On Teaching XQuery to Digital Humanists

Clifford B. Anderson, Vanderbilt University
<clifford.anderson@vanderbilt.edu>

Abstract

XQuery provides an excellent means for teaching programming to digital humanists because it works seamlessly with their existing XML data, has an elegant and simple core with a well-structured standard library, and can be used in conjunction with XML databases to develop end-to-end web applications. However, current teaching materials for XQuery do not address the needs of digital humanists, presupposing implicit knowledge of programming concepts that they frequently lack. Based on experience teaching XQuery to digital humanists (including alt-ac professionals, archivists, faculty members, graduate students, and librarians) in three distinct settings: a weekly training session for librarians, a graduate seminar on digital humanities, and a two week NEH-supported Institute for Advanced Topics in Digital Humanities, I suggest how the XML community might develop resources to widen the appeal and accessibility of XQuery.

Table of Contents

Defining the Digital Humanities

So this paper is about teaching XQuery to digital humanists, which raises the question, who are the digital humanists I have in mind? The exercise of defining the digital humanities is among the most popular and, at this point, hackneyed exercises in the digital humanities. I'm sure that I could solicit multiple definitions from the attendees of Balisage. Some of you here helped to bring about the field of the "digital humanities" from the now largely superseded concept of "humanities computing" (see [Piez 2008]). As far as it goes, I like the definition in the recently-published "Proposal for a Digital Humanities Center at Princeton University."

> 'Digital Humanities' refers to the use and application of computational tools and methods to humanist domains of study, but also the opposite, the application of humanistic questions to computer science: from history to papyrology, from comparative literature to historical geography, and to the artistic or humanities-influenced design of computational artifacts. [Martin 2013]

The definition is instructively broad. On the one hand, it encompasses the whole range of tools that humanists use in the course of their research and teaching. On the other, it indicates that digital humanities

involves taking a critical perspective on these tools. In other words, digital humanities is not just about using PowerPoint to display information to a class but also thinking about how PowerPoint as a medium shapes and influences the transmission of that information. Inevitably, digital humanities tends to divide into those who concentrate on the development of computational approaches to humanistic problems and those who prefer to interpolate digital media critically.[1] The methodologists themselves work across a wide variety of fields and with a large number of tools, including digital text editing, geographical information systems (GIS), natural language processing, network analysis, the semantic web and statistical analysis, to name only a few. Obviously, no single researcher can be competent in all these areas. Accordingly, the digital humanities has been described as a "big tent," though according to Patrik Svensson it might better be described as "a meeting place, innovation hub, and trading zone." [Svensson 2012, p. 46] In short, the digital humanities is less a discipline than a meeting place of practitioners of various disciplines with different levels of skill, different modes of production, and differing degrees of critical insight on their work.

The practical point is that one cannot assume anything when teaching "digital humanists." There are no prerequisites to calling oneself a "digital humanist" beyond, I suppose, a general interest in computation and the humanities. As a practical matter, digital humanists hail from many different parts of the academy as well as from outside its cloisters. In a broad sense, administrators, faculty members, graduate students, librarians, alt-ac scholars, and information technologists may all credibly claim to be digital humanists. Their perspectives on digital humanities will differ broadly according to discipline and social-economic location. Professors generally enter the digital humanities with specific research agendas. Librarians, by contrast, do not generally have their own research programs; more typically, they're seeking new ways to provide and improve access to the collections under their care. Graduate students may be exploring the options or working as assistants on faculty projects while contemplating alt-act alternatives to standard tenure-track careers. At best, the digital humanities provides the "meeting place" for these fellow-travellers. As Julia Flanders remarks,

> For this reason, I think one of the most interesting effects of the digital humanities upon academic job roles is the pressure it puts on what we think of as our own proper work domains. In the archetypal digital humanities collaboration, traditional faculty explore forms of work that would ordinarily look “technical” or even menial (such as text encoding, metadata creation, or transcription); programmers contribute to editorial decisions; and students coauthor papers with senior scholars in a kind of Bakhtinian carnival of overturned professional usages. [Flanders 2012, 307]

Teaching digital humanists, then, proves challenging because instructors must design curricula that do not make strong presuppositions of any form of domain knowledge. The best curricula allow for emergent collaborations among practitioners, benefiting from and perhaps also relying on the existence of complimentary strengths among students. Yet instructors must be careful not to presuppose any central vantage point. Inevitably, a curriculum will prove too slack or too demanding, too theoretical or too technical, too slanted toward one type of application or another. In such cases, teaching depends on the good will of the participants, who assist each other with learning concepts from other domains.

The lack of presuppositions in the "digital humanities" brings us to a hot button issue, namely, whether coding is necessary to the digital humanities. There is a debate in the digital humanities community between those who think all digital humanists should learn to code and those who argue that learning to program is not necessary and perhaps even harmful to scholarly productivity.[2] The slogan "More hack, less yack!"

[1] I owe this observation to Dr. Jay Clayton, William R. Kenan, Jr. Professor of English at Vanderbilt University, who presented a guest lecture on divisions within the digital humanities on Tuesday, June 10 in the context of our XQuery Summer Institute.

[2] In a blog post titled "Should Digital Humanists code? NO!," Robert Consoli writes:

> Programming is a discipline of a radically different kind from what you’re used to. It goes precisely against the grain of the way you’re using your mind now. Like oils or water color or research in old manuscripts you have to devote your life to it to be any good and programming will destroy any chance you have at being good at what you really love. Why go there? [Consoli 2013]

sometimes gets thrown around, suggesting that digital humanities is primarily about building things rather than talking about them. As Bethany Nowviskie reminds us, however, the original sense of the "yack" in that playful phrase was not aimed against academic theoretical analysis, but at wasteful committee work that frequently encompasses digital projects in the academy [Nowviskie 2014]. The XQuery Institute set out to cross this purported divide between builders and sayers in the digital humanities with as little fanfare as possible. My guiding intuition is that a complementarity exists between coding and encoding– by learning to query your markup, you become a more proficient encoder. So learning even a bit of programming should prove helpful to those working on digital markup projects. Of course, establishing this complementarity is easier with some programming languages than others, which brings me to XQuery.

Making the Case for XQuery

I may as well state upfront that I regard XQuery as a fantastic language for digital humanists. If you are involved in marking up documents in XML, then learning XQuery will pay long-term dividends. I do have arguments for this bit of bravado. My reasons for lifting up XQuery as a programing language of particular interest to digital humanists are essentially three:

- XQuery is domain-appropriate for digital humanists.
- XQuery allows for full-stack application development.
- XQuery is compact, terse, and relatively easy to learn.

Let's take each of these points in turn.

First, XQuery fits the domain of digital humanists. Admittedly, I am focusing here on a particular area of the digital humanities, namely the domain of digital text editing and analysis. In that domain, however, XQuery proves a nearly perfect match to the needs of digital humanists.

If you scour the online communities related to digital humanities, you will repeatedly find conversations about which programming languages to learn. Predictably, the advice is all over the map. PHP is easy to learn, readily accessible, and the language of many popular projects in the digital humanities such as Omeka and Neatline. Javascript is another obvious choice given its ubiquity. Others recommend Python or Ruby. At the margins, you'll find the statistically-inclined recommending R. There are pluses and minuses to learning any of these languages. When you are working with XML, however, they all fall short. Inevitably, working with XML in these languages will require learning how to use packages to read XML and convert it to other formats.

Learning XQuery eliminates any impedance between data and code. There is no need to import any special packages to work with XML. Rather, you can proceed smoothly from teaching XML basics to showing how to navigate XML documents with XPath to querying XML with XQuery. You do not need to jump out of context to teach students about classes, objects, tables, or anything as awful-sounding as "shredding" XML documents or storing them as "blobs." XQuery makes it possible for students to become productive without having to learn as many computer science or software engineering concepts. A simple four or five line FLWOR expression can easily demonstrate the power of XQuery and provide a basis for students' tinkering and exploration.

Second, XQuery allows for "full-stack" web development. What I mean by this is that XQuery can be used in conjunction with native XML databases to develop end-to-end web applications. This contrasts with other popular approaches, which require aspiring developers to learn SQL for interacting with relational databases, maybe to understand a bit about object-orientation, and to grasp how to express their outputs in HTML, perhaps with the help of a templating language. By contrast, it's perfectly feasible to write simple interactive web applications just using XQuery and a native XML database. Eventually, students

The point, I take it, is that learning to program will prove a distraction from the focus required to advance in one's academic field.

will need to learn how to incorporate CSS and JavaScript to improve the appearance and usability of their applications. But they can get things off the ground without learning anything more than XQuery.

A third advantage of XQuery is that it's easy to learn. You wouldn't think this is the case if you read questions on Stackoverflow. But my suspicion is that the majority of those who find XQuery perplexing or even execrable have not bothered to learn about functional programming and run headlong into brick walls like single-assignment. It is indeed tough for experienced programmers to switch from imperative, object-oriented programming to functional programming. A lot of the idioms and design-patterns are just different. By contrast, digital humanists are generally learning to program for the first time. They don't have to unlearn habits from imperative programming-languages. Could Edsger Dijkstra's dictum–"It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration" [Dykstra 1975]–actually put digital humanists at an advantage over experienced programmers when learning XQuery (substituting BASIC for any imperative or object-oriented language)?[3] In a more serious vein, functional programming has the advantage of not allowing beginning programmers to "mess up" their data. So long as you stay away from extension functions, you can promise students that nothing they do can corrupt their data. This brings a certain measure of relief to new programmers, who can "safely" explore the standard function library.

XQuery is fairly simple and straightforward at heart. While I wouldn't recommend that beginners tackle the *XPath and XQuery Functions and Operators 3.0* recommendation, as instructive as it is to experts, working through groups of related functions using a site like Priscilla Walmsely's *FunctX XQuery Functions*[4] is a great method to familiarize beginners with XQuery expressions.

Here's a last point about XQuery vis-a-vis other functional programming languages. I used to refer to XQuery as "functional programming light" prior to Version 3.0. However, with support for higher-order functions in XQuery 3.0 and built-in functions for mapping and folding, XQuery stacks up nicely against any other functional programming language out there. However, that doesn't mean that one should automatically jump into those deep waters with beginners.[5] In general, functional programming relies heavily on recursion, which is notoriously difficult for non-programmers to master. By contrast, FLWOR expressions make it possible to write many expressions without using recursion.[6] While I have taught recursion in all three contexts I discuss below, it's nice not to have to explain recursion too early in the learning process.

Three Case Studies

I've taught XQuery in three different contexts. Before describing my experiences, it's reasonable to ask what qualifies me to teach XQuery. My background is purely in the humanities. I was a philosophy major who eventually earned a Ph.D. in theology. But I also had the privilege of growing up around computers, following a progression from Applesoft Basic to Pascal to Visual Basic.

When I was hired as a curator in the Princeton Seminary Library, I was asked to develop a digital library program for our special collections. I started up a project in 2002 with C# and XSLT, which is still running today.[7] But I had a difficult time implementing search functionality and began looking around for other options. In 2006, I attended a software conference (now sadly defunct) called Software Development West (SD West), where I signed up for sessions on XPath by Debbie Lapeyre and XQuery by Jason Hunter. Suffice it to say, the combination of these sessions persuaded me that we should switch out digital library platform from .NET to MarkLogic. The Princeton Theological Seminary Library purchased a license for a standard edition of the MarkLogic in June 2007. In July, our metadata librarian and I took a three day

[3]I write this as someone who learned first to program in Applesoft Basic on an Apple II. Dropping line numbering was like losing the scaffolding when I switched to Turbo Pascal in high school.
[4]See http://www.xqueryfunctions.com/
[5]I once spent an evening at my kitchen table playing the Towers of Hanoi over and over again in order to grasp the concept.
[6]I owe this observation to a conversation with Jonathan Robie.
[7]See http://scdc.library.ptsem.edu/mets/mets.aspx

"Introduction to MarkLogic Server: Developer Course." We worked with MarkLogic Professional Services to rewrite our digital library platform from C# and XSLT to XQuery. Meanwhile, I spent the summer reading and re-reading Priscilla Walmsley's *XQuery* [Walmsley 2007].

So much for my qualifications. Like many digital humanists, I pieced together what I needed to know as I needed to know it. I was at least a little farther ahead of the people I planned to teach.

Teaching Librarians

The immediate impetus for our XQuery training series at Princeton Theological Seminary arose from the practical need to involve more staff members in digital library projects. We had aspirations of building a large-scale digital library and we did not want to outsource its development to others. However, we also had a larger vision of helping our library staff members transition from traditional cataloging practices to modes of work more suited to large-scale digital production. Christine Schwartz, who was the head cataloger at the time, and I had many lengthy conversations about the future direction of the library profession. As the Network Standards Office at the Library of Congress promoted new XML metadata standards such as MARCXML, METS, MIX, MODS, etc., we foresaw that catalogers would need to develop new skill sets to work with these XML formats. So our broader aim was to train librarians in XML and XQuery with the hope of reinventing at least some of our cataloging and descriptive processes.

We arranged for Priscilla Walmsley to provide our initial introduction to XQuery. Walmsley led a three day workshop in Princeton on November 12-14, 2008. Her workshop introduced our staff to XQuery and to the essentials of MarkLogic. But, like any training course, there is only so much you can absorb in a short amount of time. The key to learning anything is practice over time.

So we gathered a small group of staff members during the week following Walmsley's training course to plan what became a weekly XQuery seminar. The group included me, our Christian education librarian, collection development librarian, head cataloger, manuscript librarian, and technical services librarian. We met at 11:00 a.m. every Friday morning for an hour-long session. As a textbook for the class, we adopted Ron Hitchens' *Getting Started with XQuery* [Hitchens 2008]. For the most part, we worked through practical exercises with an eye toward addressing typical library problems. For example, our first session covered how to gather metadata from journal articles and package them for submission as DOIs to CrossRef. Other sessions addressed how to interact with the WorldCat API using XQuery and how to generate RSS feeds of newly-ingested metadata. Our focus was not entirely practical, though. We introduced concepts like recursion (with the Towers of Hanoi) and higher-order functions (using the MarkLogic implementation predating XQuery 3.0). In the beginning, I led most of the sessions. As participants gained experience with XQuery, they took turns leading sessions.

The outcome of these sessions was generally positive. One of the participants transitioned from her position in the technical services department to our digital library team. Another member subsequently became the ScrumMaster for the digital team. As we expanded our digital team, the sessions proved crucial for onboarding new members. (It's basically impossible to recruit librarians with prior knowledge of XQuery.) Other participants in the sessions learned enough XQuery to carry out basic metadata tasks. Even those who did not actively incorporate XQuery into their daily work gained a new appreciation for its role in our digital library. From another perspective, then, the sessions strengthened staff support for our digital library program as a whole.

I am glad to report that after a hiatus these sessions have resumed at the Princeton Seminary Library. Interestingly, the focus is now primarily on XSLT since most of the team members have sufficiently mastered XQuery.

Teaching Graduate Students

In spring 2014, David Michelson and I co-taught a graduate-level seminar at Vanderbilt University titled "Topics in Digital Humanities: Introduction to Digital Text Editing and Analysis." Michelson is an assistant

professor in the Divinity School at Vanderbilt with an affiliated appointment in Classics. The initiative for our course arose from the desire to offer a version of the curriculum we had proposed for the XQuery Summer Institute (described below) to graduate students on campus. Eight students registered for the course along with a number of auditors, including several librarians.[8]

We aimed to provide a balance between theory and practice in the seminar. We started with a little history of SGML and XML. (Students loved reading "What is Text, Really?" [DeRose, et al. 1990]) We also read contemporary works in the digital humanities, including Franco Moretti's *Distant Reading* [Moretti 2013], Matthew Jockers' *Macroanalysis* [Jockers 2013], and Stephen Ramsay's *Reading Machines: Toward an Algorithmic Criticism* [Ramsay 2011]. Our practical texts included Joe Fawcett, Danny Ayers and Liam Quin's *Beginning XML* [Fawcett, et al. 2012] and Priscilla Walmsley's *XQuery* [Walmsley 2007].

We set a very ambitious agenda for our class. We presupposed no knowledge of XML or any programming language. After imparting the basics of XML, we taught students how to encode humanities texts in simple TEI. From there, we moved to XPath and then to XQuery. We asked students to install eXist and populate a collection with XML documents relevant to their areas of research. Finding relevant datasets turned out to be the largest problem. In the field of art history, for example, it turned out to be surprisingly difficult to locate open access VRA Core datasets in XML. After the mid-semester break, we began teaching sessions on various useful techniques in XQuery. For example, we showed students how to generate web pages for their datasets using a recursive typeswitch expression.[9] We demonstrated the use of natural language processing using the Alchemy API[10] and the Stanford NER XQuery Module in eXist.[11] Finally, we held a session on interoperating with GeoJSON to produce simple maps of place names mentioned in documents.

We designed the course so that students would come away with an array of computational techniques for analyzing documents in their own fields. My co-instructor, David Michelson, was particularly concerned that students grapple with how the digital humanities could enhance their own evolving research agendas. He asked participants to write a brief "implementation plan," indicating their aspirations for their future work in the digital humanities and how they intended to deepen their expertise in order to meet their short-term and long-term goals.

We offered students the option of writing a traditional paper for their final project, but everybody elected to submit a hybrid final consisting of a digital project along with a critical reflection. The students' projects were uniformly excellent, demonstrating genuine creativity in the application of the techniques we imparted to their various fields of study.

Of course, we also came away with ideas about how to improve the course. The first and perhaps most obvious point is that the traditional model of the humanities seminar is not the optimal way to teach a digital humanities course, at least not one focused on teaching XQuery. A typical graduate seminar in the humanities meets weekly for several hours; students spend the week in between classes working through assigned readings, which tend to be fairly heavy. This pattern does not function well when you are teaching a programming language. While many students took advantage of our office hours, we should have held practice sessions during the week. As with learning any new language, the key is practice and an intensive seminar once a week does not work as well as shorter but more frequent sessions. We should also have assigned students more programming challenges. We asked students to turn in a number of written reflections on the assigned readings over the course of the semester, but we did not assign many programming exercises. Again, it's fairly obvious in retrospect that working through problem sets would have helped students master difficult techniques in XQuery.

On the positive side, we largely succeeded in our ambitious goal of teaching XML, TEI, XPath, XQuery, and eXist in a single semester course. We knew going in that our students had varying backgrounds in the digital humanities. We did not expect those who were completely new to the digital humanities to

[8]The syllabus for the course is available at http://paralipomena.com/schedule/.
[9]See https://en.wikibooks.org/wiki/XQuery/Typeswitch_Transformations
[10]See http://www.alchemyapi.com/
[11]See http://exist-db.org/exist/apps/stanford-ner/index.html

master all these technologies. We hoped simply that they would gain an appreciation for their possibilities and acquire a solid foundation for future growth. Students with more advanced backgrounds applied their new-found knowledge to complex questions in their domains of expertise, demonstrating in their final presentations just how powerful the combination of these technologies can be.

Teaching Digital Humanists

My final case study is the XQuery Summer Institute, which took place at Vanderbilt University from June 9 to 20, 2014.[12] The Office of Digital Humanities (ODH) at the National Endowment for the Humanities (NEH) graciously supported the XQuery Summer Institute as one of its Institutes for Advanced Topics in the Digital Humanities. The formal title of the institute was the "XQuery Summer Institute: Advancing XML-Based Scholarship from Representation to Discovery."[13] As indicated, our aspiration was to assist digital humanists to make the step (or leap?) from hand-encoding their digital texts in TEI (or any other XML-based markup language) to exploring, remixing, and enriching them with XQuery.

On the first two days of the institute, David Michelson and I reviewed the basics of XML and TEI. As our dataset for the institute, we selected the newly-released (under a CC-BY-NC license) Folger Digital Texts of William Shakespeare.[14] The Folger Digital Texts are what I might call "high-style" TEI: among other characteristics, every word, punctuation character, and space in the digital texts is marked up with an XML:ID. We had the fortunate opportunity during the institute to speak with Rebecca Niles and Michael Poston at the Folger Library to discuss their encoding decisions. An interesting feature of the institute was the opportunity it afforded to look at these texts from different perspectives, i.e. from encoding to querying to indexing in eXist. We all came away with a much stronger sense of the tradeoffs made in different contexts.

In the second half of the first week, Jonathan Robie introduced XPath and XQuery. As you all know, Robie is the lead editor of the XPath and XQuery Recommendations at the W3C. He is also a wonderful instructor. Among the many points Robie made during the institute is that encoding and querying should be considered complimentary activities. The best way to mark up documents is to encode, query, and repeat–in other words, to work iteratively rather than serially from markup to application. Robie finished the week by demonstrating the analytic power of new constructs in XQuery 3.0 such as windowing for digital humanities research.

Whereas the first week concentrated on the XQuery language itself, the second week was devoted to its implementation in eXist. Winona Salesky, an independent library consultant, and Kevin S. Clarke, Digital Library Programmer at the University of California Los Angeles, led most of the second week. Salesky and Clarke built a sample "browse and search" application for the Folger Digital Texts.[15] The sessions in the second week covered the topics necessary to design such an application, from organizing code into modules, indexing in eXist, writing Model-View-Controller (MVC) applications, to writing test suites for modules. Dale Poulter, coordinator of search and core services, showed us how to deploy our sample eXist application on Amazon AWS during the morning of our final day.

As would be evident to anyone who followed the Twitter-stream for #xqy14 during the two weeks of the institute, we had a wonderful time together and covered a very large amount of ground. A key to the success of the institute was our use of pair-programming. As we alternated between lectures and hands-on exercises, we asked participants to partner with each other to try out techniques and to talk through any problems together. We also rotated these pairings daily. This technique ensured that nobody became ineluctably stuck while attempting the exercises. Our informal motto for the institute was "No digital humanist left behind!"

[12] See http://xqueryinstitute.org/
[13] See the application for the XQuery Summer Institute on the NEH website: http://www.neh.gov/files/grants/vanderbilt_university_advancing_xml-based_scholarship.pdf
[14] See http://www.folgerdigitaltexts.org/
[15] The application is available under an open source license at https://github.com/CliffordAnderson/shakespeare-search

The diversity of the participants modeled the diversity of real world digital humanities projects. During our two weeks, we addressed nearly every angle of creating a digital humanities project *in nuce*, from articulating research questions to building full-fledged applications. Individual participants arrived with stronger background in some areas than others. By collaborating throughout the two weeks, we all gained greater appreciation for these different aspects–from the challenge of articulating a research agenda in the digital humanities to what goes into building and deploying web applications for digital humanities projects. The pacing of the institute would have been different if we had invited only faculty or librarians to apply. Inevitably, some thought we moved too quickly and others too slowly through certain portions of our agenda. But the diversity of our group mirrored the diversity of the "big tent" of the digital humanities. We are grateful to the Office of Digital Humanities for affording an all-too-rare opportunity to provide an educational event encompassing participants from all disciplines under that "big tent."

Obstacles for Digital Humanists

What are the unique challenges of teaching XQuery to digital humanists? I think we can extrapolate at least three hurdles digital humanists face when seeking to learn XQuery.

The first is simply gaining a vision for the place of programming in their scholarship. Why make the effort to learn to program? Is that not a step outside the humanities into a foreign, and sometimes hostile, world? Is it not better to leave programming to the experts? The move from word processing to markup languages is far more incremental than the leap from encoding to coding. The current enthusiasm for digital humanities has perhaps lessened this line of concern. Still, the questions remain pertinent given the demands on scholars' time.

Teaching XQuery to digital humanists requires helping them envision how their new skills and insights will advance their scholarship. We cannot assume any intrinsic motivation to learn programming for its own sake. This renders the majority of the standard teaching materials for programmers problematic, at least as standalone documents. Most programming manuals assume that readers are already motived to learn a given technology and, after perhaps a few paragraphs about its power and usefulness, immediately jump into the details. For digital humanists, a better approach would be to explain general concepts with reference to concrete research problems. For instance, using XQuery to interact with RESTful APIs might be learned in the context of teaching digital humanists how to use the Alchemy API to carry out sentiment analysis on their texts.

A second major challenge is that most digital humanists encounter XQuery without any background in computer science or software engineering. This makes for humorous "namespace" conflicts with natural language. While teaching the XQuery Summer Institute, we began keeping an informal list of technical terms in XQuery that have natural language cognates: evaluate, expression, predicate, etc. Even technical slang like "grok" needs explanation. David Michelson played our "faculty Luddite" role during the institute.[16] Whenever an unexplained term popped up, Michelson would request an explanation on behalf of the group. His role was important because, as technical terms piled up, participants tended to defer questions or pass over them altogether.

Of course, just defining terms only gets you so far. The primary issue was filling in the necessary background knowledge. Digital humanities can be more demanding than traditional computer science courses. Whereas there is a recommended sequence of courses in computer science programs, which introduce and then deepen a body of knowledge over time, digital humanities courses and training sessions generally presuppose little to nothing. Digital humanists must frequently tackle new concepts like typing and scoping while also learning new computational techniques.

A related issue is that few digital humanists have experience with software engineering. Training resources for XQuery frequently assume readers have experience with common tools like relational databases,

[16]For the record, Dave numbers among the most sophisticated digital humanists.

making analogies to SQL and relational tables, for example, when explaining joins in XQuery. This assumption makes perfect sense when teaching technologists but it doesn't work for digital humanists since you'll inevitably need to explain how a relational database functions for them to get the analogy. Best practices like using source control and writing unit tests also require clarification.[17] This problem becomes especially acute when digital humanists aspire to build systems for the project. Broadly speaking, the divide between librarians and faculty becomes wider here. Librarians frequently do need to develop systems and will thus need to master the myriad of software engineering practices and tools. Many faculty members will experience software engineering as a domain of diminishing returns. An effective digital humanities center or campus network will connect faculty to developers when they are ready to build production systems for their projects.

A third challenge is that digital humanists frequently need to push beyond the boundaries of XQuery for the purposes of their research. In our digital humanities class at Vanderbilt, for example, we wanted to reproduce some of the analytical techniques that Matthew Jockers discusses in *Macroanalysis: Digital Methods and Literary History*. As far as I am aware, there is no statistical library for XQuery. So we demonstrated how to use XQuery to gather data from documents and then how to import that data into R for statistical analysis. While the combination of XQuery and the R programming language is potentially extraordinarily powerful, switching contexts between the languages proved too difficult. We did not have sufficient time to introduce R, RStudio, ggplot2, etc. and the students who tried to incorporate this technique into their final projects were not successful. If XQuery had a data analysis library akin to Python's *pandas*,[18] for instance, it would be easier for them to use XQuery to produce the kind of digital text analysis that many have in mind when contemplating computational applications. Recent work bridging XQuery and SPARQL is promising because a significant number of digital humanities projects include a network analysis component. [See, e.g., Cagle 2011] Enriching the XQuery ecosystem with libraries to carry out basic "digital humanities" functionality would help to make the language more accessible. eXist leads the way in this respect with its package system, incorporating such useful packages as the Stanford Named Entity Recognizer.

Making XQuery More Accessible

As I wrap up, I'd like to suggest a few ways to improve the accessibility of XQuery for digital humanists.

First, we need to develop instructional materials for digital humanists. In general, the materials for learning XQuery are few compared to other programming languages, reflecting no doubt its level of usage in the general technology community. We may be thankful for Priscilla Walmsley's *XQuery* [Walmsley 2007]. Where would we be without her book? I hope that she and O'Reilly will consider a second edition now that XQuery 3.0 has reached recommendation status. We may also be grateful that Erik Siegel and Adam Retter are currently writing a book for O'Reilly on eXist; it's already an extremely valuable resource in its "rough cuts" format. [Siegel and Retter 2014] Regrettably, Ron Hitchens' *Getting Started With XQuery* has gone out of print at the Pragmatic Bookshelf, presumably because new versions of eXist and MarkLogic have rendered dated its instructions for getting started. [Hitchens 2008] Given that Hitchen's book was among the most accessible for beginners, a gap currently exists in the literature for introducing beginners to XQuery and XML databases. The excellent XQuery Wikibook provides a bunch of really useful examples of XQuery in action.[19] What's needed is a textbook for use in digital humanities workshops and classrooms. Crucially, the sample applications in this proposed textbook should center on narrative documents and, for the most part, avoid mathematical examples. In other words, if you plan to teach recursion, build an algorithm to validate palindromes rather than solve the Fibonacci sequence.

Another way to improve the accessibility of XQuery would be to encourage the sharing of datasets and code. As mentioned, the students in our digital humanities seminar at Vanderbilt experienced difficulty

[17] I've led digital humanities sessions, for instance, that have devolved into teaching git rather than the technology at stake.
[18] See http://pandas.pydata.org/.
[19] See http://en.wikibooks.org/wiki/XQuery.

finding open datasets in some fields of the humanities. The problem is especially acute in art history but there is a lot less TEI on GitHub than you might expect. (There are, of course, many notable exceptions.) Libraries and other cultural institutions that produce TEI should adopt a default policy of sharing their data under Creative Commons licenses. The same point applies, *ceteris paribus*, to sharing code. Sharing examples of XQuery on GitHub or just as Gists can really prove helpful to those getting started.[20]

A final recommendation is more ambitious. My hope is to integrate XPath and XQuery into the teaching of TEI and other markup standards in the digital humanities community. As Jonathan Robie indicated during our XQuery Institute, encoding and coding should be complimentary, iterative activities. More strongly put, the full implications of digital encoding do not become apparent until scholars learn to reorder, remix, and rework their texts computationally. We all know the effort involved in marking up a text in TEI. If the end goal is simply to transform the marked up document into HTML or PDF for publication on the web, then we cannot expect scholars to continue to make that investment of time over the long run. By providing scholars with the computational tools to analyze their documents, we repay their efforts to encode them. Moreover, teaching XQuery alongside TEI would benefit scholars' understanding of markup, allowing them deepened insight into why particular encoding decisions might be more or less efficient for particular purposes.

What would this new method of teaching TEI look like? The syllabus has yet to be worked out. However, I imagine that the basics of the approach could be accomplished in three day workshops, which aimed less at building systems with XQuery than at providing a sufficient introduction to the language for digital humanists to commence an iterative process of encoding and querying. Where they go from there would be up to them–as with any programming language, there are no limits on where XQuery can take them.

Conclusion

The pressures on XML as a markup standard in the general community of technologists also exist in the digital humanities, though in more muted fashion. While most digital humanists working on textual projects continue to work in XML, they interact with librarians and other technologists who have increasingly turned to JSON to transfer data across the wire. From my perspective, the digital humanists would be impoverished if leaders in the digital humanities community sought to rewrite TEI in JSON. What would that project look like? More likely, digital humanists might just give up on marking up documents in favor of conducting statistical analyses on datasets of unstructured texts. If we wish to avoid this future and preserve a place in the digital humanities for the kind of critical editions that XML markup affords, it behooves us to assist scholars with climbing up the XML value chain–from mastering XML languages like TEI to learning analytical technologies like XPath and XQuery.

Bibliography

[Cagle2011] Cagle, Kurt. "When XQuery and SparQL Collide," *Proceedings of Balisage: The Markup Conference* 7 (2011). http://www.balisage.net/Proceedings/vol7/author-pkg/Cagle01/BalisageVol7-Cagle01.html

[Consoli2013] Consoli, Robert. "Should Digital Humanists code? NO!" *squinches*. August 2, 2013. http://squinches.wordpress.com/2013/08/02/should-digital-humanists-code-no/

[DeRose1990] Steven DeRose, et. al. "What Is Text, Really?" *Journal of Computing in Higher Education* 1:2 (1990): 3-26.

[Dykstra1975] Dijkstra, Edsger W. "How do we tell truths that might hurt?" June 18, 1975. http://www.cs.virginia.edu/~evans/cs655/readings/ewd498.html

[20] I'd like to thank Christine Schwartz (https://gist.github.com/caschwartz) and Joe Wicentowski (https://gist.github.com/joewiz), among others in the digital humanities community, for sharing their XQuery snippets as Gists.

[Fawcett2012] Joe Fawcett, Danny Ayers, and Liam R. E. Quin. *Beginning XML*, 5th Edition. Indianapolis: John Wiley & Sons, 2012.

[Flanders2012] Flanders, Julia. "Time, Labor, and “Alternate Careers” in Digital Humanities Knowledge Work," pp. 292-308 in Matthew K.Gold, ed., *Debates in the Digital Humanities.* Minneapolis: University of Minnesota Press, 2012. http://dhdebates.gc.cuny.edu/debates/text/26

[Hitchens2008] Hitchens, Ron. *Getting Started with XQuery*. Raleigh, N.C.: Pragmatic Bookshelf, 2008. http://pragprog.com/book/xquery/getting-started-with-xquery

[Jockers2013] Jockers, Matthew. *Macroanalysis: Digital Methods and Literary History.* Urbana-Champaign: University of Illinois Press, 2013.

[Kay2014] Kay, Michael, ed. *XPath and XQuery Functions and Operators 3.0.* April 8, 2014. W3C Recommendation. http://www.w3.org/TR/xpath-functions-30/

[Martin2013] Martin, Meredith. “Proposal for a Digital Humanities Center at Princeton University.” August 2013. https://digitalhumanities.princeton.edu/files/2013/08/Proposal-for-a-Digital-Humanities-Center-at-Princeton-University3.11.pdf.

[Moretti2013] Moretti, Franco. *Distant Reading*. London: Verso, 2013.

[Nowviskie2014] Nowviskie, Bethany. "on the origin of 'hack' and 'yack.'" *Nowviskie.org*. January 8, 2014. http://nowviskie.org/2014/on-the-origin-of-hack-and-yack/.

[Piez2008] Piez, Wendell. "Something Called 'Digital Humanities,'" *Digital Humanities Quarterly* 2:1 (2008). http://www.digitalhumanities.org/dhq/vol/2/1/000020/000020.html.

[Ramsay2011] Ramsay, Stephen. *Reading Machines: Toward an Algorithmic Criticism*. Urbana-Champaign: University of Illinois, 2011.

[Siegel2014] Erik Siegel and Adam Retter, *eXist: A NoSQL Document Database and Application Platform*. Sebastopol, CA: O'Reilly. 2014.

[Svensson2012] Svensson, Patrik. "Beyond the Big Tent," pp. 36-49 in Matthew K. Gold, ed., *Debates in the Digital Humanities.* Minneapolis: University of Minnesota Press, 2012. http://dhdebates.gc.cuny.edu/debates/text/22

[Walmsley2007] Walmsley, Priscilla. *XQuery: Search Across a Variety of XML Data*. Sebastopol, CA: O'Reilly. 2007

[Walmsley2014] Walmsley, Priscilla. *FunctX XQuery Functions.* 2006-2014. http://www.xqueryfunctions.com/