Emily: A Tool for Visual Poetry Analysis

Nitin Madnani
nmadnani@umiacs.umd.edu
Department of Computer Science,
University of Maryland, College Park

May 12, 2005

Abstract

Information visualization has not been very successful in pervading areas of literary and scholarly enquiry, such as the analysis of relatively unstructured human-generated text like poetry. The work presented here proposes the use of visualization tools for such scholarly analysis and details the operation of a tool developed for this purpose.

Keywords

Information Visualization, Poetry, Emily Dickinson, Reduced Representation, Document Visualization.

1 Motivation

This goal of this project is not only to conceive and implement a tool suitable to the visualization of free-flowing natural language text, such as poetry, but also to empower the user to analyze such text, without ever leaving this visualization tool.

The users of this tool are the the researchers and the analysts of the MITH (Maryland Initiative for Technology in the Humanities) group at University of Maryland, College Park. These analysts are currently studying a selection of verse-letters and poetry, from the 19th century, by the poetess Emily Dickinson. Their objective is to determine whether there are indicators of eroticism in these documents. [1]

For this purpose, they need to work with an XML-encoded version of the above collection of poetry as maintained by the *Dickinson Electronic Archives* [2, 3]. This version of the collection contains a transcription of the poetry itself, along with other relevant tagged information such as the provenance of the poem and the condition of the original manuscript. The original manuscripts themselves also accompany the collection in the form of scanned images.

To perform such analyses, these scholars need a tool that allows them to :

1. Perform various forms of searches on multiple poems at the same time.

2. Visualize the search results, in various ways, in real time.

Section 2 discusses previous work related to this work and also some ideas that form the inspiration for this work. Section 3 enumerates, in detail, the requirements of the users of this tool, while section 4 discusses the workings of this tool in detail. Section 5 recapitulates the novel ideas behind this work and, finally, section 6 provides some ideas for future improvements.

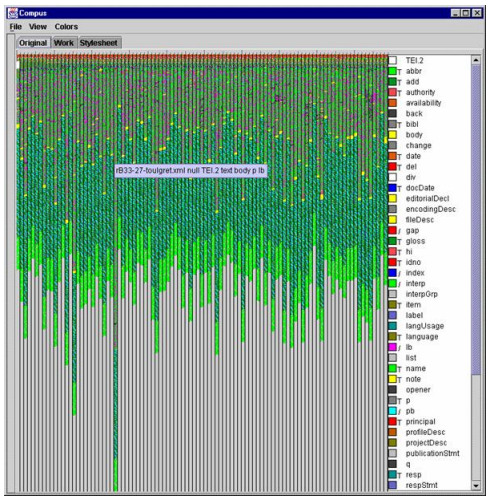

Figure 1: The Compus System showing a corpus of 100 structured documents.

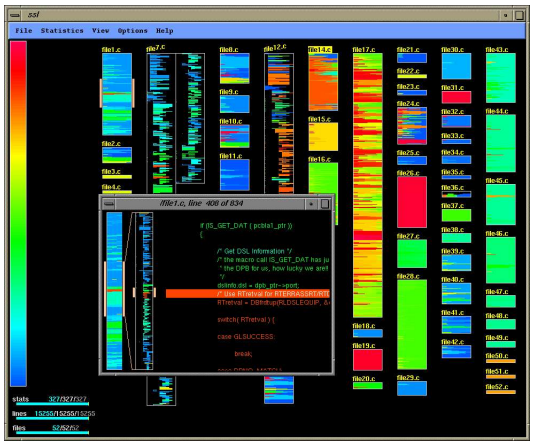

Figure 2: *SeeSoft* uses a line-based reduced representation to visualize program code.

2 Background & Related Work

It is important to note that the techniques of information visualization and human-computer interaction have not really been able to pervade the realms of literary critical enquiry. A significant reason for this is the reluctance of literary scholars to believe that such techniques – borrowing heavily from the discipline of Computer Science – are relevant to these realms [4].

Therefore, while setting up and organizing digital repositories of classical literary works – in order to provide easy access to such vast amounts of heterogeneous data – is now a major part of digital library research, the usage of such digital resources for the purpose of scholarly research is relatively less emphasized.

There has been work on integrated environments that allow users to read and annotate digitized documents from a library [5] but this does not provide support for any reasonable combination of searching and visualization that would lead the scholars down the right path of inquiry. Instead, this work focuses more on allowing scholars to transcribe manuscripts.

There is also work on studying different ways of presenting the actual text of electronic documents on the screen, in order to make such on-screen reading most useful. This work incorporates the popular fish-eye approach to document browsing and visualization but it not any form of search [6].

Other interesting document visualization systems include *TextArc* [8], a tool designed to help people discover patterns and concepts in any text by leveraging human visual processing. It represents the entire text as two concentric spirals on the screen: each line is drawn in a tiny (one pixel tall) font around the outside, starting at the top and then each word is drawn in a more readable size. Although very useful for providing an overview, it is not really suitable to the task explored in this project.

Our effort primarily derives its inspiration from three seminal works. The first one is *Compus* [7] – a visualization tool aimed at analyzing corpora or XML documents. It visualizes the actual XML encoding of the provided data by assigning a unique color to each XML element, and also allows the

users to perform structural transformations, sorting and exporting of the given XML data. It also allows filtering of XML elements but does not provide for searching of any form. Compus is at its best when being used to analyze the characteristics of the given corpora and not the actual content. Figure 1 shows the Compus system running on a structured corpus of 100 documents.

The second is *SeeSoft*, a visualization system for software statistics [9]. Although it works with a relatively more structured form of human-generated text (program code), it is still highly relevant to the task at hand, since program code and poetry are both expressed in terms of *lines* as their basic unit of composition. Thus, the idea of a line-based reduced representation as employed by SeeSoft (see figure 2) forms the inspiration for the visualization scheme adopted in this paper.

SeeSoft also uses another technique for effective visualization called *coloring by statistic* – wherein different lines in the program code are colored differently depending on what value has been assigned to that line for a pre-stipulated attribute(s). However, it is important to note that these attribute values are taken from other offline processes and are static during the execution of the program. Dynamic updating of statistics and, hence, of colors, is not supported (and possibly not warranted, as there is no search function built into the tool.)

The last work used as an inspiration, although not as significant as Compus or SeeSoft, is *DocumentLens* [10]. It proposes the idea of laying out a multi-page document (or even multiple single-page documents) on a 2D surface and using the fish-eye effect to allow the user to read these documents.

It also allows the user to search for words and automatically alters the colors of the matched words and lines – without changing the basic layout of the surface – to let the user perceive patterns, if any, that are apparent in the resulting documents. This idea of re-coloring of the text based on searches is also adopted as a step in the right direction for this work.

Therefore, the idea of dynamic visualization of results based on multiple forms of searches, on relatively unstructured text is a novel one and one that is developed on in this work.

3 Requirements

Before looking at the tool, it is important consider the basic requirements of the MITH task :

1. The presence of certain key terms in the transcribed text of the poem is an indicator of eroticism. For example, some of these terms are `seal`, `cut`, `night`, `blood` and `love` etc. It is required that these terms be weighted higher than other non-key terms and therefore, the user should be able to carry out a *weighted search*.

2. If any of the search terms occur more than once in any given line, that line should be considered more important than other lines and should be visualized accordingly.

3. The user should be able to filter out the poems that do not match the given search criteria and concentrate only on the matches.

4. The user should be able to pick and choose different forms of color assignment to the search results.

5. The tool should provide seamless and suitable linking between the electronic transcription of the poems and the original manuscripts.

4 Emily

The tool developed for this effort is titled, aptly enough, *Emily*. In the following subsections, we explore the operation of Emily in detail. Section 4.1 explains how to import data into Emily. Section 4.2 shows how the documents are represented and visualized by Emily, while section 4.3 explains how the user can access and inspect, in detail, the original

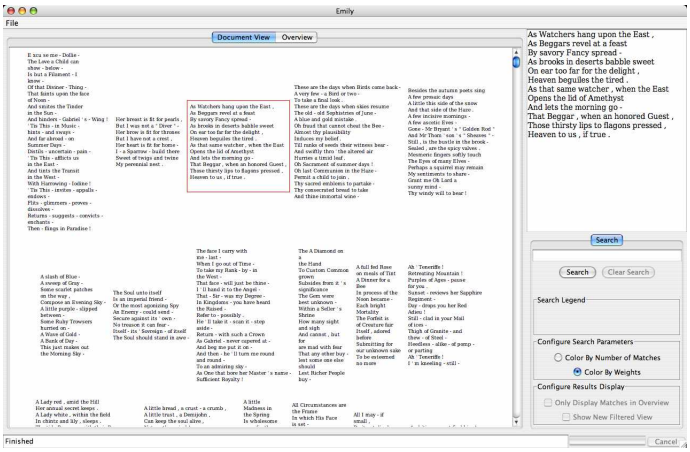

Figure 3: Emily's Document View.

manuscripts from these representations. Section 4.4 explains the idea of a search according to the given user requirements. Sections 4.5 and 4.6 clearly outline the various forms of interactive and dynamic visualization available to the user for analyzing the results of such a search. Section 4.7 illustrates these visualizations with an example search query.

4.1 Importing Data

There are two ways of importing data into Emily. One is to parse existing XML-encoded poems as provided by the MITH group. This approach is usually not very fast (about 90 seconds for 250 poems) because the underlying XML parser used is not very fast. For this reason, Emily provides a way to save the data obtained, after parsing these XML files for the first time, into a binary format so that on subsequent occasions, the binary data can be loaded directly, bypassing the parsing completely. This is quite fast (5 seconds for 250 poems).

4.2 Document Representation

Once the data has been loaded, Emily provides two different ways of visualizing the selected poems. The default view that is shown on start up is the Document View – this provides the user with a general overview of all the documents currently open (as inspired by the 2D Layout of *DocumentLens*). Figure 3 shows this view for the whole set of 250 poems in the provided collection.

The second view is the line-based reduced-representation view as shown in Figure 4. In this view, each of the poems is represented by a group of colored lines such that the lines taken together form a "bar" proportional to the poem's length itself. The width of each such graphical line is dynamically calculated depending upon the amount of space available to each poem, i.e., if a smaller number of poems are imported, then the size of the line-based representation of each poem is automatically resized to use more screen space. Figure 5 illustrates the intelligent layout property.

Also note the poem panel to the right in both figures 3 and and 4. This details-on-demand panel is coupled to both the views and allows the user to actually read the transcription of the original poem, when the user clicks on it in either view.

Figure 4: Emily's Line-Based Reduced-Representation View.

4.3 Accessing Original Manuscripts

The original manuscripts for each poem can be easily accessed by double-clicking on a poem in either the document view or the line-based view. A double-click brings up the manuscript viewer with the original manuscript and navigation buttons to go back and forth between multiple pages of the manuscript, if it has more than one.

It is also important to note that the manuscript viewer allows zooming in, zooming out and panning on the original manuscript so that the user can both closely inspect the actual written words on the manuscript and get an overview of the whole poem as written on the manuscript. Figure 6 shows the manuscript viewer with the poem at its default size. Figure 7 shows a zoomed view of the same manuscript. The zooming capability is provided by using the Piccolo toolkit [11].

4.4 Searching

The ability to perform searches and the visualization of corresponding search results form the crux of this work. To satisfy one of the core requirements of the MITH project, as outlined above, Emily supports weighted as well as unweighted searches, allows filtering of non-matches and provides multiple visualizations for the same set of search results.

Users can either search for single words (single search term) or a bag of words (multiple search terms). In either case, weights for each search term can be specified. The weights indicate how important a particular term is to the user in that given search and, therefore, the lines that contain the high–weight search terms are made more prominent in the visualization of the search results. (An unweighted search is taken to be equivalent to a weighted search with the weight for each search term as 1).

During the process of searching, each of the search terms is automatically assigned a unique color and this color is used for visualization as discussed in the next section. The search legend, visible below the search field, summarizes which colors have been assigned to which search terms.

4.5 Search Visualization

The results of any search are indicated on both the document view and the line-based view. There are two primary ways in which the results of any search

can be shown. Both of these approaches make use of colors as the primary indicator of search result attributes. The user can select either of these approaches by selecting the corresponding radio-button in Emily before starting the search.

4.5.1 Color by Search Terms

In this approach, all the search terms are assumed to be of equal weight (= 1). If the user does explicitly specify any weights for the search terms, they are ignored. For the line-based view, all lines in each poem that contain only a single search term are colored using the color assigned to that term (as shown in the search legend). If any line contains multiple search terms, that line is colored with an entirely different color (one that the user can configure - *purple* by default).

For the document view, the search results are indicated simply by coloring all occurrences of each search term in every poem with the corresponding color.

4.5.2 Color by Combined Weight

In this approach, the primary visualization attribute is *weight*. Weight can be assumed to be correlated with a degree of "hotness" (the correlation of the word *hot* with the hypothesis of eroticism is apt) of each poem. If a region in a poem contains higher weighted terms than other region, then that region is assumed to be hotter than the rest. This *degree of hotness* can naturally be indicated using different shades of the same color.

More intuitively, any search can be thought of as an exercise in assigning points to each line in the verse. If a line contains 1 word with weight 2, it is assigned 2 points; if it contains 2 words each weighted 3 and 5 respectively, the number of points then becomes $3 + 5 = 8$ points. Multiple occurrences of the same word also lead to more points being assigned to a line.

Therefore, a very simple implementation of the weighted search can be summed up as : **the more points a line has, the brighter it is.** However, this is not very useful as the MITH analysts are more interested in finding interesting regions as opposed to single lines. Therefore, an indication with lesser localization is what's needed to make weight-based visualization more useful. This is provided by the concept of **diffusion**, i.e., if a line has more weight than a pre-stipulated threshold, then the rest of that weight mass should *diffuse* or *bleed* into the surrounding lines. This bleeding can be indicated by a lighter shade of red. Therefore, with this bleeding incorporated into the visualization, the users can now easily identify interesting regions and concentrate on them.

The visualization in the document view for this approach is done in exactly the same way as the previous approach.

4.6 Filtering

It is important for the user to be able to concentrate only on the poems that contain the search terms and not be distracted by the other poems. This is especially true if the number of *hits* (the number of poems that contain the search terms) is low as compared to the total number of poems. In keeping with the multiple-options theme, Emily provides two approaches to filtering the search results.

- **Filter in place** : When this filtering option is selected, all the poems in the line-based view that do not contain any search terms effectively "turn off" or become invisible *in situ*. This is important for the situations where the original order of the poems, as they were initially laid out, is important.

- **Create filtered view** : With this option, a new filtered view is created where the hits are the only poems on the screen. Therefore, they get laid out dynamically to occupy as much space as possible. The new view is automatically purged when the search is cleared or this option is turned off.

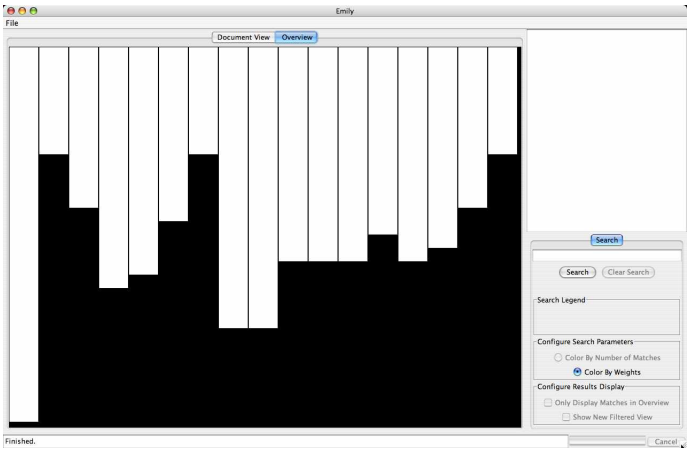

Figure 5: Illustrating the intelligent layout capabilities of Emily (compare with figure 4).

It is important to note that the above two options are completely independent of each other. The user can have either or both turned on at any time during an active search.

4.7 Search Example

Consider the search string "`hand:3, face:5, love:7`". This denotes the bag-of-words search in which the word `hand` is to be weighted 3, the word `face`, 5 and `love` 7. Figure 8 shows how Emily visualizes the search results when the option to color by search terms (as explained above) is selected.

Note that this figure shows the newly created filtered view that only contains the hits (poems where one or more search terms were found). Also notice the search legend on the right which shows the colors assigned by the system to each search term.

From this figure, the first thing that is immediately apparent (due to the absence of any purple) is that there are no poems in the entire collection, that contain any lines with more than one search term.

Figure 9 shows the results of the same search

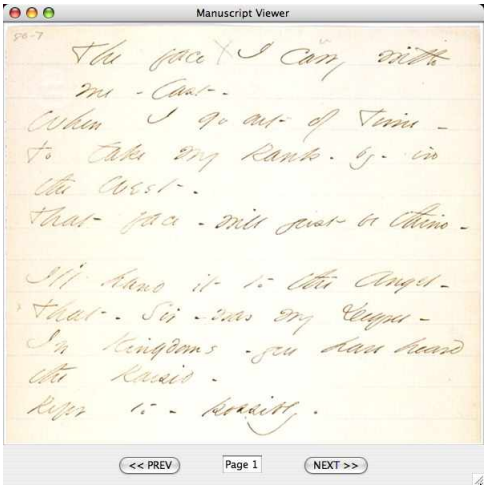

Figure 6: The original manuscript can be easily accessed with the Manuscript Viewer.

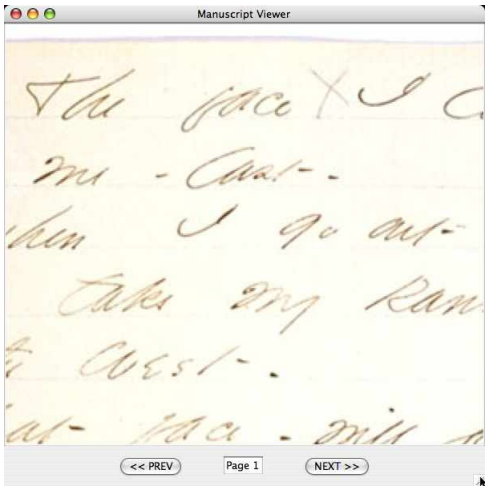

Figure 7: Users can easily zoom in and out in the Manuscript Viewer, allowing both closer inspection and overviews.

string but with the visualization option of coloring by weights, and hence shades of a single color are used.

Notice that some of the hot regions have faint colored regions below them. This is because these hot regions contain either more or higher–weighted search terms and, therefore, their hotness has bled into the lines below them, as we had postulated. This makes it possible for the more critical regions to be easily identified.

5 Discussion

It is generally assumed that literary analysis of unstructured human-generated text, such as poetry and prose, is relatively straightforward. On the contrary, the requirements of such analyses are often, if not always, almost as demanding as any other form of data analysis.

Basic string searches on a single file are simply not sufficient. What is needed is a way to let scholars analyze **multiple** documents at the same time by performing **multiple forms** of searches and allowing **multiple forms** of visualizing the results of such searches. A seamless way to integrate all the other available resources such as the original manuscripts is also essential for a fruitful enquiry.

Fundamental ideas as well as new concepts from the field of information visualization are instrumental in providing this work as the first step on that path to useful literary enquiry.

6 Future Work

The work done here only represents initial ideas and prototypical constructs. There are several areas in which much more could be accomplished :

- **Annotation** : In the longer term, this tool will be coupled to a data-mining back-end which would learn from the searches conducted by the user and suggest more "hot" results related to the search. In this scenario, there needs to be a way to annotate or mark such suggested results as **Don't Agree** or **Agree** – a sort of heuristic to active learning.

- **Refined Diffusion** : Instead of using a fixed threshold for all poems, adaptive diffusion or bleeding could be implemented – which would pick the right threshold depending on the length of a poem and the search term weights.

- **More Expressive Searches** : As of now, the search queries can only be expressed in terms of words contained in the poems. More expressive searches in terms of other XML tags and their contents are also quite useful.

- **Zooming into hot regions** : Once the hot regions have been identified with the help of diffusion, the user should be able to click on any such region and zoom into it, effectively being able to see the text of the verse lines contained in that region – but with the rest of the poem unaffected.

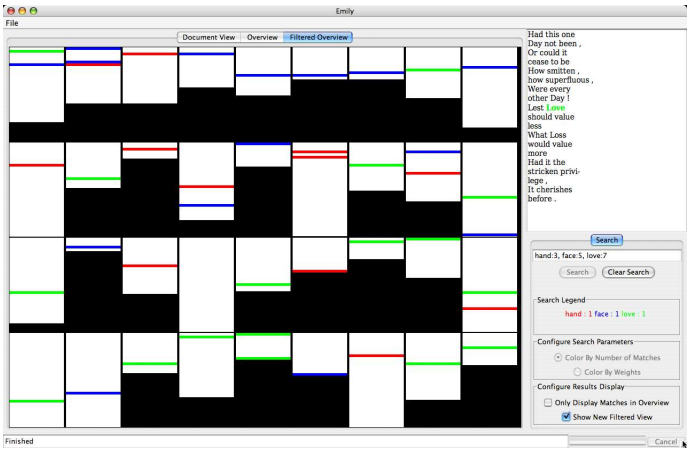

Figure 8: A filtered view of the search results for the bag-of-words search "`hand:3, face:5, love:7`" with the visualization option `color-by-matches`.

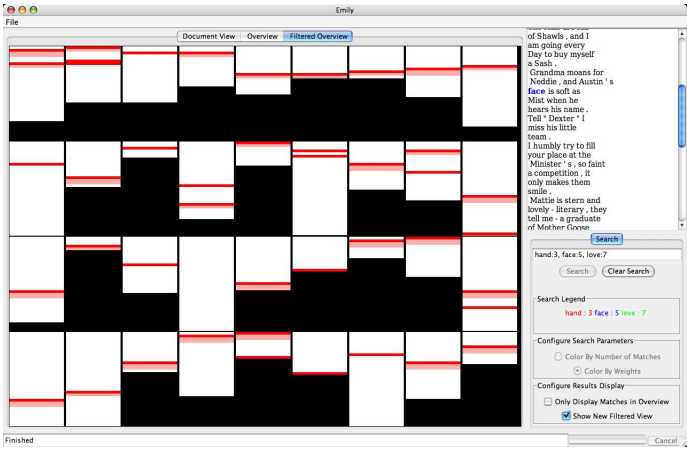

Figure 9: A filtered view of the search results for the bag-of-words search "`hand:3, face:5, love:7`" with the visualization option `color-by-weights`.

7 Acknowledgment

The authors would like to thank Catherine Plaisant for her enormous help and feedback in the design of this project. They also acknowledge the initial feedback and help received from Martha Nell Smith, Matt Kirschenbaum and Tanya Clement at MITH and last, but not least, Jean-Daniel Fekete, who provided advice for the XML parsing.

References

[1] Richard Schneider, Jr., ed., 1997, On Emily Dickinson, *The Best of The Harvard Gay and Lesbian Review*, pp. 191-197.

[2] H.T.M. van Vliet And Bodo Plachta, ed., 2002, The Dickinson Electronic Archives Project: Evolutions of a Dynamic Edition, *Editing for the New Millennium: International Perspectives*, pp. 163-180.

[3] `http://www.emilydickinson.org/`

[4] Martha Nell Smith, Computing: What Has American Literary Study to Do with IT ?, *American Literature*, 74(4), pp. 833-857.

[5] E. Licolinet, F. Role, L. Likforman-Sulem, J-L. Lebrave, L. Robert, 1998, An Integrated Reading and Editing Environment for Scholarly Research on Literary Works and their Handwritten Sources, *Proceedings of the Third ACM Conference on Digital Libraries*.

[6] Kasper Hornbaek And Erik Frokjaer, 2003, Reading patterns and usability in visualizations of electronic documents, *ACM Transactions on Computer-Human Interaction*, 10(2), pp. 119-149.

[7] Jean-Daniel Fekete And Nicole Dufournaud, 2000, Compus: Visualization and Analysis of Structured Documents For Understanding Social Life in the 16th Century, *Proceedings of Digital Libraries. ACM*, pp 47-55.

[8] `http://www.textarc.org/`

[9] Stephen G. Eick, Joseph L. Steffen And Eric E. Sumner, Jr., 1992, Seesoft-A Tool for Visualizing Line Oriented Software Statistics, *IEEE Transactions on Software Engineering*, 18(11).

[10] George G. Robertson And Jock D. Mackinlay, 1993, The Document Lens, *Proceedings of the 6th annual ACM Symposium on User Interface Software and technology*.

[11] Benjamin B. Bederson, Jesse Grosjean, and Jon Meyer, 2004, Toolkit Design for Interactive Structured Graphics, *IEEE Transactions on Software Engineering*, 30 (8), pp. 535-546.