Introducing Students to the Digital Humanities

Samantha O'Dell
Washington and Lee University – Computer Science
18009 E Fall Dr.
Independence, MO 64055
(816) 550-8041
odells15@mail.wlu.edu

Gabrielle Tremo
Washington and Lee University – Computer Science
1019 N Green Dr.
Newport News, VA 23602
(757) 256-8088
tremog15@mail.wlu.edu

1. WHY DIGITAL HUMANITIES?

Digital humanities is a fast-growing hybrid discipline of quantitative scientific analysis and more subjective disciplines. Numeric data is not the first thing that comes to mind when one thinks about literature or art, but it greatly deepens one's understanding of all the humanities and opens new avenues for analysis and learning.

Our goal was to help prepare a curriculum for a class that will be taught at Washington and Lee University during spring term 2014. The course, which will be taught by our advisors this summer, Professors Sara Sprenkle and Paul Jungman, will allow students to become acquainted with the digital humanities and explore this fascinating intersection between multiple disciplines.

2. BACKGROUND AND RELATED WORK

What is digital humanities? Answering this question is difficult because there are hundreds of valid answers. Digital humanities is not "technology for technology's sake" [1] but rather the intentional application of computing's power to other disciplines.

The class, INTR 203: You Say You Want a Revolution: An Introduction to the Digital Humanities, will be a focused, four-week liberal arts course introducing students to the digital humanities. The students will be enrolled exclusively in this class during the spring term, meaning that class time each week is higher than for a 12- or 14-week course. The students will have 8 hours of lecture and 6 hours of lab each week; however, they will not have to split out-of-class time between other courses. The course is interdisciplinary and open to all majors.

One of our main challenges was finding resources accessible to students with little to no programming background, particularly given the course's fast pace. To our knowledge, there is no prior published work on teaching a similar course for the broad undergraduate audience at a liberal arts college.

3. OUR APPROACH

We evaluated multiple tools for their suitability to this course, always keeping in mind the intended audience and the level of programming ability the students would have.

3.1 Ease of Learning

Our primary evaluation method was based on whether an average student with limited computer science familiarity could easily learn how to use a tool and get meaningful data from it quickly. Several tools were discarded because the learning curve was simply too steep for students in a four-week course.

3.2 Visualization Tools

We prioritized tools that created visualizations. Visualizations are excellent for helping students to contextualize data. They also make the data received from other tools more approachable for the students. Interpreting data from graphs or charts is often more intuitive for students unused to viewing data output in a terminal.

3.3 Guidelines and Module Creation

After deciding which tools were best suited to our purposes, we then created guidelines for the students to follow to maximize their ability to mine data. We also created customized modules for those tools that needed it, eliminating the need for students to spend time writing start-up code.

4. SELECTED TOOLS

The tools we selected to use in the class excel in either producing data or creating visualizations and some excel at both.

4.1 Gensim

Gensim [2,3] is a Python library for language analysis of a large corpus, or group of documents. It can handle thousands of documents at once and initialization of a large (15MB) corpus takes about an hour, after which analysis can be performed. The initialization process splits an incoming text file where different documents are separated by line breaks into a corpus of multiple documents. In addition, every letter is lower-cased, some punctuation is removed, stop words are removed, and unique words are left out by default. Punctuation that is removed includes periods, semi-colons, exclamation points, etc. The point is to prevent duplicate listings of words. There should be no distinction between "said" and "said.", for instance. Stop words are common words that are non-essential, such as "that," "she," etc. One .MM (MatrixMarket) file is saved and one Gensim-specific .dict file is also saved, both of which are needed for data analysis. The .MM files contains word IDs as well as the location of each word within the corpus and their frequencies. The most useful Gensim attributes include word count for each unique word in a corpus and topic analysis.

Gensim word counts reveal important information about a corpus. Here is a selection of some of the most frequent words in Washington and Lee University's News Blog [4]: {school: 4601, press: 4599, releases: 4506, &: 3874, search: 3819, alumni: 3807} The unique "&" is high-scoring, as well as common news topics, such as the school itself and our alumni. Word count data is particularly useful when applied to literary documents; authors typically strive to reuse words as infrequently as possible and often wish to repeat words intentionally. Having a list of exactly how many times every word appears throughout their text can help speed up the editing process.

Latent semantic analysis (LSA) allows the computer to scan a corpus and return topics. These topics are formed by groups of words that the computer identifies as part of a particular subject. An example LSA topic from W&L's news blog is: {site, hall, university, bell, students, 2013, robinson, education, artifacts, july}. This topic seems to be extracted from several articles on the new archeological dig site that was uncovered under Robinson hall in 2013.

4.2 NLTK: Natural Language Toolkit

The **Natural Language Toolkit (NLTK)** [5,6] is a tool that helps computers process and understand natural language. Computational languages differ from natural languages, like English, in that they are *unambiguous*. Each word in Python has a specified meaning and function. Conversely, words in English can have one or more meanings, and though some words may be spelled the same they can have separate meanings (ex. "convert," a verb; "convert," a noun). Natural language processors support digital humanities by enabling computers to read and mine text for data. They analyze text, including literature, poetry, scientific articles, newspapers, song lyrics, scripture, etc. NLTK stands out among other natural language tools because of its great range of applications. NLTK can parse through a text and either count every time a letter appears or go as far as providing a source of "understanding" to text and even having the computer respond!

We used NLTK in two ways: part of speech tagging (*pos-tagging)* and sentiment analysis. NLTK has the ability to tag individual words using a corpus, word usage, and context. In pos-tagging, we used this tagging ability to go through and tag each element with a part of speech. We used these tags to then create MadLibs [7]. We also were able to find libraries to support sentiment analysis. Using a similar parse and tag method, this NLTK tool [8,9] determined if a word or an entire text had a negative, positive, or neutral connotation. We tested the limits and accuracies of this tool by analyzing sections of literature, poetry, song lyrics, and even RateMyProfessor.com reviews. This experiment showed that, in emotive texts (such as reviews, song lyrics, and sections of dialogue from books), the sentiment analysis was very accurate. But in sections of literature or poetry where descriptions outweighed emotive phrases, the text selection was more likely to come out neutral.

4.3 Voyant

Voyant [10] is a text analysis tool that takes user-uploaded text and returns data and visualization from the data it takes from the text. Voyant is the successor to TAPoRware [11] that meshes the functions of text analysis and visualization creation that most tools do separately. User can input text, attach a document, or upload an entire corpus into the tool. After the text has been loaded, Voyant shows numerical data, such as how many unique words are in the text, as well as makes inferences, such as how words are connected. It then translates them into visualizations that are easy for the human eye to understand, like charts, graphs, and Cirrus clouds. Cirrus clouds are colorful word clouds of all of the most used words in the passage. The more often a word is used in the text, the bigger it is in the cloud. The "summary" function shows frequency and uniqueness of words. The "trends" function creates graphs with how often and where words are used in the document. Voyant can also collapse the search term and track synonyms when needed. There are also many tools that Voyant supplies to help you with more specific needs. One of these tools is Links which creates word networks based on words that are used in conjunction multiple times. All of the data is fully exportable and embeddable, making it easy to take the generated data wherever it needs to go—either in a paper or in a WordPress site.

4.4 Shiva

SHIVA is a web-based tool that combines the power of previous visualization tools like Google Maps and MIT's Timeline into one uniform application. SHIVA is both powerful and easy to use. All of its visualizations are created using Google spreadsheets. While there are many tools within SHIVA, we utilized three of them in our research. SHIVA Charts creates charts and graphs. SHIVA Maps functions like Google Maps, where markers can be placed with descriptions. Layers can also be added to show changes over time or even during two events. SHIVA Timelines juxtapose pictures, events, and descriptions together presenting them in non-text-heavy way. SHIVA visualizations are fully customizable and interactive. Additionally, SHIVA creations can be embedded into any website that supports HTML iFrames. SHIVA automatically updates visualizations when their Google Doc source is changed.

4.5 Other Work and Dead Ends

At the beginning of the summer, we refreshed our Python skills by creating a small program to format a given text file. This script, called PrettyPrint.py, was initially capable of printing out a given text with lines as close to the same length as possible. We then added several methods to increase formatting options. The methods added were random, prettyPunct (creates new lines at any punctuation mark), and setWords (creates a new line after every X number of words). We also added command line arguments to the program, allowing it to be run straight from the

terminal. PrettyPrint.py is a great example of how format greatly influences the meaning of a given text.

In addition to finding and developing guidelines for useful tools, we also found a few tools that were not suited to our goals. One of these tools was Gephi, an open-source, Python-compatible visualization tool. We chose it because it could handle a lot of data and chart it effectively. However, we discovered that while it did have a Jython console within the program (a Python module with a Java wrapper so that the Python is compatible with Java-based Gephi), it did not yet have the capability to automatically read in data and convert it to graphs. The program also had some errors in its GUI; some of the sliding bars would occasionally get "stuck" in a particular position and would no longer respond to mouse clicks. Between the difficulties in working with the program and its inability to produce the kind of results we wanted, we moved on to other tools.

Another discarded tool, CATMA, was powerful but was ultimately deemed too complex for students' use. CATMA, a web application, required that steps be taken in a very particular order and would produce no data if the user erred during the data-input process. It was similar in many ways to a much easier to use tool, Voyant, making it difficult to work with as well as redundant. It was quickly abandoned.

SEASR.org, another web application, was one tool we tried that was completely nonfunctional. After making sure that user-error was not at fault, we abandoned this tool and moved on to other things.

Also, getting large, interesting data sets to run Gensim scripts on was difficult at first. We had hoped to be able to use old issues of Washington and Lee's student newspaper, *The Ringtum-Phi*, but the digital files of the paper were poorly OCR-ed, making getting the text from the PDF files a nightmare. Instead of creating a corpus out of *Ringtum-Phi* articles, we pulled data from Washington and Lee's website and news blog as well as from websites like ESPN and CNN. The stripping of "junk" (mostly HTML/CSS code) from this data was time consuming and done by scripts written by Professor Sprenkle, but in the end we got clean data that produced intriguing results.

5. CONCLUSION

Our research this summer resulted in an interesting set of tools to be used in the INTR 203. Our work will aid these students in maximizing their 4-week spring term and allowed Professors Sprenkle and Youngman to understand what it would be like for students to use these tools. Seeing how valuable the combinations of these two disciplines can be was particularly rewarding for us as double-majors in humanity subjects. Overall, we are proud of the work we have done this summer.

6. REFERENCES

[1] Spiro, Lisa. 2011. *Getting Started in Digital Humanities*. Journal of Digital Humanities. George Mason University.

[2] Řehůřek, R. and Sojka, P. 2010. Software framework for topic modelling with large corpora. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks.* (May 2010), 45-50. http://is.muni.cz/publication/884893/en.

[3] Řehůřek, R. and Sojka, P. 2010. *GenSim.* http://radimrehurek.com/gensim/.

[4] Washington and Lee. 2013. News & Media. http://news.blogs.wlu.edu/

[5] Bird, S., Loper, E., and Klein, Ewan. 2009. *NLTK*. http://nltk.org/api/nltk.html.

[6] Bird, S., Loper, E., and Klein, Ewan. 2009. *Natural Language Processing with Python.* O'Reilly Media Inc.

[7] GitHub. 2013. Make madlibs using NLTK (Natural Language Tool Kit). https://github.com/heyhuyen/madlibs

[8] Django Project. 2010. Sentiment analysis with python NLTK text classification. http://text-processing.com/demo/sentiment/

[9] Perkins, J. 2010. Sentiment analysis. http://text-processing.com/docs/sentiment.html

[10] Sinclair, S. and Rockwell, G. 2013. *Voyant.* http://voyant-tools.org/.

[11] McMaster University and the University of Alberta. 2007. *TAPoRware Project.* http://taporware.ualberta.ca.