Git-Lit: an Application of Distributed Version Control Technology toward the Creation of 50,000 Digital Scholarly Editions

Jonathan Reeve

Columbia University

Distributed version control technologies, the most popular protocols of which are `git`, `subversion`, and `mercurial`, have long been popular among computer programmers for their abilities to track changes in a codebase and foster collaboration among coders. When combined with code management platforms such as GitHub, Bitbucket, or GitLab, they become even more powerful, enabling sophisticated bug tracking, project planning, and open-source code publication. Although these technologies have not yet been in widespread use in the humanities, their potential for use with corpus creation and textual editing is far-reaching. This paper describes Git-Lit, an open-source, community-centered initiative to parse, version control, and publish to GitHub roughly 50,000 scanned public-domain books from the British Library, thereby facilitating decentralized, open-access, and democratic scholarly editing.

The Git-Lit initiative addresses these problems:

1. **Electronic texts are difficult to edit.** Traditional text repositories like Project Gutenberg and the Oxford Text Archive maintain central, canonical versions of their texts that, in most cases, are virtually immutable. If a reader spots an OCR error in an ebook, he or she must rely on contacting the publisher to propose a correction. Even with an infrastructure such as Project Gutenberg's Distributed Proofreaders, the process of releasing a corrected edition may take months or years. Git-Lit aims to radically streamline the improvement of an electronic text in two ways. First, ease of editing is achieved through GitHub's push-button forking (making a copy of a repository in one's user account) and in-browser editing—a reader may spot a mistake, correct it, and submit a pull request for the change in mere seconds, all without leaving the browser. Second, the decentralized model ensures that no single text may be considered unquestionably canonical, although *de facto* canonicity might be democratically achieved through repository voting mechanisms such as GitHub's stars.

2. **Electronic texts often lack editorial history.** Owing, in some cases, to the age of an electronic text, its editorial provenance is often lost. Many Project Gutenberg editions, for instance, are transcribed from unknown print editions, and the history of their revisions is similarly opaque. Version control mitigates these problems by recording every edit, editor, and edition in the history of the text. When two editions diverge, git provides sophisticated tools for analyzing the differences between these editions. Sites like GitHub further provide graphical network charts, showing the genealogy of each version. Since contributions to a given text are logged according to individual contributor, credit for a given edition may be assigned according to the individual's percentage of total contributions, minimizing the danger, for instance, that a professor may take credit for his or her graduate student's work.

3. **Textual corpora are difficult to assemble.** With some exceptions, notably the `download` function of the NLTK corpus module, downloading a text corpus involves compiling texts from diverse and heterogeneous sources. A would-be text analyst must click through a sequence of web pages to find the corpus he or she wants, and then either download a number of .zip files, or email the corpus creator to request a copy. With multiple texts, this can be a labor-intensive process that is not easily scriptable or automated. Git provides an easy way to solve these problems: by making texts available through the git protocol on GitHub, anyone that wishes to download a text corpus can simply run the command `git clone` followed by the repository URL. Parent repositories can then be assembled for collections of texts using git submodules. A parent corpus repository might be created for nineteenth-century *Bildungsromane*, for instance, and that repository would contain pointers to individual text repositories. These categories would not necessarily be mutually exclusive, and would allow for arbitrary curation of custom corpora. This provides a major advantage over the traditional directory structure model, where the existence of overlapping datasets necessitates the storage and maintenance of redundant data.

4. **ALTO XML is not comfortably human-readable.** ALTO XML, the OCR output format used by the British Library texts as well as texts created by the Library of Congress, is extremely verbose. It encodes the location of each word on the page, and often gives the OCR certainty for each word. While this format is useful for archival purposes, plain text editions are more useful for reading and for most brands of computational text analysis. Git-Lit parses the British Library's ALTO XML, and creates markdown versions of each text that are easily edited. Since markdown is readily converted to other document formats using tools such as Pandoc, this allows each text to be easily exported to PDF, EPUB, LaTeX, Docbook, and others. Additionally, Git-Lit is currently working on a system that leverages GitHub's built-in Jekyll HTML compilers to convert each

text into a web page hosted on github.io, effectively creating 50,000 readable web editions. These new editions will exist as git branches in parallel with the markdown and ALTO XML editions. Since git maintains efficient copies of every historical version of the text, no information about the text is lost in these conversions. Anyone that wishes to improve the conversion script and create newer, better, editions of the original files may freely do so by branching the text from an earlier git commit.

Git-Lit software works by first parsing the XML metadata included with each text. This metadata is used to programmatically generate a repository name and a `README.md` file that describes the text, a document which GitHub will automatically render into a web page at the repository root. This file, along with standard `CONTRIBUTING` and `LICENSE` files (each text is released under the GPLv3), is then committed to local git repositories, initiating version control of the texts. The resulting local repository is then uploaded to GitHub via Python bindings to the GitHub API. Parent repositories are then created using git submodules for each collection of texts based on the their associated Library of Congress subjects. This enables a text analyst interested in 19th century poetry, for instance, to download all of the British Library's released works in this genre simply by running `git clone https://github.com/git-lit/19th-century-poetry.git && git submodule update --init --recursive`.

Since British Library texts are not the only ones being published to git-based platforms like GitHub—notable version-controlled corpora on GitHub include texts from the Text Creation Partnership and the early modern corpus Shakespeare His Contemporaries—git provides a common protocol for sharing, modifying, and distributing texts and textual corpora. Anyone may aggregate these corpora into parent repositories using git submodules. The Git-Lit project will soon launch a web application that will routinely scrape GitHub and other open repository sites for any textual corpus, thereby automating the process of discovering and indexing available corpora. This mechanism will also serve to democratize the curation of corpora, since the corpus index will be sorted by the number of GitHub "stars," or votes, a repository has engendered from the community.

The 50,000 British Library texts processed by Git-Lit, as well as many of the other open corpora described here, are currently being integrated into DHBox, the cloud-based Digital Humanities software suite. Soon, these corpora and many others will be available for download by selecting them from a web-based interface, where they will then be available for analysis using pre-installed versions of the Python NLTK, R, and other textual analytic tools.

This paper will discuss how Git-Lit's methods might be used by other digital humanities projects involved in the creation or analysis of large text corpora, and how digital humanists may contribute to the Git-Lit project. (As an open-source project, Git-Lit welcomes contributions in the form of bug reports, fea-

ture requests, or code.) The paper will also discuss some of the storage and computation limitations of electronically publishing texts via code repositories, and some of the technical problems encountered by the Git-Lit project. Finally, it will suggest pedagogical uses of git-based collaborative digital editing, such as classroom compilation of anthologies or digital scholarly editions. The applications of these technologies are wide-ranging, and are neither proprietary to this project nor to services such as GitHub, but remain concepts of openness and collaboration with powerful implications for the digital humanities.