Defactoring 'Pace of Change': Exploring Code Review Methods for Textual Scholarship and Literary Studies

Joris J. Van Zundert
joris.van.zundert@huygens.knaw.nl
Huygens Institute for the History of the Netherlands
Royal Netherlands Academy of Arts and Sciences

Matt Burton
mcburton@pitt.edu
University of Pittsburgh, United States of America

Introduction

We start from the assertion that coding and code—as the source code of computer programs that is readable to humans and which drives the performative nature of software (Ford 2015, Hiller 2015)—can be inherent parts of scholarship or scholarship by and of themselves. That is: we assert that code can be scholarly, that coding can be scholarship, and that there is little difference between the authorship of code or text (Van Zundert 2016). The dichotomy that has been often sought between on the one hand a 'pure' intellectual realm associated with scholarly writing and academic print publication, and on the other hand the 'material labour' associated with for instance instrument making or programing, is artificial.

We argue the validity of this assertion along Burgess and Hamming (2011) and Clement (2016). These scholars refer to earlier work in which Bruno Latour (1993) casts the defining characteristic of modernity as a process of 'purification' which aims to contrast the human culture of modernity to nature. Burgess and Hamming observe a congruent process in academia: "Within the academy we see these processes of purification and mediation at work, producing and maintaining the distinction between intellectual labor and material labor, both of which are essential to multimedia production" (Burgess & Hamming 2011:¶11). This process serves to distinguish between scholarly and non-scholarly activities: "The distinction between intellectual and material labor is pervasive throughout scholarly criticism and evaluation of media forms. […] In addition, any discussion of scholarly activities in multimedia format are usually elided in favor of literary texts, which can be safely analyzed using traditional tools of critical analysis." However, this distinction is based upon a technological fallacy already pointed out—as Burgess and Hamming note—by Richard Grusin in 1984. Grusin argued that Hypertext has not changed the nature of text essentially, as writing has always already been hypertextual through the use of indices, notes, annotations, and intertextual references. To assume that the technology of Hypertext has unveiled or revolutionary activated the associative nature of text, amounts to the fallacy of ascribing the associative agency of cognition to the technology, which however is of course a 'mere' expression of that agency.

Analogous to Burgess and Hamming, we argue that relegating the evaluation of scholarship to the reviewing of print publications is an equal fallacious ascribing of agency to the technology of written text. Such a narrow understanding of scholarship presupposes that something is scholarship because it is in writing, that writing makes it scholarship.

It is possible to evade all such possible technological fallacies by understanding scholarship as argument. We argue therefore that scholarship in essence is argument, and that technologies enable to shape and express that argument. This is not to say that technologies are mere inert and neutral epistemological tools, obviously different technologies shape and affect argument in different ways. Different technologies can therefore enrich scholarly argument. Scholarship is thus not bound to the use of text as an epistemological technology, but essentially is in the shaping of an argument. Text and writing may still be the most celebrated semiotic technologies to express an argument, but computer code understood as 'just another' literacy (cf. Knuth 1984, Kittler 1993, Vee 2013) can equally be the carrier of scholarly argument. However, the acceptance of code as another form of scholarly argument presents problems to the current scholarly process of evaluation because of a lack of well developed methods for reviewing and critiquing scholarly code. Digital humanities as a site of production of non conventional research outputs—digital editions, web based publications, new analytical method, and computational tools for instance—has spurred the debate on evaluative practices in the humanities considerably, exactly

because practitioners of digital scholarship acknowledge that much of the relevant scholarship is not expressed in the form of traditional scholarly output. Yet the focus of review generally remains on "the fiction of 'final outputs' in digital scholarship" (Nowviskie 2011), on old form peer review (Antonijevic 2016), and on approximating equivalencies of digital content and traditional print publication (Presner 2012). Discussions around the evaluation of digital scholarship have thus "tended to focus primarily on establishing digital work as equivalent to print publications to make it count instead of considering how digital scholarship might transform knowledge practices" (Purdy & Walker 2010:178, Anderson & McPherson, 2011). As a reaction digital scholars have stressed how peer review of digital scholarship should foremost consider how digital scholarship is different from conventional scholarship. They argue that review should be focused on the process of developing, building, and knowledge creation (Nowviskie 2011), on the contrast and overlap between the representationality of conventional scholarship and the strong performative aspects of digital scholarship (Burgess & Hamming 2011), and on the medium specificity of digital scholarship (Rockwell 2011).

The debate on peer review in digital scholarship however, has been geared much to high-level evaluation, concentrating for instance on the issue how digital scholarship could be reviewed in the context of tenure track evaluations. Very little has been proposed as to concrete techniques and methods for more practical level applied peer review of program code. Existing practical guidance pertains to digital objects such as digital editions (Sahle & Vogler 2014) or to code as cultural artefact (Marino 2006), but no substantial work has been put forward on how to peer review scholarly code. We are left with the rather general statement that "traditional humanities standards need to be part of the mix, [but] the domain is too different for them to be applied without considerable adaptation" (Smithies 2012), and the often echoed contention that digital artefacts should be evaluated as such and not as to how they might have been documented in conventional articles. The latter argument probably most succinctly put by Geoffrey Rockwell (2011): "While such narratives are useful to evaluators […] they should never be a substitute for review of the work in the form it was produced in."

Yet, the problem is growing more urgent. Increasingly, code is created and used as a mechanism of analysis in textual scholarship and literary studies—cf. for instance Enderle 2016, Jockers 2013, Piper 2015, Rybicki et al. 2014, and Underwood 2014—which leads to the need to evaluate the technical, methodological and epistemological qualities of such code, as for instance the 'Syuzhet case' showed (Swafford 2016). The algorithms, code, and software that underpins the analyses in these examples of scholarship are not standardized 'off the shelf' software productions. These code bases are nothing like a software package or product such as AntConc that can be viewed as a generic and packaged distributable tool; a tool that might be subject to a scholarly type of tool criticism explaining and opening it for reuse by other scholars. Instead these codebases are bespoke code: they are one-off highly specific and complex analytical engines, tailored to solving one highly specific research question based on one specific set of data. Reuse, scalability, and ease-of-use are, justifiably (Baldrigde 2015), not specific aims of these code objects at all. Such might be the case with generic software, but these programs have been algorithmic instruments tailor made to serve the research case at hand. As such—and following what was argued above—we must regard these code bases as an inherent part of the scholarly argument they contribute to. And as such they deserve and require specific and rigorous peer review, like any argument in humanities research. How such peer review should be conducted is, however, a large unknown.

As a contribution to the challenges of code peer review we present an experimental technique we call defactoring. Drawing on Braithwaite (2013), we have re-configured the program code that underpins a recent article by Ted Underwood and Jordan Sellers (Underwood & Sellers 2016) into a computational narrative—echoing Knuth's literate programming (1984)—to be critically analyzed and annotated. This method is intimately intertwined with the Jupyter Notebook platform, which allows for the composition of scholarly and scientific inscriptions that are simultaneously human and machine readable. The Notebook is both a document format and a platform for mixing code and prose into executable objects. We have extracted Underwood and Seller's code and defactored it into a Jupyter Notebook, available at https://github.com/interedition/paceofchange. This means we have recombined code from disparate files, linearized the execution path, demodularized function calls, and annotated code blocks with our own expository comments. As an annotated Notebook we can now engage Underwood and Seller's code directly

as a scholarly inscription and more deeply interrogate the role of data, algorithms, and code in the production of knowledge.

In the case of scholarship that uses computation, large parts of the intellectual importance are embodied in the code rather than living exclusively in the print publication. As our case study also shows, usually the method description in the print publication presents the intellectual contribution of the code development in a very reduced, and rather imprecise high-level fashion. The code itself is a more precise inscription of the analysis the researchers conducted. The methodological approach we present is a way to engage the code, and thus allows a peer reviewer to understand and interpret the intellectual narrative of the code. This results in a fuller grasp and understanding of the methods applied, and thus to a more comprehensive review of the intellectual effort associated with the publication.

After demonstrating the work in the notebook, we will conclude our paper with a critical reflection of the reviewing work that was undertaken with it. We identify the applicability, feasibility, benefits, and drawbacks of this specific approach. We also outline some possible future directions of research that could further contribute to exploring review methods for code scholarship.

Bibliography

Anderson, S., McPherson, T., (2011). Engaging Digital Scholarship: Thoughts on Evaluating Multimedia Scholarship. Profession 136–151.

Antonijevic, S., (201)5. Amongst Digital Humanists: An ethnographic study of digital knowledge production, Palgrave Macmillan.

Baldridge, J., (2015). It's okay for academic software to suck. Java Code Geeks. Available at: https://www.javacodegeeks.com/2015/05/its-okay-for-academic-software-to-suck.html [Accessed April 25, 2016].

Braithwaite, R., (2013). Defactoring. Reginald Braithwaite: via raganwald.com. Available at: http://raganwald.com/2013/10/08/defactoring.html [Accessed March 15, 2016].

Burgess, H.J. & Hamming, J., (2011). New Media in Academy: Labor and the Production of Knowledge in Scholarly Multimedia. DHQ: Digital Humanities Quarterly, 5(3). Available at: http://digitalhumanities.org/dhq/vol/5/3/000102/000102.html [Accessed September 2, 2016].

Clement, T.E., (2016). Where Is Methodology in Digital Humanities? In Debates in the Digital Humanities 2016. University of Minnesota Press, pp. 153–175. Available at: http://dhdebates.gc.cuny.edu/debates/text/65.

Enderle, J.S., (2016). A Plot of Brownian Noise. Jonathan Scott Enderle. Available at: https://github.com/senderle/svd-noise/blob/master/Noise.ipynb [Accessed September 24, 2016].

Ford, P., (2015). What is Code? Businessweek. Available at: http://www.bloomberg.com/graphics/2015-paul-ford-what-is-code/.

Grusin, R., (1994). What is an Electronic Author? Theory and the Technological Fallacy. Configurations, 2(3), pp.469–483.

Hiller, M., (2015). Signs o' the Times: The Software of Philology and a Philology of Software. Digital Culture and Society, 1(1), pp.152–163.

Jockers, M.L., (2013). Macroanalysis: Digital Methods and Literary History, Urabana, Chicago, Springfield: UI Press.

Kittler, F., (1993). Es gibt keine Software. In Draculas Vermächtmis. Leipzig: Reclam Verlag, pp. 225–242.

Knuth, D.E., (1984). Literate Programming. The Computer Journal, 27(1), pp.97–111.

Latour, B., (1993). We Have Never Been Modern, Cambridge, Massachusetts: Harvard University Press.

Marino, M.C., (2006). Critical Code Studies. Electronic Book Review. Available at: http://www.electronicbookreview.com/thread/electropoetics/codology [Accessed January 16, 2015].

Nowviskie, B., (2011). Where Credit Is Due: Preconditions for the Evaluation of Collaborative Digital Scholarship. Profession, pp.169–181.

Piper, A., (2015). Novel Devotions: Conversional Reading, Computational Modeling, and the Modern Novel. New Literary History, 46(1), pp.63–98.

Presner, T., (2012). How to Evaluate Digital Scholarship. Journal of Digital Humanities, 1(4). Available at: http://journalofdigitalhumanities.org/1-4/how-to-evaluate-digital-scholarship-by-todd-presner/.

Purdy, J.P. & Walker, J.R.,(2010). Valuing Digital Scholarship: Exploring the Changing Realities of Intellectual Work. Profession, pp.177–195.

Rockwell, G.,(2011). On the Evaluation of Digital Media as Scholarship. Profession, pp.152–168.

Rybicki, J., Hoover, D. & Kestemont, M., (2014). Collaborative authorship: Conrad, Ford and Rolling Delta. Literary and Linguistic Computing, 29(3), pp.422–431.

Sahle, P. & Vogeler, G., (2014). Criteria for Reviewing Scholarly Digital Editions (version 1.1). Institut für Dokumentologie und Editorik. Available at: http://www.i-d-e.de/publikationen/weitereschriften/criteria-version-1-1/ [Accessed October 13, 2016].

Smithies, J., (2012). Evaluating Scholarly Digital Outputs: The Six Layers Approach. Journal of Digital Humanities, 1(4). Available at: http://journalofdigitalhumanities.org/1-4/evaluating-scholarly-digital-outputs-by-james-smithies/ [Accessed September 2, 2016].

Swafford, J. (2016) 'Messy Data and Faulty Tools', in Gold, M. K. and Klein, L. F. (eds) Debates in the Digital Humanities. Minneapolis: University of Minnesota Press, p. 600. Available at: http://dhdebates.gc.cuny.edu/debates/text/100.

Underwood, T.,(2014). Understanding Genre in a Collection of a Million Volumes, Interim Report. Figshare. Available at: https://figshare.com/articles/Understanding_Genre_in_a_Collection_of_a_Million_Volumes_Interim_Report/1281251 [Accessed March 15, 2016].

Underwood, T. & Sellers, J.,(2016). The Longue Durée of Literary Prestige. Modern Language Quarterly, 77(3), pp.321–344.

Vee, A., (2013). *Understanding Computer Programming as a Literacy*. LiCS, 1(2), pp.42–64.

Zundert, J.J. van, (2016.) *Author, Editor, Engineer — Code & the Rewriting of Authorship in Scholarly Editing.* Interdisciplinary Science Reviews, 40(4), pp.349–375.