Using Poetry as a Teaching Tool for the Humanities to Learn Computer Science Concepts

Joseph Alexander Brown  
School of Computer Science  
University of Guelph  
Guelph, ON, Canada N1G 2W1  
jbrown16@uoguelph.ca or  
jb03hf@gmail.com

Terry Trowbridge  
Brock University  
500 Glenridge Ave.  
St. Catharines, ON, Canada L2S 3A1  
ttrowbridge@brocku.ca

ABSTRACT
The emerging fields of digital humanities and humanities computing require new approaches in the area of their education by computer scientists. Currently, the role of educators in computer science has been to ready computer science students to obtain employment and research within the sciences, however, this must be changed if these two new fields are to be adequately served. We look at four areas where poetry can be used as an introduction to computer science: the analysis of poetry which uses the computer as its media, the analysis of concrete poems, the creation of poetry using a computer or using the computer as an aid to its creation, and professional development for instructors.

Categories and Subject Descriptors
J.5. [Arts and Humanities]; [Literature]; K.3.2 [Computer and Information Science Education]; [Curriculum]

General Terms
Theory

Keywords
Humanities, Curriculum, Poetry

1. INTRODUCTION
We propose that college level courses that teach Canadian twentieth-century poetry and twenty-first century poetry need to begin including lessons about computer code. Since the 1980’s, Canadian poets have been experimenting with computer code in their work. The earliest example of a poet actually coding we have found is bpNichol’s First Screening [11] which he coded using his Apple II home computer. Additionally, some earlier poems by bpNichol that appeared in the 1960’s traditional paper formats [16][18] can be analysed using principles from computer programming.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 2012 ACM 978-1-4503-1084-0/12/06 ...$15.00.  

Using examples from bpNichol’s work (and others), we will demonstrate that lessons in simple examples of coding should be taught to students of Canadian literature so that they can understand the experiments that Canadian poets have produced. Likewise, these poets can deepen the appreciation of literature students for coding, and also deepen the appreciation of computer science students for literature and the arts. We will compare our proposals for lessons that use bpNichol poetry to a small sample of literature in teaching and learning in different classroom settings to show that more discussion about curriculum like ours is interesting across diverse disciplines and student cohorts.

In order to show how elements of computer science can be introduced to humanities majors we look at three case studies from our own experiences and give preliminary suggestions for how they can be used in practice as well as background resources for the educator. These lessons are problem solving-based and goal-oriented rather than focusing on the particular language used by poets. This goal oriented approach has been shown in the literature to lead to overall improvements in learning. These ideas for projects have been linked to the concepts in programming languages which are required for their implementation. Each of the three examples use different tools of the programming languages but all require a teaching strategy directed towards problem solving.

The remainder of the paper is organised as follows: Sections 2-4 are case studies from our own experience as educators and examinations of the findings of others. Section 2 looks at how computer code is necessary in order to understand the previous creations of poets looking primarily at bpNichol’s disk of poems, First Screening, and why humanities will require knowledge of code. Section 3 looks at how the aspects of mathematics used in computer science can be used in the analysis of even poems which were not created with computational assistance. Section 4 examines how the creation of poetry via computers and computer-human creation can be used as a teaching tool by creating assignments for beginning coders that both computing and humanities majors will find valuable. Section 5 looks at where these suggestions fit into current pedagogical trends in computational humanities and digital humanities. Section 6 briefly relates the proposed lessons to broader problems in pedagogy. Finally, Section 7 gives concluding remarks about use of computers for poetry as an educational tool and an overview of future directions in using computing as an aid to the analysis and appreciation of poetry.
2. CASE STUDY 1 - COMPUTER CODE FOR POETRY

A working knowledge of code is becoming necessary not only for the analysis tools for large texts but in order to understand the texts themselves. Looking at bpNichol’s First Screening we see an example of how a working knowledge of coding is necessary. This disk, created as programs for the Apple II, allowed the poet to expand his normal concrete poetry from a static into a moving form by display on the screen. The code has a number of interesting features, such as a hidden poem which can only be accessed by reading the open source of the poetry. The poem is indicated by the statement:

110 REM FOR THE CURIOUS VIEWER/READER THERE’S AN ‘OFF-SCREEN ROMANCE’ AT 1748. YOU JUST HAVE TO TUNE IN THE PROGRAMME.

This section of poetry, and thus analysis, is locked off for those with no knowledge of the source.

The analysis of such poetry has been undertaken by Jim Andrews and Geof Huth who have not only been giving criticisms of the work [2] but have also been part of a team dedicated to its preservation [3][4]. Such works of bpNichol have not traditionally been seen as important as his physical paper works, such as the Martyrology [17]; he was known for his Avant Garde nature and would often give performances of works which were ephemeral in nature, such as his sound poetry [16]. The preservation techniques for software, such as updating and maintaining archives, are not as well formed as those for print media/sound recordings. They require someone familiar with not only the medium, but with the method of generation. Jim Andrews sees readership of this poem as requiring “a slightly deeper level of computer literacy;” and, since bpNichol’s comment is only visible in the source code, it “encourages us not only toward some basic interactivity, but toward looking into the programming source code itself” [2]. Issues exist due to the ephemeral nature of hardware and software. The Apple II is long out of date, yet one can still learn the language in order to view the source as well as see it presented in terms of a video taken or emulation online [3].

Such a coding project would be excellent for a student to convert one of these poems from the Apple II basic into another learning language. The poems are quite simple in terms of the language concepts used and thus would made an excellent beginning assignment to work on the programming concepts of loops, output to a terminal, conditionals, counters/variables, and character strings. This would also be a fine introduction to the ideas of open source, reading source code created by another programmer, and legacy code. An interesting project would be for a small class or a class working in pair programming teams to each take a different poem of the First Screening series and from this an entire computer language X version of the work could be constructed as a contribution to the preservation process.

3. CASE STUDY 2 - ANALYSIS OF ‘TRADITIONAL’ POEMS

In early 2009, the authors carried a copy of the latest bpNichol anthology The Alphabet Game: a bpNichol reader [18] to one of the undergraduate Computer Science workrooms at Brock University. We did this in order to play around on the chalkboards with several of the poems inside in order to develop our theories about bpNichol and cryptography. Joseph Brown was already well-known by the students who were in the room as a Teaching Assistant in Computer Science as well as a graduate student. Terry Trowbridge was an unlikely person to find working on math in the computer science department. A couple of the students knew him as a poet [43], but stranger still was that he was a graduate student in the social sciences, located on the other side of Brock University’s campus. It was explained to a group of curious undergraduates that we were working on a mathematical problem in poetry, this being bpNichol’s “Probable Systems” series [12]. They questioned why we were looking at poetry. When we explained that we were treating the poems in “Probable Systems” as cryptography problems, they immediately set about trying to solve the poems we were analyzing.

They recognized potential graphs [21, 15], potential encryption [18], and other mathematical challenges in bpNichol’s poems. As more students walked past the room, some poked their heads in to listen to the bizarre problems like ‘dividing giraffe by sailboat’ [12] and a handful of them joined the impromptu tutorial in looking for integers which will satisfy an alphametic, identifying graphs and paths, along with other skills that normally belong to disparate parts of computer programming.

In order to gain this same experience with other classes we propose that a brainstorming seminar with some of the poems found in The Alphabet Game would actually benefit undergraduate students in Computer Science by giving them an unusual context to test some of their basic skills. Examined is one particular assignment that makes use of Probable Systems 15.

Assigned to the students would be three algebraic problems in the form of a rebus or alphametic / cryptogram. Cryptarithms are mathematical games with arithmetical operations where numbers are substituted for letters or other symbols. An Alphametic is a subtype of cryptarithm where each of the letters form words of a natural language. Perhaps the most well know example was the “Send More Money” problem in Strand magazine created by Henry Ernest Dudeney [26]. The idea was to find the unique solution to SEND + MORE = MONEY by the substitution of each letter with a unique number in the set $\{0, 1, \ldots, 9\}$.

One of the problems would be Probable Systems 15 and the other two would have solutions in some low radix, like base 9 and 10. The students’ tasks would be to test each one to figure out which radix the division problems belong to, and to identify the bpNichol poem. We would assign them a ceiling, such as not to test beyond base 10. After they applied themselves and came up with answers (or not) we would discuss the work and further applications of the skills and approaches they used.

Such an application would engage their problem solving skills, their knowledge of math, the ability to link variables such as ‘giraffe divided by sailboat equals church’ into actual literal values. This skill of being able to link a variable to value is of foremost importance in programming; it is also, in personal experience of the authors, as well as other educators[35][37], one of the hardest concepts to impart into new programmers. The concept is so elementary to the prac-
tioner that one often has a blind spot for their misuse. Often, students will come up with very creative methods to avoid using a concept of which they are unclear, which in turn leads to the phenomenon of “spaghetti code” where a programmer is liable to use a structure which is tangled with problems in flow and unnecessary statements and commands [30].

The lesson that we propose should supplement the skills workshop would be that there are times when programmers will encounter data and problems that are clearly mathematical, but from a disciplinary standpoint of the sciences, also nontraditional. Those challenges might require programmers to venture away from the typical problem solving examples of programming and investigate alien or interdisciplinary concepts in literary criticism [35] and philosophy to know what they are dealing with. On the other hand, the students would also be encouraged to consider their skills as artistic tools. By adding occasional problems in poetics to undergraduate curricula, instructors might help students make lateral connections that offer a new level of legitimacy to the arts that they had not considered.

Secondly, we found this to be an interesting problem seeing as how the authors themselves (Brown and Trowbridge) were rather blindsided by the results of Probable Systems 15. In an examination of it using a solver for alphametics no solutions were found in base 10 [21]. After publication of this result, Dr. Marc Bender of McMaster University reviewed this result and found its solution. A small bug in the way a carry was made was enough to totally obscure this result. This posed an interesting problem in that not only was the mathematics incorrect but an alternative reading of the poem was required. At first we thought bpNichol managed to trick us by providing what we believed originally to be an unsolvable problem in mathematics, and thus what he would have considered to be a “pataphysical joke, see [9] for other examples of such jokes. Dr. Bender’s review in fact demonstrated a solution, thereby compelling us to have to make other speculations about why bpNichol made his poem such that it can, and should be, solved. Our conversation with academics in engineering fields was yielding results in literary criticism while still being focused on computational methodologies.

Though this was a rather annoying and embarrassing incident for the authors, it did show us the interest in both alphametics which could be solved, and the unsolvable kind. An unsolvable alphametic is, in the poetical terms of a writing theory called ’pataphysics, a mathematical joke about variables. In his 1985 essay “The “Pata of Letter Feet, or, the English Written Character as a Medium for Poetry” bpNichol discussed the ways that poetry can be a vehicle for creating humor in scientific or quasi-scientific activities[16]. In this essay, bpNichol cited earlier writing by the philosopher Roger Shattuck, the poet Alfred Jarry, and the writer Rene Daumal. Other authors that discuss the quasi-scientific humour in ’pataphysics include one of bpNichol’s collaborators [34] Steve McCaffery [32], and the poet and literary critic Christian Bök [7][9]. ’Pataphysics is a recurring theme in the University of Windsor’s journal Rampike and in the writing of Rampike’s editor Karl Jirgens [29].

The ordinal value of a letter has been used in alphametics by both bpNichol (hope=faith [13] and a proof that the difference between prose and poetry is 1 2 [14]) and Christian Bök (word=diamond [10]). This allows an interesting method for the introduction of how a computer stores data such as characters as bits. The presentation of these alphametics and the Socratic question of how are these two words are equal forces a student to think about the issues associated with representation in a computer, ASCII characters and operations on characters as if they were integer values.

4. CASE STUDY 3 - COMPUTER ASSISTED GENERATION OF POETRY

In Virtual Muse : Experiments in Computer Poetry, poet Charles O. Hartman gives various methods for creating poems using a computer poet [28]. These techniques which he uses are rather advanced in terms of necessary programming skill. However, the techniques are accessible to the humanities as a method of explaining such concepts as trees and random generation.

A number of technical issues would have to be handled in terms of the scale of assignments given the ability of the students. In order to use concepts already gained via a first year programming class the data structures used should be limited to arrays, strings, and some simple classes. A number of the data structures, such as trees, used in the creation of automatic poetry are not examined until a second, and in some cases, an advanced class in computer science. However, the majority of these structures are preprogrammed, usually as part of the language’s standard libraries. This allows for their use without having to bog down in details as the students with experience in a first year class should be familiar with the use of abstraction of functions. This could also be avoided via the use of provided code where a student is given a set of functions which can be modified by changes to parameters. This would allow for concepts which are not covered until later classes of computer science to still be covered with only a need for the theoretical knowledge, while they still gain an appreciation for the practical application by seeing the investment of source code required to show the concept.

This would be incredibly useful for the portion of evolutionary algorithms and generative techniques where these structures are highly complicated. We would not expect, for example, a student after this class to program their own algorithm poetry creator, nonetheless we would expect them to know the concepts of how the algorithm functions, what the limitations on such an approach are, and what would be the prerequisite information required for such an algorithm to function.

A few other methods, such as Jean Lescure’s S + 7 (aka N + 7) procedure [33] are trivially able to be implemented with some knowledge of strings. In this method a set of words, usually nouns, in a given text are moved seven places down a dictionary list creating nonsensical and whimsical sentences which are still grammatically correct (usually). For example the sentence “I went to the store to buy some bread” would become after moving the nouns seven places down in a dictionary “I went to the storm to buy some breakdown”. Changes in the number of places used, as well as the dictionary, will make differences in the text produced; instead of N + 7, N + 15 would give “I went to the straitjacket to buy some breastplate”. This method is normally given in creative writing scenarios for constricted writings and as a brainstorming method for poets. Such a program would
give introductions to string methods (especially tokenizing),
string matching, and searching for which words to replace,
as well as file IO in order to read and parse the dictionary file.

Such an application is a hallmark of the ideas of Ouvroir
de Littérature Potentielle (Oulipo), a twentieth century po-
etry collective formed by French writers and mathematicians
interested in constrained writing [33]. Constrained writing
can be defined (for our purposes) as the creation of an in-
teresting work of writing which has a number of constraints
in terms of the words, letters, rhyming scheme, etc. Con-
strained writing can be as simple as a haiku or as complex as
writing a poem using all words with only one vowel in them
[8], or only the letters in their original order, from an existing
sonnet [5]. The Canadian poet Christian BAAk argues that
Oulipo sees poetry as an ars combinatoria, wherein a poem
is a selection of words from the formulaic permutations of
all words of a language, and indeed multiple languages in
some cases [9].

Another possible application created by Hartman uses the
ideas of modeling population growth in order to act as a
method of random generation for word choices [28]. The it-
 erated function of \( p_{n+1} = rp_n(1-p_n) \), where \( r \in [0, 4.0] \)
and \( p_0 \) is in \([0.0, 1.0] \), provides a interesting random function.
By mapping ranges of output to a small set of words, the
function acts as a stream of pseudo random numbers. As \( r \)
value tends towards 4, the results become more chaotic, and
even small changes of initial parameters leads to interesting
changes in the output. The same list of words produces a
wide variety of results. Using other interesting and perhaps
more complicated iterated functions would be ideal in such
an example. A reader might ask if there is there perhaps a
theme apparent in the function, or what would the ap-
pearance of themes say about the underlying values of the
function.

This is a fine introduction to students about randomness,
by looking at the differences between what a human and
computer will produce, given the same set of words and be-
ing told to make as random a poem as possible. Humans
will perhaps avoid long strings of the same word, seeing this
as more random. However, as each of the strings of words
for a computer using a random distribution is as likely as
another, there is a large number of random strings which
have a single word repetition. This is sometimes called the
Gambler’s Paradox where one believes, even though each
toss of a coin is independent, there must be a causal connec-
tion between them which will present over time. A simple
process to demonstrate this technique would be to have each
of the students provide the “most random poem” which they
can create using the words, then having the program give
an equal number of poems based on a random number gen-
erator. By counting the number of runs, consecutive uses of
the same words, of both there will be a tendency of students
to underestimate the length of a run. (Note that this pro-
cess would be best using either a small set of words and a
long text as the statistical properties would be more likely to
emerge). From this initial experimental method, the math-
ematics behind the random generation can be shown. The
runs in a string of words are a multinomial distribution, a
generalization of the binomial distribution for multiple in-
dependent mutually exclusive events. This would allow stu-
dents to then form a basis to test how random a poem gen-
erator is objectively, by analyzing how closely it conforms to
the expected outcome of a given multinomial distribution.

Hartman’s work can be used as the basis for engaging
students in crossover activities between programming and
literary criticism. For example, in humanities research, a
simple thesis to test would be that poems which are more
random are more pleasing to a sample of humans (under-
graduate students); Hartman shares this thesis [28]. This
could be tested by first letting the students generate a set
of poems using varying degrees of randomness, and then
having the class rank them, not knowing the generator re-
sponsible, using their own criteria for aesthetic success. The
criteria, being aesthetic, could be formally operationalized
or it could be something more casual.

In each of these possible classroom exercises, looking to
see if the scores given for each of the criteria graphed by
the randomness values forms a monotonic increasing func-
tion would show if a trend exists. By using Spearman’s rank
relation coefficient between the randomness value and hu-
man rankings this hypothesis could be examined further, if
there was a need for mathematical rigor.

In fact, this is a typical exercise done without comput-
ers in courses on creative writing, avant garde poetry, and
contemporary poetry. Students are encouraged to combine
words, phrases, and sounds in random or methodical pat-
terns and offered more sophisticated ways of judging the
results. What we propose is that Hartman, by claiming
that humans enjoy poetry the more random the poems be-
come, has introduced the means for a significant change to
this pedagogical model in creative writing. The tools are
now available to operationalize what makes a selection of
words random, and also the tools are available to measure
how random they are. Computing offers a different way to
focus on the selection of words rather than the words them-
selves. To a certain extent, all poetry can be examined not
by asking what words mean but why the poet chose these
particular words, even when the poet chose random words.

We think that the broader point of Hartman’s thesis is that
if a poet chooses words randomly, then poetics and comput-
ing overlap in ways that poets and programmers both find
significant.

For an excellent example of this kind of literary experi-
ment by students, we suggest reading a small, 18 page stap-
led book by Jeremy Colangelo titled The Limits of My
Language. In this book, Colangelo has collected various ex-
periments from his undergraduate writing courses at Brock
University. The Limits of My Language contains a cipher,
a formal graph, a poem based on randomized words, and
a meditation on number theory, among other experiments
[23].

More extensive examples of constrained writing and ele-
ments of randomness appear in two books by the contem-
porary Canadian poet Gregory Betts. For his 2005 book If
Language Betts took a paragraph of text written by Steve
McCaffery, and then rearranged the letters into fifty six po-
ems that are perfect anagrams of McCaffery’s original [5].
The fifty six anagrams are not exhaustive, for example num-
ber 52 is all names of famous people, and surely there are
other appropriate lists of names. Almost all of the anagrams
make syntactical sense or can easily be read with some kind
of syntactical meaning projected onto them by the reader
with minimal effort.

In Betts’ 2009 book of poetry The Others Raised in Me, he
takes Shakespeare’s “Sonnet 150” and turns it into 150 new
texts by crossing out letters, spaces, words, and punctuation [6]. This book is a particularly interesting text to test Hartman’s thesis because while some of Betts’ results are clearly legible in English, others require hard work by the reader to make sense. Depending on different criteria, the poems can be ranked from least random to most random (and it is interesting to ask where the original “Sonnet 150” by Shakespeare would appear in that order). Betts did not use a computer when he crossed letters out and that is precisely why comparing his results to students using computational techniques would be interesting.

5. POSTSECONDARY PEDAGOGY

We think there is a growing market for teaching modules that combine introductory level programming concepts and literary analysis. Our own example supposes a target audience of students and instructors in a college level course. At the college level, there is an obvious need for interdisciplinary courses that might help students improve their own literacy, or courses that might pique their interest for required context credits or electives. Humanities programs in general are taking serious interest in training graduate students to work with digital technologies. To that end, funding is being diverted by colleges in Canada and the USA, and major organizations like the National Endowment for the Humanities [36], into thesis projects and laboratories [19][38][39][45][46] dedicated to what is known as digital humanities. Digital humanities is the trendy and more accurate name for what was considered “humanities computing” in the last decade [41].

Certain disciplines in humanities are expanding with great success into the digital realm. In particular, the study of history is experiencing a kind of renaissance thanks to immersive environments, digital archives, and even non-traditional thesis projects [20]. Some English-speaking masters and doctoral programs in history that traditionally require students to have a second language (like French, Latin, etc.), now allow students with digital humanities projects to use programming language instead [27]. The market for class work in the humanities and computing is obviously growing.

On the other hand, by moving away from humanities computing and toward digital humanities, the sort of literacy we are trying to demonstrate by analyzing bpNichol’s poems and Gregory Betts’ poems is at risk of being sidelined. John Unsworth, a Professor of English at the University of Virginia’s Institute for Advanced Technology in the Humanities, has tried to answer the questions “What is Humanities Computing and What is Not?” [47]. He lists six different applications for humanities computing, but the sixth application, “Humanities Computing and Formal Expression” appears in his essay as only a little more than an honourable mention. Unsworth explains that, “There is also one other feature of knowledge representations that humanities computing authors don’t mention, because their discussion takes it for granted. That feature is the formal language in which any such representation must be expressed” [47]. Nonetheless, Unsworth promotes exactly the kind of ability to read in two different academic worlds at once that we are advocating. Unsworth summarizes this aspect of humanities computing like so:

In short, once we begin to express our understanding of, say, a literary text in a language such as XML, a formal grammar that requires us to state the rules according to which we will deploy that grammar in a text or texts, then we find that our representation of the text is subject to verification — for internal consistency, and especially for consistency with the rules we have stated [47].

This is exactly what our analysis of bpNichol’s poems has been doing so far. We have been treating bpNichol’s poems as open to interpretation using computational methods. The result is that his traditionally ambiguous and therefore mysterious poems are also being subjected to testing for the consistency and rules of computer programming and principles from applied mathematics. The potential discussions of Unsworth’s “representation of the text” and “consistency with the rules we have stated” also appears to be consistent with our suggestion for studying randomness in poetry and the choices made by poets such as Gregory Betts and student works, like Jeremy Colangelo’s poems [23].

6. OTHER PEDAGOGIES

What we also found is a diverse conversation about interdisciplinary pedagogy in secondary school classrooms. The articles written about pedagogy seem to focus on a few main professional conversations between teachers. They work toward justifying the need to combine math and science curricula with the arts. We will briefly comment on four articles, with the assumption that further research will be necessary because the meeting of computation and poetry has been long overdue in the classroom (since the 1980’s when bpNichol was working with the Apple II). Don Mainprize’s “Ouchless Poetry” [31] is about navigating the difficulties that poetry based lessons present for secondary school teachers, and a complementary article about mathematics teaching by Heinz Steinberg [40]. Shelia Tobias and Lynne S. Abel [42], focus on the need for identifying how comfortable science students are with taking courses that include humanities curricula or combining such divergent disciplines. David Donahue [24] gives us some direction for successful peer mentoring and collaborative projects from the perspective of science and humanities instructors.

Despite the different disciplinary language used in their articles, Mainprize and Steinberg [31] are essentially discussing the comfort level necessary for students to find meaningful engagement with poetry and mathematics (respectively) in the classroom. Mainprize offers suggestions for making the study of poetry as inviting to students and instructors as possible. An ouchless poetry teacher achieves this through careful consideration of how to preface poetry reading, how to construct an interesting classroom, and how to address students sympathetically. An undergraduate science student that went to highschool with an ouchless poetry teacher might therefore already be willing to read any sort of poetry in an academic setting, but not necessarily find the exercise appealing to their own aptitude. What Mainprize does not discuss deeply, and what we suspect many college instructors are concerned with, is that student interest is often geared to aptitude. We suspect that many students enter college to study computer science and engineering because they feel their natural abilities are alphanumeric and therefore not found in poetry. Mainprize’s essay is evidence that English and literature instructors are already trying to accommo-
date science and math students. What we would like to add to Mainprize’s discussion is that some students enjoy school because they are good at math, therefore English and literature classes should include teaching modules about poetry that is interesting because of its connection with applied mathematics.

The same material used in a lesson about poetry and mathematics could work the other way as well. Steinberg’s *communication and epistemological constraints* in mathematics classrooms must necessarily be difficult to break down because many students who enjoy school find that their aptitude is in poetry, which they assume means experimenting with words and not experimenting with computation [42]. We propose that with examples like the poems of bpNichol and Gregory Betts, these verbally motivated students can study mathematics in a way that suits them. The final product of their lessons can include poems instead of traditional equations, and computation can become part of literary criticism as a final product, and thereby sidestep the sort of graded schoolwork that makes some literary students allergic to computing.

Sheila Tobias and Lynne S. Abel have tested a short teaching module about the poetry of Chaucer and Wordsworth to fourteen science instructors from Cornell University, including engineers, applied mathematicians, chemists, geologists, materials scientists, and physicists. The objectives of their experiment is, in certain ways, opposite to ours. The significant difference in Tobias and Abel’s curriculum is that they chose texts that would be foreign to computational analysis instead of sympathetic. Translation of Chaucer’s Middle English is much different than translation of bpNichol’s poems into computational analysis, in that Middle English is less consistent and more foreign to twenty-first century readers of any formal disciplinary background. Tobias and Abel explain:

The choice of Chaucer and Wordsworth was partly deliberate and partly serendipitous. We wanted to choose texts that were part of the established canon, yet unfamiliar, texts we thought would be taken seriously, but might be hard and not a part of scientists’ everyday life. We wanted to choose texts in English and not introduce the language of translation between students and teacher and between students and text. We decided to read poetry because we knew we could not ask participants to read masses of material in one short week; poetry can be read carefully and studied in short segments. Also, the “poetic” view of the world is commonly opposed to the scientific view [42].

Therefore Tobias and Abel are not concerned with using poetry as a way to demonstrate computational methods, but their findings are still relevant to the curriculum we propose. They found that one of the most difficult barriers between scientific learning and humanities learning was that:

Scientists and engineers tend to pay attention to structure and hierarchical orderings. Humanists tend to pay attention to similarities and differences. ... The problem seemed to be one of sequence. The lectures and discussions appeared to be non-linear and without direction. ... Scientists routinely explain to their students why they are studying some topic and what they expect to accomplish in a particular lesson. In the poetry seminar, the teachers did not stop to explain either what was going to happen or why. They just began [42].

We believe that bpNichol’s poetry would help to bridge the gap between hierarchical scientific learning and the open-endedness of humanities learning. bpNichol’s *Probable Systems 15* can be approached in the hierarchical, problem based fashion of a science or engineering lecture. However students who are introduced to his work through the hierarchical mode of science lectures might also be interested to know that he has much more traditional poetry, such as *The Martyrology* [17], and therefore academically discussing bpNichol’s poetics requires the ability to think in both modes. We hope that teaching bpNichol in this way would also dispel the persistent myth in science and humanities settings that authors and students with aptitude in one domain cannot operate successfully in the other. Tobias and Abel argue that, despite this challenge, they have achieved the same goal, but have done so by challenging students to step away from their comfort zones, whereas we would like to see this proven by uniting scientific and humanities pedagogies with a single comprehensive poet.

Creating a curriculum that demonstrates concepts in computing using poetry is not only for the benefit of students, but also for the challenge it presents to instructors. Our own disciplinary backgrounds are significantly different, Brown in science and math, Trowbridge in humanities and social science. We encourage the design and teaching of an interdisciplinary curriculum using multidisciplinary teaching teams. Our experience is that explaining to each other how we read poems is difficult and rewarding theoretical work, and explaining how we understand each other to students, peer reviewers, and conference audiences presents another set of important and rewarding challenges.

David Donahue began this conversation in his article “Reading across the Great Divide: English and Math Teachers Apprentice One Another as Readers and Disciplinary Insiders” by discussing how secondary school teachers can learn from reading each other’s texts and discussing methods for teaching them. Donahue identified five common ways in which the teachers who participated in his study benefited professionally: by identifying disciplinary similarities in “searching for truth in math and literature” [24]; by "exploring social justice in math and the humanities" through recurring themes that appeared in texts like W. E. B. DuBois’ *The Souls of Black Folk* [25] and Edwin Abbott’s *Flatland* [1]; by having “metacognitive conversations” about “their strategies for making meaning, their tools for identifying and fixing problems, and their motivation for sticking with texts that proved difficult or initially uninteresting”; by “preparing subject area teachers as teachers of reading”; and by “breaking down subject area divisions” [24]. Donahue’s results are interesting and suggest tantalizing potential pedagogical opportunities for instructors that are willing work in curriculum design and teaching teams in an undergraduate teaching and learning environment.

7. CONCLUSIONS

The idea of using a computer as a method for creation of poetry is far from being new, however, we have not been
effective as educators providing students with a motivation for using the skills provided in an introductory programming class. The majority of science degree requirements have an element of context courses, as well as the new emerging field of humanities computing and digital humanities, requiring growing numbers of students to develop skills in the humanities as well as appreciation of the discipline of computer science. In order to better provide for such an education, the choice of course materials that demonstrate areas of reflection in students’ own disciplines is prudent. Further, even students who are seeking a degree in one of the sciences can gain from a more liberal education showing how they can apply their skills to a variety of problems for which the application of computers is not normally seen by the public. A practitioner of the *ars technica* should not be bound however by these preconceived notions of what a computer is allowed to do and should be freed to produce interesting and creative works of their own.

In future we plan on continuing our explorations into the methods of using computers to assist in the creation, analysis, conservation, and appreciation of the poet’s art. We have thus far primarily looked at the work of bpNichol for our analysis, however, there is a number of works by other Canadian and world poets which would benefit from such analysis. In terms of creation we will look into making concrete poetry, poetry where not only the written word but the arrangement of the words typographically gives meaning, via a computer. This form is uniquely suited to automatic and human assisted generation as the form relies on the typographical arrangement of the work, thus an algorithmic approach can be evaluated based on metrics gleaned from an objective view of matching a proposed form, rather than relying on a subjective method of evaluation, such as if a human enjoyed the poem.

The separations which hold academic disciplines apart, even in multidisciplinary faculties, are starting to fade away due to the requirements of technology and the available glut of information provided to practitioners, students, and educators. We have not done well thus far in changing the methods of education to account for the need for a more multidisciplinary and holistic approach to learning. The examples of computer science and humanities can be brought together for the benefit of students forms a current case study which is in no way exhaustive of the techniques and lessons which can and should exist. They are only a short overview, and in many ways a call to action for other educators in computer science and in the humanities to work in order to allow for an emerging practice to be codified through the sharing of teaching and learning experiences and course syllabi.

8. ACKNOWLEDGMENTS

The first author wishes to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for their support of this work. We would like to thank Dr. Marc Bender of McMaster University for his finding of some corrections to our previous work which have now aided in a better understanding of the 'pataphysical poetry of bpNichol. Also, we would like to thank MA candidate Spencer Roberts of Brock University’s Centre for Digital Humanities for help with Omeka and pedagogical definitions.

9. REFERENCES


[38] Roy Rosenzweig Center for History and New Media. George mason university. http://omeka.org/.


