NCWIT EngageCSEdu TEACHING PAPER

Computing and the Digital Humanities

MARK D. LEBLANC, Wheaton College

This paper introduces three assignments—each with their own "starter kits" for students—for those with a love of the written (and digital) word. These assignments are part of a 'Computing for Poets' course that exposes students to leading markup languages (HTML, CSS, XML) and teaches computer programming as a vehicle to explore and "data mine" digitized texts. Recent advances in computer software, hypertext, and database methodologies have made it possible to ask novel questions about a poem, a story, a trilogy, or an entire corpus. Programming facilitates top-down thinking and practice with computational thinking skills such as problem decomposition, algorithmic thinking, and experimental design, topics that humanities students in our experience rarely see. Programming on and with digitized texts introduces students to rich new areas of scholarship including stylometry (i.e., the statistical analysis of variations in literary style between one writer or genre and another), applied to, for example, authorship attribution.

1. COMPUTING FOR POETS

At Wheaton College (Massachusetts), we offer a semester-long CS-1-like elective course named 'Computing for Poets' to introduce students to programming within one area of the digital humanities: the application of computing to the study of digitized texts. Course material development was done *with* humanities scholars. The 'Computing for Poets' course discussed here is connected with two courses by Michael Drout in the English department at Wheaton: 'J.R.R. Tolkien' (ENG 259) and 'Anglo-Saxon Literature' (ENG 208).

The course exposes students to leading markup languages (HTML, CSS, XML) and teaches computer programming (Python) as a vehicle to explore and "data mine" digitized texts. A learning objective for students in this course is to articulate how computational analyses of digitized texts enables both a "close reading" of a single text as well as a "distant reading" of many texts across time. The goal for each student is to master enough programming to process digitized texts in order to conduct a computational experiment that explores a question of a single novel, set of texts, or entire corpora.

The time has never been better for computing to impact the humanities. Digital humanities represent a growing specialization on many campuses, and the glut of digitized texts, many in languages from around the globe that match a scholar's areas of expertise, has radically altered what it means to be a scholar of texts [Gold 2012]. Computational explorations of texts, sometimes referred to as "computational stylistics" is a subfield within the Digital Humanities and one where students need exposure to and practice with searching and analyzing large digitized corpora.

The Poets course is offered in the spirit of EngageCSEdu's effort for creative offerings that might increase the number of women who consider computing. In the six offerings of Poets since 2004, women enroll in higher percentages than 20 of the 22 offerings of our traditional CS1 course. Since 2000, women enroll in other sections of CS1 at an average of 36%. In contrast, from 2004 to the present, on average 58% of the students in Poets were women [LeBlanc 2015]. In the most recent semester (Spring 2016), 63% of the 27 students are women.

2. SOME GENERAL ADVICE FOR IMPLEMENTING THE POETS-FOCUSED ASSIGNMENTS

(1) Regularly allocate time in your class sessions to bring in colleagues. To find good matches, walk across campus and buy a faculty member a cup of coffee and listen as they discuss their area of expertise. You will be surprised at how quickly their area intersects with computing. In my case, for example, English professor Michael Drout showed up at my office wanting to ask a question across the entire, now digitized, Anglo-Saxon corpus. This conversation results in our on-going collaboration.

(2) Give students a sense of the real demand for people who can work in multidisciplinary groups on computationally-intensive problems. Don't be shy when sharing that the skills learned in this type of course can “get students a job” at the start of an exciting career [Sinha and Rauscher 2014; Rumsey 2012].

(3) Use a “flipped classroom” where students watch lectures before/after class in order to maximize the amount of hands-on play. For example, leverage programming practice sites for students by requiring practice outside of class (*cf*. Codecademy's Python course and/or Runestone Interactive’s, How to Think Like a Computer Scientist [Miller and Ranum 2014]). As much as you are able, leverage the growing number of online resources as a means to protect your vital classroom time for hands-on work.

3. HOW TO IMPLEMENT THREE CORE ASSIGNMENTS IN THE POET SEQUENCE

The EngageCSEdu collection contains five of the assignments in the Poet sequence. For purposes of this paper, I focus on implementation of three of the core assignments:

(1) Reading Poetry Backwards
https://www.engage-csedu.org/find-resources/2poets-reading-poetry-backwards-%E2%80%93-rpb-v10

(2) Regular Expressions for a Puzzle Master
https://www.engage-csedu.org/find-resources/3poets-regex-play

(3) Asking a Question Over an Entire (Anglo-Saxon) Corpus
https://www.engage-csedu.org/find-resources/5poets-only-poetry-searching-anglo-saxon-corpus)

3.1 Reading Poetry Backwards – RPB v1.0

This initial programming assignment requires students to study, understand, and augment a Python program that (re)writes or “breaks” poems in various “deformed” manners, including printing the lines of a poem in reverse (last line to first line), randomized orders, and printing the individual words of a poem in complete reverse order. “Breaking poems” is one method to help the reader more easily read poems. In an undated fragment on a leaf of stationery, Emily Dickinson wrote:

> Did you ever read one of her Poems backward, because the plunge from the front overturned you? I sometimes (often have, many times) have -- a Something overtakes the Mind.

The learning goals include problem decomposition (functions), extending existing code, problem solving with multiple solutions, and building a program to handle a wide range of input texts.

3.1.1 Capturing Student Interest

This programming assignment leverages students love of literature and leads them to a space where literature and computation are merging in collaborative and interdisciplinary groups. The assignment shows by example some of the faculty-student interactions that occur in our interdisciplinary research group of literary scholars and programmers. Student choice is immediately encouraged as students use their program to break the poems written by their own favorite writers.

I have no level of expertise with or even comfort with reading poetry, therefore, I invite an experienced colleague—for example, Reference Librarian Amy Barlow, a lover of poetry—to my class to talk about deforming poems [Leithauser 2013]. Students appreciate the fact that their program might help others read poetry in new ways.

3.1.2 Starter Code Builds Confidence and Persistence

Prior to this assignment, students have practiced with Python for at least two weeks, including finishing a number of the beginning lessons in Codecademy's Python course and finishing in-class, hands-on labs.

In all of the programming assignments, students are given a "starter kit" that includes a detailed specification and parts of a programming solution. My experience shows that asking students to augment an existing code-set mirrors their future interactions with scripting (e.g., modify code from a GitHub repository) and avoids an early discouragement of not knowing how/where to start and/or feelings of being overwhelmed. Of course, instructors may augment the amount of code provided in this or any of the starter kits.

3.1.3 Student challenges

In general, students do not have a good handle on directory (folder) and file structure, therefore, teaching file input and output requires some extra sensitivity on a topic that the instructor considers as second nature. In addition, Python's write statement, along with formatting output, like many languages, is terse, at best. I recommend that the instructor provide sufficient time in class for practice on file IO.

3.2 Regular Expressions for a Puzzle Master

This is the third of five assignments in the Computing for Poets course. This assignment requires students to write regular expressions (regex) to match patterns in words that solve word puzzles. A number of the puzzles are taken from Will Shortz' books [1996, 2003]. Shortz is National Public Radio's (NPR) puzzle master. This assignment is a stand-alone exercise for practice with the powerful pattern-matching syntax of regular expressions. The assignment involves no programming.

A web-based CGI (http://cs.wheatoncollege.edu/regexplay) compares student regex with a dictionary of words and returns a table of resulting word matches. Pairing students on this exercise generates wonderful conversations as students collaborate to design and test regular expressions that will help solve the word puzzles! As students are practicing their regex with this assignment, in-class time is spent showing students how to apply regex in their Python scripts.

3.3 Asking a Question Over an Entire (Anglo-Saxon) Corpus

This fifth of five assignments requires students to consider a collection of Old English poetry and prose texts, including tests using the entire corpus. They are asked to explore whether any words appear *only* in the poetry collection? If so, how many times do these words occur? Students use a Python dictionary (also called a "hash table" or "map") to keep track of all words in the poetry and then remove words from that dictionary that appear in the prose. Learning goals include problem decomposition (functions), extending existing code, technical writing, and writing scripts to produce HTML output. The series of assignments has come full circle: the student's Python script produces HTML pages as output.

4. CONCLUSION

As noted in new frameworks for information literacy [2016], computer science faculty must strive to provide students with opportunities to be information creators and not only consumers. The intersection of digitized corpora and programming enable our students to be creators and makers, and to ask original questions. Students benefit when they recognize that their solutions on one set of texts can be applied to other corpora and collections of texts. For example, in the last assignment that isolates words found only in the poetry of an entire corpus, a student might ponder: "Are there any words used by the Brontë sisters that are never used by Jane Austen?" What do you think?

ACKNOWLEDGMENTS

The breaking poems assignment was inspired by a conversation with Amy Barlow, a Reference and Instruction Librarian, and a number of other library colleagues, specifically Mason Brown, helped with the regular expression exercise. The regular expression work stems from wonderful English and DNA facing page translations when applying regex to English words or DNA (with Betsey Dyer, Biology, Wheaton College; LeBlanc and Dyer, 2007). English professor Michael Drout (Wheaton) is actively involved in the design of assignments for the Poets course as we continue our research collaboration together (http://lexomics.wheatoncollege.edu).

REFERENCES

Codecademy's Python Course. Retrieved April 1, 2016 from https://www.codecademy.com/learn.

Flipping the Classroom. Center for Teaching and Learning, University of Washington. Retrieved April 1, 2016 from http://www.washington.edu/teaching/teaching-resources/engaging-students-in-learning/flipping-the-classroom/.

Framework for Information Literacy for Higher Education (2016). Association of College and Research Libraries. Retrieved April 1, 2016 from http://www.ala.org/acrl/standards/ilframework.

Gold, M.K. 2012. *Debates in the Digital Humanities*. University of Minnesota Press, Minneapolis, MN, U.S.A.

LeBlanc, M.D. and Drout, M.D.C. 2015. "DNA and 普通話 (Mandarin): Bringing introductory programming to the Life Sciences and Digital Humanities. *Procedia Computer Science*, v51, International Conference on Computational Science, Reykjavik, Iceland, p1937-1946.

LeBlanc, M.D. and Dyer, B.D. 2007. *Perl for Exploring DNA.* Oxford University Press.

Leithauser, B. 2013. Reading Poems Backward. *The New Yorker*. July 11, 2013. Retrieved April 1, 2016 from http://www.newyorker.com/books/page-turner/reading-poems-backward.

Miller, B. and Ranum, D. 2014. How to Think Like a Computer Scientist. Runestone Interactive. Retrieved April 1, 2016 from http://interactivepython.org/runestone/static/thinkcspy/toc.html.

Rumsey, A.S. 2013. Creating Value and Impact in the Digital Age Through Translational Humanities. Council on Library and Information Resources: Ruminations. Retrieved April 1, 2016 from http://www.clir.org/pubs/ruminations/03smithrumsey/translational_humanities.

Shortz, Will 1996. *The Puzzle Master Presents: 200 Mind-Bending Challenges*. Random House.

Shortz, Will 2003. *The Puzzlemaster Presents: Will Shortz's Best Puzzles from NPR*. Volume 2. Random House.

Sinha, N. A., & Rauscher, B. M. 2014. Preparing digital natives for industry. *Proceedings of WorldComp 2014*. Retrieved April 1, 2016 from http://worldcomp-proceedings.com/proc/p2014/FEC3353.pdf.