Visualizing Poetry: Creating Tools for Critical Analysis

Luis Meneses and Richard Furuta

Introduction

Current debates over “distant reading” (Moretti) seem to imply that digital tools are suited to nothing else. However, we can and should develop more digital tools and methods to read closely and carefully analyze literary documents. Moreover, these analyses can be visualized. In distant reading as well as close, visualization tools can be used to “amplify cognition” (Ware). Although too little work has been done in poetry visualization, it can be used to emphasize the structure of the narrative, the organization of the poem, the language elements, and the metaphors employed (Chaturvedi). We have developed a set of visualization tools that aim to help scholars carry out the critical analysis of poetry. More specifically, the purpose of the visualization tools is to help synthesize and bring forward specific key elements found in poems. Our visualization tools were developed using open–source programming languages; we used poems encoded in TEI from *The Poetess Archive* to create the visualizations.

The tools we developed purposely bring forward specific elements of poetry through graphical representations by carefully layering annotations (Tufte) based on literary criticism, thus creating new methods to analyze and create poetry. Additionally, our tools place special emphasis in viewing the poems from different perspectives (Meneses, Monroy, Furuta, et al.) and visualizing the textual representation of the poetic texts in their formal structure (Audenaert et al.). These visualizations will be made available to scholars of all sorts, whether they are theoretically or historically engaged in the poetic texts that we use. In the following sections we will review the previous research related to poetry visualization, describe the rationale behind the tools we have developed and discuss their characteristics. Finally we will provide conclusions and recommendations for future work.

Previous Work

Little work has been done in creating tools to visualize and analyze poetry. Most of the work that has been has focused on creating new forms of expression using deformation and transformation techniques. These techniques are focused on creating new art forms using poems as an input source. Examples of these transformations include Visualizing Text by Diana Lange (Lange), Poetry on the Road (Schaffors, Müller and Pfeffer), Text Universe (Rapati) and Ira Greenberg's Syntactic Arthropod (Greenberg). However, we have found that in many cases the transformations that are applied to a poem do not provide a rendering of the original text. In these cases, the visualizations are successful in transforming the text but ultimately hinder access to the original material. To avoid this problem, we have developed,visualizations that transform the original poem to create a new representation, but keep the poem visible and available for study and analysis.

The visualizations in Understanding Shakespeare (Thiel) provide an overview of different plays by displaying the most commonly used words grouped for each character. In this visualization, blocks of text represent scenes that are scaled depending on their number of terms. Additionally, the speeches of major characters are highlighted and characters are ordered by their order of appearance in the plays. Although the results of Understanding Shakespeare are displayed on the web, a final iteration of the project was a printed book version that provides a different way of reading the plays.

Frameworks have been built with the specific purpose of analyzing poetry. Myopia is a framework that enables its users to analyze poetry by visually emphasizing the structure of the narrative, the organization of the poems, the language elements, and the metaphors employed (Chaturvedi). Myopia was built in Python and used open source libraries for displaying multidimensional graphics, which could be manipulated using an intuitive user interface. Additionally, The Mandala Browser is a rich prospect interface that can be used to analyze the different relationships occurring within a document corpus. Because of its flexibility, the Mandala Browser is widely used within the humanities (Gainor et al.; Ruecker, Radzikowska and Sinclair).

Visualizing Poetry

Our visualization tools were developed using Processing ("Processing.org"), an open-source programming language designed specifically for visual artists and designers with the purpose of teaching the fundamentals of programming. Casey Reas and Ben Fry started Processing as a project in 2001. However, Processing has evolved through the years and it is used beyond its original pedagogical scope. Nowadays, Processing is also widely used by visual artists and designers to create new ways of displaying of data, animations and digital artworks.

Using Processing has some obvious advantages related to the execution of the source code and its portability. Processing is an extension of the Java Imaging Library, so Processing files can be executed as Java programs. Additionally, the Processing language is also available as a JavaScript port ("Processing.js"), making it possible to run our visualizations in a modern web browser. We believe that the main advantage derived from using Processing is the simplification of the development process by encapsulating the complex data structures into simpler objects and methods. However, Processing is not a perfect programming solution. For example, parsing complex XML documents and creating complex user–interactions can be difficult. Because of this, we also relied on external libraries and other programming languages to achieve our end results.

In this paper we will describe the 4 different tools we have developed to visualize and create poetry: Graphwave, SentimentGraph, SentimentWheel and Ambiances. These visualization tools were created independently, but they all share similar characteristics. In this section we will elaborate on their characteristics, advantages, and specific uses.

Graphwave

Our initial idea for this visualization was to create a wave-like pattern using geometrical shapes that would allow us to highlight certain terms within a given poem. In this specific case, we chose to highlight the unique terms that could be found within a poem. To create this visualization, we used a process that borrows concepts from the term “concordance” in Picasso’s poetry (Meneses, Monroy, Mallen, et al.). This process can be described in four steps. First, we identified the unique terms in a selected poem. Second, each unique term was placed in a node in a Red Black Tree. A Red Black Tree is a binary data structure where operations can be performed *O(lg n)* time, where *n* is the number of nodes or terms. Third, each term occurrence along with its metadata was placed in a linked list attached to each unique node in the tree. Finally, the poem transcription and its binary tree are traversed simultaneously, drawing a square in the canvas for the terms in the poem. The vertical position of each square is calculated based on the frequency of the term it represents, and its color is more saturated if the term is used less frequently. Thus, unique terms exhibit a more saturated color and are placed higher along the canvas while common terms have a more transparent surrogate and have a relative lower position.

The resulting visualization creates a colorful wave-resembling pattern, where terms are easily identifiable and can be referenced to the different parts of the poem. Additionally, the visualization keeps the poem readable by following the pattern from left to right. Longer poems are split into separate vertical waves. Figure 1 shows a visualization of "When I Have Fears" by John Keats. More importantly, GraphWave allows scholars to create new readings by focusing on a specific aspect from a poem. For example, the visualization can be modified to highlight terms belonging to places, time periods, figures of speech, etc. Certain words are used in poems because of their aesthetic qualities and their semantic content. Therefore, unusual words with less frequency can be used to the poet's advantage when trying to convey a message. In "When I Have Fears", John Keats expresses fear of dying before finishing his life's work and fear that he might never experience love again. In the specific case of this poem, giving more importance to terms by placing them higher on the vertical axis highlights these details.

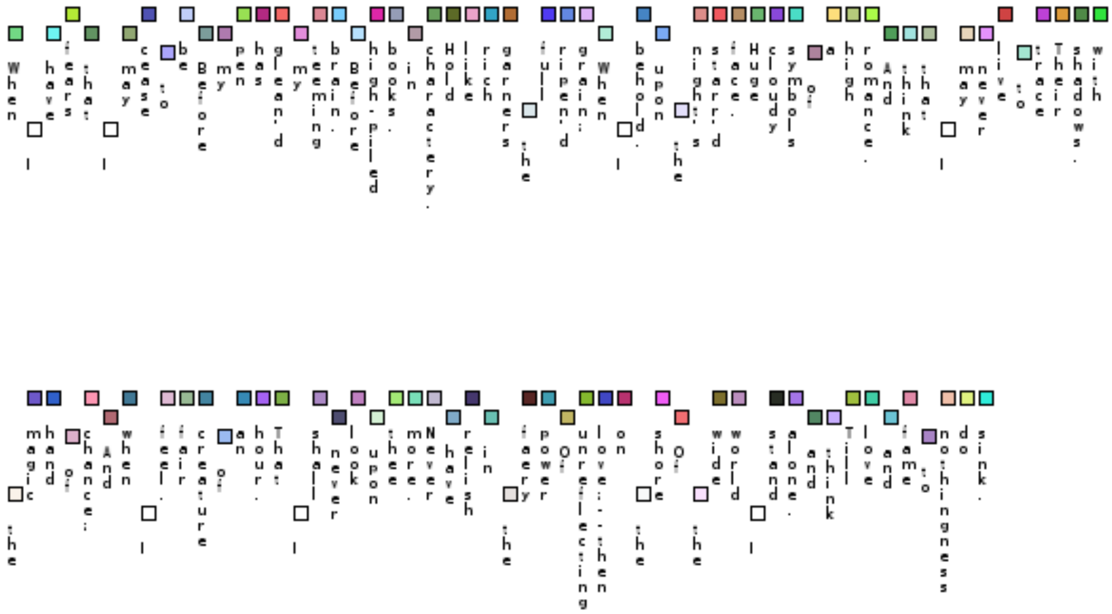


Figure 1: Viewing “When I Have Fears” through GraphWave.

SentimentGraph and SentimentWheel

For our next visualizations, we analyzed how specific elements of the language and their inherent connotations are used in a poem. In the specific case of SentimentGraph, we analyzed the sentiments behind each term in a poem. We created this visualization using a two-step process. First, we used a Python script to query the Sentiment Analysis API ("Sentiment Analysis - text-processing.com API v1.0 documentation"), which labeled each term as positive, neutral or negative. Second, we used the sentiments for each term and we correlated them against the poem transcription. Terms with a negative connotation are represented as red squares, whereas neutral terms are gray and positive are blue. Additionally, we included information regarding line numbers and a transcription of the poem. However, we found some disadvantages that are derived from the procedure used for this visualization. First, the visualization algorithm ignores the surrounding context of each term; and second, the sentiments extracted from the web API reflect only the most common use of each word, which might not reflect the emotion that the author wanted to express. Figure 3 shows the visual result after feeding “Romance” by Edgar Allan Poe to the visualizing algorithm. We addressed some of the shortcoming of this visualization in SentimentWheel.

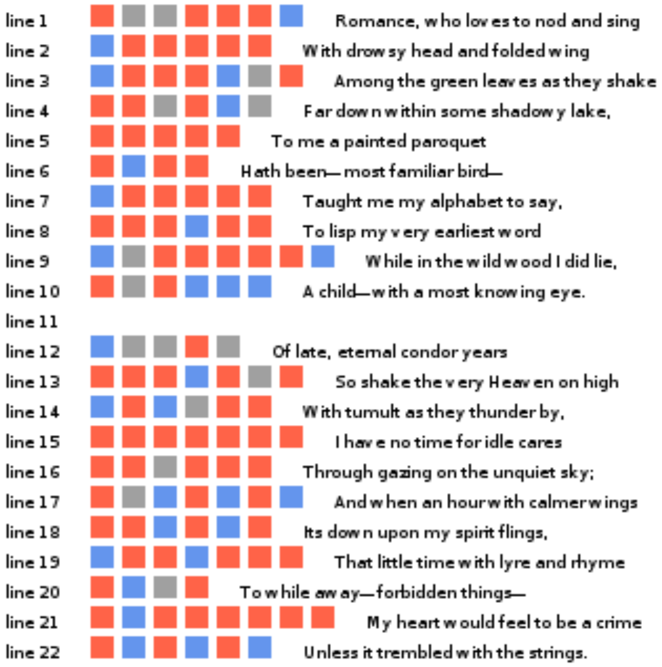


Figure 2: Viewing “Romance” through SentimentGraph.

The visualization behind SentimentWheel borrows some concepts from SentimentGraph. Creating this visualization also involves a similar process. First, we used a Python script to analyze the sentiments expressed. This time, we analyzed the sentiments expressed by each line of the poem. As before, the sentiments for each line were extracted using the Sentiment Analisis API. Second, we ran the sentiments we obtained with the API along the lines and individual terms form each poem. For this visualization, we decided to arrange the lines of a poem as a wheel. The line sentiments are expressed as consecutive circles: red for negative, gray for neutral and blue for positive feelings. The main advantage of this visualization is that the representation of the sentiments in each line correspond more accurately to the emotion that the author wanted to express. Another advantage is that the poem is readable by following the terms clockwise. Figure 3 shows the resulting visualization of “When I Have Fears” by John Keats. The contribution of SentimentGraph and SentimentWheel is producing new readings of poems by embedding a visual representation of the author’s perceived emotion during certain parts or passages of a poem. However, these visualizations can also be extended to accommodate the different representations of certain semantic passages. For example, the visualizations can be modified to represent important places, time periods or characters.

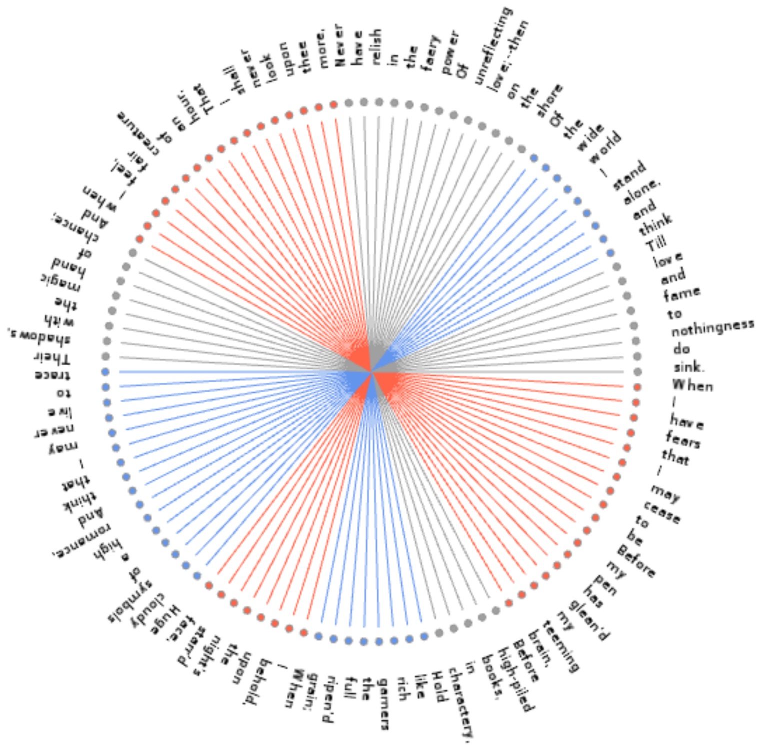


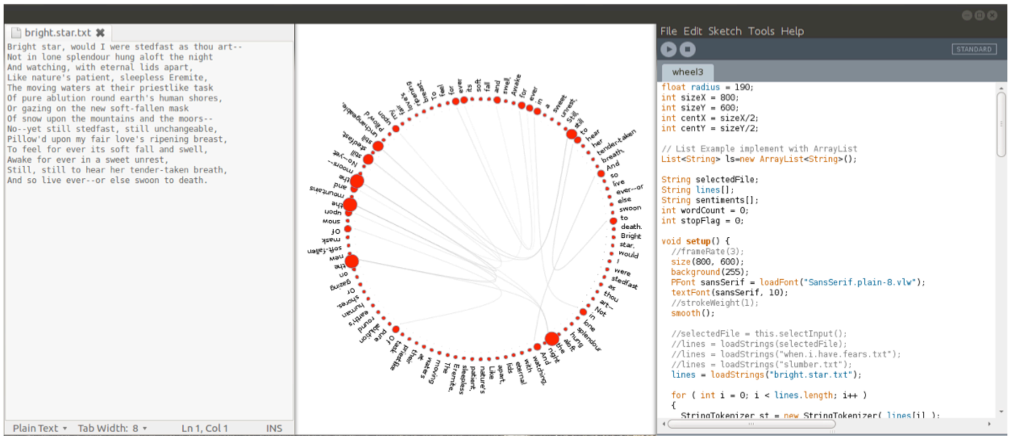
Figure 3: Viewing “When I Have Fears” through SentimentWheel.

Ambiances

For Ambiances we took a different approach. Our approach was based on the premise that the current tools used to visualize poetry have three important characteristics. First, the visualizations are created after the poem has been published. Second, the scholars who create and use the visualization usually do not have any relationship to the author who created the original literary work. And third, the visualizations are a direct consequence of the transformations applied to the original text. Taken together, the three characteristics mean that visualization tools serve their purpose when highlighting certain passages in a poem, but do not have any effect or significant influence on the author as the poem was written beforehand.

Ambiances, our interactive framework to write and visualize poetry, challenges these three characteristics and changes how poetry is visualized (Meneses, Furuta and Mandell). The main goal of the Ambiances framework is to provide a tool that affords a symbiotic relationship between writing and visualizing a poem. In our framework, the process of writing a new poem influences its resulting visualization and the visualization also affects the process of writing.

The first prototype of Ambiances was composed of three different areas or "environments": a text editor where an author composes the poem, a minimalist-programming environment optimized for writing Processing code, and an environment where the resulting visualization was displayed. Figure 4 shows a screen capture of an early prototype for Ambiances.



```
Bright star, would I were stedfast as thou art--
Not in lone splendour hung aloft the night
And watching, with eternal lids apart,
Like nature's patient, sleepless Eremite,
The moving waters at their priestlike task
Of pure ablution round earth's human shores,
Or gazing on the new soft-fallen mask
Of snow upon the mountains and the moors--
No--yet still stedfast, still unchangeable,
Pillow'd upon my fair love's ripening breast,
To feel for ever its soft fall and swell,
Awake for ever in a sweet unrest,
Still, still to hear her tender-taken breath,
And so live ever--or else swoon to death.
```

```
float radius = 190;
int sizeX = 800;
int sizeY = 600;
int centX = sizeX/2;
int centY = sizeY/2;

// List Example implement with ArrayList
List<String> ls=new ArrayList<String>();

String selectedFile;
String lines[];
String sentiments[];
int wordCount = 0;
int stopFlag = 0;

void setup() {
  //frameRate(3);
  size(800, 600);
  background(255);
  PFont sansSerif = loadFont("SansSerif.plain-8.vlw");
  textFont(sansSerif, 10);
  //strokeWeight(1);
  smooth();

  //selectedFile = this.selectInput();
  //lines = loadStrings(selectedFile);
  //lines = loadStrings("when.i.have.fears.txt");
  //lines = loadStrings("slumber.txt");
  lines = loadStrings("bright.star.txt");

  for ( int i = 0; i < lines.length; i++ )
  {
    StringTokenizer st = new StringTokenizer( lines[i] );
```

Figure 4: Screen capture of an early prototype for Ambiances.

We soon realized that allowing the use of a programming language in one of the environments created a steep learning curve and introduced some overhead for the authors. To solve this problem we eliminated the programming environment and introduced a Microsoft Kinect sensor in its place. A Microsoft Kinect is a motion sensing input device originally created as a peripheral for the Xbox 360 game console. When used in conjunction with open-source libraries, a Kinect sensor enables users to interact with computers through a natural user interface using hand gestures and body postures. Figure 5 shows the interaction diagram for Ambiances.

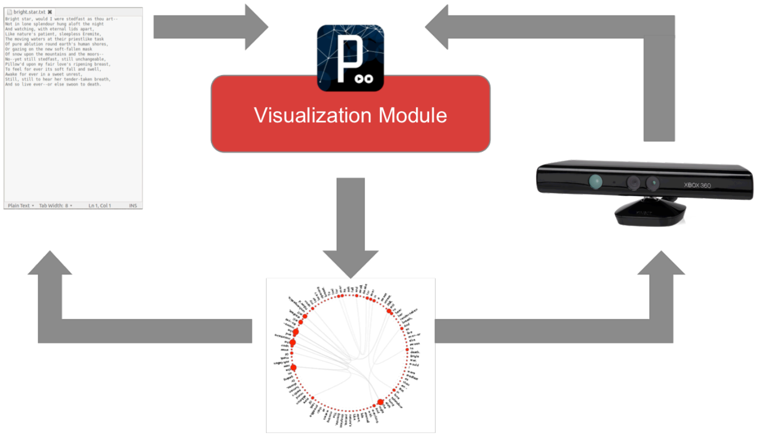


Figure 5: Interaction diagram for Ambiances.

In the current prototype of Ambiances, the layout of the environments allows users to collaborate synchronously: authoring the poem and the visualization at the same time. Additionally, the interface encourages the authors to receive instant feedback through the visualization. Given that the authors cannot critique each other directly and can communicate through the visualization, Ambiances provides an unobtrusive way of writing poetry collaboratively that encourages unexpected interactions. For example: in the specific case where the visual elements are developing in syncopated opposition, we believe that the visualization and the interactions will provide hints that will allow the author of the poem to modify certain figures of speech accordingly. Figure 6 shows the resulting visualization obtained during the creation of a poem through Ambiances.

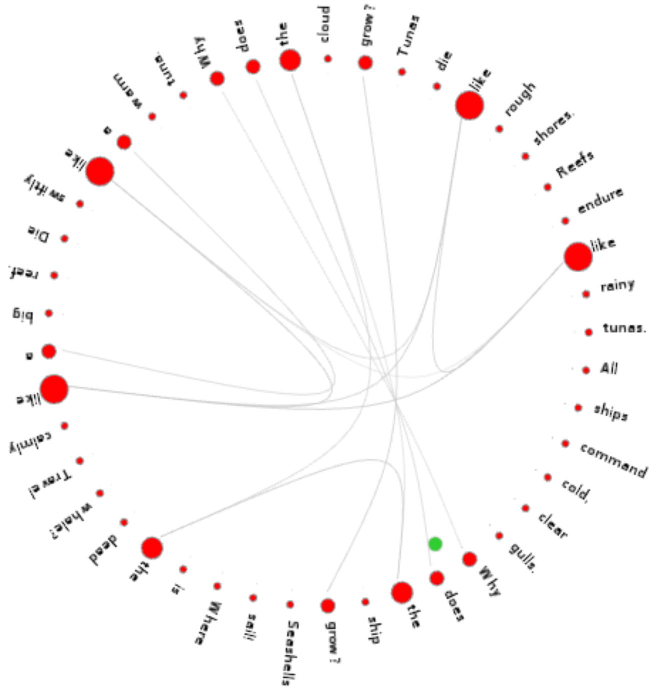


Figure 6: Visualizing the creation of a poem through Ambiances.

The contribution of Ambiances is introducing a framework for writing and visualizing poetry collaboratively. Ambiances also introduces authorship issues that should be taken into consideration. From our point of view, the users involved in the creative process are considered authors as they contribute equally towards the poem and the visualization, which are both considered as the overall end result. Additionally, this framework introduces new creative methods in the humanities for artifacts that can be composed of different artworks. We believe that a symbiotic relationship is established when using Ambiances where the poem cannot be separated from the visualization. Finally, the creative process in Ambiances can also be considered a performance if we document and record the procedure that was used and the interactions that occurred during a specific session.

Conclusions and Future Work

We believe that the visualization tools we have developed allow different approaches and methods to close read documents, which open up new possibilities for their analysis. In a scholarly environment, comparing documents in a collection is a common undertaking. However, this comparison is limited to two documents. In our case, we can compare different documents by using multiple instances of the interface. In addition, poems can be analyzed from different perspectives by using different visualizations from different poems. We believe that through these visualizations scholars can carry out new forms of analysis, explore new theories, and discover hidden relationships among different poems and authors.

Additionally, developing Ambiances left us with questions that we still need to answer. One of them is if Ambiances is a general tool or if the authors' specific genres, writing styles, languages, and cultural backgrounds influence its use. For this purpose, we are gathering a diverse group of authors to participate in a user study. The feedback that we will collect from the authors and their interactions will be included in the new iterations of the system.

We must also address questions regarding capturing, storing and replicating the end result in the prototype for Ambiances. These become complicated issues, since we consider the end result to include the poem and its revisions, the interactions between the authors and the multiple resulting visualizations. Most text editors can store the multiple revisions made to a text. However, we must devise and implement mechanisms to store the interactions with the sensors and synchronize them with the visualizations. Our aim is to capture and recreate the performance that the authors were involved with when creating a poem.

In this paper, we have described approaches that use visualization algorithms and techniques to help scholars carry out the critical analysis of poetry. We believe that we achieved our goal by synthesizing and highlighting specific elements from selected poems. However, there are research possibilities open in this area. We summarize our future work with three points. First, we will investigate if integrating specific annotations from TEI markup into our visualizations helps scholars during their research. Second, we will strive to create a system that will enable scholars and researchers to annotate and share their resulting visualizations. Finally, we will attempt to create methods that will allow scholars to store their visualizations and retrieve them for later use.

References

The Poetess Archive. Web. 9/8/2014.
"Processing.js." Web. 8/22/2012.
"Processing.org." Web. 4/2/2012 2012.
"Sentiment Analysis - text-processing.com API v1.0 documentation." Web. 8/22/2012.

Audenaert, Neal , et al. "Viewing Texts: An Art-Centered Representation of Picasso's Writings." *Digital Humanities 2007*. 2007. Print.

Chaturvedi, Manish "Visualization of TEI Encoded Texts in Support of Close Reading." Miami University, 2011. Print.

Gainor, Rhiannon, et al. "A Mandala Browser User Study: Visualizing XML Versions of Shakespeare's Plays." *Visible Language* 43.1 (2009): 60-85. Print.

Greenberg, Ira. "syntactic_arthropod: Built with Processing." Web. 8/22/2012.

Lange, Diana. "Visualizing text - OpenProcessing." Web.

Lentricchia, Frank, and Andrew DuBois. *Close reading : the reader*. Durham N.C.: Duke University Press, 2003. Print.

Meneses, Luis, Richard Furuta, and Laura Mandell. "Ambiances: A Framework to Write and Visualize Poetry." *Digital Humanities 2013*. 2013. Print.

Meneses, Luis, et al. "Computational Approaches to a Catalogue Raisonné of Pablo Picasso's Works." *Interdisciplinary Journal for Germanic Linguistics and Semiotic Analysis* (2011). Print.

Meneses, Luis, et al. "Picasso's Poetry: The Case of a Bilingual Concordance." *Digital Humanities 2008*. 2008. Print.

Moretti, Franco. *Distant Reading.* London: Verso, 2013. Print.

---. *Graphs, Maps, Trees: Abstract Models for a Literary History*. London, New York: Verso, 2005. Print.

Rapati, Tiemen "Text Universe." Web. 8/22/2012.

Ruecker, Stan, Milena Radzikowska, and Stéfan Sinclair. *Visual Interface Design for Digital Cultural Heritage*. Surrey, United Kingdom: Ashgate Publishing, 2011. Print.

Schaffors, Andrea, Boris Müller, and Florian Pfeffer. "esono.com - Poetry on the Road 2006." Web. 8/22/2012.

Thiel, Stephan. "Understanding Shakespeare." Web. 8/22/2012.

Tufte, Edward. *Envisioning Information*. Cheshire, CT: Graphics Press, 1990. Print.

Ware, Colin. *Information Visualization: Perception for Design*. San Francisco: Morgan Kaufman, 2004.