
Interactive Visualizations for Text Exploration

Using SVG to navigate large collections of unstructured documents

Mr. Seth T. Raphael, National Institute for Technology and Liberal Education [<http://www.nitle.org/>] <
raphael@nitle.org>

Abstract

The task of navigating large unstructured collections of documents to find information that is relevant, related, or connected to a topic can be very difficult. At the National Institute for Technology and Liberal Education (NITLE) we develop tools that use statistical analysis to correlate similar documents and to facilitate the exploration and discovery of connections between them.

Information visualization is a growing and relevant field. Data-driven visualization graphics created with the SVG format can interact robustly with the surrounding web-based environment, have the power of a built-in programming language, and can dynamically update themselves over a network. These abilities allow rich visualizations of content to not only be created on the fly, but navigated on the fly as well. The ability of an SVG image to interact with the underlying systems it represents makes it a very powerful solution for content and relationship visualization.

Table of Contents

Overview	1
Our Visualizations	2
Clusters	2
Dendrograms	3
Content Stars	5
Character Diagrams	6
NSE SVG Toolkit: NST	8
Reusable Content (widgets)	9
Debugging	9
Windowing System	9
Server Talk	10
Further Research	10
Bibliography	10

Overview

The task of navigating large unstructured collections of documents to find information that is relevant, related, or connected to a topic can be very difficult. At the National Institute for Technology and Liberal Education (NITLE [<http://www.nitle.org/>]) we develop tools that use statistical analysis to correlate similar documents and to facilitate the exploration and discovery of connections between them. The tools range from probabilistic models, to computational linguistics, and from clustering algorithms to graph-theoretic models of text representation. These tools by themselves lead to interesting interactions with the text, but the experience can be improved upon through the use of interactive visualizations.

The use of dynamically generated representative graphics allows users of the NSE to understand a

collection of documents in a new way. The visualizations can help elucidate the connections between documents, show the relationships between key concepts, and help characterize documents' content. Dynamically generated graphics can be created using a number of different technologies, but SVG has several features that make it particularly suited to these purposes.

Data-driven graphics created with the SVG format can interact robustly with the surrounding web-based environment, have the power of a built-in programming language, and can dynamically update themselves over a network. These abilities allow rich visualizations of content to not only be created on the fly, but navigated on the fly as well. The ability of an SVG image to interact with the underlying systems it represents makes it a very powerful solution for text and relationship visualization.

Our Visualizations

Information visualization is a growing field with many different areas of research. Within the context of the NITLE Semantic Engine, a suite of tools for managing and discovering knowledge in unstructured collections, there are many potential uses of visualization. From depicting an entire collection with thousands of documents and their relationships, to revealing the way content in a single document fluctuates and changes over time, these graphical representations can reveal both macro and micro patterns.

The goal of our visualizations is to allow new understandings of the body of text being navigated and also foster a greater understanding of the algorithms used in our suite of tools, the NITLE Semantic Engine (NSE). In designing and implementing the visualization aspect of the NSE the aim is to have each visualization add to the user's own mental model of the collection, thereby improving their ability to manipulate and harness the existing content. The algorithms we use lend themselves naturally to intuitive visualizations, which makes the task of designing them slightly easier. The central algorithm of our project is a graph-theoretic search algorithm. In this model, each document is treated as a node in a graph, which is connected to each of the terms contained therein. Documents are connected through the term-nodes they share, and searching entails a spreading activation algorithm whose feedback and Monte Carlo aspects create relevant search results that may not contain the exact words from the original query. This graph structure would seem to be a natural candidate to be visualized, unfortunately, direct visualizations of this structure are generally too dense to be meaningful.

However, we have experimented with several different visualizations of this process, which are presented here. These include showing how different clustering algorithms reveal the structure of the collection, seeing graphically how many distinct topics are discussed in a collection, and visualizing the interactions between characters in a document and their changes over time.

Clusters

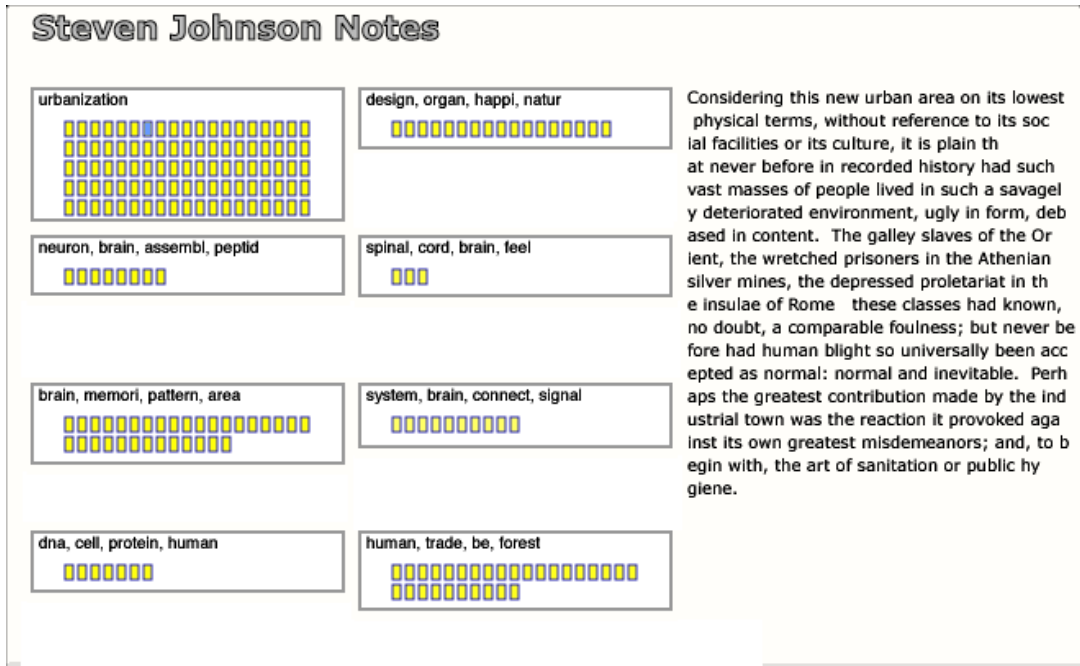
A given set of documents to be searched by the NSE may cover many and disparate topics. In a collection of highly structured documents marked up with meta-information such as keywords, or tags, the task of categorization is simple. Unfortunately the cost of creating and maintaining such structured collections is very high and often unrealistic. Using various techniques of statistical analysis and dimension reduction, common themes and topics between documents can be discovered algorithmically. The NSE includes a variety of clustering algorithms, which attempt to put groups of similar documents together and to characterize their similarities.

The problem with such algorithms, from the point of view of visualization, is their variability. There are many different algorithms, each with many different tunable settings, and they each yield different groupings. One benefit of the NSE to clustering is a well-defined distance metric: the distance measure between two documents is equal to the relevance of two documents in a given search. For our purposes, this provides a very good basis from which to begin clustering documents. However, it is still important to be able to evaluate the effectiveness of the algorithms to determine if the results they yield are useful.

Visualization can help facilitate this task greatly. By observing how many documents are in each cluster and what the main topics are for the groups, it is possible to identify clusters that are inappropriate and to combine clusters that may overlap or be semantically equivalent. By graphically dis-

playing these groups of automatically clustered documents along with other information, new information can be gleaned about the nature of the collection. The relative sizes of clusters and documents, the size of the entire collection, and the strength and number of clusters are all small factors which can contribute to an expert's understanding of the global nature of a large number of heterogeneous documents.

Figure 1. Visualizing Clusters



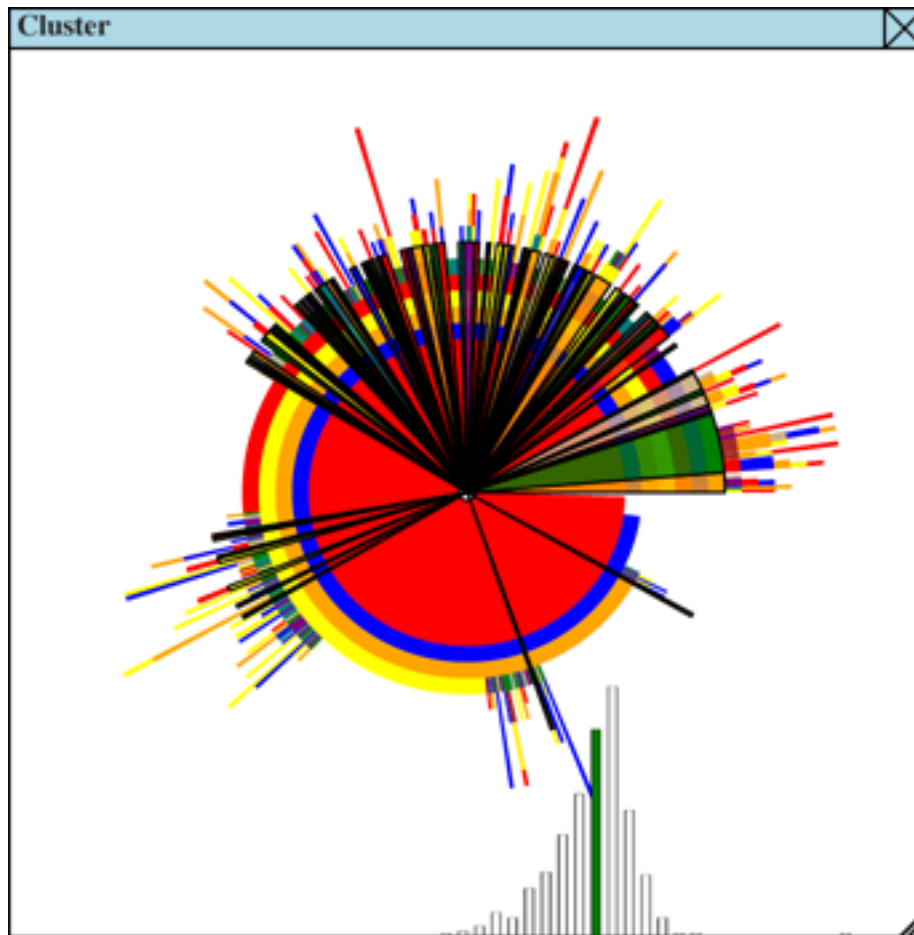
Various automatically generated clusters are shown with their constituent documents. The content of one document is displayed.

The system we designed for navigating clusters is very straightforward (Figure 1, "Visualizing Clusters"). A number of boxes are displayed on the screen, one for each cluster. Each box is sized just large enough to contain an icon for each document that falls into that cluster. At the top of the box is a summary of the top keywords for those documents, which are represented therein. At this point, a mouse-over of the icons will display the title of the document, and a click will load the text of the document for review. In this way the quality of each cluster can be assessed. Clicking on the summary of a cluster allows the user to edit it and rename the cluster as appropriate. If necessary, documents can be dragged from one cluster to another to augment the automatic results, and two clusters can be dragged together to combine their results. These actions are all sent back to the server and stored on the database as a modified version of the original clustering.

Dendrograms

Some of the clustering algorithms we use are called hierarchical clustering algorithms. These algorithms work recursively, first clustering some documents, then adding more, and eventually clustering smaller clusters together. Eventually everything is conglomerated into one giant cluster encompassing all of the documents. This structure is a binary tree. The root is the largest cluster, and its two branches are the next two distinct clusters. Each of these branches and so on until the leaves, each of which is a single, unclustered document

Considering it from the opposite end helps elucidate the algorithm. It starts by taking the two most similar documents and clustering them, and then it takes the next two most similar documents and clusters them. At each stage, the existing clusters are treated as one document so that similar documents are attracted to them. The problem with this type of algorithm is deciding when to stop the cycle, in other words, where to cut the tree. The number of clusters decreases by one with each iteration.



Information is condensed, the more relevant information being given more space in a radially-organized dendrogram.

In our SVG implementation of radial dendrograms to visualize cluster space, we added several features. The data displayed was collected at regular intervals instead of continuously. This not only reduced processing time for the algorithm, but also created "slices" to display the general level of clustering throughout the process. Below the dendrogram, we displayed a histogram showing how many clusters existed at each threshold. In this way, the user could click on the largest bar to find the most clusters, which would then be highlighted in the radial view. Clicking on a particular wedge would load that cluster from the server-side database to be viewed in the SVG.

Content Stars

Even after documents have been clustered it is still difficult to determine their meaning without spending some time reviewing their actual content. To this end we have developed summarization techniques which complement the other components of the system. These summarization techniques still require reading small blurbs of text, which makes it difficult to scan the summarization of many documents at once. For this purpose we designed visual summarization icons, or content stars. These are based on the work "Seeing Meaning [<http://www.media.mit.edu/cogmac/projects/seeingmeaning/index.html>]" done at MIT[MIT]. In our version, we arrange the strongest clusters radially and generate an icon for each document reflecting its individual strength in each particular topic. These star-shaped blobs (Figure 4, "Content Stars") characterize the overall content of each document.

Figure 4. Content Stars



These color-coded icons represent each document's component along five cluster-axes. A larger point indicates a stronger correlation to that particular cluster.

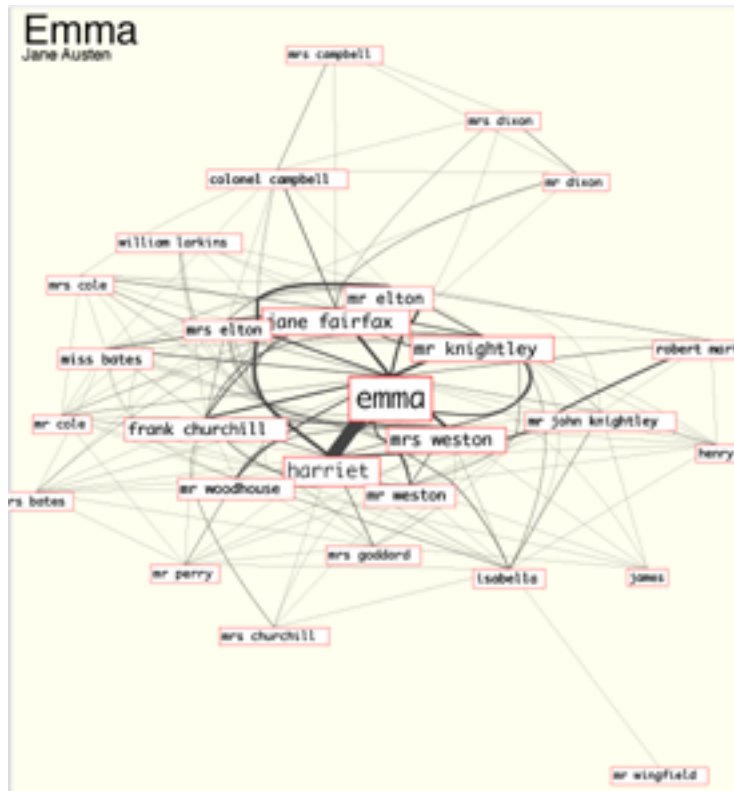
The first step is to run the clustering algorithm on a set of search results to sort them into the five strongest clusters. Each document is then color coded to a key (not shown) that describes the content of each cluster. The background color of each title indicates its primary cluster. Each document is then measured to see how relevant each of the five clusters is to it, and this is rendered as a star-like shape representing the content of the document.

This technique not only allows you to see the content of given document at a glance, but it also allows you to compare large numbers of documents. Using the idea of small multiples, many documents can be shown on a page[Tufte, E 2001]. This particular visualization can also aid in discovering anomalous documents as in the last result in Figure 4, "Content Stars" where though the clustering algorithm placed the document in the red cluster, it is more related to the yellow cluster. This can be for any number of reasons, and the existence of a high number of anomalous cases could indicate a failure of the clustering algorithm. This could be particularly interesting with relation to the technique of Scatter/Gather clustering for search results.

Character Diagrams

As mentioned earlier, the full graph representation of a document collection is too populous to be of use as a visualization. However, a graph visualization is still a very intuitive and useful technique for displaying information. One of our tools is designed specifically for use with literary texts. We applied this graph visualization to the characters in several novels and their interactions, based on co-occurrences in the text. This allows one method of viewing patterns of character development that occur, and by comparing different points in the book, the visual changes in the story may be manifested.

Figure 5. Visualizing Characters

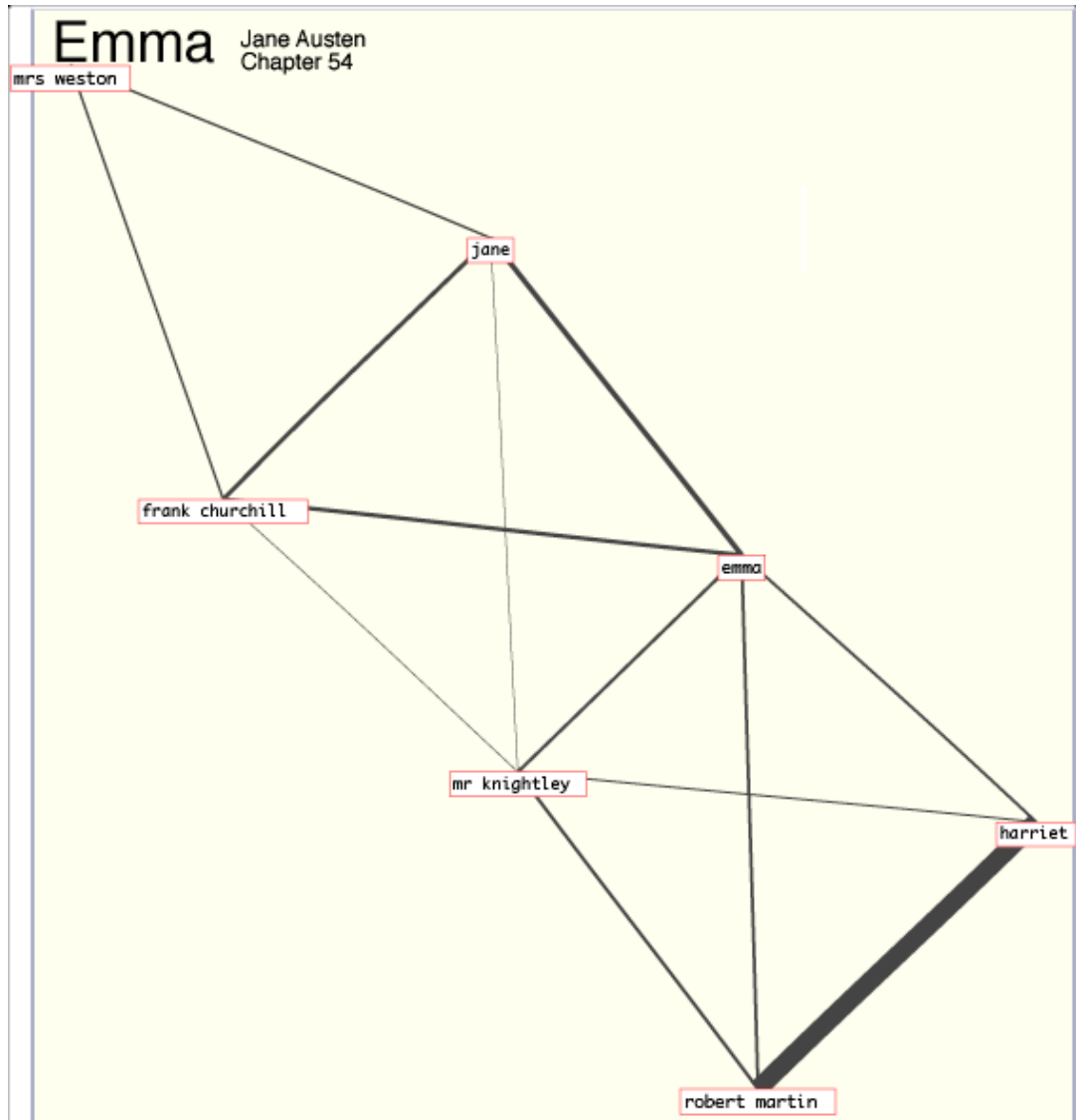


This diagram shows the relationships of the main characters in Jane Austen's Emma

The characters are displayed scaled by importance. The more frequently a character appears in the text, the larger their name is displayed, hence Emma is the largest character in Figure 5, "Visualizing Characters". The lines connection characters represent their interactions: the thicker the line, the more interactions between those two characters. The geographic placement is determined by a graph layout algorithm that uses an annealing model to determine optimal positions, therefore the distance between two characters has no meaning, nor does their location. Emma's central placement in this diagram, however, does reflect her high degree of connectivity as interpreted by the layout engine, SVG-neato [<http://research.nitle.org/downloads/SVG-Neato/>][Coburn, A.].

There is an interactive aspect to this visualization as well. At the lowest level, mousing over a connection between two characters yields a visual stimulus to clarify which characters are reflected by that line. Clicking on any character will reveal all of the instances of that character in the novel (or chapter). Clicking on the connecting lines will perform a search on the two characters, yielding the places in the text that they interact.

Figure 6. Chapter-level Visualization



The penultimate chapter of Austen's *Emma* reveals the pairing off of characters. In previous chapters, the graphs have been fully connected. In this chapter, however, the pairs indicated in the diagram each get married: Robert Martin marries Harriet, Mr. Knightley marries Emma, and Frank Churchill marries Jane.

NSE SVG Toolkit: NST

Through the course of implementing our various experiments in SVG, we have discovered tricks, developed toolkits and designed workflows to solve our problems. In our work, we have run into the need to reuse custom content to reduce the complexity of our SVG files and to keep them consistent. In the process of designing a toolkit for that purpose, we discovered a need for debugging tools. We also had to discover the best process for quickly delivering dynamically generated SVG images in a production environment in a way that dealt with server load, bandwidth, and site design. We developed a number of open source libraries that deal with these varying issues and will discuss them in this paper.

The NITLE SVG Toolkit is a collection of tools that we have developed to facilitate our development of interactive SVG. These include drag and drop libraries, debugging utilities, libraries for getting user text input, and corresponding with a server. The main library is the RCC library which allows the creation of a framework of widgets using their own namespaces. Content is automatically replaced in a simple SVG allowing the re-use of SVG code and objects across projects and the scaffolding of code. The library not only allows the instantiation of objects, but also allows robust inter-

actions, and event-handlers to be defined in a manageable way.

Reusable Content (widgets)

The goal of Reusable Content is to reduce the amount of time needed to create and deploy complex SVG-based applications. By designing a framework, reusable content can be created and re-used in an encapsulated way. Key to this is that different parties can create their own content, extend existing content, and build widget sets that do not interfere with each other. Another goal in our design was to make it easy to insert custom content into regular SVG easily. While there are several open-source widget toolkits like CGUI [<http://homepage.usask.ca/~ctl271/cgui/>][Lewis, C.], none of them met the last requirement.

The solution we arrived at is an ECMAScript library that manages custom component loading, management, and replacement. Once this library is loaded, the SVG DOM is traversed looking for custom content to be replaced. When a recognized namespace and object type is discovered, the appropriate action is applied, replacing it with the custom content as specified in separate ECMAScript files.

Debugging

As our SVG applications became more robust, it became apparent that some debugging environment was required to facilitate development. The first step was creating a method for viewing the output of scripts. our NSTglobals.es adds to the global scope, two functions, syslog() and syserror(). A script can use these functions to log messages, or errors. In normal circumstances they have no visible effect. If a group is added to the document tree with an id of "stdout" these messages will be logged for review and display in the SVG. The next step is to add a javascript console. This allows an object called "stdin" to be created allowing the developer to run arbitrary ECMAScript in the current SVG file. These two additions, while rather crude and rudimentary, allow at least a beginning at a developer environment for debugging more complex scripts.

Windowing System

Drag and Drop

One of the first pieces of interactivity we needed to implement was drag and drop. The foundation for our code came from Antoine Quint's tutorial Doing That Drag Thing [<http://www.xml.com/pub/a/2002/02/27/drag.html>] on dragging and dropping [Quint, A. 2002]. The problem of drag and drop has been solved many times, addressing issues like custom zoom levels, nesting of objects, and turning off inappropriate mouse events. In designing our drag and drop library, the goal was to have it be easy to use, robust, and extensible. The syntax to add drag functionality to an element is very simple, you merely add the attribute drag='true' to your element. It then instantly becomes draggable.

The library also supports other features such as dropping. This means an element can receive other objects dropped on it. They then act as a group. You can also implement call-back functions which are called when an item is being dragged, when it is dropped, or when an object is dropped on it. This allows robust activities to be built from this simple framework.

Windows

Windows are just a convenient piece of reusable content. We designed them so that we could provide various pieces of visualizations on screen at the same time allowing a user to move them around using the familiar window paradigm. A window is a rectangle with a titlebar with buttons, and a resize triangle at the lower left. A few features of our simple windowing system make it useful. Each window can have a colored title bar with customizable styles. They also sport a close button, and they can be minimized like the mac "window shade" behavior. The content in a window can be resized independently of the surrounding content with the resize triangle. Each window can be dragged around so that content can be compared, and rearranged at the user's desire.

Server Talk

Not only are our visualizations often dynamically generated, but they sometimes need to be live. This means that the data reflected in the graphics is not static. In fact, the SVG's themselves might even effect changes in the underlying models. Bugs in current browsers have made some of this functionality difficult to implement, but there are methods of working around them described herein.

As described in other places, the key to talking with the server is the `geturl()` function. Jim Ley has excellent information on this technique. [<http://jibbering.com/2002/5/dynamic-update-svg.html>]

Further Research

There are many many possible continued areas of research in this field. The framework we have developed is still in development and is not yet mature, or robust enough for large-scale deployment. The visualizations we have created and described in this paper can be further developed and integrated into other tools. Following are some ideas we have just begun to explore, or are ripe for other areas of research.

- **Animation:** The appropriate use of animation could elucidate temporal relationships. Adding the ability to animate the character map visualization over chapters could prove both interesting and very useful.
- **Tile Bars:** Tile-Bars are a way of visualizing the distribution of specific content through one document [Marti A. Hearst]. This visualization could be combined with content stars, or animation to allow the flow of the document to be visualized.
- *Mapping:* It is possible to pare a graph to a point where visualizations of sets, a result set, for example, could be visualized. This may allow another interpretation of clustering.

With such a new field of research, many unexplored directions, and such interesting content, continued exploration of these and other techniques can prove fruitful at many levels. SVG provides an excellent platform from which to quickly prototype and produce visualizations and also to design robust means of interaction.

Thanks to those who made significant contributions but are not listed as authors.

Bibliography

[Coburn, A.] *SVG-Neato* A perl module for generating SVG graphs from neato output. <http://research.nitile.org/downloads/SVG-Neato/>

[Lewis, C.] *CGUI*. A widget toolkit conforming to the SPARK standard. <http://homepage.usask.ca/~ct1271/cgui/>.

[Marti A. Hearst] . *TileBars: Visualization of Term Distribution Information in Full Text Information Access*. ACM SIGCHI Conference on Human Factors in Computing Systems, Denver, CO, ACM, May 1995. <http://elib.cs.berkeley.edu/tilebars/about.html>.

[MIT] *Seeing Meaning*. <http://www.media.mit.edu/cogmac/projects/seeingmeaning/>.

[Schonlau, M 2002] "*The clustergram: A graph for visualizing hierarchical and non-hierarchical cluster analyses*", *The Stata Journal*, 2002, 3, pp 316-327. <http://www.schonlau.net/clustergram.html>

[Stasko, J 2000] *An evaluation of Space-Filling information visualizations for depicting hierarchical structures* *Int. J. Human-computer Studies* 533, 663-694. <http://www.cc.gatech.edu/gvu/ii/sunburst/>

[Tufte, E 2001] *The Visual Display of Quantitative Information*. Graphics Press, Cheshire Connecticut, 2001.

[Quint, A. 2002] *Doing That Drag Thing*. XML.com 2002. An example of drag-and-drop functionality for SVG.
<http://www.xml.com/pub/a/2002/02/27/drag.html>