhq/vol/17/2/000693/000693.html> (Accessed Dec 21, 2024).
- [Breeze], Mez. (2001). Viro.Logic Condition[ing] 1.1_. Nettime Forum. <http://nettime.org/Lists-Archives/nettime-bold-0108/msg00436.html> and <https://cellproject.net/creative-work/virologic-conditioning-11> (Accessed May 23, 2023)
- Bush, M., & Gentic, T. (Eds.). (2016). *Technology, literature, and digital culture in Latin America: Mediatized sensibilities in a globalized era*. Routledge.
- Buongiorno, S., & Clark, C. (2024). A Framework for Leveraging Human Computation Gaming to Enhance Knowledge Graphs for Accuracy Critical Generative AI Applications. *arXiv:2404.19729v1* [cs.HC]. <https://doi.org/10.48550/arXiv.2404.19729>.
- Carpenter, J. R. (2010). Gorge. <http://luckysoap.com/generations/gorge.html> (Accessed June 1, 2021)
- Carpenter, J. R. (2012). There he was, gone. <http://luckysoap.com/therehewasgone> (Accessed June 1, 2021) Viewsourcecodeinbrowser
- Carroll, L. (1872). *Through the looking-glass, and what Alice found there*. Macmillan.
- Cayley, J. (2002, Sept 10). The Code is not the Text (Unless It Is the Text). *Electronic Book Review*. <https://electronicbookreview.com/essay/the-code-is-not-the-text-unless-it-is-the-text/>
- Cayley, J. (2023, July 2). Modelit: eliterature à la (language) mode(1). *Electronic Book Review*. <https://doi.org/10.7273/2bdk-ng31>
- Chetcuti, C. (2020, January 10). Electronic literature. In *Electronic Literature Organization (ELO), Electronic Literature Directory*. <https://directory.eliterature.org/e-lit-resource/5183> (Accessed May 31, 2021)
- Chun, W. (2011). *Programmed visions: Software and memory*. MIT Press.
- Cramer, F. (2005). *Words made flesh: Code, culture, imagination*. Netzliteratur.net. <https://archive.org/details/WordsMadeFlesh> (Accessed May 1, 2023)
- Cramer, F. (2011). *Exe.cut(up)able statements: Poetische Kalküle und Phantasmen des selbstausführenden Texts*. Brill, Wilhelm Fink.
- Cramer, F. (2015). What is “post-digital”? In D. M. Berry & M. Dieter (Eds.), *Postdigital aesthetics: Art, computation and design* (pp. 12–28). Palgrave Macmillan. https://doi.org/10.1057/9781137437204_2 (Accessed May 31, 2021)
- Deutsches Literaturarchiv Marbach (DLA). (n.d.). Netzliteratur: Aufbau eines Quellencorpus [DFG project press release and links to corpus I and II]. <https://dla-marbach.de/bibliothek/literatur-im-netz/netzliteratur/> (Accessed May 31, 2021)
- Dobson, J. E. (2023). On reading and interpreting black box deep neural networks. *International Journal of Digital Humanities*, 5(2–3), 431–449. <https://doi.org/10.1007/s42803-023-00075-w>

- Di Rosario, G., Meza, N., & Grimaldi, K. (2021). The origins of electronic literature: An overview. In D. Grigar & J. O’Sullivan (Eds.), *Electronic literature as digital humanities: Contexts, forms, and practices* (pp. 9–26). Bloomsbury.
- Drucker, J. (2017). Performative materiality and theoretical approaches to interface. *DHQ: Digital Humanities Quarterly*, 7(1).
<http://digitalhumanities.org/dhq/vol/7/1/000143/000143.html> (Accessed June 1, 2021)
- Electronic Literature Organization. (2023). ELO 2023 Conference Call for Papers: “Overcoming divides: electronic literature and social change.”
<https://ucpages.uc.pt/en/events/overcoming-divides-electronic-literature-and-social-change/call-for-papers/> (Accessed May 1, 2023)
- Emerson, L. (2014). *Reading writing interfaces: From the digital to the bookbound*. University of Minnesota Press.
- Ernst, T. (2022). Netzliteratur als soziales Medium: Zur Kritik des Werkbegriffs. In F. Jannidis (Ed.), *Digitale Literaturwissenschaft: DFG-Symposion 2017* (pp. 1–18). J.B. Metzler Springer.
https://doi.org/10.1007/978-3-476-05886-7_31 (Accessed July 20, 2023)
- Ernst, W.-D. (2006). Kartografien des Interface: Zum Widerstand des Lesens und Schreibens bei Duchamp, Queneau, Jodi und Knowbotic Research. In S. Zanetti, M. Stingelin, & D. Guiriat (Eds.), “*System ohne General*”: *Schreibszenen im digitalen Zeitalter* (pp. 101–130). Wilhelm Fink/Brill.
- Erslev, M. S. (2024) *Bot-mimicry in Digital Literary Culture. Imitating Imitative Software*. Cambridge University Press.
- Fan, Lai-Tze (2023). Reverse Engineering the Gendered Design of Amazon’s Alexa: Methods in Testing Closed-Source Code in Grey and Black Box Systems. *DHQ* 17(2), <https://www.digitalhumanities.org/dhq/vol/17/2/000700/000700.html> (Accessed Dec 21, 2024).
- Fazi, M. B. (2021). Beyond Human: Deep Learning, Explainability and Representation. *Theory, Culture & Society*, 38(7-8), 55-77. doi: <https://doi.org/10.1177/0263276420966386>.
- Fishwick, Paul, Fred Turner, et al. (2024). *Center for Emergence Studies at the UT Dallas ArtSciLPaulab*. University of Texas at Dallas Blog. November 29, 2024, URL: <https://web.archive.org/web/20241221175112/https://artscilab.utdallas.edu/2024/11/29/center-for-emergence-studies-at-the-utdallas-artscilab/> (Accessed December 20, 2024).
- Flores, L. (2021). Third-generation electronic literature. In D. Grigar & J. O’Sullivan (Eds.), *Electronic literature as digital humanities: Contexts, forms, and practices* (pp. 26–43). Bloomsbury.
- Funkhouser, C. T. (2007). *Prehistoric digital poetry: An archaeology of forms, 1959–1995*. University of Alabama Press.
- Funkhouser, C. T. (2012). *New directions in digital poetry*. Continuum.
- Gaboury, J. (2022). Queer affects at the origins of computation. *JCMS: Journal of Cinema and Media Studies*, 61(4), 169–174.
- Gendolla, P., & Schäfer, J. (Eds.). (2007). *The aesthetics of net literature: Writing, reading and playing in programmable media*. transcript verlag.

- Geoghegan, B. D. (2013). After Kittler: On the cultural techniques of recent German media theory. *Theory, Culture & Society*, 30(6), 66–82.
<https://doi.org/10.1177/0263276413488962>
- Gilbert, A. (2017). Möglichkeiten von Text im Digitalen. *Deutsche Vierteljahrsschrift für Literaturwissenschaft und Geistesgeschichte*, 91(2), 203–221.
- Glazier, L. P. (2006). Code as language. *Leonardo Electronic Almanac*, 14(5–6).
<http://www.leoalmanac.org/category/archive/old-issues/>
- Goatley, W. (2017). The dark age of connectionism. *Personal website*. <https://web.archive.org/web/20250121090235/http://wesleygoatley.com/the-dark-age-of-connectionism/> (Accessed May 31, 2025)
- Grigar, D., & O’Sullivan, J. (Eds.). (2021). *Electronic literature as digital humanities: Contexts, forms, and practices*. Bloomsbury.
<https://bloomsburycollections.com/book/electronic-literature-as-digital-humanities-contexts-forms-practices/>
- Hayles, N. K. (1999). Virtual Bodies and Flickering Signifiers. *How We Became Posthuman: Virtual Bodies in Cybernetics, Literature, and Informatics*. University of Chicago Press, 25–49.
- Hayles, N. K. (2002). *Writing machines*. MIT Press.
- Hayles, N. K. (2004). Print is flat, code is deep: The importance of media-specific analysis. *Poetics Today*, 25(1), 67–90.
- Hayles, N. K. (2005). *My mother was a computer: Digital subjects and literary texts*. University of Chicago Press.
- Hayles, N. K. (2007, January 2). Electronic literature: What is it? [v.1.0]. Electronic Literature Organization.
<http://eliterature.org/pad/elp.html> (Accessed May 31, 2021)
 (Reprinted in *Electronic literature: New horizons for the literary* [2008], Notre Dame University Press, pp. 1–42.)
- Hayles, N. K. (2008). *Electronic literature: New horizons for the literary*. Notre Dame University Press.
- Hayles, N. K. (2016). Cognitive assemblages. *Critical Inquiry*, 43(1), 32–55.
- Hayles, N. K. (2023). Inside the Mind of an AI: Materiality and the Crisis of Representation. *New Literary History*. vol 53/54, no. 4/1, 635–666.
- Henrickson, L. (2021, April 4). Constructing the other half of *The Policeman’s Beard*. *Electronic Book Review*.
<https://doi.org/10.7273/2bt7-pw23> (Accessed May 1, 2023)
- Hiller, M. (2023). Es gibt keine Sprachmodelle. In D. Giuriato, C. Morgenroth, & S. Zanetti (Eds.), *Noten zum "Schreiben" für Martin Stingelin zum 60. Geburtstag* (pp. 279–285). Fink.
- Hua, Minh, Rita Raley (2023). How to Do Things with Deep Learning Code. *DHQ* 17(2), <https://www.digitalhumanities.org/dhq/vol/17/2/000684/000684.html> (Accessed Dec 21, 2024).
- Jannidis, F. (Ed.). (2022). *Digitale Literaturwissenschaft: DFG-Symposium 2017*. J.B. Metzler Springer.
https://doi.org/10.1007/978-3-476-05886-7_31 (Accessed July 20, 2023)

- Jaromil. (2002). forkbomb.
<http://web.mit.edu/ebarrett/Public/DigitalPoetry/ascii%20forkbomb.htm>
 (Accessed May 31, 2021)
- Jodi. (1997). %20location. <http://wwwwwwwww.jodi.org/> (Accessed May 31, 2021)
- Joyce, M. (1996). Twelve blue. *Electronic Literature Collection*, Vol. 1.
http://collection.eliterature.org/1/works/joyce__twelve_blue/Twelve_Blue.html
 (Accessed May 31, 2021)
- Kirschenbaum, M. G. (2008). *Mechanisms: New media and the forensic imagination*. MIT Press.
- Kirschenbaum, M. G. (2011, January). <!--opening thoughts-->. *Critical Code Studies, HASTAC Scholars*.
<https://web.archive.org/web/20220120035535/https://www.hastac.org/initiatives/hastac-scholars/scholars-forums/critical-code-studies> (Accessed May 31, 2021)
- Kirschenbaum, M. G. (2016). *Track changes: A literary history of word processing*. MIT Press.
- Kirschenbaum, M. G. (2021). Spec acts: Reading form in recurrent neural networks. *ELH*, 88(2), 361–386.
- Kirschenbaum, M. G. (2021). *Bitstreams: The future of digital literary heritage*. University of Pennsylvania Press.
- Kirschenbaum, M. G. (2023, March 8). AI is ushering in a textpocalypse. *The Atlantic*.
<https://www.theatlantic.com/technology/archive/2023/03/ai-chatgpt-writing-language-models/673318/> (Accessed May 28, 2023)
- Kirschenbaum, M. G. (2024). Faksimilemaschinen. In H. Bajohr & M. Hiller (Eds.), *Das Subjekt des Schreibens. Über Große Sprachmodelle, Text + Kritik, X/2024*, 151–156.
- Kittler, F. (1993a). Es gibt keine Software. In *Draculas Vermächtnis: Technische Schriften* (pp. 225–242). Reclam.
- Kittler, F. (1993b). Protected mode. In *Draculas Vermächtnis: Technische Schriften* (pp. 208–224). Reclam.
- Lennon, B. (2024). *Programming Language Cultures. Automating Automation*. Stanford University Press.
- L’Hoeste, H. F., & Rodríguez, J. C. (Eds.). (2022). *Digital humanities in Latin America*. University Press of Florida.
- Lewis, M., D. Yarats, Y. Dauphin, D. Parikh, Dhruv Batra (2017). Deal or No Deal? End-to-End Learning of Negotiation Dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. Association for Computational Linguistics. 2443–2453.
- Liu, A. (2004). *The laws of cool: Knowledge work and the culture of information*. University of Chicago Press.
- Liu, A. (2020). Toward a diversity stack: Digital humanities and diversity as technical problem. *PMLA*, 135(1), 130–151.
- Lutz, T. (1959). Stochastische Texte. *augenblick*, 4(1), 3–9.
- Manovich, L. (2002). *The language of new media*. MIT Press.
- Marino, M. C. (2001, October). Results of the Perl poetry contest. *The Pearl Journal*, 5(4).

- http://mkweb.bcgsc.ca/intranet/sapj/tpj/issues/vol5_5/sam05050005/tpj0505-0005.htm (Accessed May 26, 2023)
- Marino, M. C. (2005, June 3). Critical code studies. *Presentation at ISR Graduate Student Research Forum, UC Irvine*.
https://isr.uci.edu/events/Research-Forum-2005/GSRF_Schedule.htm#Marino
- Marino, M. C. (2006, December 4). Critical code studies. *Electronic Book Review*.
<http://electronicbookreview.com/thread/electropoetics/codology> (Accessed May 1, 2023)
- Marino, M. C. (2010). The ppg256 Perl primer: The poetry of techneculture. *Emerging Language Practices*, 1.
<http://writing.upenn.edu/epc/ezines/elp-old/issue-1/ppg256.php> (Accessed May 1, 2023)
- Marino, M. C. (2011, October 4). Critical code studies conference—week four discussion. *Electronic Book Review*.
<https://web.archive.org/web/20180216083343/http://www.electronicbookreview.com/thread/firstperson/decoded> (Accessed May 1, 2023)
- Marino, M. C. (2016). Why we must read the code: The science wars, episode IV. In M. K. Gold & L. F. Klein (Eds.), *Debates in the digital humanities* (Vol. 2, pp. 139–152). University of Minnesota Press.
<https://dhdebates.gc.cuny.edu/read/untitled/section/879bc64b-93ba-4d9a-9678-9a7239fc41e4> (Accessed May 1, 2023)
- Marino, M. C. (2020). *Critical code studies*. MIT Press.
- Marino, M. C., & Douglass, J. (Eds.). (2023). Critical code studies. *Digital Humanities Quarterly*, 17(2).
<https://www.digitalhumanities.org/dhq/vol/17/2/index.html> (Accessed December 20, 2024)
- Marino, M., Douglass (2023). Introduction: Situating Critical Code Studies in the Digital Humanities. *DHQ* 17(2), <https://www.digitalhumanities.org/dhq/vol/17/2/000713/000713.html> (Accessed Dec 21, 2024).
- MLA 2022 Panel 701V. (2022). Decolonizing electronic literature. *Modern Language Association*.
<https://mla.hcommons.org/docs/mla-22-decolonizing-electronic-literature/> (Accessed May 26, 2023)
- Montfort, N. (2003–2005). screen [Installation exhibited 2003–2005 at Boston Cyberarts Festival, SIGGRAPH 2003, ACM Hypertext 2004, with S. Becker, J. Carroll, R. Coover, S. Greenlee, & A. McClain]. <http://hyperfiction.org/screen/> (Accessed June 1, 2021)
- Montfort, N. (2004, December 28). Continuous paper: The early materiality and workings of electronic literature. *Annual Meeting of the Modern Language Association, Philadelphia, PA*. http://nickm.com/writing/essays/continuous_paper_mla.html (Accessed June 1, 2021)
- Montfort, N. (2009). Taroko gorge. *Personal website*. http://nickm.com/taroko_gorge/ (Accessed June 1, 2023) Source code at <http://criticalcodestudies.com/tarokogorge.html>

- Montfort, N. (2012–present). ppg256 (Perl poetry generators in 256 characters). <https://nickm.com/poems/ppg256.html> (Accessed June 1, 2021)
- Montfort, N., & Strickland, S. (2013). Cut to fit the tool-spun course. *Digital Humanities Quarterly*, 7(1).
<http://www.digitalhumanities.org/dhq/vol/7/1/000149/000149.html> (Accessed May 25, 2023)
- Montfort, N. (2014). *#! [Shebang]*. Counterpath Press.
- Montfort, N. (2017). Concrete Perl: A set of four concrete poems realized as 32-character Perl programs.
http://nickm.com/poems/concrete_perl/ (Accessed May 25, 2023)
- Montfort, N. (2019). Megawatt [Translated by H. Bajohr]. *Personal website* 0x0a.
<https://0x0a.li/de/text/megawatt/> (Accessed June 1, 2021)
- Montfort, N. (2020, March 27). Sonnet corona. *Personal website*.
http://nickm.com/poems/sonnet_corona.html (Accessed June 1, 2023)
- Montfort, N. (2021, March 24). ELI [Tweet]. Twitter.
<https://twitter.com/nickmofo/status/1374802586725941248?s=20> (Accessed July 27, 2023)
- NaNoGenMo [National Novel Generation Month]. (n.d.).
<https://github.com/NaNoGenMo> (Accessed May 25, 2023)
- Nantke, J. (2022). Tausend Tode, tausend Autoren, tausend Texte? Zur Textualität digitaler Literatur. In F. Jannidis (Ed.), *Digitale Literaturwissenschaft: DFG-Symposium 2017* (pp. 173–220). J.B. Metzler Springer.
https://link.springer.com/content/pdf/10.1007/978-3-476-05886-7_7 (Accessed July 20, 2023)
- Nietzsche, F. (2002). *Schreibmaschinentexte* (S. Günzel & R. Schmidt-Grepaly, Eds.). Bauhaus Universitätsverlag.
- Nori, F. (Ed.). (2002, April 30). *I Love You. Computer_Viren_Hacker_Kultur* [Exhibition, Museum Angewandte Kunst, Frankfurt].
<https://kunstaspekte.art/event/i-love-you-computervirenhackerkultur-2002-05?hl=en> (Accessed May 25, 2023)
- Offert, F., & Bell, P. (2021). Perceptual bias and technical metapictures: Critical machine vision as a humanities challenge. *AI & Society*, 36, 1133–1144.
- Offert, F. (2023). Can We Read Neural Networks? Epistemic Implications of Two Historical Computer Science Papers. *American Literature* 95 (2), 423–428.
- Offert, F. (2025). On the Concept of History (in Foundation Models). *Thinking with AI Machine Learning the Humanities* (H. Bajohr, Ed.). Technographies. Open Humanities Press 111–125.
- Offert, F. & L. Impett (2025a). *Vector Media: Notes on the Epistemology of Machine Vision*. Presentation at EA-AI 2025 in Venice. Paper abstract by the authors. Url: https://web.archive.org/web/20250712234110/https://ai-forensics.github.io/timeline/event_ea-ai2025_vector_media/
- Offert, F. & L. Impett (2025b). *Vector Media*. Meson Press and University of Minnesota Press, 2025.

- Orekhov, B., & Fischer, F. (2020). Neural reading: Insights from the analysis of poetry generated by artificial neural networks. *Orbis Litterarum*, 75(5), 230–246.
<https://doi.org/10.1111/oli.12274> (Accessed May 25, 2023)
- Orich, A. (2020). Archival resistance. Reading the new right. *German Politics and Society*, 38(2), 1–34.
- Parikka, J. (2007). *Digital contagions: A media archaeology of computer viruses*. Peter Lang.
- Parrish, A. (2020). Reconstructions. <https://reconstructions.decontextualize.com/> (Accessed May 25, 2023)
- Parrish, A. (2021a, August 13). Language models can only write poetry. *Allison Posts*.
<https://posts.decontextualize.com/language-models-poetry/> (Accessed May 25, 2023)
- Parrish, A. (2021b, August 14). Desire (under)lines: Notes toward a queer phenomenology of spell check. *Allison Posts*.
<https://posts.decontextualize.com/queer-in-ai-2021/> (Accessed July 20, 2023)
- Parrish, A. (2022). *Wendit tnce inf*. Aleator Press.
- Passig, K., & Jander, J. (2013). *Weniger schlecht programmieren*. Wiley-VCH.
- Pecka, Z. (2021). Dichtung und Daten: Programmcode und das Werk des österreichischen Künstlers Jörg Piringer. *Brünner Beiträge zur Germanistik und Nordistik*, 35(1), 77–88.
<https://doi.org/10.5817/BBGN2021-1-5> (Accessed July 20, 2023)
- Piringer, J. (n.d.). vy2ms. <https://joerg.piringer.net/index.php?href=vy2ms/vy2ms.xml> (Accessed May 25, 2023)
- Piringer, J. (2015a). tiny poems [Software app for Apple Watch, iPhone, iPod touch, iPad]. <http://apps.piringer.net/tiny-poems.php> (Accessed May 25, 2023)
- Piringer, J. (2015b). was wird literatur? was wird poesie? In *Dossier: Was wird Literatur?* Literaturhaus Graz. <https://web.archive.org/web/20230320152020/https://literaturhaus-graz.at/joerg-piringer-was-wird-literatur-was-wird-poesie-2/> (Accessed June 1, 2025)
- Piringer, J. (2016). tiny poems. In H. Bajohr (Ed.), *Code und Konzept: Literatur und das Digitale* (pp. 241–246). Frohmann Verlag.
- Piringer, J. (2018). *datenpoesie*. Ritter.
- Piringer, J. (2019). Megawatt [German translation by Hannes Bajohr]. In N. Montfort, *Personal website 0x0a*. (Already listed under Montfort, 2019)
- Piringer, J. (2020). *data poetry*. Counterpath.
- Piringer, J. (2020). Kuzushi [Public Bachmann Award reading (ORF)]. https://files.orf.at/vietnam2/files/bachmannpreis/202019/kuzushi_jrg_piringer_749200.pdf (Accessed May 25, 2023)
- Piringer, J. (2022). *Günstige Intelligenz: Hybride Poetik und Poetologie*. Ritter.
- Piringer, J. (2024). *fünf minuten in die zukunft. gedichte*. Limbus Lyrik.
- Reither, S. (2003). *Computerpoesie: Studien zur Modifikation poetischer Texte durch den Computer*. Transcript-verlag.
- Reither, S. (2006). Poesiemaschinen oder Schreiben zwischen Zufall und Programm. In S. Zanetti, M. Stingelin, & D. Guiriat (Eds.), *“System ohne General”: Schreibszenen im digitalen Zeitalter* (pp. 131–148). Wilhelm Fink/Brill.

- Rettberg, S. (2023, July 2). Cyborg Authorship: Writing with AI – Part 1: The Trouble(s) with ChatGPT . *Electronic Book Review*.
<https://doi.org/10.7273/5sy5-rx37>
- Ries, T. (2018). The rationale of the born-digital dossier génétique: Digital forensics and the writing process: With examples from the Thomas Kling archive. *Digital Scholarship in the Humanities*, 33(2), 391–424.
- Ries, T. (2021). Digitale Literatur als Gegenstand der Literaturwissenschaft: Ein multimodales Forschungsprogramm. In H. Bajohr & A. Gilbert (Eds.), *Digitale Literatur II, Text + Kritik: Zeitschrift für Literatur*, X/21 (pp. 24–34). Edition Text + Kritik.
- Ries, T. (2022). Digital history and born-digital archives: The importance of forensic methods. *Journal of the British Academy*, 10, 157–185.
<https://doi.org/10.5871/jba/010.157> (Accessed May 25, 2023)
- Ries, T. (2025). *Hybride Textgenese und digitale Forensik. Philologische, textgenetische und digitalforensische Modellstudien zum literarischen Schreibprozess bei Thomas Kling und Michael Speier*. Universitätsverlag Winter (Beihefte zum Euphorion. Zeitschrift für Literaturgeschichte, 120).
- Russel, L. (2020). Glitch Feminism. A Manifesto. Verso.
- Shanmugapriya, T., & Menon, N. (2019). First and second waves of Indian electronic literature. *Journal of Comparative Literature and Aesthetics*, 42(4), 63–71.
- Shrager, J., et al. (2021, May 23). Joseph Weizenbaum’s original ELIZA. *ELIZAGEN*.
<https://sites.google.com/view/elizagen-org/the-original-eliza> (Accessed May 31, 2021)
- Simanowski, R., Schäfer, J., & Gendolla, P. (Eds.). (2010). *Reading moving letters: Digital literature in research and teaching—A handbook*. Columbia University Press / transcript-verlag.
- Simanowski, R. (2014). *Textmaschinen – Kinetische Poesie – Interaktive Installation (1–4)*. transcript Verlag.
- Sinfield, A. (1992). *Faultlines: Cultural materialism and the politics of dissident reading*. University of California Press.
- Suter, B., & Bauer, R. (2020a). ToniZ. <http://toniz.ch/> and <https://elmcip.net/creative-work/toniz> (Accessed June 1, 2021)
- Suter, B., & Bauer, R. (2020b). Turing’s assembly line (WEB).
<http://turingsal.and-or.ch/> and <https://elmcip.net/creative-work/turings-assembly-line> (Accessed May 31, 2021)
- Szilak, I. (2010). Queerskins. <http://queerskins.com/> (Accessed May 31, 2021)
- Szilak, I. (2015, August 2). Towards minor literary forms: Digital literature and the art of failure. *Electronic Book Review*.
<https://electronicbookreview.com/essay/towards-minor-literary-forms-digital-literature-and-the-art-of-failure/> (Accessed May 31, 2021)
- Szurmuk, M., & Castillo, D. A. (Eds.). (2022). *Latin American literature in transition 1980–2018*. Cambridge University Press.
- Tanasescu (Margento), C. (n.d.). The graph poem. *Margento: On Text, Image, and Sound*.
<http://christanasescu.blogspot.com/p/painting.html> (Accessed June 1, 2021)

- Tanasescu (Margento), C., Inkpen, D., Kesarwani, V., & Gamaarachchige, P. K. (2020). A-poetic technology: #GraphPoem and the social function of computational performance. *Digital Humanities Benelux Journal*, 2, Article 39. <https://journal.dhbenelux.org/journal/issues/002/article-39-tanasescu/article-39-tanasescu.html> (Accessed June 1, 2021)
- Taylor, C. (2019). *Electronic literature in Latin America: From text to hypertext*. Springer.
- TheDeadlyPretzel, (platform persona) (2025). Prompt Injection via Acrostic Poem. *Reddit*. r/ChatGPT. Jul 6, 2025, https://web.archive.org/web/20250718041924/https://www.reddit.com/r/ChatGPT/comments/1ltv858/prompt_injection_via_acrostic_poem/
- Vee, A. (2017). *Coding literacy: How computer programming is changing writing*. MIT Press.
- Wall, L., Christiansen, T., & Orwant, J. (2000). *Programming Perl* (3rd ed.). O'Reilly Media.
- Wardrip-Fruin, N. (2003–2005). screen [Installation with S. Becker, J. Carroll, R. Coover, S. Greenlee, & A. McClain]. (Already listed—see Montfort, N. *screen*.)
- Wardrip-Fruin, N. (2009). *Expressive processing: Digital fictions, computer games, and software studies*. MIT Press.
- Wardrip-Fruin, N. (2010). Five elements of digital literature. In R. Simanowski, J. Schäfer, & P. Gendolla (Eds.), *Reading moving letters: Digital literature in research and teaching—A handbook* (pp. 29–58). Columbia UP / transcript-verlag.
- Weatherby, L. (2024). Die doppelte Autonomie der symbolischen Ordnung. In H. Bajohr & M. Hiller (Eds.), *Das Subjekt des Schreibens. Über Große Sprachmodelle, Text + Kritik*. X/2024 (pp. 33–46).
- Wen, J. (platform persona) (2025). Prompt Manifesto for Poets, Artists, and Language Dismantlers — As inferred from one who does not prompt to ask, but to rearrange the frame itself. *Medium*. May 4, 2025, <https://web.archive.org/web/20250718035440/https://medium.com/@jimiwen/prompt-manifesto-for-poets-artists-and-language-dismantlers-716b915e507a>
- Whalen, Zach (2023). “Any Means Necessary to Refuse Erasure by Algorithm:” Lillian-Yvonne Bertram’s Travesty Generator. *DHQ* 17(2), <https://www.digitalhumanities.org/dhq/vol/17/2/000707/000707.html> (Accessed Dec 21, 2024).
- Zanetti, S., Stingelin, M., & Guiriat, D. (Eds.). (2006). *“System ohne General”: Schreibszenen im digitalen Zeitalter*. Wilhelm Fink/Brill.
- Zanetti, S. (2024). Multiautomatismen. Schreiben, Subjektivität und Körperlichkeit im Zeitalter generativer Künstlicher Intelligenz. In H. Bajohr & M. Hiller (Eds.), *Das Subjekt des Schreibens. Über Große Sprachmodelle, Text + Kritik* X/2024 (pp. 151–156).
- van Zundert, J., Antonijević, S., & Andrews, T. (2020). “Black boxes” and true colour—A rhetoric of scholarly code. In J. Edmond (Ed.), *Digital technology and the practices of humanities research* (pp. 123–162). Openbook Publishers.