Showcase Browsing with Texttiles 2.0 and BubbleLines

Alejandro Giacometti
University College London
Stan Ruecker
Illinois Institute of Technology
Carlos Fiorentino
University of Alberta

Scholarly and Research Communication
VOLUME 3 / ISSUE 2 / 2012

Abstract
In this article, we argue that it is possible to effectively expand interactions with a subset of a collection of text files by using an online interface based on rich-prospect browsing principles, coupled with task-specific visualization tools. In this context, we will present the revisions carried out to create the next iteration of the Texttiles browser, based on results from a user study in 2009. These changes have produced a much more flexible platform for showcasing significant items in text collections, based on user selections of display details, arrangement of items into groups, and annotation marks. In addition, we introduce BubbleLines, an interactive visualization that sits on the Texttiles 2.0 Application Programming Interface (API) and allows users to see search results across multiple documents simultaneously.

Keywords
Showcase browsing; Texttiles 2.0; BubbleLines; Application Programming Interface; Visualization tools; Rich-prospect browsing; Collection management

Alejandro Giacometti is Doctoral Research Student in the Department of Medical Physics & Bioengineering at University College London, 2nd Floor, Medical Physics, Malet Place Engineering Building, Gower Street, London, UK WC1E 6BT. Email: alejandro.giacometti.09@ucl.ac.uk .

Stan Ruecker is Associate Professor at the Institute of Design, Illinois Institute of Technology, 350 North La Salle Street, 4th Floor, Chicago, IL, USA 60610. Email: sruecker@id.iit.edu .

Carlos Fiorentino is Instructor in the Department of Art & Design at the University of Alberta, 3-71A FAB, Edmonton, AB, Canada T6G 2C9. Email: carlosf@ualberta.ca .

CCSP Press
Scholarly and Research Communication
Volume 3, Issue 2, Article ID 020121, 8 pages
Journal URL: www.src-online.ca
Received August 17, 2011, Accepted November 15, 2011, Published August 15, 2012

Giacometti, Alejandro, Ruecker, Stan, & Fiorentino, Carlos. (2012). Showcase Browsing with Texttiles 2.0 and BubbleLines. *Scholarly and Research Communication, 3*(2): 020121, 8 pp.

The INKE Research Group comprises over 35 researchers (and their research assistants and postdoctoral fellows) at more than 20 universities in Canada, England, the United States, and Ireland, and across 20 partners in the public and private sectors. INKE is a large-scale, long-term, interdisciplinary project to study the future of books and reading, supported by the Social Sciences and Humanities Research Council of Canada as well as contributions from participating universities and partners, and bringing together activities associated with book history and textual scholarship; user experience studies; interface design; and prototyping of digital reading environments.

Introduction

In this article, we argue that it is possible to effectively expand interactions with a subset of a collection of text files by using an online interface based on rich-prospect browsing principles, coupled with task-specific visualization tools. In this context, we will present the revisions carried out to create the next iteration of the Texttiles browser, based on results from a user study in 2009. These changes have produced a much more flexible platform for showcasing significant items in text collections, based on user selections of display details, arrangement of items into groups, and annotation marks. In addition, we introduce BubbleLines, an interactive visualization that sits on the Texttiles 2.0 Application Programming Interface (API) and allows users to see search results across multiple documents simultaneously. Both tools can read RSS feeds, XML files, or databases for documents selected from a larger collection, providing a flexible means of working with purpose-built subsets. The contents of documents are also immediately accessible in reading panels. BubbleLines is an example of the kind of visual showcase exploration tools that can be readily added to Texttiles 2.0 to extend the affordances of corpus browsing by applying specific processes to multiple collection items.

Rich-prospect browsing

The principles of rich-prospect browsing were laid out by Stan Ruecker (2003): a meaningful representation of each item in a collection and controls to manipulate the display. Meaningful representation refers to the display of each item in a collection using a metaphor or characteristic of the item that uniquely identifies that item in relation to all other items in that collection, and associates that item with the user's purpose in using the collection. Rich-prospect browsing allows a user to place any acquired knowledge in context – to recognize the circumstances around any new finding. A user is not only provided with a notion of the magnitude of the collection, but also with a rough idea of some other inherent properties of the collection such as range, diversity, and structure.

The second principle is in place in order to further this initial perception. Tools that allow the user to manipulate how the collection is displayed can help advance this exploration and aid research tasks. These tools are in the form of controls that allow visual selection and grouping of the items in a collection based on their properties. For example, a large collection of books can be grouped by author, year of publication, topic, or even the colour of their covers. Grouping allows a user to reveal further details about the topography of a collection. It shows the range and diversity: how many authors are represented, are the authors represented by more than one work? It also uncovers the relative weight of characteristics of items in a collection: how many books by one author versus another, a breakdown of the collection by date of

Giacometti, Alejandro, Ruecker, Stan, & Fiorentino, Carlos. (2012). Showcase Browsing with Texttiles 2.0 and BubbleLines. *Scholarly and Research Communication, 3*(2): 020121, 8 pp.

publication, and so on. Selecting, on the other hand, permits a researcher to look into a subset of the collection in order to explore aspects within the collection. The user can find the same characteristics of the whole collection within subsets. The reduction of items on display can help uncover deeper associations within the items. For example, by exploring a subset of works by a single author, a researcher could explore other characteristics that those works share, or how those works differ. By combining "grouping and selecting," a researcher can uncover new patterns within the collection and find unusual associations between individual items.

Several experimental interface prototypes have been developed based on these principles (e.g., Given, Ruecker, Simpson, Sadler, & Ruskin, 2007; Ruecker, 2006; Giacometti, Ruecker, Craig, Dersken, & Radzikowska, 2008). These prototypes have served as a confirmation of the value of rich-prospect browsing and have also brought us closer to understanding its use, and highlighted some important concepts to complement the original theory. The Pill browser is a browser designed to inspect a collection of medication tablets and capsules according to their physical appearance. A user study with senior citizens revealed the necessity of displaying alternative images (Given et al., 2007). Users were unable to discriminate between pills due to the pills having very similar faces, even though they had different profiles. A meaningful representation of an item in the collection is better achieved by displaying multiple depictions of the item. Exhibiting a pill through a variety of pictures taken from different perspectives can help the user to better identify it.

The delegate browser is another rich-prospect browser, designed to allow conference attendees to review the details of new acquaintances at an academic conference. The user can browse through the physical appearance of conference attendees, their professional affiliations and the topics of their talks. A preliminary study showed that users will use any and all available metadata (Ruecker, 2006). Users as a whole showed no preference for a particular piece of metadata. This study stresses the necessity of gathering and using as much data as possible, since users will find and make use of all available data points.

Similarly, the Texttiles browser was designed to present collections of text. In the absence of a simple concomitant pictorial representation of each work, a meaningful representation of each item was achieved by displaying their individual textual details inside of a tile. Those details are interchangeable, adding another layer of control over the display. A study on the Texttiles browser revealed some interesting factors: users will always attach meaning to the order in which items are displayed, and the exploratory process often involves actively manipulating the state of items in a collection (Giacometti, 2009). When displaying a collection of texts, the Texttiles browser would often list the works in the order that they were received from the server. It was found that users tried to figure out the order that the browser was using to sort the items – was it alphabetical, chronological, etc. Even though the order of the items was of no particular importance, users attached some meaning to it. The browser also included a marking feature, which users could use to colour a tile and keep track of it while the tiles were being grouped, selected, and moved around. Users would use this tool as a marker for tiles they were interested in, or the ones that they had already read, or just simply to be sure that the browser was always keeping the tile on display. It was important to them to be able to

alter the state of the tile. These marks helped users track down relations between tiles, formulate hypotheses, test them, and discover patterns.

Texttiles 2.0

We started development of Texttiles 2.0 by incorporating the lessons learned from the previous prototypes and user studies. We believe we now have a better understanding of rich-prospect browsing. We have acquired considerable experience from the previous prototypes in dealing with the infrastructure needed to create a collection browser. In addition, we believe the transfer of rich-prospect browsing principles to text collections with the Texttiles browser was a successful one, and we want to continue exploring this new paradigm.

We used JavaScript and HTML to develop the first Texttiles. The Web nature of the application means that it is instantly accessible to anyone using any computer with a modern browser. However, that approach also meant that much of the data processing needed to power Texttiles was done at the client's side, on the user's computer. The collection would be downloaded into the browser, processed, and displayed every time a user wanted to see it. Our study participants liked Texttiles, but complained that its response time was too slow. At the same time, we did not want to create a stand-alone application that needed to be tested for different operating systems, or that would support only a limited set of computers. The browser environment served as a great delivery mechanism that allowed us to show our interface in many situations without the use of specific testing environments or distribution systems. We wanted to create an application that could be accessed by a wide variety of users.

For Texttiles 2.0, we decided to separate the browser into its natural components: one layer that can support collection management and another interface layer to support interaction with the collections. We also needed to create a communication mechanism between these two layers: an Application Programming Interface (API). We informed our design by looking at the current standards of communication in Web applications.

Collection management

We have consequently spent some time developing a server-side service that will manage the collections, process data, and serve it to the Texttiles 2.0 browser in small digestible chunks. The Texttiles 2.0 back-end can ingest text collections and store them in a database. We used BigTable, an object-oriented database within Google App Engine, a cloud-based framework for developing Web applications.

Data is stored as a collection of objects and properties. This model for storing data fits nicely with text collections. Objects in BigTable are mutable, which means that their properties are not rigid, and can be manipulated when the need arises (Google, n.d.). Metadata can be stored in the database by referencing an object and storing a value using a property key. This key can represent any property or metadatum of the object. Properties can be added to objects at any time.

Within the collection manager, this dynamic works very well. For example, we can ingest an RSS feed from a blog. Each item in the feed has a title and text. In BigTable we can create a collection for the blog, and `item` objects for each article. The `item` object will have a

Giacometti, Alejandro, Ruecker, Stan, & Fiorentino, Carlos. (2012). Showcase Browsing with Texttiles 2.0 and BubbleLines. *Scholarly and Research Communication, 3*(2): 020121, 8 pp.

`title` property and a `text` property where the title and text of the article will be stored accordingly. Additionally, the RSS feed also includes metadata about each article: date of publication, the name of the author, and keywords that identify the article. BigTable allows us not only to update the original `item` object but also add new properties to it. We can add a `created_on` property to store the date of publication and so on.

Google's App Engine is a framework for developing applications in the cloud. It allows us to use Google's server-distributed infrastructure. It facilitates the reliability of the service. Its simple building style should also permit future collaborators to easily participate in the development of this project. Additionally, this design will simplify adding extensions to the system in the future, such as text analysis tools for further processing the text collections.

At the moment, these collections need to be encoded in XML and have an accessible URL, but we have plans for adding other formats such as JSON, CSV, and plain text translators, as well as the ability of uploading files. Users will be able to create, update, and manage their collections, and the items within collections. Users will also be able to edit properties of the items and add new ones if they need to.

Application Programming Interface (API)

A simple API allows the collections to be accessed through a common language. After a collection is ingested by the system, the data store provides unique URLs for creating, retrieving, updating, and deleting the collections and each item within the collections using a JSON REST API. Representational State Transfer (REST) is a style of communication modelled to facilitate interaction within a client-server environment (Fielding, 2000). It is based in the same principles as the protocol that powers most communication on the Internet, HyperText Transfer Protocol (HTTP). Its general principle is a client sending a request to a server to access or manipulate a resource, the server, which in turn processes the request and responds to the client. Among other things, a REST API provides a stateless, uniform communication protocol between the interface and storage, where the data is persistent but can be manipulated (Rodriguez, 2008).

The REST API of Texttiles works by providing a unique URL for a resource like a collection: `http://texttiles/collection/<collection_id>`, where `<collection_id>` is a unique identifier for a collection; and a variety of methods: `GET`, `POST`, `PUT`, and `DELETE`. Each method performs a different task, in order: retrieve a collection, create a new collection, update the properties of a collection, and finally, delete a collection. The API also provides URLs for similar actions to individual items within the collection: `http://texttiles/collection/<collection_id>/item/<item_id>`, where `<item_id>` is a unique identifier for an item in a collection.

This simple communication protocol exposes the resources of the collection manager to the interface in a structured manner. It allows the interface development to be exclusively concerned with the presentation of those resources.

Texttiles 2.0 interface

The Texttiles 2.0 browser interface communicates with the collection manager in the server through the API. It can continue to be developed in HTML and JavaScript and

enjoys the advantages of being a tool that can be accessed by the majority of computers. It also benefits from the performance improvements of receiving all the information in small digestible chunks in a standard format that is ready to be displayed.

The new Texttiles interface will include the findings from the previous studies on rich-prospect browsing, including diversity of metadata to facilitate browsing, variability of data representation to facilitate identification, sorting that unambiguously conveys some order, and the ability to change the state of items. These principles will be supported by the new platform that we have created. The architecture facilitates the storage of the diverse properties of each item. By separating the storage and processing of the collection, we can now request data in an order that will convey meaning. Finally, the design of the storage and the communication API enable manipulation of the data, where we can modify the state of each item.

A platform

This modular architectural approach is similar to that of the Just in Time Research (JiTR) platform, currently in development (Rockwell, 2007), and its predecessor, the Text Analysis Portal of Research (Rockwell, 2003). TAPoR provides an accessible resource for text analysis tools available to humanities scholars. It is successful in offering an environment where researchers can access a variety of different text analysis tools and utilize them to process their own text collections. Research with JiTR is now attempting to provide an environment for collection management. JiTR aims to become a repository of collections where researchers can maintain their collections in a format that is compatible with the TAPoR tools, and new digital tools provided within the JiTR environment, including document management, open source crawling and scraping, tools for linguistic processing, and text analysis (Rockwell, 2007). TAPoR and JiTR are ambitious projects. They provide not only collection management, but are equipped with a series of tools that were developed in conjunction with the platform. TAPoR is a great research as well as teaching tool, as it is often the way that new scholars are introduced to text analysis.

Within the development of Texttiles 2.0 we realized that we would be creating a generalized platform that could support not only the Texttiles 2.0. browser but also a series of visualizations that would require the use of a similar information communication and maintenance infrastructure. The work in collection management and the API has started to pay off, as we have realized that this same data crunching, combined with simplified communication, can be the basis for a number of small experimental interfaces. Every prototype that we developed required some form of data management and processing, which was developed at the time with the current prototype in mind, and the purpose of solving immediate needs. With Texttiles 2.0, we took a step back and considered the potential long-term outcome of the project. We planned the collection management and API with future projects in mind. Building this part of the project as infrastructure that can be re-used in other visualization projects, we have developed BubbleLines as an example of what this technology can do.

BubbleLines

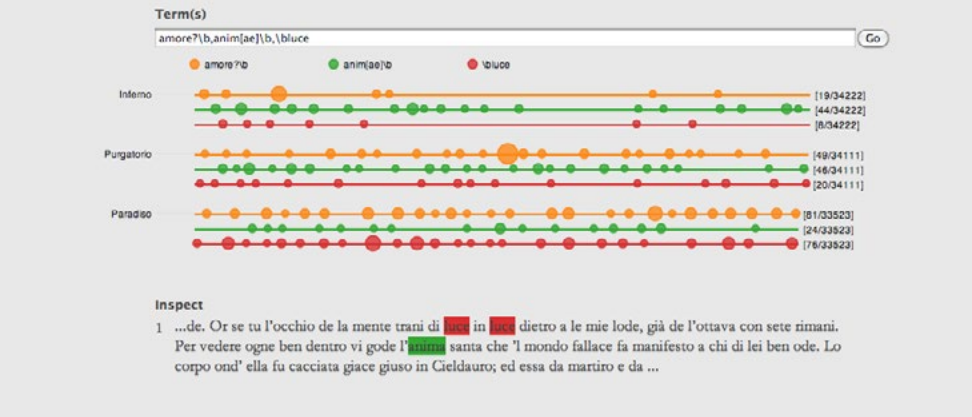
A simple but powerful visualization like BubbleLines can retrieve specific documents within the database. BubbleLines only deals with the display of that information in its

Giacometti, Alejandro, Ruecker, Stan, & Fiorentino, Carlos. (2012). Showcase Browsing with Texttiles 2.0 and BubbleLines. *Scholarly and Research Communication, 3*(2): 020121, 8 pp.

specific way. The visualizations can be developed exclusively in HTML, and JavaScript still allows them to run in any web browser.

Specifically, BubbleLines makes a request for a number of individual texts, and performs a search for a particular term on each one of them. The visualization contains several kinds of information. The horizontal line shows the relative length of a text as compared to each other. Bubbles are displayed on the line where occurrences of the queried term appear. Finally, the size of the bubble represents the concentration of occurrences at that location. The user can click on each bubble to see the context of the occurrences (Figure 1).

Figure 1: The BubbleLines interface



We believe that BubbleLines represents a new type of task-specific visualization that can aid research tasks by providing an additional layer of representation, in combination with a platform that supports management and exploration of collections. The Texttiles browser allows for general purpose exploration of the collection, making use of available metadata. BubbleLines and visualizations that present additional analysis of texts will allow researchers to acquire knowledge about items in the collections that might not be representable in metadata.

Conclusion

We envision a scenario where users perform a series of research tasks using the rich-prospect interface. Through a series of display manipulations that present aspects of the collection, the user will uncover patterns and reveal associations within items in a collection. The user would finally arrive at a subset of texts that hold particular interest for the task at hand. The researcher would subsequently use a variety of different task-specific visualizations to learn and understand this subset. These visualizations would present the subset in a fresh perspective, highlighting characteristics of each item that are not readily specified within the item's metadata.

We would like to offer simple visualizations that would further allow the researcher to investigate this subset of works by envisioning and comparing them in a variety of ways. BubbleLines, for example, would allow a researcher to compare the distribution of frequent terms within a limited set of texts. This investigation would lead to the acquisition of a very specific kind of new knowledge about a set of texts. Our hope is that these specific

additional visualizations will help to make Texttiles into a rich-prospect platform for doing algorithmic criticism (Ramsay, 2003), where scholars subject document collections to an interpretation that is based on emergent patterns from the text.

The back-end service, the API, Texttiles 2.0, and BubbleLines are all being developed concurrently. As we proceed, we hope to learn the challenges and needs of each one of the parts. Our aim is to add an array of visual experiments like BubbleLines in order to expand both the back-end service and the API.

References

Fielding, Roy Thomas. (2000). *Architectural Styles and the Design of Network-based Software Architectures* (Doctoral dissertation). Irvine, CA: University of California.

Giacometti, Alejandro, Ruecker, Stan, Craig, Ian, Dersken, Gerry, & Radzikowska, Milena. (2008). Introducing the Ripper Interface for Text Collections. In *Canadian Symposium on Text Analysis (CaSTA) Conference: New Directions in Text Analysis*. Saskatoon, SK: University of Saskatchewan.

Giacometti, Alejandro. (2009). *The Texttiles browser: an experiment in rich-prospect browsing for text collections* (Master's thesis). Edmonton, AB: University of Alberta.

Given, Lisa M., Ruecker, Stan, Simpson, Heather, Sadler, Elizabeth, & Ruskin, Andrea. (2007). Inclusive interface design for seniors: Image-browsing for a health information context. *Journal of the American Society for Information Science and Technology, 58*(11), 1610-1617.

Google. (n.d.). *The Expando Class. Google App Engine*. URL: http://code.google.com/appengine/docs/python/datastore/expandoclass.html [November 30, 2010].

Ramsay, Stephen. (2003). Toward an Algorithmic Criticism. *Literary and Linguistic Computing, 18*(2), 167-174.

Rockwell, Geoffrey. (2003). What is Text Analysis, Really? *Literary and Linguistic Computing, 18*(2), 209-219.

Rockwell, Geoffrey. (2007). Mashing Texts: New Methodologies in Digital Textuality. URL: http://tada.mcmaster.ca/Main/MashTextsProposal [November 7, 2010].

Rodriguez, Alex. (2008). *RESTful Web services: The basics. IBM developerWorks*. URL: https://www.ibm.com/developerworks/webservices/library/ws-restful/ [November 30, 2010].

Ruecker, Stan. (2003). *Affordances of prospect for academic users of interpretively tagged text collections* (Doctoral dissertation). Edmonton, AB: University of Alberta.

Ruecker, Stan. (2006). Experimental Interfaces Involving Visual Grouping During Browsing. *Canadian Journal of Library and Information Practice and Research, 1*(1). URL: http://journal.lib.uoguelph.ca/index.php/perj/article/viewarticle/142/177 [November 7, 2010].