SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Date: June 24, 2009

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Peter Bergström

ENTITLED

Augmenting Digital Libraries Using Web-Based Visualizations

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN COMPUTER ENGINEERING

Thesis Advisor

Thesis Reader

Department Chair

Augmenting Digital Libraries Using Web-Based Visualizations

by

Peter Bergström

Submitted in partial fulfillment of the requirements
for the degree of
Master of Science in Computer Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 24, 2009

Augmenting Digital Libraries Using Web-Based Visualizations

Peter Bergström

Department of Computer Engineering
Santa Clara University
June 24, 2009

ABSTRACT

Web-based digital libraries have sped up the process that scholars use to find new, important research papers. With powerful searching capabilities and rich bibliographic meta data, a digital library is a great resource. Unfortunately, current digital libraries are limited by their inadequate webpage-based paradigm, and it is easy for even the most experienced scholar to get lost in the vast quantity of published material available. A paper and its immediate references are shown on a webpage, but it is not obvious where that paper belongs in the larger context of a field of research.

The main goal of our research was two-fold. First, to develop and test the effectiveness of an application that can augment a scholar's interaction with a digital library and explore bibliographic meta data using a defined set of visualizations. These visualizations needed to provide different levels of visibility into a paper's citation network without losing focus of the currently viewed paper. The application, called *PaperCube,* needed to support navigation between papers and, to a limited extent, the authors of those papers. The application needed to be able to provide insight into author to author citation networks as well as collaboration relationships.

The hypothesis was that by replacing the traditional webpage-based paradigm with a suite of visualizations that expose various dimensions of bibliographic meta data, the a researcher could gain new insights and find relationships that were not previously apparent. Furthermore, by making the experience spatial, the aim was to see if researchers were able to find what they were looking for more quickly and intuitively.

The second goal was to push the limits of modern web browsers. By using web standards-based technologies, the goal was to explore the possibility of creating a dynamic, desktop-like experience that incorporates rich, interactive visualizations.

PaperCube was written using the SproutCore JavaScript framework, which is geared towards the creation of highly interactive, cloud, or thick-client applications in the web browser. Using only JavaScript and standards-based rendering technologies such as Scalable Vector Graphics, Canvas tag, HTML and CSS, PaperCube allows users to browse a version of the CiteSeer digital library and view paper and author relationships using a set of dynamic visualizations.

Leveraging the powerful features in SproutCore including bindings and observers, PaperCube aims to deliver unparalleled interactivity within the confines of a web browser. Bindings and observers enabled PaperCube to easily implement resolution independence in its visualizations. Resolution independence was key because citation networks can be very large. By adjusting a slider control in the UI, a visualization can be zoomed in using vector-based transforms without needing to explicitly redraw. Furthermore, through the use of bindings, slider controls can dynamically alter the display parameters of the visualizations, permitting the depth of a paper citation network to be adjusted or the thresholds that determine if a node should be displayed as part of the graph changed.

Stemming from the need for a flexible graph API for various views in PaperCube, the SVG-based `NodeGraph` class was created as a generalized solution that can display any type of relational data as an undirected graph. The class is not PaperCube-specific and could be easily integrated into other applications.

PaperCube was validated through a user study which showed that it was very useful when it comes to augmenting digital library search by reducing the "cognitive load" put on a scholar and aiding the "discoverability" of new research material. Furthermore, it was shown that participants thought that it was "visually exciting and intuitive" application and an "amazing example of the apps that well be seeing on the web in a couple of years."

Acknowledgements

I would like to thank my graduate advisor at Santa Clara University, Darren Atkinson, for all his help and support over the past year. From the idea that I had a year ago, to the final product, he has been able to give me perspective and focus. I thank Darren for all his work reading, editing, and giving me valuable feedback on my writing. The resulting application has taken on a life of its own, well beyond what I originally thought possible mainly due to his suggestion to make PaperCube web-based. Originally I pressed to make it a desktop application, but was convinced to make it web-based and in turn, pushed the limits of web browser technology further than I thought possible.

I also want to thank everyone who participated in the user survey—Seth Shostak, May-Li Khoe, Alex Wright, Nan Shostak, Ben Krasnow, Ben Blake, Dan Lewis, Stuart Hastings, Melissa Chan, August Joki, Mark Slater, Peter Leroe-Munoz, Joshua Dickens, Bengt Bergstrom, Sally Scrutchin, Lisa Shaner, Teresa Tsui, Onar Vikingstad, David van der Bokke, Ryan Schenk, Juan Pinzon, Brian Dote, Bill Humphries, Cameron Chrisman, Keri Ahlgren, Jason Levy, Drew Johnson, Diana Higuera, Grace Chung, JoAnne Holliday, Retesh Shah, Rosie Wacha, James Mauss, Georg Aptiz, Shashi Ramchandani, Ernest Prabhakar, and Katie Mihaly. You all gave me very thoughtful, detailed, and honest insights into PaperCube and how you view digital libraries. Not only were there validation of my hypotheses and surprising results, but also, unique ideas about how to use PaperCube in ways that I never imagined.

I would also like to thank my mother, Margaretha, and my father, Bengt, for their encouragement to pursue my own interests throughout my life. You have never pushed me to do anything that I did not want and by giving me the freedom and support to do what I have wanted, I am forever grateful. I love you both very, very much.

I also want to thank my friends, and coworkers who have put up with me being distracted and steadfastly focused on getting my thesis done. If I have bored anyone by talking incessantly about my research, I apologize.

Table of Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Problem and Motivation

When a scholar develops new concepts, ideas, and inventions, he or she most often builds upon, and expands beyond, previously published research. Therefore, a scholar needs to find relevant research material, most commonly in the form of articles, papers, or books, and then forage for other works through references. Even if the person is an experienced and established scholar in a field of research, finding relevant material can be a difficult and time-consuming task. Even worse, for a newcomer to a field, without proper guidance, finding research material can be near impossible.

Furthermore, the rate at which scholarly papers are published is accelerating. The National Science Foundation National Science Board and United States (2008) has shown that the world-wide publication rate of science and engineering research has grown at a steady annual rate of 2.3% between 1995 and 2005. Also, between 1988 and 2005, the amount of cross-institutional author collaboration grew from 40% to 61%. Since the amount of published research material is growing and the interconnectedness of author collaboration is becoming increasingly tangled, it is important to create new methods to aid the discovery of relevant research.

1.1.1 Digital Libraries to the Rescue

During the past forty years, the main mechanism for finding relevant research has shifted from the printed page to the computer screen. Even before advent of the World Wide Web, digital libraries in other forms were available either as stand-alone systems or through proprietary means via the early Internet. Services such as DIALOG, BRS, and Chemical Abstracts (Lynch, 2005) were developed beginning in the 1960s. However, these services were specialized and, due to the nature of the technology at the time, inaccessible to most.

Once the web took off in the mid 1990s, the way that information and knowledge was disseminated changed drastically, not only for digital libraries, but in general. Digital libraries have taken full advantage of the web to make scholarly research easy to access. In the field of computing, web-based services such as the ACM Digital Library (http://portal.acm.org) and IEEExplore (http://ieeexplore.ieee.org) have revolutionized how researchers access the vast quantity of available publications. Service covers other fields such as PubMed (http://www.ncbi.nlm.nih.gov/pubmed) for medicine, Westlaw (http://www.westlaw.com) for law, PsychINFO (http://www.apa.org/psycinfo) for psychology. These services offer powerful search capabilities that allow scholars to find academic papers by filtering across a multitude of bibliographic meta data fields such as title, abstract, publication venue, publication year, keywords, authors, among many others. However, one downside of many of these digital libraries is that they are backed by professional organizations that disallow access without a subscription fee.

More recently, as the web has become more widely used, it has created an age of more open and free knowledge. The web has enabled everyone to find information easily, but also, it has empowered people to create and disseminate knowledge in a far easier manner than ever before. Due to the effort of search engines such as Google and collaborative ventures such as Wikipedia, much of the almost unlimited quantity of information created by humanity is now accessible and searchable by all. The web may offer freely available information, but not all of it should be trusted. Fee-based digital libraries do have one advantage over the services that are freely accessible; they are reviewed for accuracy and correctness, something that is very important to scholars.

In terms of publicly available digital library services, Google Scholar (http://scholar.google.com) is the most well-known. It allows anyone to search the web for scholarly publications and also gives a limited amount of insight into the previously mentioned fee-based digital libraries. Another service is CiteSeerX (http://citeseerx.ist.psu.edu), a newly developed version of the original CiteSeer (http://citeseer.ist.psu.edu; Giles et al., 1998), a web-based digital library from Penn State University. This new version of CiteSeer contains more articles—at the time of writing, 1,366,867 articles indexed and 26,435,805 citations—and more importantly, more accurate bibliographic meta data than the previous version.

1.1.2 The Limitations of Digital Libraries

Unfortunately, although the web has made it easy to access publications, it has not solved one major problem: with the amount of knowledge available, sensory overload is a very real possibility.

Due to this, the actual information a scholar is searching for can be lost in the clutter of irrelevant material. Searching for papers in a digital library makes it easy to narrow down the search space quite effectively, but the webpage-based user interfaces currently in use are not always adequate. The current paradigm of webpage-based searching, viewing, and foraging has its limitations; while focused on a paper, represented as a webpage in the DL with links to its referenced papers, it is non-trivial for a user to see where the paper belongs within the wider context. Without navigating away from the original paper, it is not possible to view the references of a referenced paper, and so on, or in other words, the overall citation network. In order to view a reference, a researcher has to click on a link that either makes the browser navigate away from the current page or opens a new browser window. Neither of these solutions are ideal because the researcher has left the original paper behind.

1.1.3 We Need A New Paradigm

Therefore, it is important to allow for the navigation of bibliographic meta data in such a manner that allows a paper to be focused yet allow for the easy and seamless access to its surrounding citation network without losing focus and context. To create a practical system that allows for this and is web-based would be quite useful to scholars. Such a system does not necessarily need to replace currently existing web-based digital libraries, but rather augment them. Simply by replacing the webpage that represents a paper within the typical digital library of today with a rich, interactive visualization-based user interface would be compelling. However, it is important to remain web-based so that it is possible to seamlessly switch back and forth from the webpage in the digital library to the visualization-based interface.

What does it mean to view a paper and its citation network without losing focus and context? Imagine that the researcher using a digital library service is a pilot inside of a helicopter and the ground is made up of a paper and its citation network. A webpage-based service can be thought of as sitting in the helicopter on the tarmac; out the window, the pilot can see the pavement with all its cracks and imperfections. If there are no obstructions, the pilot can even see almost three miles to the horizon. However, sitting on the ground, the pilot does not know what lies beyond the immediate area and does not necessarily know what the surrounding area looks like. Going back to the digital library, if the researcher want to see more of the paper's citation network, he needs to step back. The helicopter ascends straight up to a thousand feet. The pilot can still see where he took off from, but he can also see more of the surrounding the area. He may not be able to see the small cracks in the tarmac, but the pilot sees the pavement well enough to know what it is. It is now possible to see thirty-nine miles in every direction. Now the researcher can not only see the paper, but its references and those references' references, and beyond. If the pilot wants to see more, he can ascend to ten thousand feet. Now he can see 123 miles to the horizon and everything on the ground is small, but details are still visible. In terms of a citation network, the papers are not very detailed, but it is still possible to figure out what they are. If the researcher want to see the detailed information about any paper, he can always descend, just like the helicopter pilot, towards the paper to reveal additional detail.

1.2 General Goals and Requirements

The main goal of this work was two-fold. First, to develop and test the effectiveness of an application that allows a scholar to interact with a digital library and explore bibliographic meta data using a defined set of visualizations. These visualizations had to provide different levels of visibility into a paper's citation network without losing focus of the currently viewed paper. The visibility into the paper's citation network needed to be accomplished without losing focus of the paper that is being viewed. This concept is commonly known as *Focus+Context* (Furnas, 1986) in visualization. The application needs to support navigation between papers and, to a limited extent, the authors of those papers.

The hypothesis was that, by replacing the traditional webpage-based paradigm with a suite of visualizations that expose various dimensions of bibliographic meta data, a researcher could gain new insights and find relationships that were not previously apparent. Furthermore, by making the experience spatial, the aim was to see if researchers were able to find what they were looking for more quickly and intuitively. The available bibliographic meta data in a digital library is very interesting to explore and by looking at papers' references and citations, a researcher should be able to easily explore a field of study and get acquainted with what papers and authors are the most important in a given field.

The second goal was to push the limits of modern web browsers. By using web standards-based technologies, the goal is to explore the possibility of creating a dynamic, desktop-like experience that incorporates rich, interactive visualizations.

Below is a detailed list of the specific goals for this work that includes the design principles of the experimental application, called *PaperCube*, as well as validation and technical requirements.

- **Navigate Paper Citation Network Relationships**

  The paper citation network had to be navigable in both directions of the citation relationship, which means that it is possible to see the papers that a paper as referenced, but also what papers reference it. Also, PaperCube needed to allow users to navigate the author citation and collaboration relationships that can be implied via the available bibliographic meta data.

- **Provide a Suite of Visualizations**

  PaperCube had to incorporate a defined set of visualizations of papers and authors that provide for different perspectives and levels of detail. Citation network relationships needed to be rendered using various visualizations that allow for the easy representation of not only a paper's immediate references or citations, but also its extended citation network organized in various ways including chronologically. For author relationships, the visualizations allow for showing interconnected author to author citation and collaboration relationships. Therefore, the goal was to develop at least five paper views and a smaller number for authors.

- **Switch Between Views Without Losing Focus**

  Researchers using PaperCube needed to seamlessly switch between views that visualize the different dimensions of the same author or paper. Furthermore, when switching views, the same paper or author needed to remain in focus automatically without any additional user action.

- **Allow User to Adjust the Amount of Data Shown**

  Since visualizations of networked information have the potential to render a large amount of data, the visualizations had to be easily customizable with a set of threshold parameters that put constraints on what data should be displayed. Instead of devising an advanced system to automatically determine what should be shown, PaperCube deferred to the user. The user is most likely at least somewhat familiar with the meta data and should be able to determine what should be shown. A system can be engineered to be smart, but people are more intelligent than any metric that could be devised. Therefore, PaperCube needed to expose UI elements such as slider controls that a researcher could adjust and, in turn, have the views update automatically in real time.

  The thresholds that needed to be incorporated included the number of levels of the paper or author citation or collaboration network should be explored and rendered, the number of references or citations a paper or author had, and the number of collaborators for an author.

- **Resolution Independence**

  Even through the views allowed the amount of data rendered to be adjusted, there are cases when that is not enough. Therefore, all of the views needed to be able to be resolution independent so that any portion of a visualization can be dynamically zoomed in to reveal more detail. If technically possible, the views were not to redraw when zoomed in, out, or panned around.

- **Provide Basic Search Functionality**

  Even though the goal was to use an existing digital library, some basic searching functionality in PaperCube itself was required. However, the search needed only to be advanced enough to return papers or authors based based on searching against a paper's title, abstract, subject, publication year, or an author's name.

- **Use Existing Bibliographic Meta Data**

  The purpose of this work was not to create a new set of bibliographic meta data. Therefore, the use of an existing digital library with publicly available meta data was desired. CiteSeer was ultimately selected as the data source.

- **Access to Source Digital Library**

  PaperCube needed to provide access to the currently viewed paper at the source digital library as well as provide a link to the full-text PDF or PostScript file of the paper, if possible.

- **Thick-client Application On The Web**

  It was paramount to create a solution that was entirely web-based. The web is the best way to disseminate information today and it would be a great disservice not to use the web. Since the goal of this work was to explore new ways to represent bibliographic meta data, a desktop-like experience would be the most appropriate, yet still remain web-based. Even on the desktop, it can be challenging to develop an application with rich visualizations that performs well.

  In pursuit of creating something that was forward-looking, it was decided that PaperCube would utilize technologies backed by web standards. Therefore, the client component of PaperCube was written using JavaScript, Scalable Vector Graphics, Canvas tag, CSS, and HTML. However, due to cross browser concerns, modern browsers that support the previously mentioned technologies were supported. Therefore, Firefox, WebKit, Safari, and Google Chrome are fully supported but Internet Explorer is not. Since PaperCube is experimental, this was an acceptable compromise in order to not be limited by the lowest common denominator.

- **Validation Through A User Study**

  PaperCube needed to be validated through a user study that assessed the effectiveness of its design goals and methods. The study should assess the efficacy of the major design goal as well as the individual views and visualization methods. The participant pool should consist of past or present graduate students, researchers, and professors from several fields of study.

1.3 Overview of Remaining Chapters

The second chapter in this work gives the reader with some specific terms that are used in this work as well as background into hypertext, user interface design, visualization methods, and web technologies. The third chapter steps through two use case scenarios showing how PaperCube works when searching, viewing, and browsing paper citation networks and author collaboration networks. The fourth chapter details some of the major common features of PaperCube that is used throughout all the different views and visualizations such as searching, the toolbar, zooming and panning, the fan popup menu, and more.

The fifth chapter goes through the motivation, goals, and functionality of the paper views and visualizations. The sixth chapter follows the same pattern and steps through all the author views.

The seventh chapter is a high-level architectural overview of PaperCube as a whole followed by the eight chapter that discusses the server-side architecture including the database, the client-server interface, and the JSON data format used for client-server communication. Chapter nine covers the client-side architecture including the SproutCore JavaScript framework, implementation details, and the generic `NodeGraph` API developed as part of this work.

The tenth chapter presents a study of one of the visualization methods, Circle View, in terms of cognitive psychology and discusses how the design of Circle View visualization was altered taking into account the findings. The eleventh chapter presents the user study, its goals, and results followed by a discussion.

The twelfth chapter discusses related work in the field of digital library visualization and navigation. The final chapter presents the conclusion of this research.

Chapter 2

Background

The purpose for this chapter is to cover background information that can be informative for readers who are not familiar with hypertext, digital libraries and citation networks, user interfaces and visualization methods, and web technologies. Throughout the remaining chapters, in order to avoid confusion, a small set of terms are used to describe concepts related to viewing and browsing digital libraries.

PaperCube builds upon previous research in visualization, human-computer interaction and hypertext. Hypertext, as explained by Wright (2008), is most often narrowly defined as the technology behind the *World Wide Web,* or more colloquially, the *web.* The system was originally proposed in 1989 and developed in the early 1990s by Tim Berners-Lee while at CERN (1989). The explosive growth and the significance of the web over the past two decades has made the term hypertext enter the public consciousness as being synonymous to the web itself. However, hypertext is more far-reaching and older. The web is a hypertext system, but hypertext is not only made up of the web.

The concept of hypertext traces to Vannevar Bush who wrote the forward-looking article, "As We May Think" (1945) illustrating a theoretical machine called the *Memex.* The Memex was depicted as a device that a person could use to explore a large microfilm-based research library. As the person foraged for information and found interesting material, associative trails of links and notes could be added as well as existing trails created by others could be explored. The device also allowed a person to go back and forth in the trails and create new side trails that could later be followed. This concept is very similar to modern web browsers that allow users to open tabs to view new information and the visited pages are seamlessly recorded in the browser's history. However, Bush's vision of allowing users to publish and share these trails has not been fully realized yet.

Twenty years later, in 1965, the term *hypertext* was coined by Ted Nelson (1990) to describe the idea of going beyond static text and offering interactivity to easily jump from one idea to another, something that was included in the Memex, but not explicitly named.

Three years later in 1968, Douglas Engelbart at the Augmentation Research Center at the Stanford Research Institute in Menlo Park, California, took the first step to creating the UI paradigms that we all use today. Engelbart and English (1968) developed the On-Line System or NLS, a networked hypertext system that had a basic graphical interface with bit-mapped elements and allowed system interaction with a mouse. This system was demonstrated at a conference in San Francisco that featured the newly invented mouse, teleconferencing, email, and hypertext.

2.1 Specialized Terminology Used in this Work

When describing the structure and implementation of citation networks and digital libraries, a set of domain-specific terms are used throughout this work that may not be in common usage. The most important disambiguation of terms is regarding the directionality of paper relationships. Papers list other papers in their bibliography which represents one direction of the relationship. Within the context of this work, the phrase "paper A *references* B" means that that paper A lists paper B in its bibliography. The inverse is "paper B *is cited by* paper A" which means that paper B is listed in paper A's bibliography. Therefore, this work uses the terms "references" and "citations" as the words to determine the direction of these relationships. However, when referring to the general network of papers that can be implied from a paper's bibliography, regardless of direction, it is referred to as a "citation network" since its is a generally accepted term.

Certain terms used when describing PaperCube's user interface elements may cause confusion. First, PaperCube contains a set of different visualizations that are used to display paper or author relationships. Each of those visualizations are referred to as *views* in order not to confuse them with *visualization* methods, which may be shared between different views. Also, for simplicity, when referring to a specific rendered element that represents a paper or author as part of a visualization, it is referred to as a *node.* In other cases, when talking about the specific instance of an author's or paper's bibliographic meta data, the instance, regardless of type, is referred to as an *item.*

2.2 Visualization Methods

PaperCube uses several commonly known user interface paradigms and visualization methods as the inspiration for the views created to show paper and author relationships. Most of these methods have been used in previous applications that visualize bibliographic meta data as well as in a diverse array of applications that show things such as disk usage and calendars. Due to the specialized needs of digital library visualizations, the following visualization methods do not necessarily map directly on to the implementations used in this work but they were used as a starting point.

One pivotal concept used in many of these visualizations is what is called *Focus+Context* which was first described by Furnas. Card, Mackinlay, and Shneiderman (1999) best describe the concept:

> Focus+Context start from three premises: First, the user needs both overview (context) and detail information (focus) simultaneously. Second, information needed in the overview may be different from that needed in detail. Third, these two types of information can be combined within a single (dynamic) display, much as in human vision.

2.2.1 Fisheye Views

In 1986, Furnas introduced the concept of using a *fisheye* view as the basis for a UI in which the center contains more detailed information than the periphery. A fisheye view follows closely how the human eye works—objects placed at the center of our vision are naturally more detailed than objects that are at periphery. Also, the idea of dropping the level of detail when moving further away from the central focus point is analogous to how humans view the real world when it comes to information: The closer to home, the more it matters. For example, news papers or programs on television will feature less important local area stories prominently but only cover highly important news stories from other areas.

Calendaring applications use fisheye views to rendered more detail on the selected date. In Figure 2.1, the left screenshot is from Furnas' 1986 paper and on the right, the web-based Yahoo Calendar application as it exists in 2009.

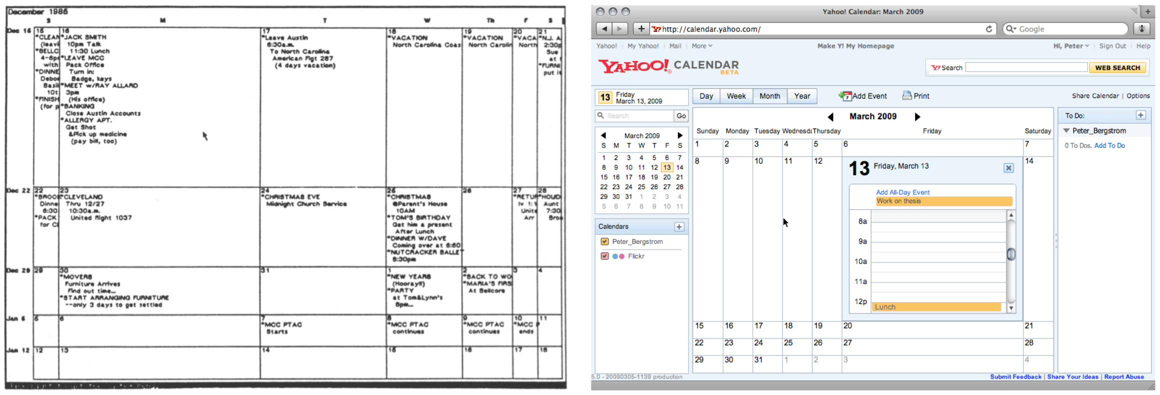


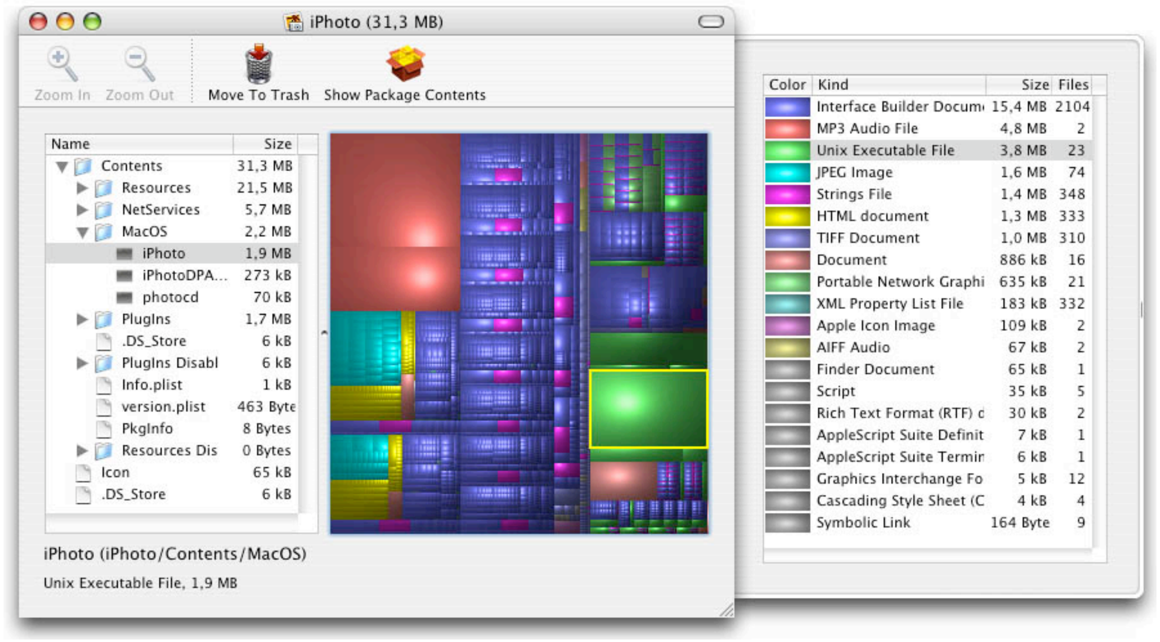
Figure 2.1: Example of a fisheye calendar featured in Furnas' paper compared to Yahoo Calendar.

A common use of fisheye views specifically within the realm of digital library visualizations is to use it to focus on a paper and show where it belongs within the context of its citation network. The visualization has a focused paper at the center and subsequent levels of paper references or citations are shown in decreasing detail. This makes it natural to pay more attention to the focused paper in the middle yet see enough context of where it belongs. From that information, it is possible to judiciously choose where in the citation network to go next.

2.2.2 Tree Maps

The concept of a *tree map* was introduced by Johnson and Shneiderman (1991) as a novel visualization method to show hierarchies in a two dimensional, space efficient manner. The visualization method maps out a hierarchy onto the largest possible rectangular region inside of a containing rectangle. Subtrees are contained within its parent rectangle and each subsequent level of the tree gets smaller and smaller. Due to this space efficient layout method it is possible to show very large sets of data and infinitely zoom in on parts of the tree map to reveal more detail.

Some of the very diverse uses for tree maps include *Disk Inventory X*, a disk usage utility (http://www.derlien.com) shown in Figure 2.2. Other uses for tree maps include photo browsing using the *PhotoMesa Image Browser* (Bederson, 2001), the innovative and influential *Newsmap* web application that aggregates Google News stories in real time (Ong et al., 2005), and they have been used by the New York Times for informational graphics (Quely and McClain, 2008).



Name	Size
▼ Contents	31,3 MB
▶ Resources	21,5 MB
▶ NetServices	5,7 MB
▼ MacOS	2,2 MB
iPhoto	1,9 MB
iPhotoDPA...	273 kB
photocd	70 kB
▶ PlugIns	1,7 MB
.DS_Store	6 kB
▶ PlugIns Disabl	6 kB
Info.plist	1 kB
version.plist	463 Byte
PkgInfo	8 Bytes
▶ Resources Dis	0 Bytes
Icon	65 kB
.DS_Store	6 kB

iPhoto (iPhoto/Contents/MacOS)
Unix Executable File, 1,9 MB

Color	Kind	Size	Files
🟦	Interface Builder Docum	15,4 MB	2104
🟥	MP3 Audio File	4,8 MB	2
🟩	Unix Executable File	3,8 MB	23
🟦	JPEG Image	1,6 MB	74
🟪	Strings File	1,4 MB	348
🟨	HTML document	1,3 MB	333
🟦	TIFF Document	1,0 MB	310
🟥	Document	886 kB	16
🟩	Portable Network Graphi	635 kB	21
🟦	XML Property List File	183 kB	332
🟪	Apple Icon Image	109 kB	2
🟨	AIFF Audio	67 kB	2
⬜	Finder Document	65 kB	1
⬜	Script	35 kB	5
⬜	Rich Text Format (RTF) d	30 kB	2
⬜	AppleScript Suite Definit	7 kB	1
⬜	AppleScript Suite Termir	6 kB	1
⬜	Graphics Interchange Fo	5 kB	12
⬜	Cascading Style Sheet (C	4 kB	4
⬜	Symbolic Link	164 Byte	9

Figure 2.2: A screenshot of Disk Inventory X, a Mac OS X application that shows disk usage using a tree map visualization.

2.2.3 Hyperbolic Tree

Lamping, Rao, and Pirolli (1995) use a fisheye display of hierarchical information in the form on a tree. Their approach, called a *hyperbolic tree*, uniformly lays out the tree around a central point where the root node is located and the branches lead to other nodes. The node at the center is more emphasized and easy to read than the nodes at the periphery. The circular layout allows for a very efficient way to display a large amount of information and it allows the focus to be smoothy shifted from one node to the next. The hierarchical nature of trees make hyperbolic trees in particular very useful for showing things such as organizational charts, as was one of the examples in Lamping and Rao's paper shown in Figure 2.3.

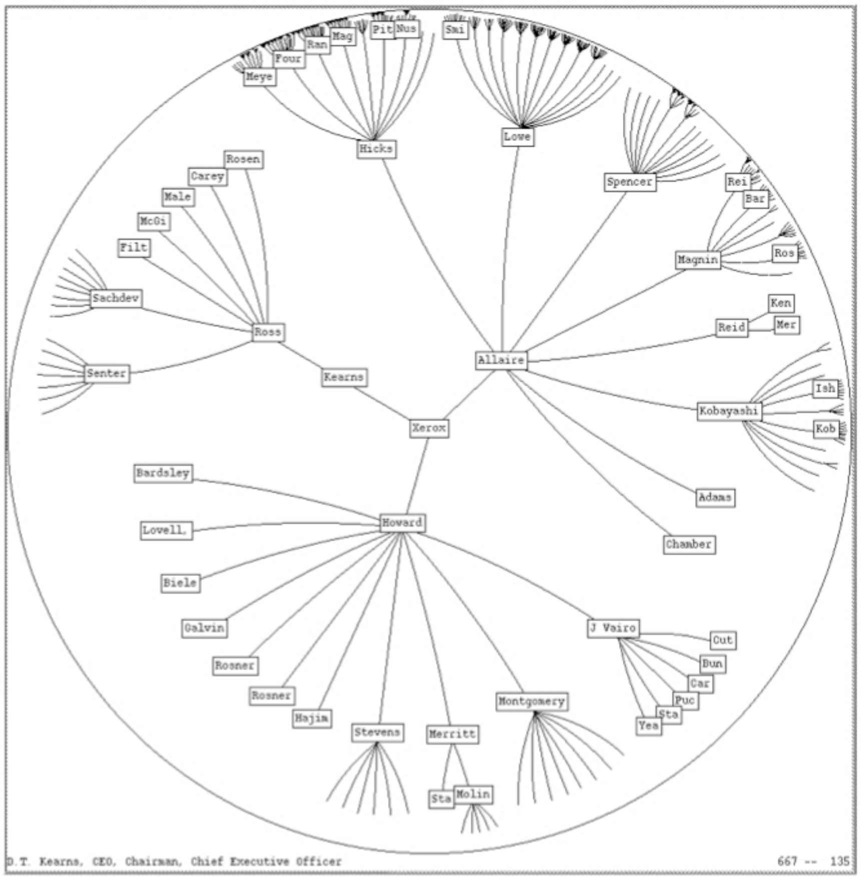


Figure 2.3: The organization chart example pointed out by Lamping and Rao in their paper.

The hyperbolic layout method is very effective when displaying a hierarchical tree, but it does not perform well when displaying citation networks because it contains cycles and cross references. Therefore, a strictly defined hyperbolic tree does not work for citation network visualizations but with a minor modification of the data, it is quite effective—by allowing repeated references to be treated as separate nodes, the citation network can be rendered.

2.3 Web Technologies

This work not only describes the features, concepts, and validation of PaperCube as a way to augment digital libraries, but also touches on technical implementation details. PaperCube is web-based and in order to provide rich interactivity, it needs not only HTML and CSS but also uses a rich programming language. In this case, instead of using proprietary technologies such as Flash (http://www.macromedia.com/software/flash) or Flex (http://www.adobe.com/products/flex) from Adobe Systems or Silverlight (http://silverlight.net) from Microsoft, it is written entirely using JavaScript. Furthermore, its visualizations are created using standards-based rendering technologies such as Scalable Vector Graphics and the Canvas tag.

2.3.1 JavaScript

The most common method to enable interactivity within the web browser is through the use of JavaScript. JavaScript is a scripting language that can be natively executed by web browsers and allows for the dynamic manipulation of elements within a HTML document. It is interesting to note that since all web browsers since the mid-1990s have shipped with a JavaScript engine, is the most ubiquitous programming language with almost every single personal computer in use today being able to run it. No matter what operating system, it is more or less guaranteed that there will be an installed web browser that can run JavaScript-based applications.

JavaScript was originally developed by Brendan Eich (Champeon, 2001) at Netscape and was included into the Netscape web browser in 1995. Although the name JavaScript implies a direct connection with Java, the name was chosen for marketability to get developers to use it. Also, in terms of syntax, JavaScript was designed to look similar to Java. However, in reality, it is more closely related to the Scheme and Self programming languages. Also, rather than being class-based like Java, JavaScript is prototype-based with some class-like syntactic sugar added on to make it easy for Java developers to transition to the language. Although almost always referred to as JavaScript, the language is officially defined as the ECMA-262 (ECMA International, 1999) specification, also known as ECMAScript.

Over the years, JavaScript has been used augment HTML-based web pages with additional functionality, but the extent has been limited to form validation, basic widgets such as date pickers, and AJAX-based server communication. Many JavaScript-based and frameworks exist such as Prototype (http://www.prototypejs.org), jQuery (http://www.jquery.com), Dojo (http://www.dojotoolkit.org), or YUI (http://developer.yahoo.com/yui) that allow developers to augment webpages with additional functionality.

However, in the past couple of years, JavaScript has started to be a viable solution for so-called Rich Internet Applications or RIA. The main reason for this is two-fold. First, web browsers such as Firefox (http://www.getfirefox.com), Google Chrome (http://www.google.com/chrome/), WebKit (http://www.webkit.org), and Safari 4 (http://www.apple.com/safari) have greatly increased their JavaScript execution speeds through just-in-time (JIT) compilation (Resig, 2008). Second, during the past two years, several JavaScript-based RIA frameworks have been developed such as ExtJS (http://www.extjs.com), Cappuccino (http://www.cappuccino.org), and SproutCore (http://www.sproutcore.com). PaperCube uses the SproutCore framework.

2.3.2 Scalable Vector Graphics

Scalable Vector Graphics, or SVG (Ferraiolo et al., 2003), is an open standard defined by the World Wide Web Consortium (http://www.w3.org) for describing two-dimensional vector-based graphics expressed in XML format. It was initially proposed in 1999 but has only recently been natively supported in web browsers. As of 2008, all major browsers except for Microsoft Internet Explorer support native SVG rendering. SVG allows developers to describe shapes, text, color gradients, paths, animation, and interactivity using XML or through programmatic creation using JavaScript.

Since SVG is vector-based, it makes resolution independence possible. This means that defined elements can be scaled without having to explicitly redraw. A conventional bitmap-based graphic cannot be scaled larger without losing fidelity because only so many pixels make up the graphic and by scaling it up, the pixels are made larger and more noticeable. On the other hand, a vector-based graphic does not consist of a matrix of pixels but rather sets of commands that describe to the rendering engine how to display the graphic. Therefore, a vector-based graphic can be transformed and without having to explicitly describe the graphic to the rendering engine, be updated automatically by the browser and as a result, it will look crisp at any resolution.

2.3.3 Canvas Tag

The Canvas tag was initially introduced by Apple, Inc as part of its WebKit browser rendering engine in 2004 as a method to programatically draw bitmapped-based graphics in JavaScript. It was originally created to be used in Mac OS X Tiger's Dashboard application. One example within

Dashboard was the clock widget that used the Canvas tag to draw its analog hour, minute, and second hands. It is now part of the HTML5 recommendation (Hickson, 2009) and is implemented by other browsers in addition to WebKit.

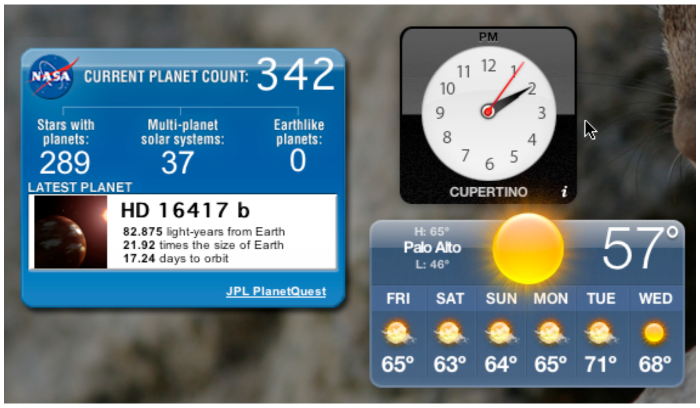


Figure 2.4: Screenshot Mac OS X Leopard's Dashboard showing off several widgets including the clock widget that uses the Canvas tag to animate the second, minute, and hour hands.

The main difference between the Canvas tag and SVG is that it is bitmap, consisting of a grid of pixels, which means it cannot scale without losing fidelity. Also, although it is possible to create most of the same shapes using Canvas as in SVG, it does not support animation and text strings, without the use of font sprites—images that contain letters of the alphabet and are positioned dynamically to make up the strings. Since the Canvas tag is essentially an image, it cannot easily be transformed with redrawing. It is possible to easily rotate, scale, or perform other types of transformations as well as do additive manipulations on the whole or a part of a Canvas element, but more advanced alterations may require a complete redraw.

Chapter 3

Use Case Scenarios

This chapter describes two use case scenarios that demonstrate how PaperCube can be used by researchers. The first scenario describes the case of a researcher searching for papers and browsing paper citation networks. The second shows how, from the found papers in the first scenario, a researcher can find new authors through an author's collaboration networks.

3.1 Scenario 1: Searching and Browsing For Papers

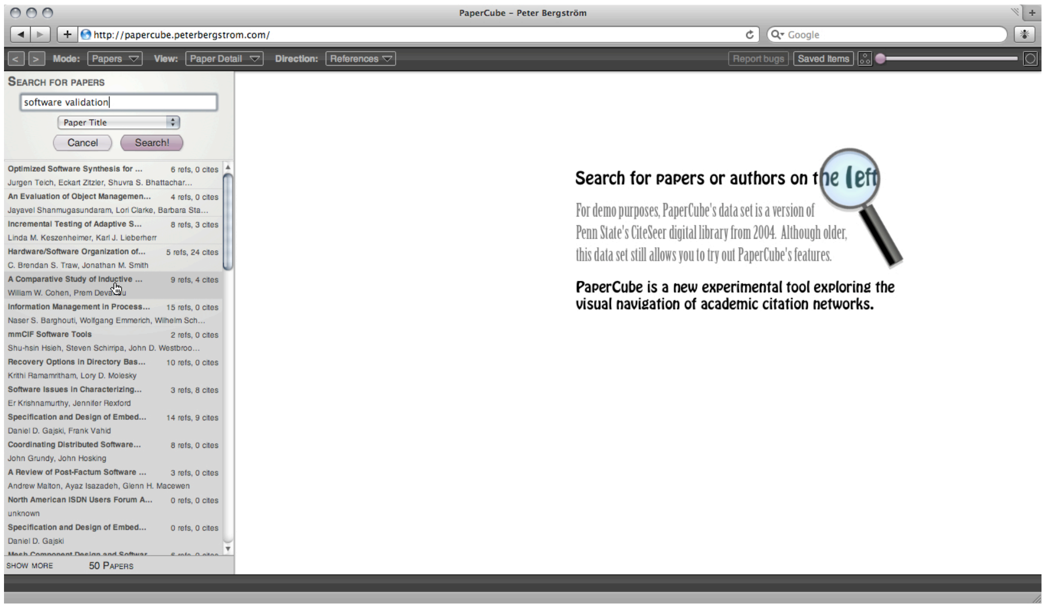
Billy Joe is a Computer Science student at Savannah State University, and under the supervision of his academic advisor, is doing research into software fault prediction.

3.1.1 Search for Papers

Billy Joe launches his web browser and goes to PaperCube's web site. After it loads, he opens the search pane and searchers for papers that contain the term "software validation" in their titles. As shown in Figure 3.1, more than fifty papers show up in the search results. Billy Joe looks over the summary entries for the papers retuned in the search and finds a promising paper, "A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction."

3.1.2 Using Paper Detail View

To view the paper, Billy Joe clicks on the paper summary in the search results list. This makes the paper, "A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction," the focused paper. By default, Paper Detail view is displayed first as shown in Figure 3.2. Billy Joe finds that the paper has two authors, Prem Devanbu and William Cohen. Also, he sees that the paper has nine references and four citations.



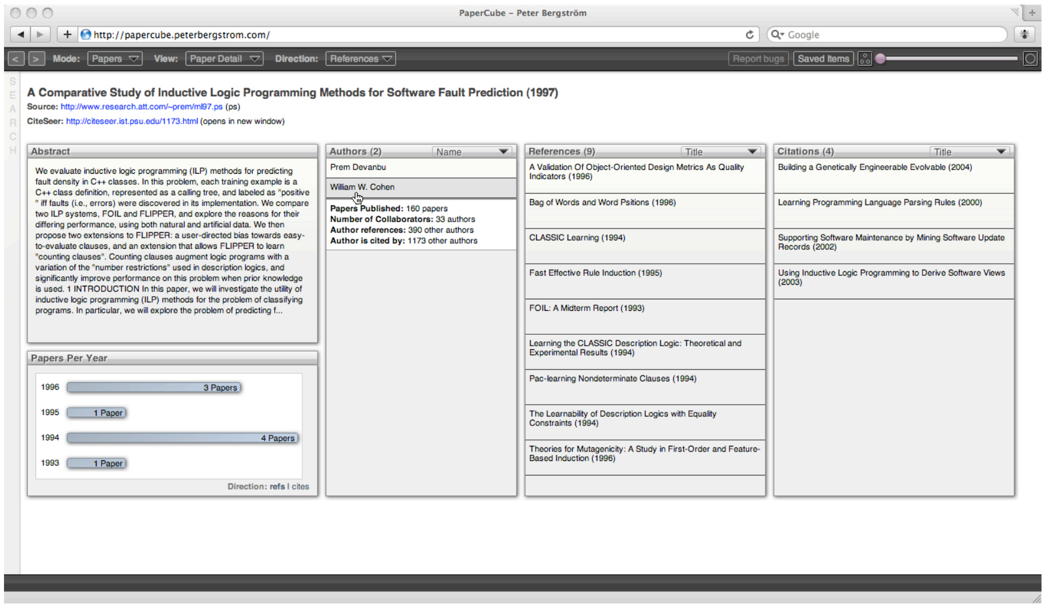
Optimized Software Synthesis for ... 6 refs, 0 cites
Jurgen Teich, Eckart Zitzler, Shuvra S. Bhattachar...
An Evaluation of Object Managemen... 4 refs, 0 cites
Jayavel Shanmugasundaram, Lori Clarke, Barbara Sta...
Incremental Testing of Adaptive S... 8 refs, 3 cites
Linda M. Keszenheimer, Karl J. Lieberherr
Hardware/Software Organization of... 5 refs, 24 cites
C. Brendan S. Traw, Jonathan M. Smith
A Comparative Study of Inductive ... 9 refs, 4 cites
William W. Cohen, Prem Devanbu
Information Management in Process... 15 refs, 0 cites
Naser S. Barghouti, Wolfgang Emmerich, Wilhelm Sch...
mmCIF Software Tools 2 refs, 0 cites
Shu-hsin Hsieh, Steven Schirripa, John D. Westbroo...
Recovery Options in Directory Bas... 10 refs, 0 cites
Krithi Ramamritham, Lory D. Molesky
Software Issues in Characterizing... 3 refs, 8 cites
Er Krishnamurthy, Jennifer Rexford
Specification and Design of Embed... 14 refs, 9 cites
Daniel D. Gajski, Frank Vahid
Coordinating Distributed Software... 8 refs, 0 cites
John Grundy, John Hosking
A Review of Post-Factum Software ... 3 refs, 0 cites
Andrew Malton, Ayaz Isazadeh, Glenn H. Macewen
North American ISDN Users Forum A... 0 refs, 0 cites
unknown
Specification and Design of Embed... 0 refs, 0 cites
Daniel D. Gajski

Search for papers or authors on the left

For demo purposes, PaperCube's data set is a version of Penn State's CiteSeer digital library from 2004. Although older, this data set still allows you to try out PaperCube's features.

PaperCube is a new experimental tool exploring the visual navigation of academic citation networks.

Figure 3.1: Searching for papers containing the term "software validation" and finding a paper.



A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction (1997)
Source: http://www.research.att.com/~prem/ml97.ps (ps)
CiteSeer: http://citeseer.ist.psu.edu/1173.html (opens in new window)

Abstract

We evaluate inductive logic programming (ILP) methods for predicting fault density in C++ classes. In this problem, each training example is a C++ class definition, represented as a calling tree, and labeled as "positive " iff faults (i.e., errors) were discovered in its implementation. We compare two ILP systems, FOIL and FLIPPER, and explore the reasons for their differing performance, using both natural and artificial data. We then propose two extensions to FLIPPER: a user-directed bias towards easy-to-evaluate clauses, and an extension that allows FLIPPER to learn "counting clauses". Counting clauses augment logic programs with a variation of the "number restrictions" used in description logics, and significantly improve performance on this problem when prior knowledge is used. 1 INTRODUCTION In this paper, we will investigate the utility of inductive logic programming (ILP) methods for the problem of classifying programs. In particular, we will explore the problem of predicting f...

Papers Per Year

1996	3 Papers
1995	1 Paper
1994	4 Papers
1993	1 Paper

Direction: refs | cites

Authors (2)

Prem Devanbu
William W. Cohen

Papers Published: 160 papers
Number of Collaborators: 33 authors
Author references: 390 other authors
Author is cited by: 1173 other authors

References (9)

A Validation Of Object-Oriented Design Metrics As Quality Indicators (1996)
Bag of Words and Word Psitions (1996)
CLASSIC Learning (1994)
Fast Effective Rule Induction (1995)
FOIL: A Midterm Report (1993)
Learning the CLASSIC Description Logic: Theoretical and Experimental Results (1994)
Pac-learning Nondeterminate Clauses (1994)
The Learnability of Description Logics with Equality Constraints (1994)
Theories for Mutagenicity: A Study in First-Order and Feature-Based Induction (1996)

Citations (4)

Building a Genetically Engineerable Evolvable (2004)
Learning Programming Language Parsing Rules (2000)
Supporting Software Maintenance by Mining Software Update Records (2002)
Using Inductive Logic Programming to Derive Software Views (2003)

Figure 3.2: Viewing the paper in Paper Detail view.

Billy Joe reads the abstract and thinks that the paper still looks interesting but he wants to see where it belongs in the greater context of the field of software validation.

3.1.3 Using Tree Map View

To see more of the paper's citation network, Billy Joe switches to Tree Map view and by using the sliders at the bottom only shows the papers with at least fifteen citations. Although his focused paper does not have that many citations itself, many of its references have a lot citations. Since those papers have a lot of citations, it may suggest that they are highly regarded and are important.

As Billy Joe moves his mouse cursor over the references, bibliographic meta data is displayed next to the cursor in a hovering window and duplicate instances of the paper being cited is highlighted in green as shown in Figure 3.3.

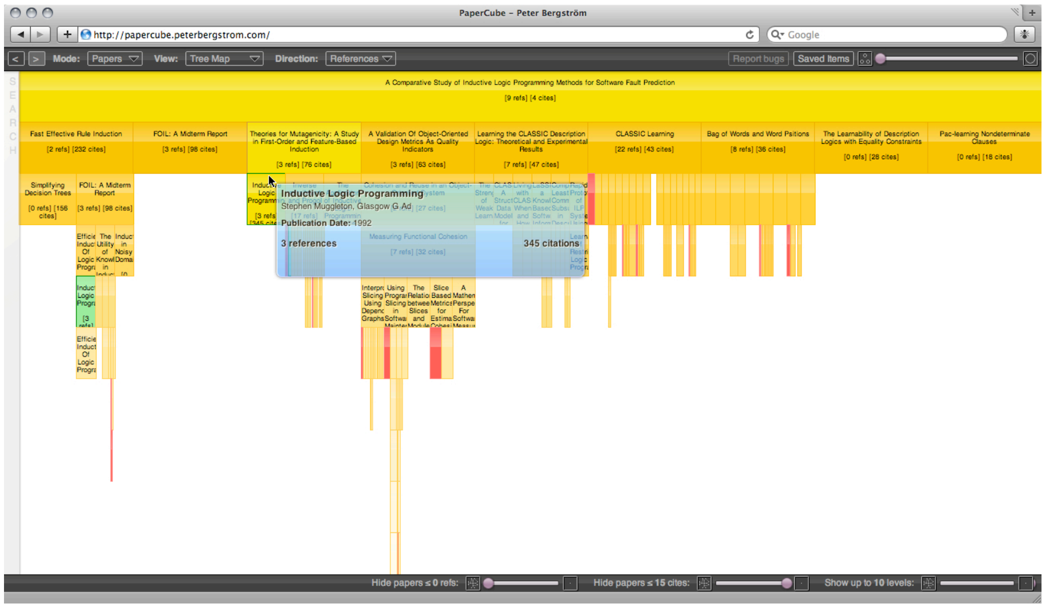

PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Google
Mode: Papers
View: Tree Map
Direction: References
Report bugs
Saved Items
A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction
[9 refs] [4 cites]
Fast Effective Rule Induction
[2 refs] [232 cites]
FOIL: A Midterm Report
[3 refs] [98 cites]
Theories for Mutagenicity: A Study in First-Order and Feature-Based Induction
[3 refs] [76 cites]
A Validation Of Object-Oriented Design Metrics As Quality Indicators
[3 refs] [63 cites]
Learning the CLASSIC Description Logic: Theoretical and Experimental Results
[7 refs] [47 cites]
CLASSIC Learning
[22 refs] [43 cites]
Bag of Words and Word Psitions
[8 refs] [36 cites]
The Learnability of Description Logics with Equality Constraints
[0 refs] [28 cites]
Pac-learning Nondeterminate Clauses
[0 refs] [18 cites]
Simplifying Decision Trees
[0 refs] [156 cites]
FOIL: A Midterm Report
[3 refs] [98 cites]
Inductive Logic Programming
Stephen Muggleton, Glasgow G Ad
Publication Date: 1992
3 references
345 citations
Measuring Functional Cohesion
[7 refs] [32 cites]
Hide papers ≤ 0 refs:
Hide papers ≤ 15 cites:
Show up to 10 levels:

Figure 3.3: Displaying meta data for "Inductive Logic Programming," part of the focused paper's network of citations in Tree Map view.

As he scans Tree Map view for papers, Billy Joe finds a paper, "Fast Effective Rule Induction" also written by William Cohen which has 232 citations. Since the paper has so many citations, Billy Joe decides that he wants to look at the paper later. To save the paper for later viewing, he clicks on the paper's node in the visualization to bring up the fan menu that allows him to save the paper as shown in Figure 3.4.

Billy Joe then finds another paper in the citation network that looks interesting, "CLASSIC Learning" by Michael Frazier. Without zooming in, all of the paper's references are not rendered as

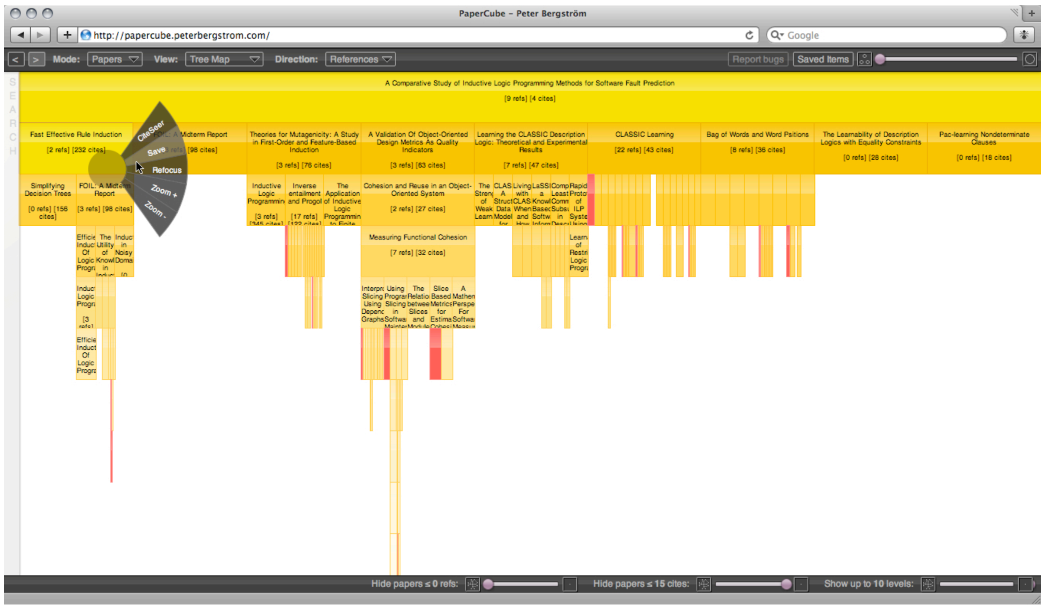

PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Google
Mode: Papers
View: Tree Map
Direction: References
Report bugs
Saved Items
A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction
[9 refs] [4 cites]
Fast Effective Rule Induction
[2 refs] [232 cites]
Midterm Report
[98 cites]
Theories for Mutagenicity: A Study in First-Order and Feature-Based Induction
[3 refs] [76 cites]
A Validation Of Object-Oriented Design Metrics As Quality Indicators
[3 refs] [63 cites]
Learning the CLASSIC Description Logic: Theoretical and Experimental Results
[7 refs] [47 cites]
CLASSIC Learning
[22 refs] [43 cites]
Bag of Words and Word Psitions
[8 refs] [36 cites]
The Learnability of Description Logics with Equality Constraints
[0 refs] [28 cites]
Pac-learning Nondeterminate Clauses
[0 refs] [18 cites]
CiteSeer
Save
Refocus
Zoom +
Zoom -
Simplifying Decision Trees
FOIL: A Midterm Report
Cohesion and Reuse in an Object-Oriented System
[2 refs] [27 cites]
Measuring Functional Cohesion
[7 refs] [32 cites]
Hide papers ≤ 0 refs:
Hide papers ≤ 15 cites:
Show up to 10 levels:

Figure 3.4: Using the pie menu control to save the paper “Fast Effective Rule Induction” for later.

shown in Figure 3.5.

To reveal the paper’s references, Billy Joe zooms in on the paper, “CLASSIC Learning”. As shown in Figure 3.6, when he zooms in, the direct references for that papers are automatically fetched from the server and rendered.

However, Billy Joe is still not satisfied, so he decides to refocus PaperCube to that paper. To refocus on the paper, he uses the fan menu again to refocus as shown in Figure 3.7.

3.1.4 Using Papers Per Year View

After refocusing, Billy Joe first sees the paper in Tree Map view. However, he wants to see when the paper’s references were published. To do this, he switches over to Papers Per Year view to more clearly see the evolution of the previous research that the paper builds upon. If Billy Joe moves his cursor over the focused paper, he sees all its references and citations color coded. Shown in Figure 3.8, the blue lines signify reference links and red, citation links.

At first, Billy Joe is overwhelmed with the quantity of information but using the slider controls, he is able to narrow down the display parameters to show papers with a large number of citations because he thinks they may be more important. Billy Joe plays with the display parameters and hides all papers that have less than twenty citations and saves some of the papers that he thinks may be seminal in the field, especially some of the papers published in the 1980s. The result of the display parameter changes is shown in Figure 3.9.

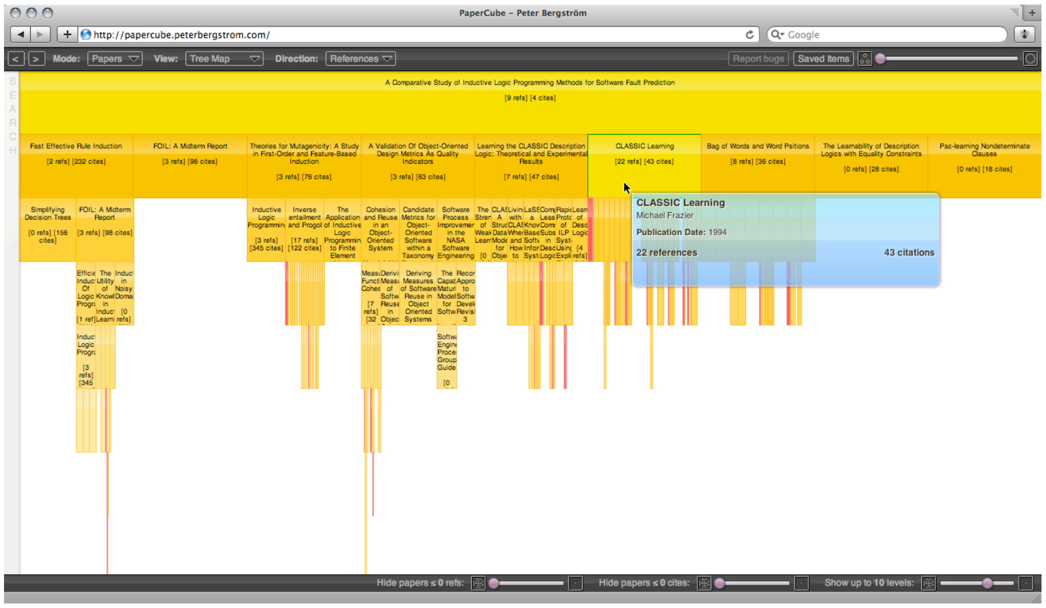


Figure 3.5: Zooming in and the references for “CLASSIC Learning” are not clearly visible.

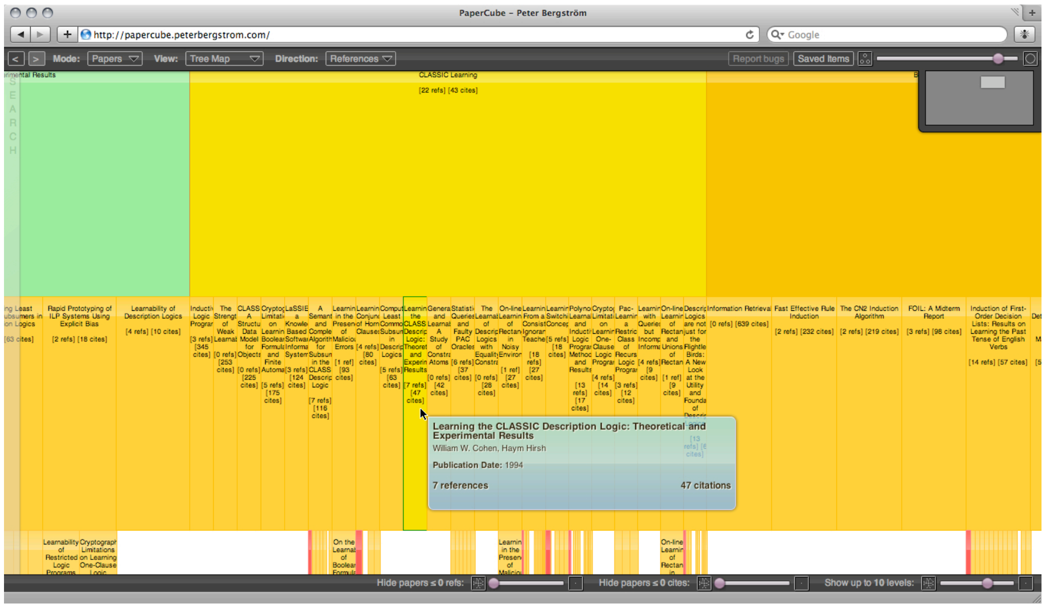


Figure 3.6: Zoomed in on “CLASSIC Learning” while still focused on the original paper.

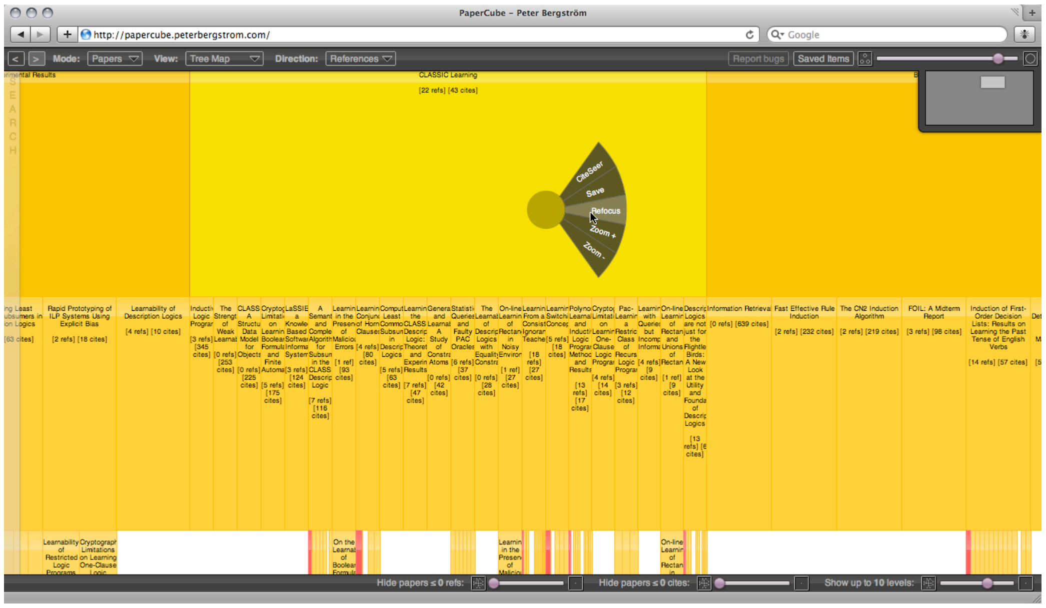


Figure 3.7: Refocusing the view to the new paper, “CLASSIC Learning.”

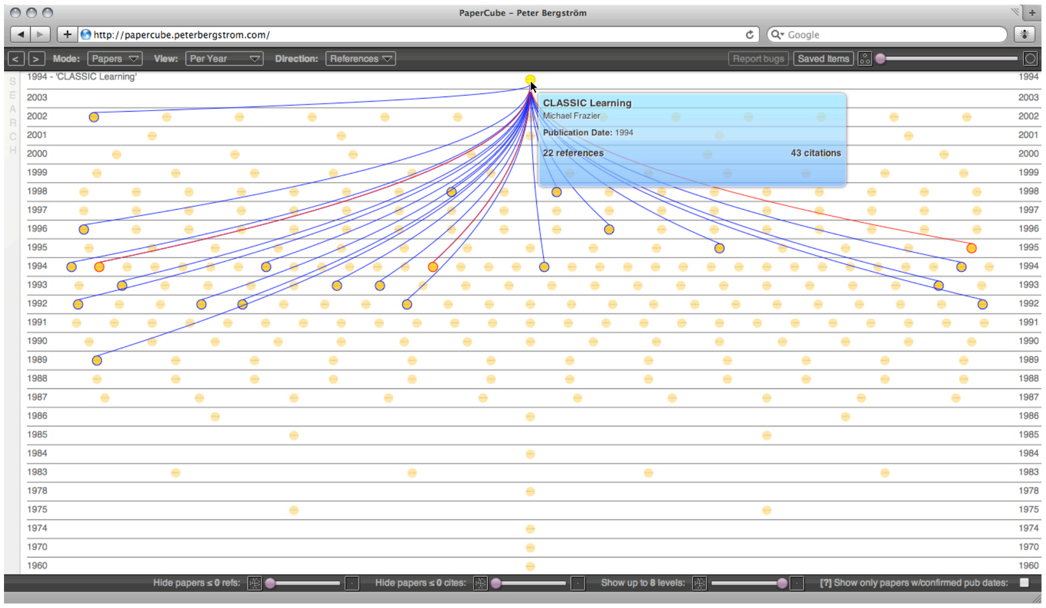
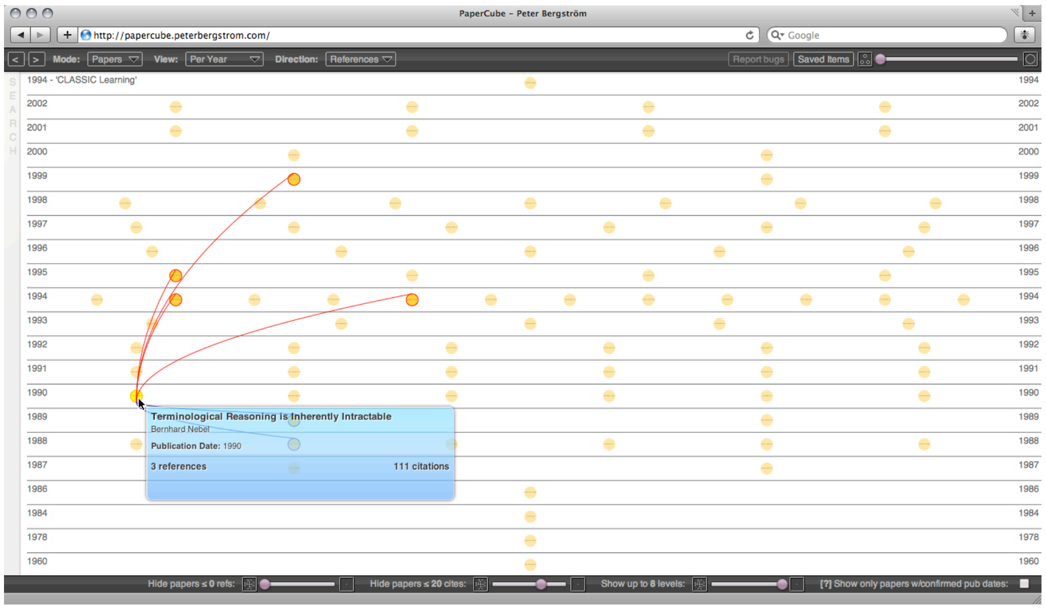


Figure 3.8: Viewing the paper “CLASSIC Learning” in PAPERS PER YEAR view.


PaperCube – Peter Bergström

http://papercube.peterbergstrom.com/

Mode: Papers View: Per Year Direction: References

Terminological Reasoning is Inherently Intractable
Bernhard Nebel
Publication Date: 1990
3 references 111 citations

Hide papers ≤ 0 refs: Hide papers ≤ 20 cites: Show up to 8 levels: [?] Show only papers w/confirmed pub dates:

Figure 3.9: After changing the display parameters, the amount of data is smaller but more relevant.

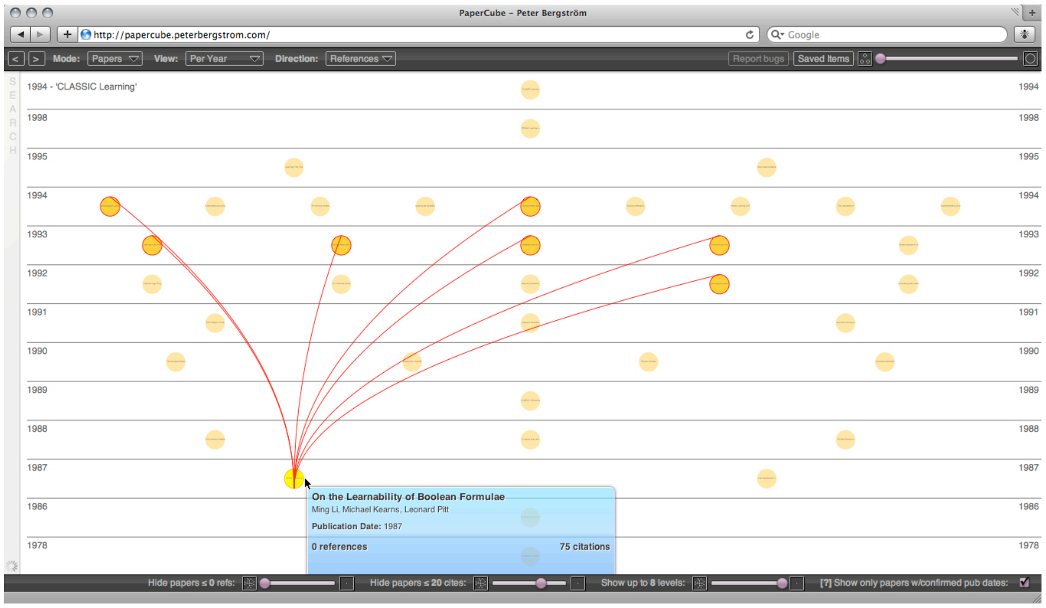
Billy Joe notices that some of the papers may not have confirmed years, including one of its references. In order to only shown papers with confirmed years, he checks the box at the bottom right of the window “Show only papers w/confirmed pub dates.” The data set is reduced further, but it is even more accurate. He sees that the paper “On the Learnability of Boolean Formulae” is cited many times over the years as shown in Figure 3.10. Since it seems important, Billy Joe saves it for later.

3.2 Scenario 2: Finding Collaborators

Later, Billy Joe returns to PaperCube to find some collaborators of the authors that have written the papers that he looked at previously. Since the papers he found were interesting, he wants to find more related work.

3.2.1 Loading a Paper From Saved Items

Billy Joe loads PaperCube and opens the “Saved Items” panel (Figure 3.11) to see the paper’s that he previously saved. He clicks on “Fast Effective Rule Induction,” the first paper he found, which looked really interesting.


PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Google
Report bugs
Saved Items

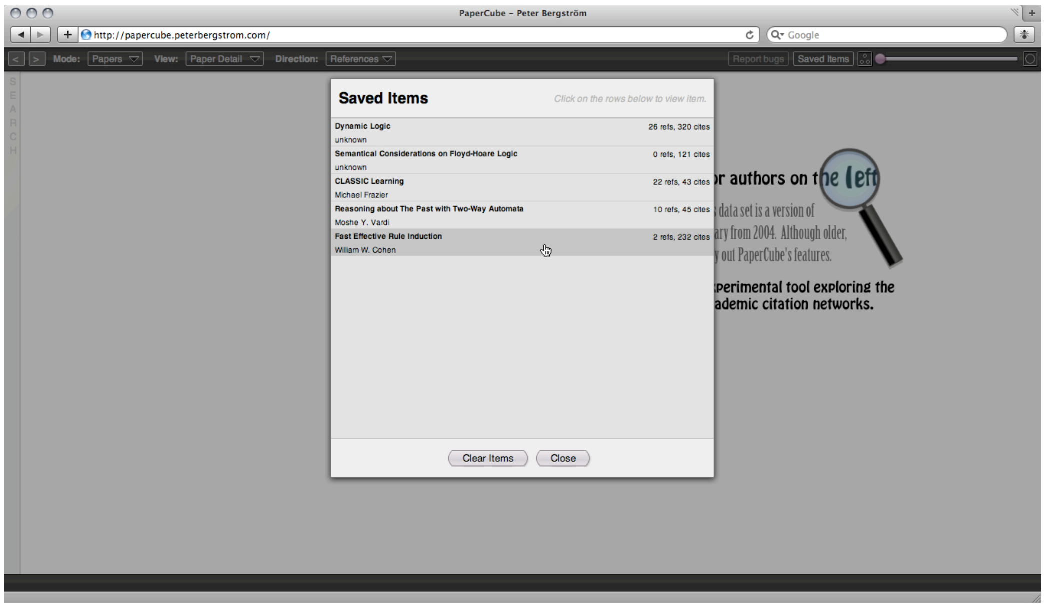
Mode: Papers View: Per Year Direction: References

1994 - 'CLASSIC Learning'

On the Learnability of Boolean Formulae
Ming Li, Michael Kearns, Leonard Pitt
Publication Date: 1987
0 references 75 citations

Hide papers ≤ 0 refs: Hide papers ≤ 20 cites: Show up to 8 levels: [?] Show only papers w/confirmed pub dates: ✓

Figure 3.10: Only showing confirmed years narrows the data set down even further.


PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Google
Report bugs
Saved Items
Clear Items
Close

Mode: Papers View: Paper Detail Direction: References

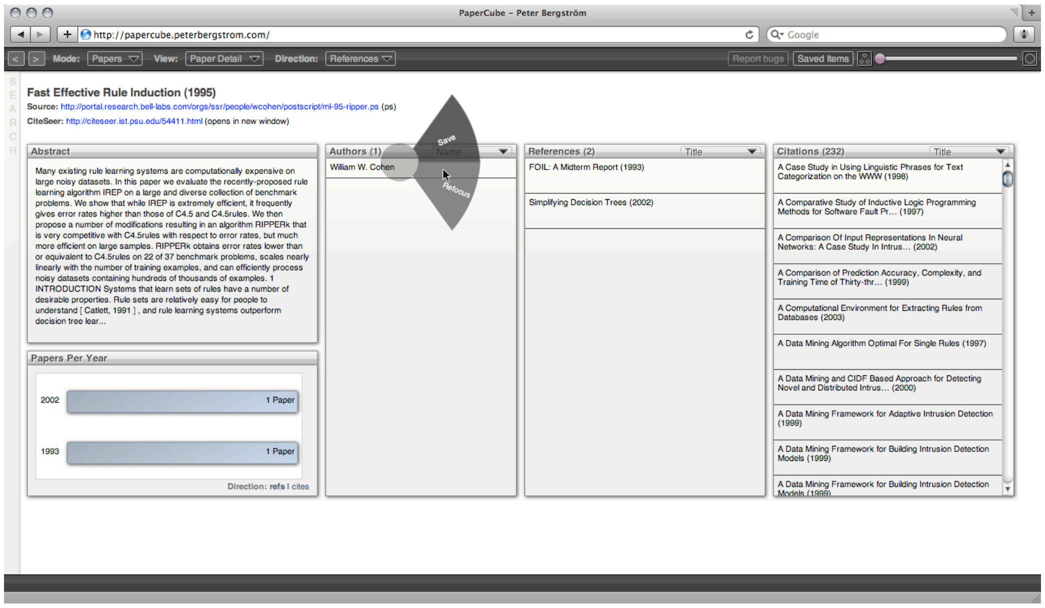
Saved Items *Click on the rows below to view item.*

Dynamic Logic unknown	26 refs, 320 cites
Semantical Considerations on Floyd-Hoare Logic unknown	0 refs, 121 cites
CLASSIC Learning Michael Frazier	22 refs, 43 cites
Reasoning about The Past with Two-Way Automata Moshe Y. Vardi	10 refs, 45 cites
Fast Effective Rule Induction William W. Cohen	2 refs, 232 cites

Figure 3.11: Saved items pane showing all the papers that were saved during the last session.

3.2.2 Using Author Detail View

After clicking on the paper in the "Saved Items" list, Billy Joe is brought to Paper Detail view. Since Billy Joe now wants to look at this paper's authors, he uses the fan menu to refocus to view the author, William Cohen as shown in Figure 3.12.



Fast Effective Rule Induction (1995)
Source: http://portal.research.bell-labs.com/orgs/ssr/people/wcohen/postscript/ml-95-ripper.ps (ps)
CiteSeer: http://citeseer.ist.psu.edu/54411.html (opens in new window)

Abstract

Many existing rule learning systems are computationally expensive on large noisy datasets. In this paper we evaluate the recently-proposed rule learning algorithm IREP on a large and diverse collection of benchmark problems. We show that while IREP is extremely efficient, it frequently gives error rates higher than those of C4.5 and C4.5rules. We then propose a number of modifications resulting in an algorithm RIPPERk that is very competitive with C4.5rules with respect to error rates, but much more efficient on large samples. RIPPERk obtains error rates lower than or equivalent to C4.5rules on 22 of 37 benchmark problems, scales nearly linearly with the number of training examples, and can efficiently process noisy datasets containing hundreds of thousands of examples. 1 INTRODUCTION Systems that learn sets of rules have a number of desirable properties. Rule sets are relatively easy for people to understand [Catlett, 1991] , and rule learning systems outperform decision tree lear...

Papers Per Year

2002 1 Paper
1993 1 Paper

Authors (1)
William W. Cohen

References (2)
FOIL: A Midterm Report (1993)
Simplifying Decision Trees (2002)

Citations (232)
A Case Study in Using Linguistic Phrases for Text Categorization on the WWW (1998)
A Comparative Study of Inductive Logic Programming Methods for Software Fault Pr... (1997)
A Comparison Of Input Representations In Neural Networks: A Case Study In Intrus... (2002)
A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-thr... (1999)
A Computational Environment for Extracting Rules from Databases (2003)
A Data Mining Algorithm Optimal For Single Rules (1997)
A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrus... (2000)
A Data Mining Framework for Adaptive Intrusion Detection (1999)
A Data Mining Framework for Building Intrusion Detection Models (1999)
A Data Mining Framework for Building Intrusion Detection Models (1999)

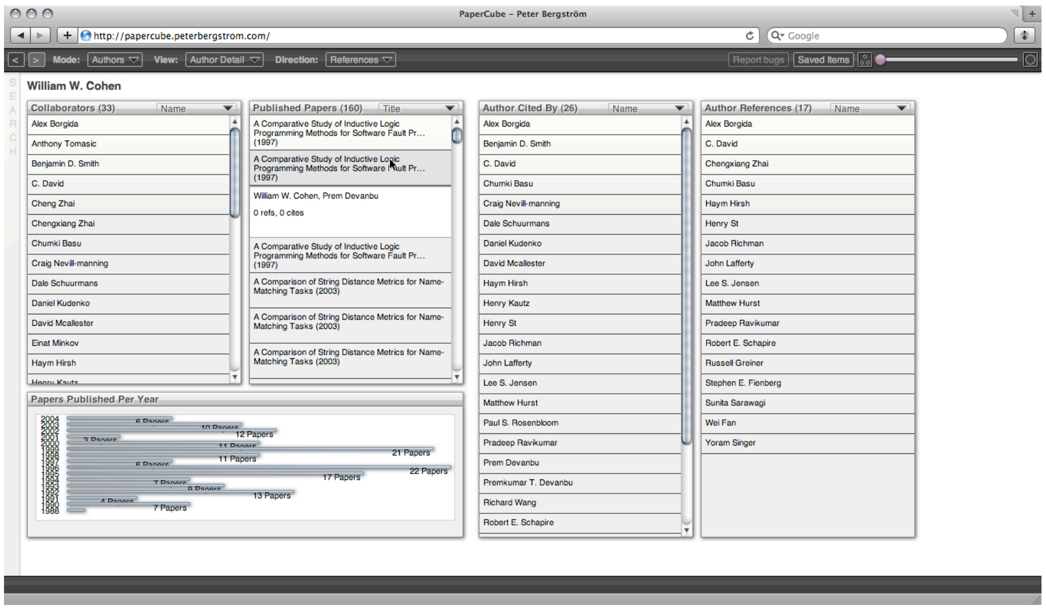
Figure 3.12: Using the fan menu to refocus on the author William Cohen.

Now PaperCube is focused on William Cohen and PaperCube defaults to show his information in Author Detail view. Here Billy Joe can see that per the CiteSeer data set, Cohen has worked with thirty-three other authors, published 160 papers, cited by twenty-six other authors, and references seventeen other authors as shown in Figure 3.13.

3.2.3 Using Collaborators View

Billy Joe wants to see with whom Cohen has worked on papers so he switches to the Collaborators view. This view shows all the authors with whom Cohen has written papers in the past. Furthermore, the view shows which of his collaborators have worked together. In this case, Billy Joe sees that some of the authors have worked with Cohen more than twenty times. One of the authors is Haym Hirsh as shown Figure 3.14.

In order to narrow down to Cohen's most frequent collaborators, Billy Joe sets the number collaborations threshold to five to only show authors that Cohen has written papers with at least five times and the visualization animates the transition smoothly to reflect the new threshold as shown in Figure 3.15.



William W. Cohen

Collaborators (33) — Name

Alex Borgida
Anthony Tomasic
Benjamin D. Smith
C. David
Cheng Zhai
Chengxiang Zhai
Chumki Basu
Craig Nevill-manning
Dale Schuurmans
Daniel Kudenko
David Mcallester
Einat Minkov
Haym Hirsh
Henry Kautz

Published Papers (160) — Title

A Comparative Study of Inductive Logic Programming Methods for Software Fault Pr… (1997)
A Comparative Study of Inductive Logic Programming Methods for Software Fault Pr… (1997)
William W. Cohen, Prem Devanbu
0 refs, 0 cites
A Comparative Study of Inductive Logic Programming Methods for Software Fault Pr… (1997)
A Comparison of String Distance Metrics for Name-Matching Tasks (2003)
A Comparison of String Distance Metrics for Name-Matching Tasks (2003)
A Comparison of String Distance Metrics for Name-Matching Tasks (2003)

Papers Published Per Year

2004	6 Papers
2003	10 Papers
2002	12 Papers
2001	3 Papers
2000	11 Papers
1999	21 Papers
1998	11 Papers
1997	6 Papers
1996	22 Papers
1995	17 Papers
1994	7 Papers
1993	9 Papers
1992	13 Papers
1991	4 Papers
1990	7 Papers
1988	

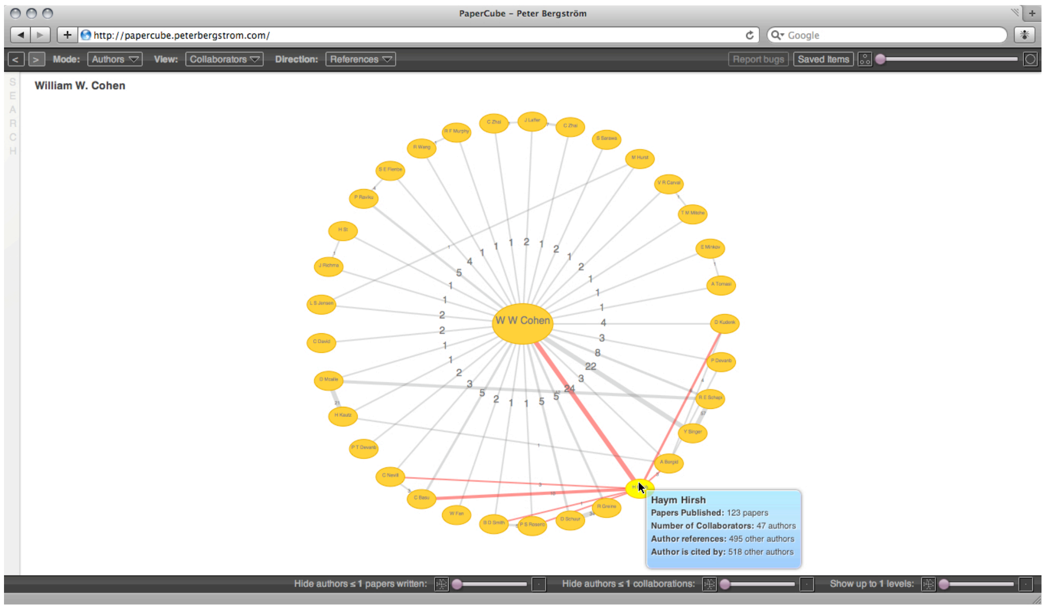
Author Cited By (26) — Name

Alex Borgida
Benjamin D. Smith
C. David
Chumki Basu
Craig Nevill-manning
Dale Schuurmans
Daniel Kudenko
David Mcallester
Haym Hirsh
Henry Kautz
Henry St
Jacob Richman
John Lafferty
Lee S. Jensen
Matthew Hurst
Paul S. Rosenbloom
Pradeep Ravikumar
Prem Devanbu
Premkumar T. Devanbu
Richard Wang
Robert E. Schapire

Author References (17) — Name

Alex Borgida
C. David
Chengxiang Zhai
Chumki Basu
Haym Hirsh
Henry St
Jacob Richman
John Lafferty
Lee S. Jensen
Matthew Hurst
Pradeep Ravikumar
Robert E. Schapire
Russell Greiner
Stephen E. Fienberg
Sunita Sarawagi
Wei Fan
Yoram Singer

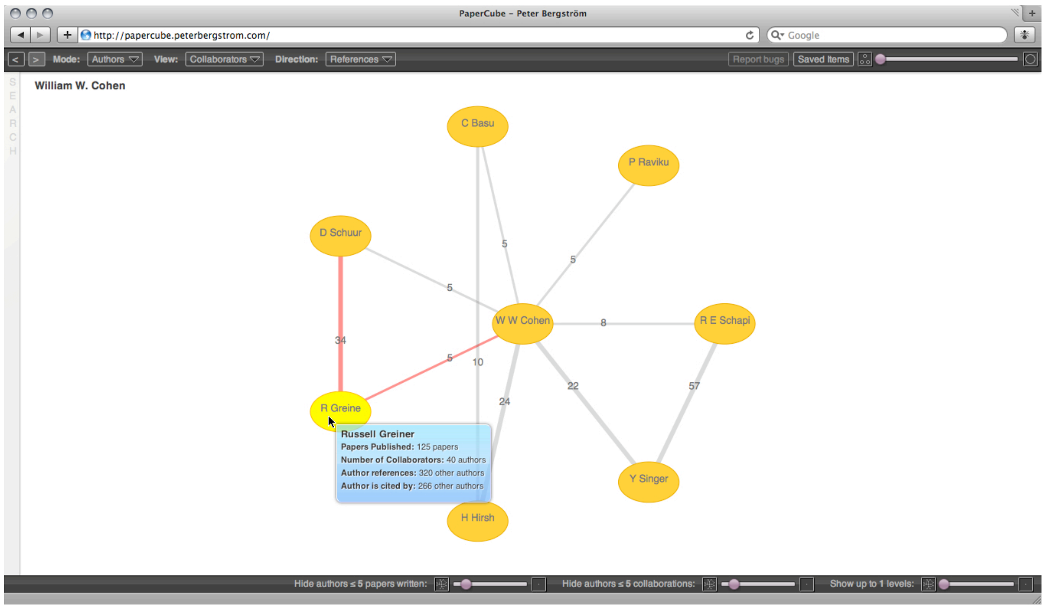
Figure 3.13: Author Detail view for William Cohen.



William W. Cohen

Haym Hirsh
Papers Published: 123 papers
Number of Collaborators: 47 authors
Author references: 495 other authors
Author is cited by: 518 other authors

Figure 3.14: Collaborators view for William Cohen shows a strong relationship with Hayn Hirsh, with whom he has worked 24 times.



Mode: Authors
View: Collaborators
Direction: References

William W. Cohen

Russell Greiner
Papers Published: 125 papers
Number of Collaborators: 40 authors
Author references: 320 other authors
Author is cited by: 266 other authors

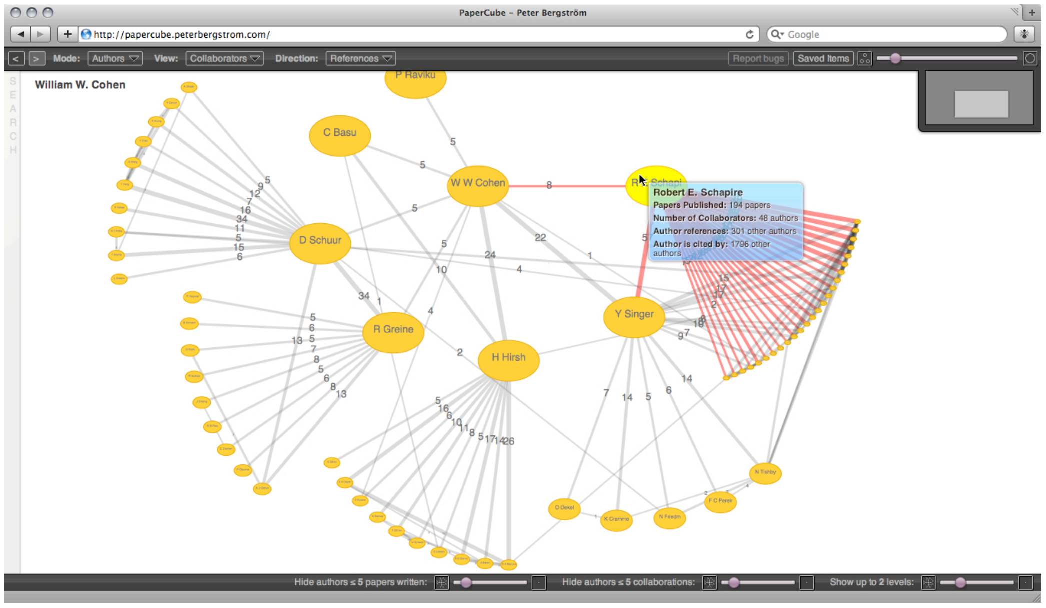
Hide authors ≤ 5 papers written:
Hide authors ≤ 5 collaborations:
Show up to 1 levels:

Figure 3.15: COLLABORATORS view with thresholds to show less authors on the screen.

As shown in Figure 3.16, Billy Joe then wants to see the collaboration network up to two levels so he changes the depth display setting. More authors are displayed they have a rich set of collaboration relationships not only with Cohen, but with each other. Billy Joe zooms in to see the smaller circles that represent the authors. Billy Joe can pan around using the zoom widget at the top right corner of the window or by grabbing the white space in the view.

In order to narrow the set of collaborators rendered, Billy Joe increases the thresholds higher to show only authors who have written at least thirty papers and collaborated at least seven times. Shown in Figure 3.17, the graph is more manageable.

Billy Joe wants to look at Thomas Ellman a bit closer, an author that Hirsh, a frequent collaborator with whom Cohen has worked eleven times. Therefore, Billy Joe refocuses COLLABORATORS view on Ellman using the fan menu as shown in Figure 3.18. After finding Ellman and Hirsh, Billy Joe has found new authors that may have conducted research that he would be interested in learning about.

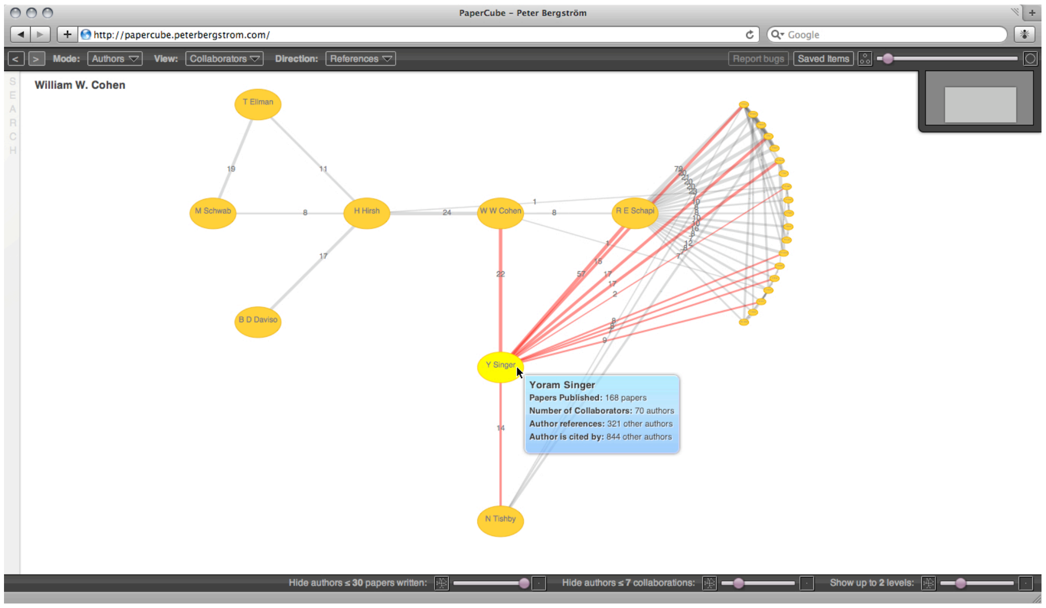


Mode: Authors View: Collaborators Direction: References

Robert E. Schapire
Papers Published: 194 papers
Number of Collaborators: 48 authors
Author references: 301 other authors
Author is cited by: 1796 other authors

Hide authors ≤ 5 papers written: Hide authors ≤ 5 collaborations: Show up to 2 levels:

Figure 3.16: Collaborators view showing two levels of collaborators for Cohen.



Mode: Authors View: Collaborators Direction: References

Yoram Singer
Papers Published: 168 papers
Number of Collaborators: 70 authors
Author references: 321 other authors
Author is cited by: 844 other authors

Hide authors ≤ 30 papers written: Hide authors ≤ 7 collaborations: Show up to 2 levels:

Figure 3.17: Collaborators showing two levels of collaborators but with more constraints.

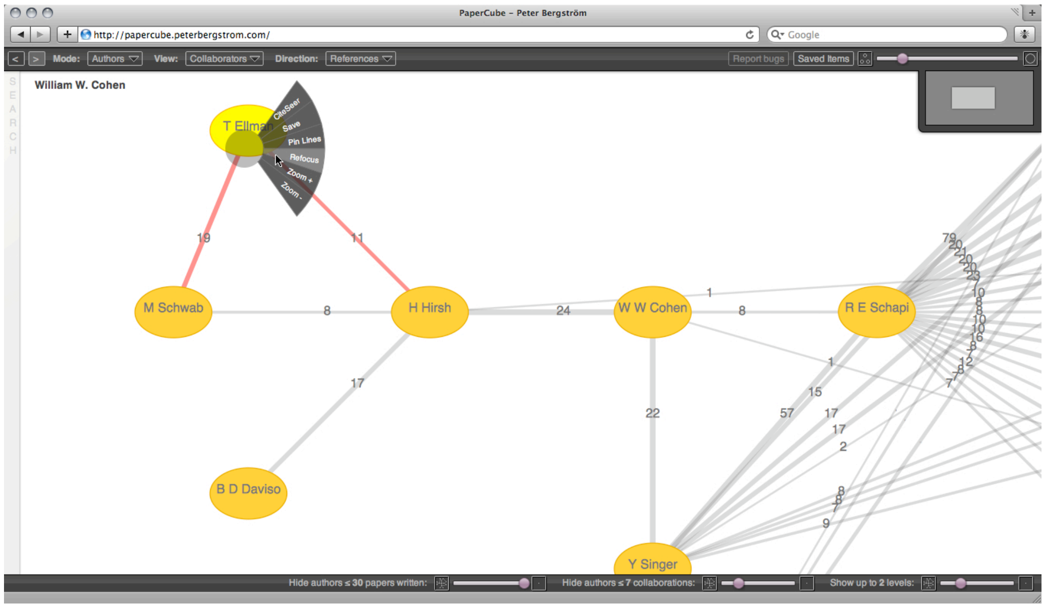


Figure 3.18: Using the fan menu to refocus COLLABORATORS view to view Ellman.

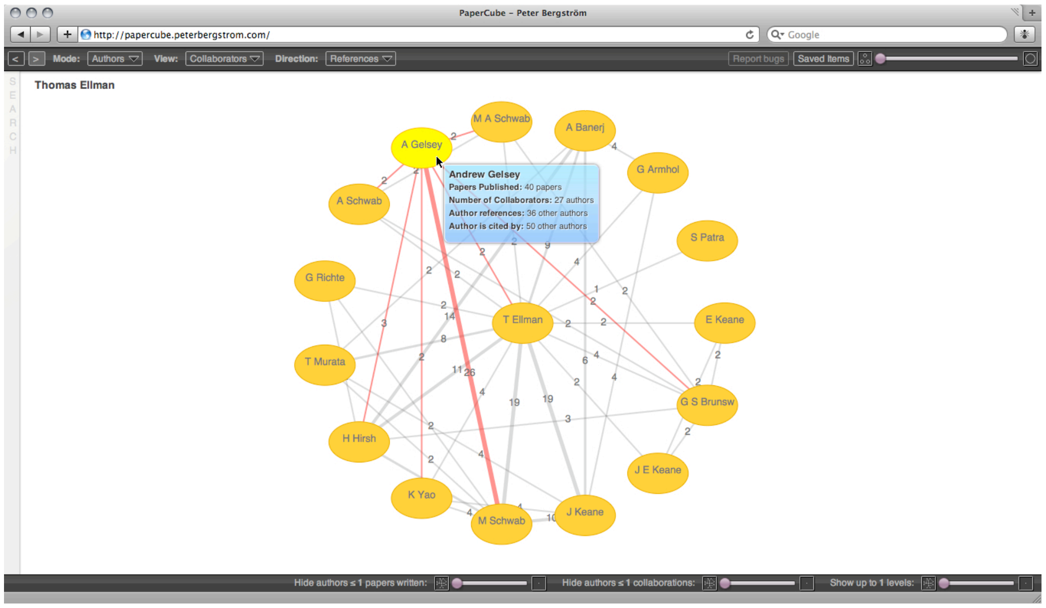

PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Mode: Authors
View: Collaborators
Direction: References
Report bugs
Saved Items
Thomas Ellman
A Gelsey
M A Schwab
A Banerj
G Armhol
S Patra
E Keane
G S Brunsw
J E Keane
J Keane
M Schwab
K Yao
H Hirsh
T Murata
G Richte
A Schwab
T Ellman
Andrew Gelsey
Papers Published: 40 papers
Number of Collaborators: 27 authors
Author references: 36 other authors
Author is cited by: 50 other authors
Hide authors ≤ 1 papers written:
Hide authors ≤ 1 collaborations:
Show up to 1 levels:

Figure 3.19: COLLABORATORS view focused on Thomas Ellman.

Chapter 4

General Features of PaperCube

This chapter is a walk-though of the general features of PaperCube. The main focus is to describe the included functionality and features that are commonly used across the set of paper and author views. The paper and author views and visualizations themselves will be covered in the two following chapters.

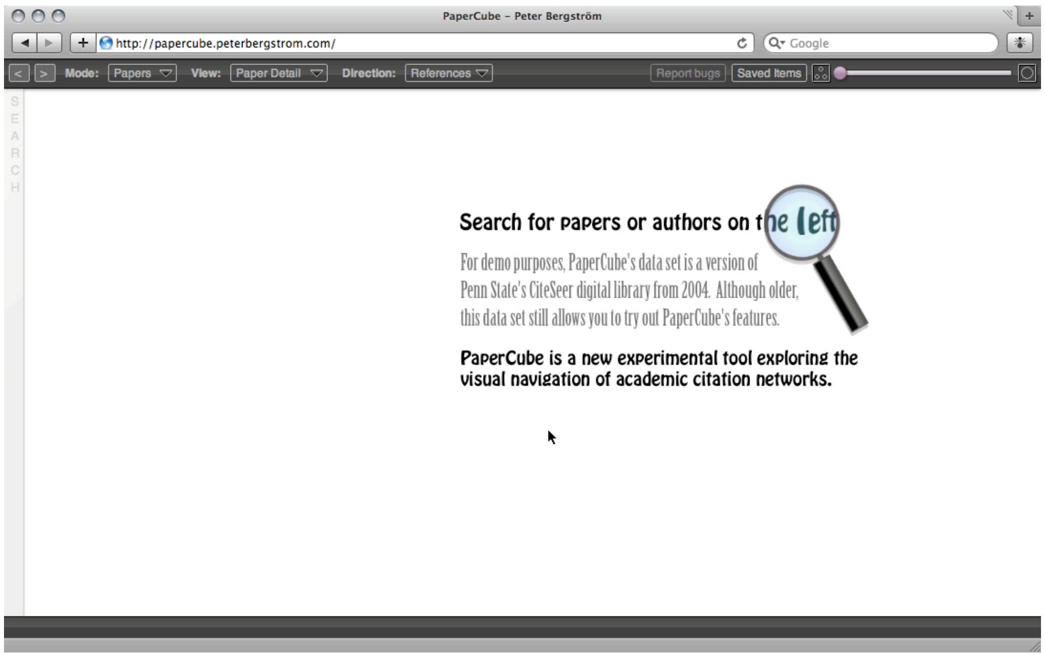
Although PaperCube is web-based, the aim was to create a highly interactive application that allows researchers to interact with it in a similar manner to a desktop application. The idea is to make the fact that PaperCube resides with the confines a web browser almost impossible to tell. This cloud—or more technically—thick-client web application architecture is possible through the use of SproutCore, a JavaScript framework that is specifically tailored to deliver desktop-class user interfaces on the web. The technical implementation will be covered in detail in chapter ten.

Just as a desktop application, PaperCube loads once per session and any subsequent actions are handled without reloading. This allows for rich, immersive user interaction that is fast and seamless. Communication with the server takes place asynchronously in the background and allows a researcher to interact with PaperCube without interruption.

PaperCube was designed to deliver a common user experience regardless of the currently visible view and the bibliographic meta data being displayed. Therefore, many universal UI elements were created and are present throughout.

4.1 Test Data Set

The need for a large set of bibliographic meta data was obvious for an application such as PaperCube. Initially, a basic programatically generated data set was used to begin development. However, a synthetic data set does not necessarily model the makeup the bibliographic meta data contained in an actual digital library. In order to test PaperCube's effectiveness at augmenting cur-



Search for papers or authors on the left

For demo purposes, PaperCube's data set is a version of Penn State's CiteSeer digital library from 2004. Although older, this data set still allows you to try out PaperCube's features.

PaperCube is a new experimental tool exploring the visual navigation of academic citation networks.

Figure 4.1: Screenshot of PaperCube when initially loaded.

rent webpage-based digital libraries, it was necessary to find a real data set. The most logical choice was CiteSeerX because it provides an OAI (Lagoze et al., 2002) interface for harvesting its meta data. However, the harvested CiteSeerX bibliographic meta data did not work correctly because the reference relationship links between papers were incorrect or incomplete. During Circle View's development in 2004, the original CiteSeer meta data was explored but dropped in favor of a limited set of bibliographic meta data from the ACM Digital Library.

However, for this work, it was necessary to test a larger set of bibliographic meta data and therefore, the original CiteSeer data from 2004 was used. It was augmented for correctness by using additional information from CiteSeerX. Also, in order to allow for the exploration of authors, the bibliographic meta data set was mined to get additional author relationships such the papers published, collaborators, the authors that an author has referenced and the authors that have cited the author. Even though the harvested CiteSeerX meta data did not yield usable reference links, it did provide correct information regarding publication year for half of the data set.

Currently, PaperCube allows a researcher to search and explore the bibliographic meta data for 716,772 papers, 411,034 authors, and 1,764,929 reference links. Although outdated, the set of bibliographic meta data used should be sufficient for the purpose of validating PaperCube.

4.2 Fan Popup Menu

One challenge that was encountered during development was that in information dense visualizations it is paramount to reduce the amount of additional clutter that may distract from the visualizations themselves. In each view, when a node is moused over, associated meta data is displayed in the form of a hovering information pane that follows the mouse cursor. In order to allow for actions to be applied to nodes in a visualization, these actions would have to be implemented by clicking on the node.

As a result, it was decided to create a pop-up menu that provides a set of actions that can be applied to an item. Several UI design concepts were explored. First, a drop down menu that listed choices in a list was considered but it was thought to be relatively space inefficient. Also, it was thought that such a menu would be slow since the mouse has to move far to select a menu choice. Instead, the most interesting solution seemed to be to use a pie menu, also known as a marking or radial menu. Pie menus are very efficient because the mouse cursor does not have to move far to select an option in the menu. The menu is positioned so that it surrounds the mouse cursor. Around the mouse cursor, a set of actions are displayed that can be quickly selected.

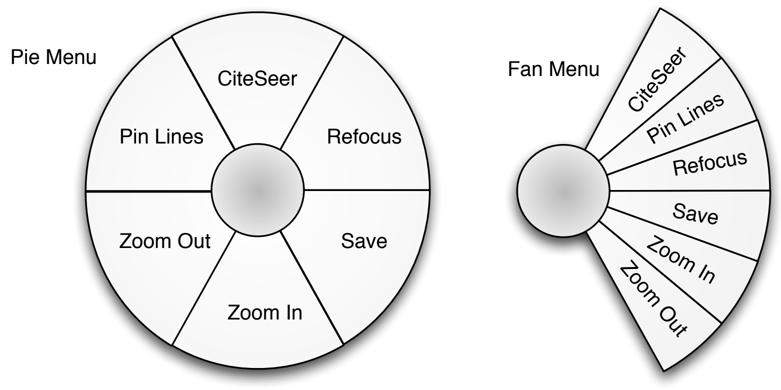


Figure 4.2: Comparison between a pie menu and the fan menu.

However, a traditional pie menu control that surrounds the mouse cursor entirely seemed like it would have to be too large in order to spell out complex actions the with limited screen real estate available. Instead, a derivative of a pie menu was developed. We call this element a *fan* menu because it looks like an opened hand-held folding fan. The fan menu allows slices to contain longer strings of text yet be more compact than a comparable pie menu because it spreads out horizontally and in the writing direction of left to right languages. On the left, Figure 4.2 contrasts a sample pie menu to the derivative fan menu. The fan menu surrounds up to 120 degrees of the right hand side of the mouse cursor and takes up a significantly smaller area than a comparable pie menu.

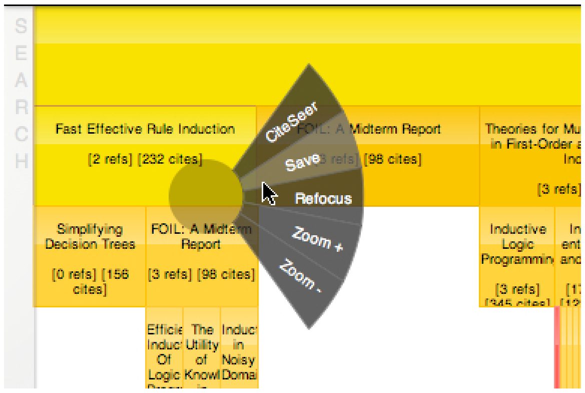


Figure 4.3: The fan menu activated on a paper in Tree Map view.

In PaperCube, when a node is clicked with the left mouse button, the fan menu is displayed. The options in the views depend on the type of data being displayed. As shown in Figure 4.3, the available actions for a paper allow for the zooming in or out of the visualization, saving the clicked paper, refocusing the view on the paper, and view the paper at the source digital library, CiteSeer.

4.3 Slider Controls to Manipulate the Visualization

Instead of having the views attempt to automatically determine the optimal amount of data to be shown for a paper or author in a view, PaperCube defers to the user to determine the rules that determine how much is rendered on the screen. These display parameters are threshold parameters that are used to constrain the amount of items that are visible in the screen. The hope was that the researcher using PaperCube knows best when it comes to determining what bibliographic meta data is the most important to show on the screen.

At the bottom of the browser window, with the exception of Paper Detail and Author Detail, views have a set of one or more slider controls. These controls allow the researcher to easily adjust the threshold parameters to hide or show more bibliographic meta data in order to tailor the view to show what is most useful to the individual. Some of the common thresholds that can be adjusted via the sliders include: the retrieval depth of a paper's citation network or an author's relationship network, the visibility of papers based on its number of references or citations and the visibility of authors based on the number of papers written or the number of collaborations with other authors. In addition to controlling the visual representation of the viewed bibliographic meta data, the sliders also determine the amount of data retrieved from the server, and the more strict your display parameters are, the less needs to be loaded from the server. If more information is revealed by changing a display parameter, only the bibliographic meta data that has not previously been loaded into the SproutCore data store will be retrieved from the server.

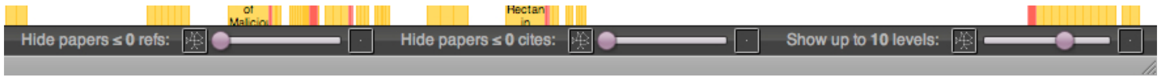


Figure 4.4: The slider controls for TREE MAP view showing three different display parameters that can be adjusted.

4.4 Toolbar

At the top of the screen, PaperCube has a toolbar that exposes a diverse set of actions that make it possible to navigate and manage the type of bibliographic meta data displayed. Please refer Figure 4.5 throughout this section as the various toolbar actions are described. Starting from left to and going to the right, the toolbar contains the navigation history buttons (1), the viewing mode menu (2), the view choice menu (3), the view direction menu (4), the saved items (5), and the zooming widget (6). The zoom widget is a very important component of PaperCube and is heavily used in most views to navigate around a zoomed in visualization.

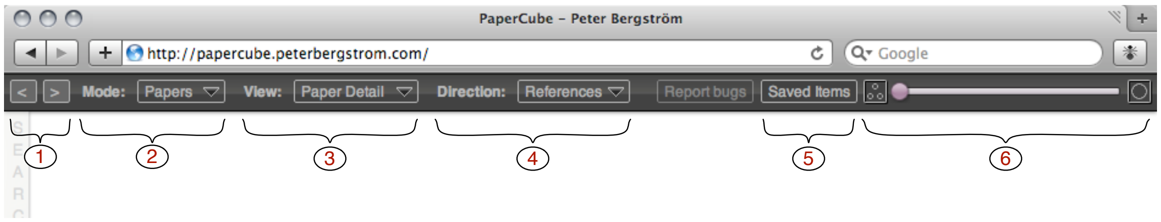


Figure 4.5: Screenshot of PaperCube's toolbar showing all its imporant elmements labelled for reference in this section.

Navigation History Buttons and Menus

Since PaperCube was designed to have a similar interaction model as a desktop application, it does not refresh as actions such as navigating from one item to the next are performed. In the top toolbar, the two left most buttons labelled (1) in Figure 4.5 make it possible to go backwards and forwards in the history of viewed papers and authors. Since the page does not reload, one can not directly rely on the browser's history management without overriding it in JavaScript. The saved information not only consists of the item viewed, but also the viewing mode, view used, and the viewing direction. Therefore, PaperCube manages the browsing history using JavaScript but it does not save it into the web browser's history object.

As papers and authors are viewed, items are added to the history. The history can be sequentially accessed to go back and forth between previously viewed items. Items can be accessed non-sequentially through a menu showing all items in the back or forward history by pressing either the forward or back button for a half second. That menu makes it possible to jump to any item in the history directly and the view mode, view choice, and viewing direction will change accordingly.

Viewing Mode Menu

PaperCube shows both paper and author relationships. To switch between them, the "Mode" needs to be changed using the drop down (2) in Figure 4.5. When the viewing mode is changed, the associated view choices change and PaperCube will switch to the last used view in the newly selected mode. Also, if a researcher searched for papers or authors when using the previous viewing mode, the search will automatically be resubmitted to find matching items in the new viewing mode. For example, if a researcher previously searched for papers with the word "hypertext" in the title in papers mode and then switch to viewing authors, the search will automatically return the authors that have written papers with the word "hypertext" in the title.

View Choices Menu

One of the main goals of PaperCube was make it possible to easily switch from one view to another without losing focus on the item currently in view. The currently selected view can be changed with view choice menu as shown as (3) in Figure 4.5. Thus, PaperCube allows the context in which an item is being viewed to be changed but yet remain focused on it. As stated earlier, when the viewing mode option changes, the available views change based on which of the two choices, "Papers" or "Authors" is selected. When the visible view is switched, the currently focused item will be redisplayed in the new view. By default, when PaperCube loads, the views PAPER DETAIL and AUTHOR DETAIL are the default for both modes because they serve as the jumping off point from which users can select any of the visualization-based views.

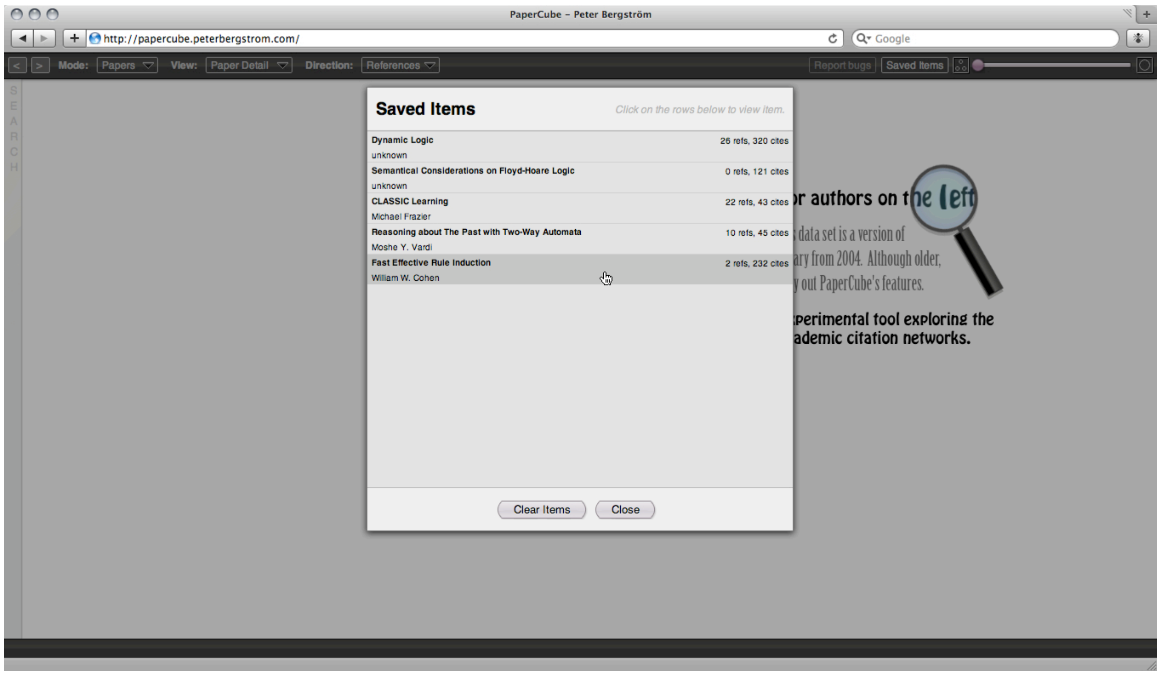
For example, if a paper is being viewed in using CIRCLE VIEW, viewing more levels of its citation network can be done by switching to TREE MAP view. It is important to note that when switching between views of the same item, the new view will only retrieve bibliographic meta data from the server that has not previously been loaded into memory from being displayed in a previous view.

View Direction Menu

When viewing paper relationships, either forward or reverse relationships can be displayed. As previously mentioned, a forward link is called a *reference* and a reverse link is called a *citation.* When the selection is changed in menu shown labelled as (4) in Figure 4.5, the currently visible view will recalculate what is shown based on the new relationship direction and automatically retrieve any required bibliographic meta data from the server.

Saved Items

As papers and authors are discovered through the use of PaperCube, there may be items that should be saved for later. Using the fan menu, items can quickly be saved. To view them later, there is a button located on the toolbar labeled, “Saved Items”, (5) in Figure 4.5. This button reveals a light box that lists all saved paper or author items. Figure 4.6 shows the light box with several items saved. Any of the listed items can be clicked and viewed.



Saved Items

Click on the rows below to view item.

Dynamic Logic unknown	26 refs, 320 cites
Semantical Considerations on Floyd-Hoare Logic unknown	0 refs, 121 cites
CLASSIC Learning Michael Frazier	22 refs, 43 cites
Reasoning about The Past with Two-Way Automata Moshe Y. Vardi	10 refs, 45 cites
Fast Effective Rule Induction William W. Cohen	2 refs, 232 cites

Figure 4.6: Screenshot of PaperCube showing the “Saved Items” light box showing several saved papers.

The items are saved to a cookie that PaperCube reads when it starts up and seamlessly restores any saved items from previous browsing sessions. The decision to use a cookie and not a back end server to store items saved was purely for practicality. This work was not meant to be a full-fledged service with user management. Using a JavaScript cookie allows items to be saved without having complicated user management. One problem with this cookie-based implementation is that if the web browser's cache is cleared or cookies are cleared, the saved items will be deleted.

Zoom Widget

Most of the views in PaperCube are designed with resolution independence in mind. Resolution independence was a top goal when creating the tool. Since the visualizations can be complex, zooming in to get more information can be very useful. Labelled (6) in Figure 4.5, the zooming widget is located at the top right corner of the screen. It consists of two buttons and a slider control. The buttons either zoom in and or out by one step of the slider, and the slider can be used to dynamically zoom in and out by dragging the handle with the mouse. It is also possible to zoom in and out using the mouse scroll wheel or through the fan menu.

When zoomed in, a preview UI slides down that shows the full area of the visualization and the currently visible area. With the mouse, one can zoom and pan the whole visualization from the preview pane. When panning using the preview pane, the visualization does not update until the user lets go of the mouse button. It is possible to pan the visualization from the main window by grabbing the available white space and dragging the visualization. This method is not as fast in terms of rendering as using the preview window since it updates the visualization in real time but it is more precise and gives immediate feedback.

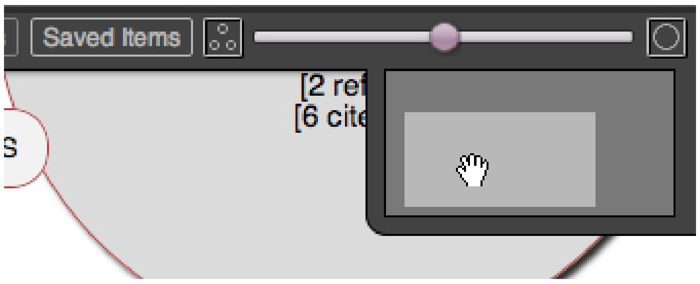


Figure 4.7: The zoom preview window is visible when zoomed in and can be used to pan around the visualization.

4.5 Meta Data View

In the majority of views, screen real estate is at a premium. However, the goal is to show as much bibliographic meta data about an item as possible. In the visualizations themselves, only limited information can be displayed, but a lot of other important meta data is present that should not missed about the items. Therefore, when the mouse cursor is moved above a node, a box containing additional meta data is revealed that follows the mouse cursor. As shown in Figure 4.8, a paper item's meta data consists of the title, authors, publication year, and the number of references and citations.

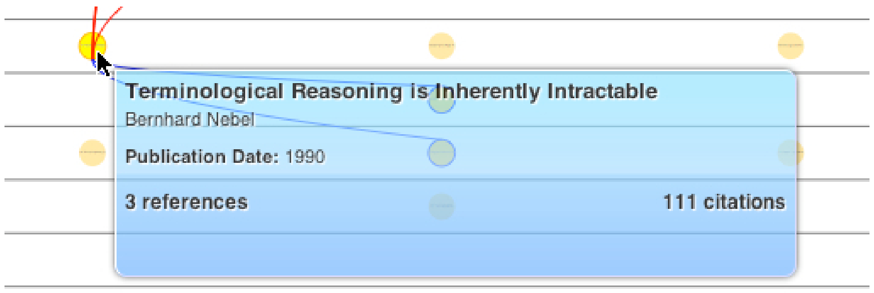
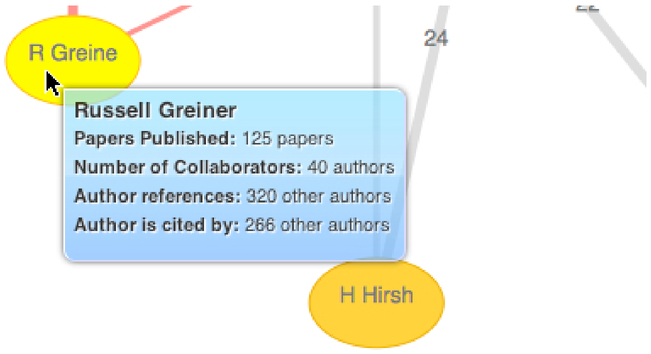

Terminological Reasoning is Inherently Intractable
Bernhard Nebel
Publication Date: 1990
3 references **111 citations**

Figure 4.8: Meta data for the paper "Terminological Reasoning Is Inherently Intractible" shown in Papers Per Year view.

An author's bibliographic meta data consists of the author's name and other high-level statistics. The statistics consists of the number of papers published, the number of collaborators, the number of times the author references others, and the number of times that other authors reference the author. The reason those statistics were selected was that they seemed like good indicators of the author's importance.



Russell Greiner
Papers Published: 125 papers
Number of Collaborators: 40 authors
Author references: 320 other authors
Author is cited by: 266 other authors

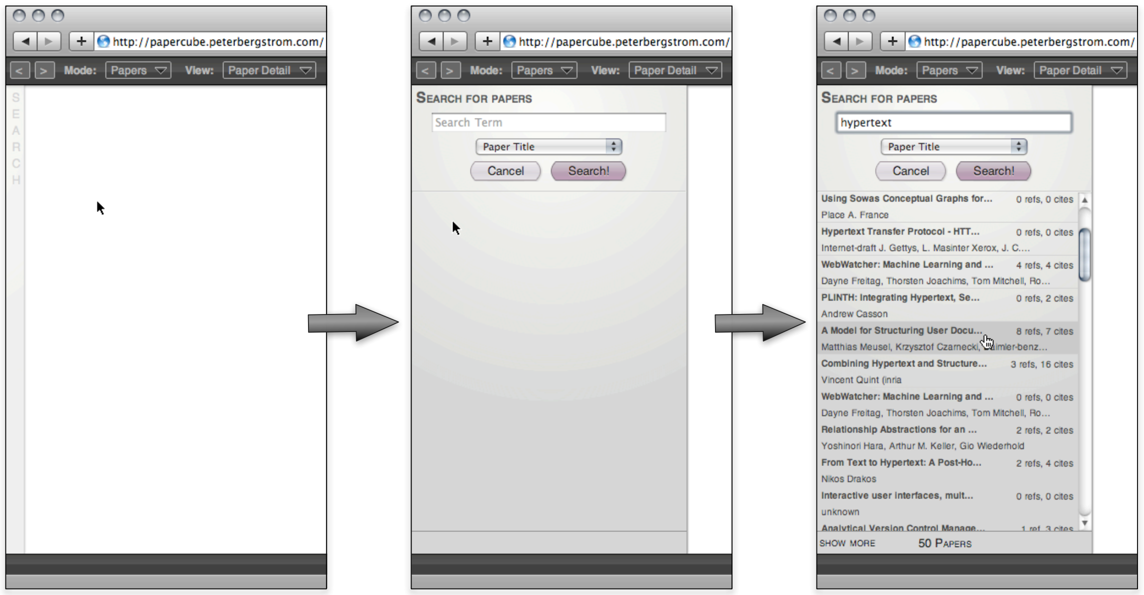
Figure 4.9: Author meta data for Russell Greiner shown in Collaborators view.

4.6 Search Pane

While the actual search for papers and authors is not the main focus of PaperCube, some limited searching functionality was required so that the tool is usable in its experimental form. Following the flow shown in Figure 4.10, to search, move the mouse cursor to the hot zone on the left edge of the browser window and the search pane will be automatically revealed.

The searching functionality allows bibliographic meta data about papers or authors to be retrieved from the database based on a paper's title, abstract, publication date, subject or an author's name. Most of searchable fields allow the use of MySQL "full-text" query syntax (MySQL Boolean Full-Text Searches) that allows for powerful and precise searches. However, as previously mentioned, the functionality is limited and should not be compared to a fully developed digital library service.

The search pane returns either papers or authors and allows for paging of results. The maximum results that can be shown at any time is fifty items, but at the bottom there is a button, "SHOW MORE", which reveals the next page of search results.



Using Sowas Conceptual Graphs for... 0 refs, 0 cites
Place A. France
Hypertext Transfer Protocol - HTT... 0 refs, 0 cites
Internet-draft J. Gettys, L. Masinter Xerox, J. C....
WebWatcher: Machine Learning and ... 4 refs, 4 cites
Dayne Freitag, Thorsten Joachims, Tom Mitchell, Ro...
PLINTH: Integrating Hypertext, Se... 0 refs, 2 cites
Andrew Casson
A Model for Structuring User Docu... 8 refs, 7 cites
Matthias Meusel, Krzysztof Czarnecki, Daimler-benz...
Combining Hypertext and Structure... 3 refs, 16 cites
Vincent Quint (inria
WebWatcher: Machine Learning and ... 0 refs, 0 cites
Dayne Freitag, Thorsten Joachims, Tom Mitchell, Ro...
Relationship Abstractions for an ... 2 refs, 2 cites
Yoshinori Hara, Arthur M. Keller, Gio Wiederhold
From Text to Hypertext: A Post-Ho... 2 refs, 4 cites
Nikos Drakos
Interactive user interfaces, mult... 0 refs, 0 cites
unknown
Analytical Version Control Manage... 1 ref, 3 cites

Figure 4.10: The search pane is revealed by putting the mouse cursor on the right hand side of the screen. Then search terms can be entered that will be matched against the CiteSeer digital library.

Chapter 5

Paper Views

The main purpose of this work is to explore the augmentation of digital library search and browsing through the use of visualizations of paper citation network relationships. As a result, a set of tasks were envisioned and views were created to accomplish those tasks.

The first task is the exploration of the paper's immediate citation network in both direction of relationships, references and citations. This task would be useful once a paper has been found to find other papers that were immediately referenced by the authors. These papers are usually strongly related to the subject matter of the focused paper.

The second task is the exploration of the paper's extended citation network viewed hierarchically. This task allows the researcher to explore deeper into the citation network of a paper. The papers that are cited on the shallowest levels will be very targeted to the focused paper's subject, but the more hops away from the focused paper, the less the papers may be directly related, yet still important to a researcher's paper search. Papers that may seem unrelated at first can ultimately be very beneficial to a user's research.

The third task is looking at a focused paper's citation network as it has developed over time, which may perhaps show temporal relationships that may otherwise be difficult to notice. Charting the evolution of a field from the most recent research back to a seminal work can be very useful not only to find other papers in the field, but also to see where it belongs in the overall tree of knowledge.

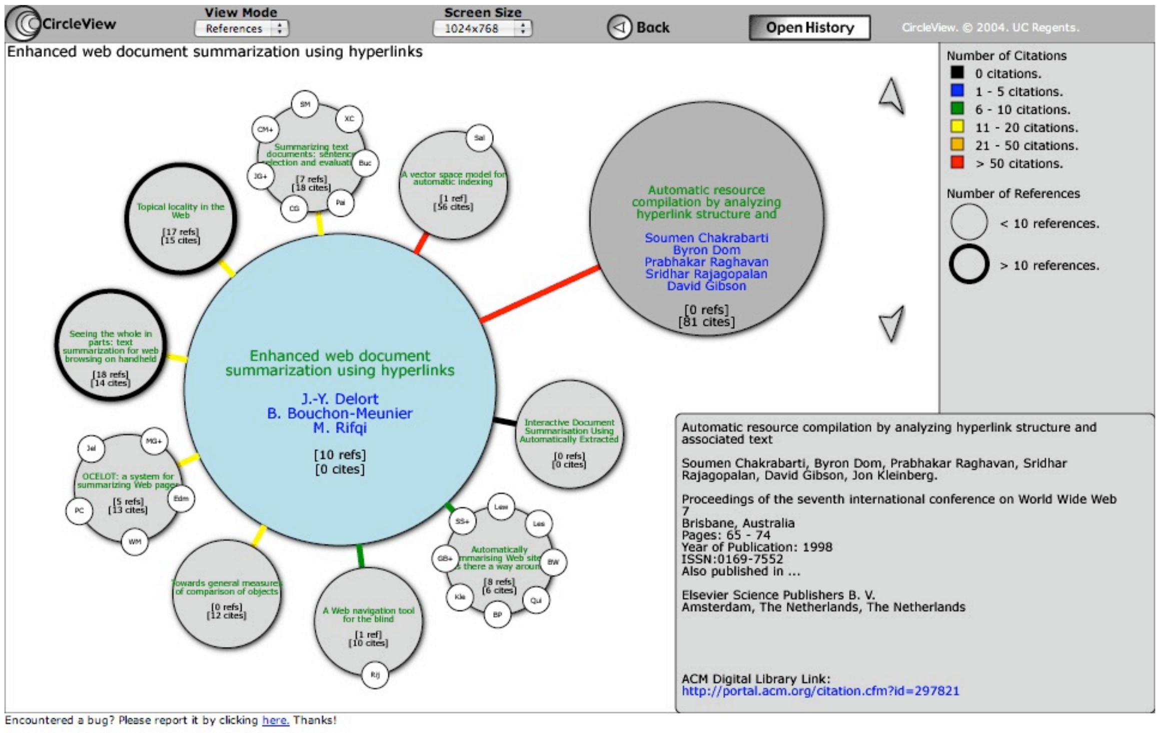
The hypothesis is that creating a set of visualizations to accomplish those tasks will make Paper-Cube very effective helping the users of digital library services find what they are looking for faster and more intuitively. Therefore, a set of five views were created to accomplish the tasks. All of these views are related because when switching between views, the same paper is in focus. However, the interpretation and exact bibliographic meta data rendered are different. The names for the five paper views were selected to describe their goals in the least amount of verbiage.

5.1 Showing a Paper's Immediate Citation Network

Two views were created to show a paper's immediate citation network, Circle View and Paper Graph. The basis behind having two views is that Circle View is a very rigid visualization method that only shows two levels of a papers immediate citation network and Paper Graph was created using the `NodeGraph` class which allows for more a more dynamic user experience. Also, the views represent the relationships between papers differently: Circle View does not show edges but duplicates nodes, and Paper Graph shows multiple edges to the same node.

5.1.1 Circle View

Circle View (Bergström and Whitehead, 2006) was originally created at University of California at Santa Cruz in 2004 as part of Peter Bergström's undergraduate thesis research under the supervision of professor E. James Whitehead. A screenshot of the original version of Circle View is shown in Figure 10.1 below.


CircleView
View Mode
References
Screen Size
1024x768
Back
Open History
CircleView. © 2004. UC Regents.
Enhanced web document summarization using hyperlinks
Number of Citations
■ 0 citations.
🟦 1 - 5 citations.
🟩 6 - 10 citations.
🟨 11 - 20 citations.
🟧 21 - 50 citations.
🟥 > 50 citations.
Number of References
○ < 10 references.
● > 10 references.
Enhanced web document summarization using hyperlinks
J.-Y. Delort
B. Bouchon-Meunier
M. Rifqi
[10 refs]
[0 cites]
Automatic resource compilation by analyzing hyperlink structure and
Soumen Chakrabarti
Byron Dom
Prabhakar Raghavan
Sridhar Rajagopalan
David Gibson
[0 refs]
[81 cites]
Summarizing text documents: sentence selection and evaluati
[7 refs]
[18 cites]
A vector space model for automatic indexing
[1 ref]
[56 cites]
Topical locality in the Web
[17 refs]
[15 cites]
Seeing the whole in parts: text summarization for web browsing on handheld
[18 refs]
[14 cites]
OCELOT: a system for summarizing Web pages
[5 refs]
[13 cites]
Towards general measures of comparison of objects
[0 refs]
[12 cites]
A Web navigation tool for the blind
[1 ref]
[10 cites]
Automatically summarising Web sites: is there a way around
[8 refs]
[6 cites]
Interactive Document Summarisation Using Automatically Extracted
[0 refs]
[0 cites]
SM
XC
CM+
Buc
JG+
CG
Pal
Sal
Jal
MG+
Edm
PC
WM
Rij
Law
SS+
Les
GB+
BW
Kle
BP
Qui
Encountered a bug? Please report it by clicking here. Thanks!

Automatic resource compilation by analyzing hyperlink structure and associated text

Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, Jon Kleinberg.

Proceedings of the seventh international conference on World Wide Web
7
Brisbane, Australia
Pages: 65 - 74
Year of Publication: 1998
ISSN:0169-7552
Also published in ...

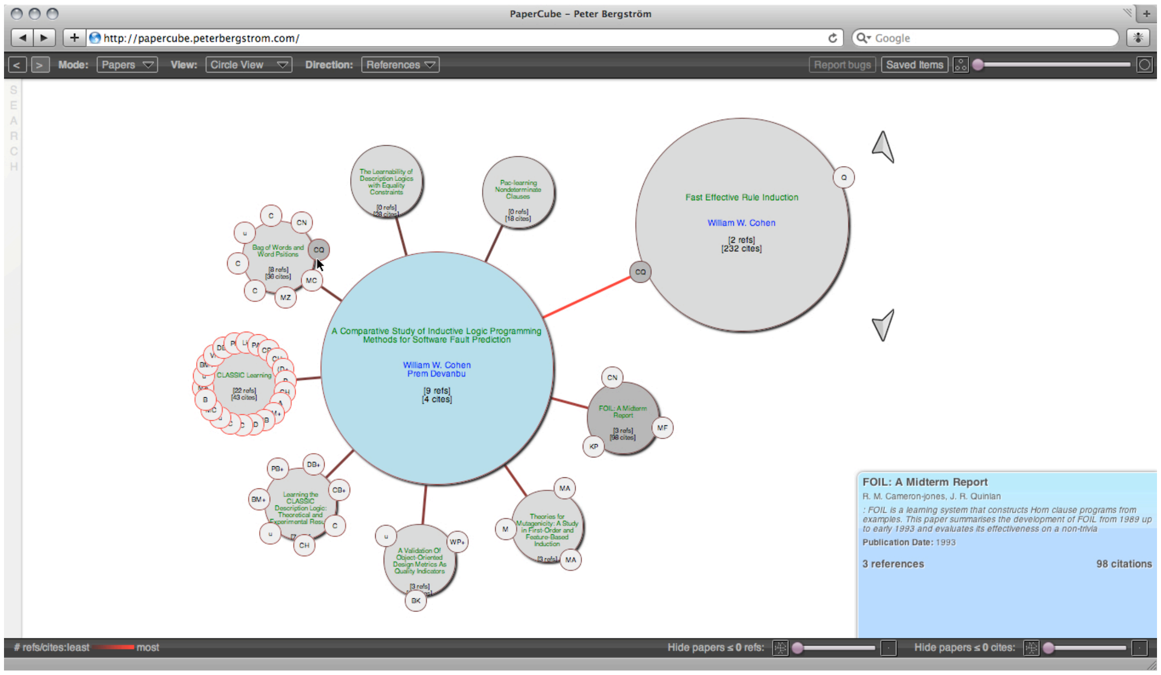
Elsevier Science Publishers B. V.
Amsterdam, The Netherlands, The Netherlands

ACM Digital Library Link:
http://portal.acm.org/citation.cfm?id=297821

Figure 5.1: Screenshot of the original Circle View application from 2004.

When the original research was presented at IVICA 2006, it was thought to have a lot of potential, but no user study had been conducted to formally validate CIRCLE VIEW's visualization method. Therefore, by including it in this work, it could be validated and compared against other visualizations of the same citation network meta data. The hypothesis was that users will find this view quite effective and the user study results supported that quite strongly. Participants selected CIRCLE VIEW as the most effective visualization of paper citation networks.

The original implementation of CIRCLE VIEW was rendered with Scalable Vector Graphics generated by the server written in PHP and required the web browser to reload when any user interaction changed the state of the view. This work expands upon the original CIRCLE VIEW by making it rendered in real time in the web browser as well as adding the ability to dynamically alter its the visualization based on a set of display thresholds. By using the slider controls, thresholds can be specified for the papers visible based on their number of references or citations. Other optimizations in the visualization are reflected based on research based on cognitive psychology. One of the improvements between the versions is that the color scale that signifies the number of references or citations is a non-temperature based scale. The new version of CIRCLE VIEW, as shown in Figure 5.2 adapted for this project is rendered using HTML instead of SVG.



FOIL: A Midterm Report

R. M. Cameron-jones, J. R. Quinlan

: FOIL is a learning system that constructs Horn clause programs from examples. This paper summarises the development of FOIL from 1989 up to early 1993 and evaluates its effectiveness on a non-trivia

Publication Date: 1993

3 references **98 citations**

Figure 5.2: PaperCube's version of CIRCLE VIEW showing data for a paper.

The goal of this novel visualization method is to create away to easily show a paper, its references and its references' references. The visualization used circles around circles and showed papers that occurred more than once in the graph as distinct nodes in order to eliminate the need to show edges crossing all over the screen. The visualization is arranged in a fisheye configuration with the focused paper showing a lot of meta data including its title, authors, number of references and citations. The subsequent levels of related papers are smaller and show less information.

References are ordered by the number of citations they have, and citations are ordered by the number of references. Depending on the view direction, the paper with the most references or citations is the default detailed paper and subsequent papers are arranged clockwise. In order to provide a fast way to assess significance, color coding is used to signify the number of references and citations for the papers relative to the rest in the visualization. The color gradient ranges from black to red where bright red signifies that the paper has the highest number of relations. In the case of viewing references, the color around the circles' border signifies the paper's number of references and the lines that go from the focused paper to its first-level references signify the paper's number of citations. When the relation direction is reversed, the meaning of the lines change accordingly.

When interacting with the view and mousing over a paper circle, bibliographic meta data pertaining to the paper is displayed and any additional occurrences of that reference are highlighted. Since the visualization allows for the same paper node to be rendered more than once, it is easy to count multiple occurrences on the screen. Circle View's visualization method is very rigid and only allows for two levels of references or citations to be shown at any given time. Other views in PaperCube are much more flexible and allow for up to fifteen levels to be displayed.

5.1.2 Paper Graph

The motivation for creating Paper Graph view was to create a view of a paper's immediate citation network that offers the flexibility that Circle View explicitly eliminates by its design. Paper Graph shows either references or citations at any given time, not both. The visualization uses the `NodeGraph` class that renders relationships in the form of an undirected graph. Each paper is represented as a node and any reference or citation relationships to other papers are shown as edges. The widths of the edges is variable depending on its implied relevance. In this case, the relevance is determined by the number of citations, when viewing its references, or the number of references, when viewing its citations.

As with the other views, it is possible to dynamically adjust the thresholds that determine the parameters of the visualization by hiding nodes that fall below the specified reference or citation thresholds. Also, in contrast to Circle View, the number of levels of the citation network is customizable. However, the view is not designed to be effective beyond five levels. Figure 5.3 shows, the focused paper's references up to three levels deep and with the display thresholds set to only show papers with at least three references and thirty citations.

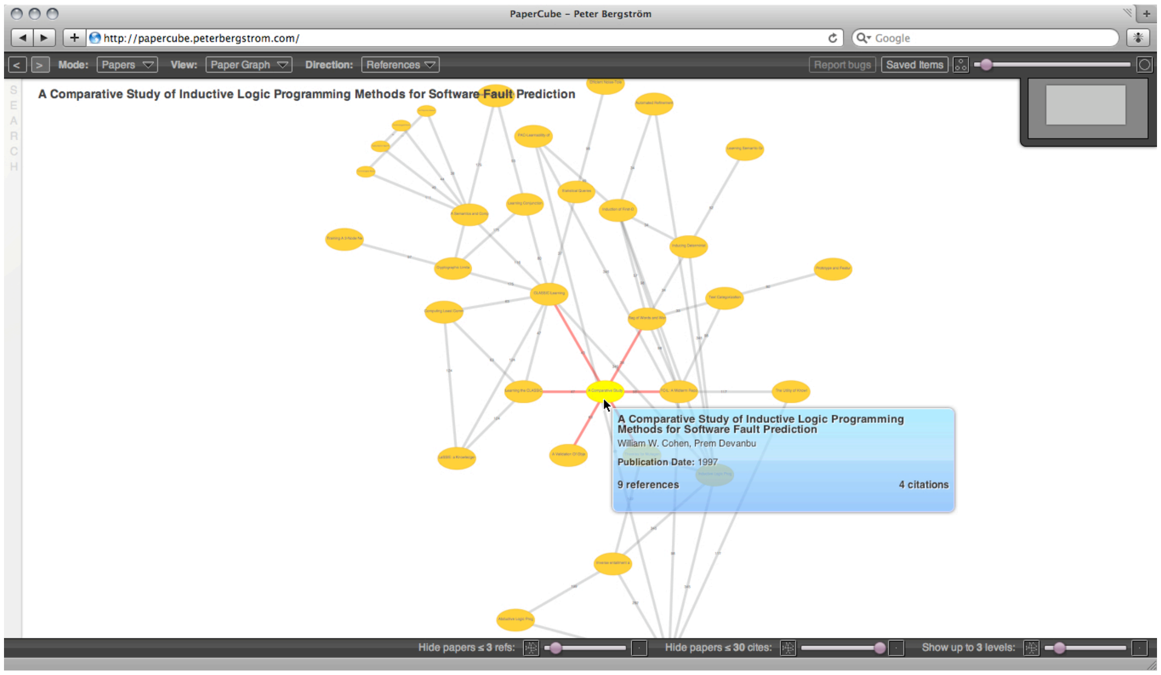

PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Google
Mode: Papers
View: Paper Graph
Direction: References
Report bugs
Saved Items
SEARCH
A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction
A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction
William W. Cohen, Prem Devanbu
Publication Date: 1997
9 references
4 citations
Hide papers ≤ 3 refs:
Hide papers ≤ 30 cites:
Show up to 3 levels:

Figure 5.3: A paper's citation network shown to three levels of references using Paper Graph view.

The hope for this view was to use it to contrast Circle View in the way that it presents the relationships between papers as well as adding some more flexibility that Circle View lacks. The user study did validate the fact that the two views are very similar, but Paper Graph view was shown to rate much lower than "Circle View" and thought to be redundant.

5.2 Visualizing An Extended Citation Network

When exploring ways to show a large number of levels in a paper's citation network, the best visualization method seemed to be a tree map. Tree maps are very powerful when it comes to showing deep, intricate hierarchical relationships. Therefore, the idea of using a tree map seemed logical.

The hope for this view was to show as much bibliographic meta data as possible and let the user adjust thresholds that determine the display parameters using slider controls to narrow down the potentially large data set to a very focused set of relevant information. The results from the user study pointed out that participants were overwhelmed by the large amount of information rendered.

This view, even though it was hoped to be one of the most liked views, turned out to be the most disliked due to its data density.

Since a citation network may contain relationships that can cause reference loops that break the simple definition of a tree-based graph, papers that are cited multiple times are only explored deeper once in the visualization and subsequent occurrences are rendered only to show the reference itself and not its subtree. Therefore, just as with Circle View, papers that occur more than once in the graph are rendered as multiple nodes without showing edges.

Furthermore, in order to make the visualization more readable, the height of all the nodes are the same and as result, does not visually fill the whole window since citation subtrees may have different depths. The Tree Map view has a relatively basic rendering algorithm that reads the width of the root node and subdivides subsequent levels of the tree equally to hold its children. The height is determined by looking at the maximum explored tree depth of the tree and making each node the same height to fill the height of the screen.

When viewing a paper's references, the focused paper is on top and its references are below it. When viewing a paper's citations, the reverse is done and the focused paper appears at the bottom. In the case of viewing a focused paper's references, the focused paper is located at the top and below are the referenced papers, and below those papers' references, and so on.

When hovering over a node with the mouse cursor, the path from that paper to the focused paper will be highlighted. This is done to make it easy to find where a paper belongs in the larger context of the citation network. Also, any other nodes representing the paper will be highlighted in green so that they can be easily identified.

In Figure 5.4, the references for the paper "A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction" are displayed. When hovering over a paper, the path back up to the focused paper is shown in yellow and two other instances of the hovered over paper are highlighted in green.

In Figure 5.5, the same focused paper as above is shown but in this case, the viewing direction option has been changed to view citations instead of references. Since the focused paper only has four citations, the citation network visualized is much smaller. Even though the option was to show up to fifteen levels of citations, the height of each level of the visualization is optimized to give the maximum height for the three citation levels actually present.

Since the view can potentially show a very large amount of data, it was important to be able to determine what should be visible. One the most powerful features of this view that it allows papers to be shown or hidden based on the number of references and citations they have and also show

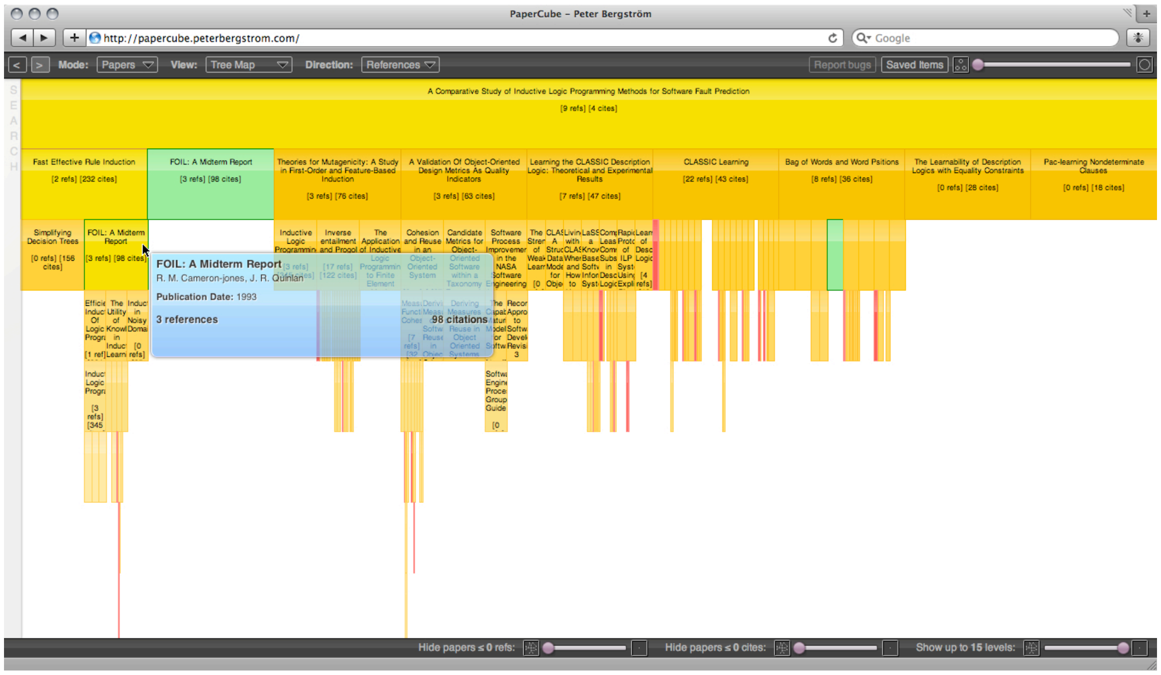

PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Mode: Papers
View: Tree Map
Direction: References
Report bugs
Saved Items
A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction
[9 refs] [4 cites]
Fast Effective Rule Induction
[2 refs] [232 cites]
FOIL: A Midterm Report
[3 refs] [98 cites]
Theories for Mutagenicity: A Study in First-Order and Feature-Based Induction
[3 refs] [76 cites]
A Validation Of Object-Oriented Design Metrics As Quality Indicators
[3 refs] [63 cites]
Learning the CLASSIC Description Logic: Theoretical and Experimental Results
[7 refs] [47 cites]
CLASSIC Learning
[22 refs] [43 cites]
Bag of Words and Word Psitions
[8 refs] [36 cites]
The Learnability of Description Logics with Equality Constraints
[0 refs] [28 cites]
Pac-learning Nondeterminate Clauses
[0 refs] [18 cites]
Simplifying Decision Trees
[0 refs] [156 cites]
FOIL: A Midterm Report
[3 refs] [98 cites]
FOIL: A Midterm Report
R. M. Cameron-jones, J. R. Quinlan
Publication Date: 1993
3 references
98 citations
Hide papers ≤ 0 refs:
Hide papers ≤ 0 cites:
Show up to 15 levels:

Figure 5.4: Screenshot showing TREE MAP view displaying the references for a paper up to fifteen levels deep.

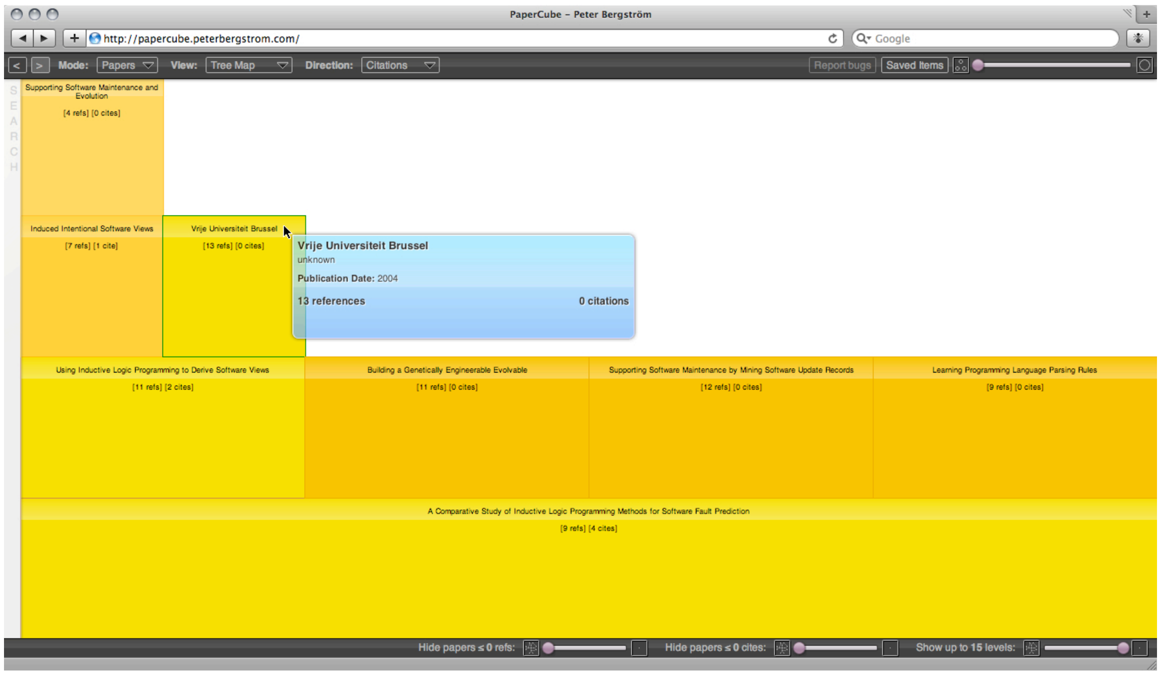

PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Mode: Papers
View: Tree Map
Direction: Citations
Report bugs
Saved Items
Supporting Software Maintenance and Evolution
[4 refs] [0 cites]
Induced Intentional Software Views
[7 refs] [1 cite]
Vrije Universiteit Brussel
[13 refs] [0 cites]
Vrije Universiteit Brussel
unknown
Publication Date: 2004
13 references
0 citations
Using Inductive Logic Programming to Derive Software Views
[11 refs] [2 cites]
Building a Genetically Engineerable Evolvable
[11 refs] [0 cites]
Supporting Software Maintenance by Mining Software Update Records
[12 refs] [0 cites]
Learning Programming Language Parsing Rules
[9 refs] [0 cites]
A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction
[9 refs] [4 cites]
Hide papers ≤ 0 refs:
Hide papers ≤ 0 cites:
Show up to 15 levels:

Figure 5.5: Screenshot of showing TREE MAP view displaying the citations for a paper up to fifteen levels deep.

more or less levels deep into the citation network. The maximum depth of paper retrieval is fifteen levels, which seems to be a good balance between breadth and performance. If the visualization is too dense, the rendering algorithm will abstract out the parts that are too small to show at the current zoom level. If then zoomed in, that section that was previously abstracted away will be automatically fetched from the server and displayed.

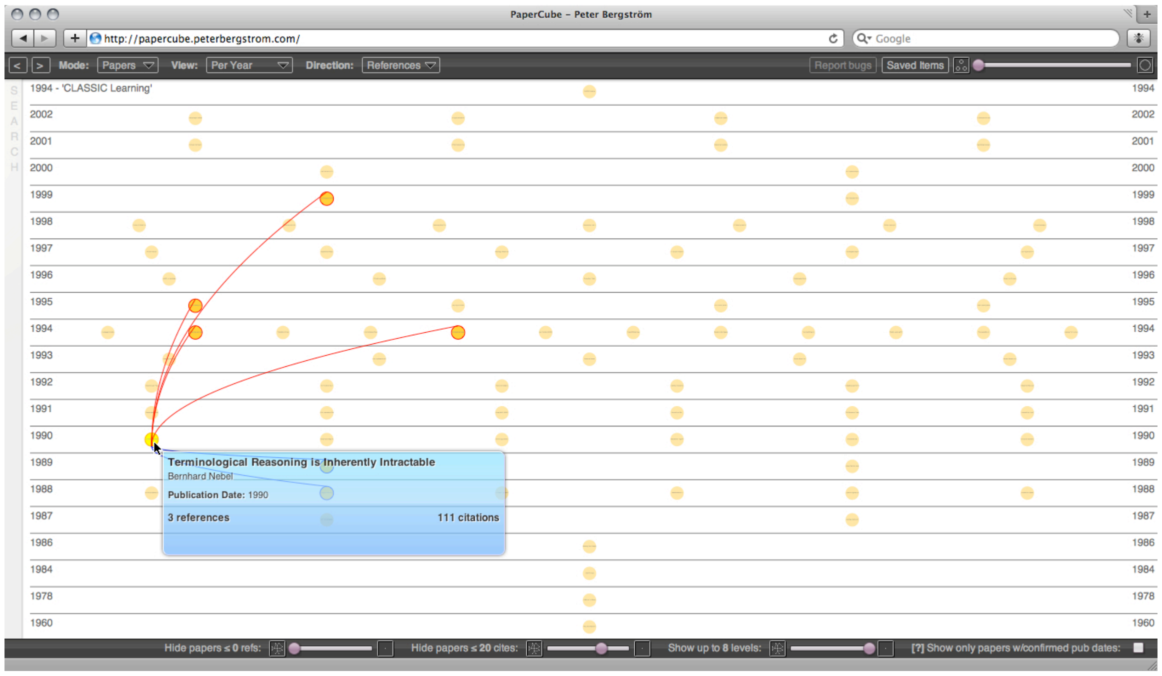
5.3 Visualizing Citation Networks Chronologically

Showing a paper's citation network chronologically was one of the early goals of this work. The ability to look a paper and trace back to seminal works seemed very appealing and useful. In order to do this, Papers Per Year view was created to test this concept and resulted in perhaps the most interesting visualization of papers that this tool offers. Participants rated this view very highly and besides Circle View, it was the highest rated visualization method for displaying paper citation network relationships.

When viewing a paper's references, the focused paper is at the top of the browser window and when viewing a paper's citations, the focused paper is at the bottom. The view allows for the exploration of a paper's citation network and all the papers are represented as nodes organized chronologically by publication year.

When the mouse hovers over a paper, its bibliographic meta data is shown as well as any reference and citation relationships are highlighted with lines to the related papers. The references are shown as blue lines and citations are shown as red lines. It is possible, by using the "pin lines" option in the fan menu to lock a given paper's relationship lines so that a those papers can easily be found and their bibliographic meta data be read without losing where they are located on the screen. It is possible adjust the thresholds using slider controls that determine the number of references and citations papers have to have in order to be rendered on the screen. As is the case with some other views, it is possible to specify the maximum depth to recursively explore.

In Figure 5.6, the citation network for the paper "CLASSIC Learning" is rendered up to eight levels deep and hiding papers with less than twenty citations. Since this example shows the references view direction, the paper is at the top the screen.


PaperCube – Peter Bergström

http://papercube.peterbergstrom.com/

Mode: Papers   View: Per Year   Direction: References

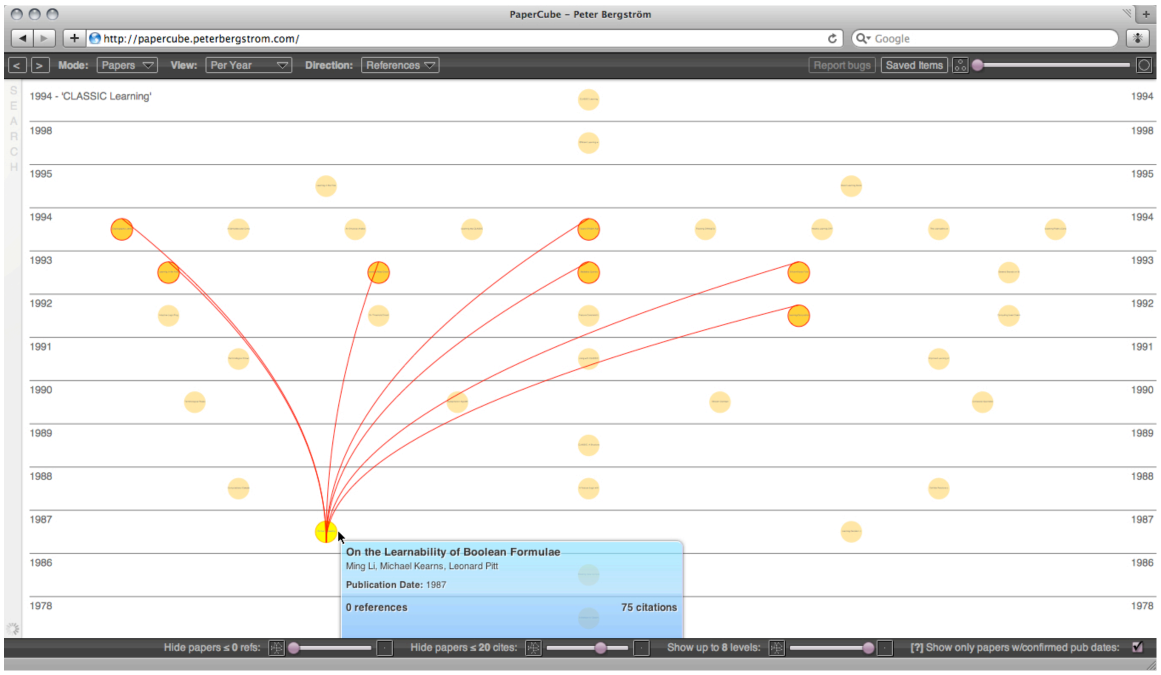
Terminological Reasoning is Inherently Intractable
Bernhard Nebel
Publication Date: 1990
3 references   111 citations

Hide papers ≤ 0 refs: Hide papers ≤ 20 cites: Show up to 8 levels: [?] Show only papers w/confirmed pub dates:

Figure 5.6: Screenshot of the PAPERS PER YEAR show a paper's references up to eight levels deep and with at least twenty citations. In this case, the view shows papers with both confirmed and unconfirmed publication years.

Since the paper in the example was published in 1994, it is impossible for the paper to have references that were written after this date. One deficiency of the CiteSeer data set is that a lot of years are incorrect. Therefore, there is the ability to hide papers with unconfirmed publication years. The publication years were confirmed by cross referencing the papers in the limited CiteSeer data set with the bibliographic meta data harvested through CiteSeerX. As shown in Figure 5.7, when the checkbox the bottom right of the screen, "Show only papers w/confirmed pub dates" is checked, the rendered data set is reduced and the chronology of the visible papers works out correctly. By looking at a parameter for each paper that indicates the confirmation status of the publication year, unconfirmed papers are hidden.

The visualization originally posed some interesting technical challenges that should be mentioned since it impacts the features. The nature of the visualization method makes it possible that a very large number of papers can be retrieved and shown to the point where the computer may run out of memory. There was one instance where a paper was explored without a depth limit and retrieved 40,000 papers. Interestingly enough, the browser was able to handle that amount of data until the computer ran out of memory. Therefore, the adjustable depth parameter has a maximum of fifteen levels deep which in some cases may still yield a very large amount of papers.


PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Google
Report bugs
Saved Items
1994 - 'CLASSIC Learning'
1998
1995
1994
1993
1992
1991
1990
1989
1988
1987
1986
1978

Mode: Papers View: Per Year Direction: References

On the Learnability of Boolean Formulae
Ming Li, Michael Kearns, Leonard Pitt
Publication Date: 1987
0 references 75 citations

Hide papers ≤ 0 refs: Hide papers ≤ 20 cites: Show up to 8 levels: [?] Show only papers w/confirmed pub dates: ✓

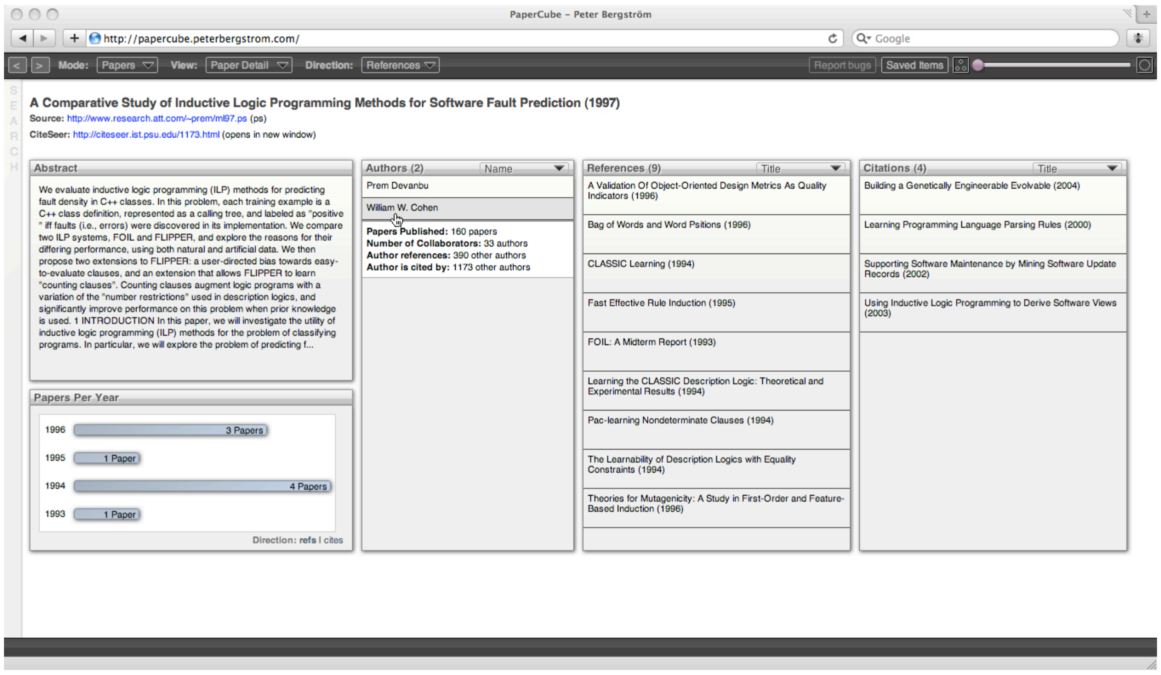
Figure 5.7: Screenshot of Papers Per Year view showing the same paper as in Figure 5.6, but now with the '"Show only papers w/confirmed pub dates" turned on.

If a year in the visualization contains more than 300 papers, the papers are not rendered. If the thresholds are adjusted to reduce the number of papers contained in that year, papers in that year may be rendered. The decision to do this was twofold; first, to preserve the user experience and not allow the browser to negatively impact the system in terms of CPU and memory usage that PaperCube is running on. Second, unless used on a very large display, showing 300 papers in a year results in very small nodes that are hard to read.

5.4 Bringing All of the Visualizations Together in Paper Detail View

Although PaperCube's primary focus is on the visualization of paper citation network relationships, it needed an initial entry point. It was thought that starting PaperCube directly in one of the visualization-based views would perhaps bias users to that view or be potentially confusing. Therefore, Paper Detail view was created as an information dense entry point from where users can make a choice as to where to go next. Surprisingly, this view was rated the most liked view by the participants of the user study because it was a good way to enter and go to subsequent other views.

Paper Detail view shows a the focused paper's authors, references, citations, as well as other information including a link to the paper itself. If any of the items in the list is clicked, PaperCube will refocus. If an author is selected, PaperCube will switch viewing mode to view that author in the default or last used author view. In Figure 5.8, all the details for "A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction" are shown.



A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction (1997)
Source: http://www.research.att.com/~prem/ml97.ps (ps)
CiteSeer: http://citeseer.ist.psu.edu/1173.html (opens in new window)

Abstract

We evaluate inductive logic programming (ILP) methods for predicting fault density in C++ classes. In this problem, each training example is a C++ class definition, represented as a calling tree, and labeled as "positive " iff faults (i.e., errors) were discovered in its implementation. We compare two ILP systems, FOIL and FLIPPER, and explore the reasons for their differing performance, using both natural and artificial data. We then propose two extensions to FLIPPER: a user-directed bias towards easy-to-evaluate clauses, and an extension that allows FLIPPER to learn "counting clauses". Counting clauses augment logic programs with a variation of the "number restrictions" used in description logics, and significantly improve performance on this problem when prior knowledge is used. 1 INTRODUCTION In this paper, we will investigate the utility of inductive logic programming (ILP) methods for the problem of classifying programs. In particular, we will explore the problem of predicting f...

Papers Per Year

1996	3 Papers
1995	1 Paper
1994	4 Papers
1993	1 Paper

Direction: refs | cites

Authors (2) — Name

- Prem Devanbu
- William W. Cohen

Papers Published: 160 papers
Number of Collaborators: 33 authors
Author references: 390 other authors
Author is cited by: 1173 other authors

References (9) — Title

- A Validation Of Object-Oriented Design Metrics As Quality Indicators (1996)
- Bag of Words and Word Psitions (1996)
- CLASSIC Learning (1994)
- Fast Effective Rule Induction (1995)
- FOIL: A Midterm Report (1993)
- Learning the CLASSIC Description Logic: Theoretical and Experimental Results (1994)
- Pac-learning Nondeterminate Clauses (1994)
- The Learnability of Description Logics with Equality Constraints (1994)
- Theories for Mutagenicity: A Study in First-Order and Feature-Based Induction (1996)

Citations (4) — Title

- Building a Genetically Engineerable Evolvable (2004)
- Learning Programming Language Parsing Rules (2000)
- Supporting Software Maintenance by Mining Software Update Records (2002)
- Using Inductive Logic Programming to Derive Software Views (2003)

Figure 5.8: Screenshot of Paper Detail view showing the meta data for a paper.

The lists showing paper information can be sorted by different parameters: title, number of references, number citations, number of authors, and publication year. The list of authors can be sorted by name, number of papers written, number of collaborators, and the number of times the author has been cited. When the mouse hovers over an item summary in a list, the summary will expand to reveal additional meta data. The view also contains a chart that shows a papers references or citations organized per year which was inspired partially by CiteSeerX.

Chapter 6

Author Views

As a secondary goal, this work wanted to explore how the author relationships could be interpreted through visualization as a way to allow researchers to find papers via a different dimension on the available bibliographic meta data. Most existing digital libraries are solely focused on paper relationships and being able derive and visualize author relationships from the existing bibliographic meta data would give researchers a novel way to find new papers.

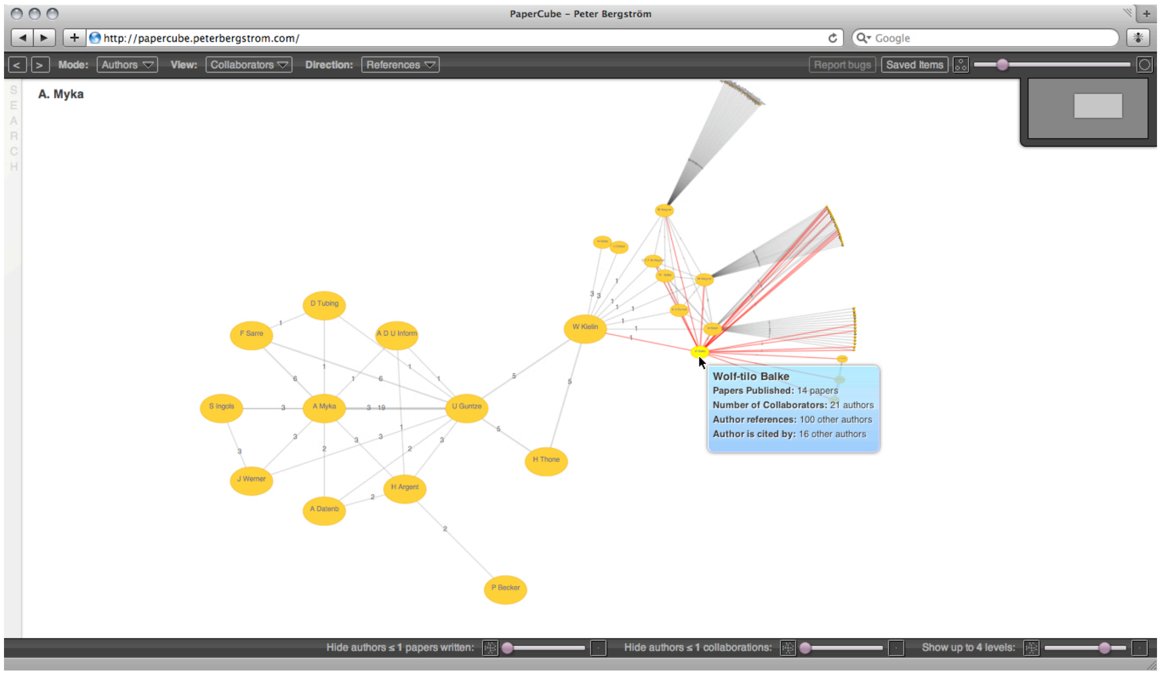
The goal was to show author relationships that could be directly inferred from paper relationships that exist in digital libraries. A paper contains information about its authors and from this, the relationships between authors can be derived. Therefore, the goal was to visualize simple relationships and do not use any calculations to determine what authors are more significant other than looking at the number of relationships between authors and the papers that they have written.

The relationships that were thought to be useful were showing collaboration between authors, the authors that a given author has referenced or the authors that have cited that author, and the published papers for a given author. As with the "Paper" viewing mode, an entry point for the "Author" viewing mode was created.

6.1 Finding Collaborators

COLLABORATORS view is the primary reason why authors were included PaperCube. The hope was that showing collaboration relationships between authors could be an appreciated and powerful feature and appreciated by researchers. By finding a paper and looking at its authors, it should be very simple to look at the author's collaborators and then find other important papers. The participants in the user study rated this view the best visualization of author relationships because it shows information not accessible through most current digital library services.

Collaborators view shows all of the authors that the focused author has coauthored with based on the bibliographic meta data. As with some of the other views, it is based on the `NodeGraph` class that renders relationships as undirected graph with a basic radial fisheye layout.



Wolf-tilo Balke
Papers Published: 14 papers
Number of Collaborators: 21 authors
Author references: 100 other authors
Author is cited by: 16 other authors

Figure 6.1: Collaborators view showing the collaborators for A. Myka up to 4 levels deep.

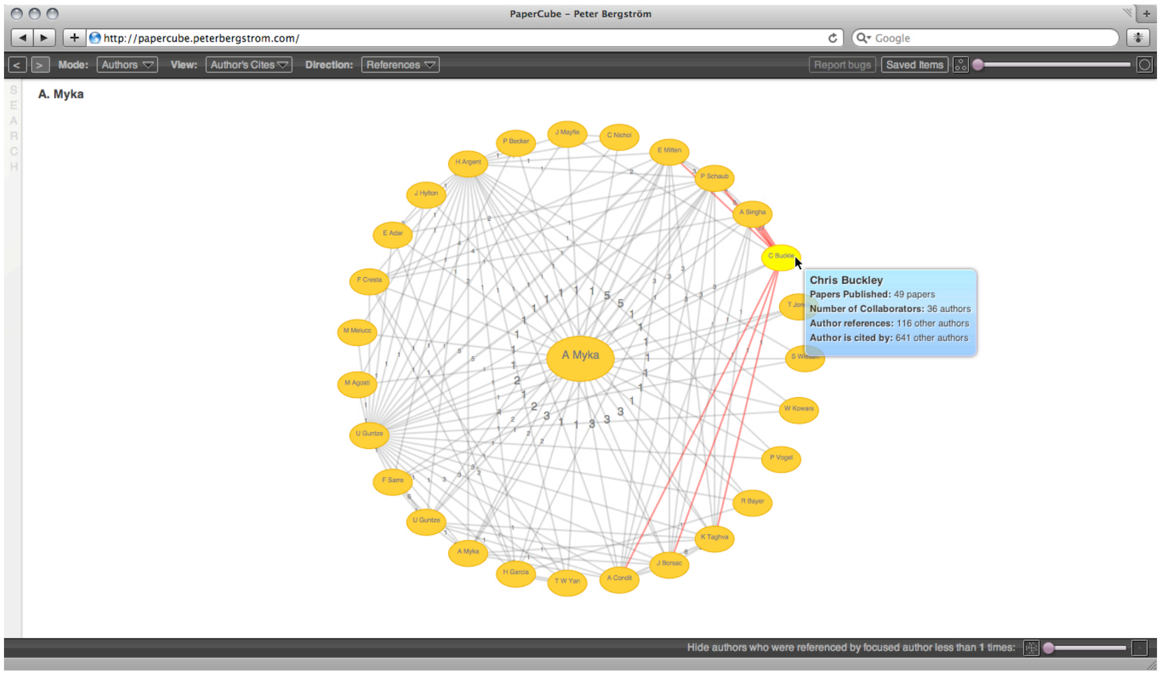
In Figure 9.3, the author A. Myka is shown with a collaboration network up to four levels. Although not all of the authors that Myka has worked with have other collaborators, the ones that do radiate outwards and become smaller the farther away they are from the focused author.

It is easy to limit the number of collaborators shown by increasing the threshold of the number of papers or by increasing the threshold of the number of times the collaborator has collaborated with the given author. However, the most interesting aspect of this view is that it is possible to expand the reach of visualization and show multiple levels of collaboration. Up to five levels of collaboration relationships can be shown that expand radially from the focused author. Therefore, it is easy to see where the focused author belongs in the larger context of a community of authors, and from there a scholar can easily find who is more important than others. If an author has collaborated with many others, that author may be very important to look at, but more significantly, Collaborators view allows a researcher to pick out strong collaboration relationships between authors. If an author has worked with another many times, the edge connecting their nodes will be thicker. As mentioned before, the user study validated that Collaborators view is a very compelling way to view author relationships and discover new authors and papers.

6.2 Finding Author Citations and References

The Author Cites view is another way of showing author relationships. It was thought that it would be interesting to know what authors have cited a given author and what authors that author had referenced across all his or her published papers. The hope was to create a view that gives a general overview of the citation space throughout the career of an author. The potential for this view was that it would be easy to see if an author influences many authors or vice versa. In general the participants found this view to be useful and it was rated as the second most liked author visualization.

In Figure 6.2, all the authors that A. Myka has referenced in all of the works that he has written throughout his career are shown through one visualization.



Mode: Authors View: Author's Cites Direction: References

A. Myka

Chris Buckley
Papers Published: 49 papers
Number of Collaborators: 36 authors
Author references: 116 other authors
Author is cited by: 641 other authors

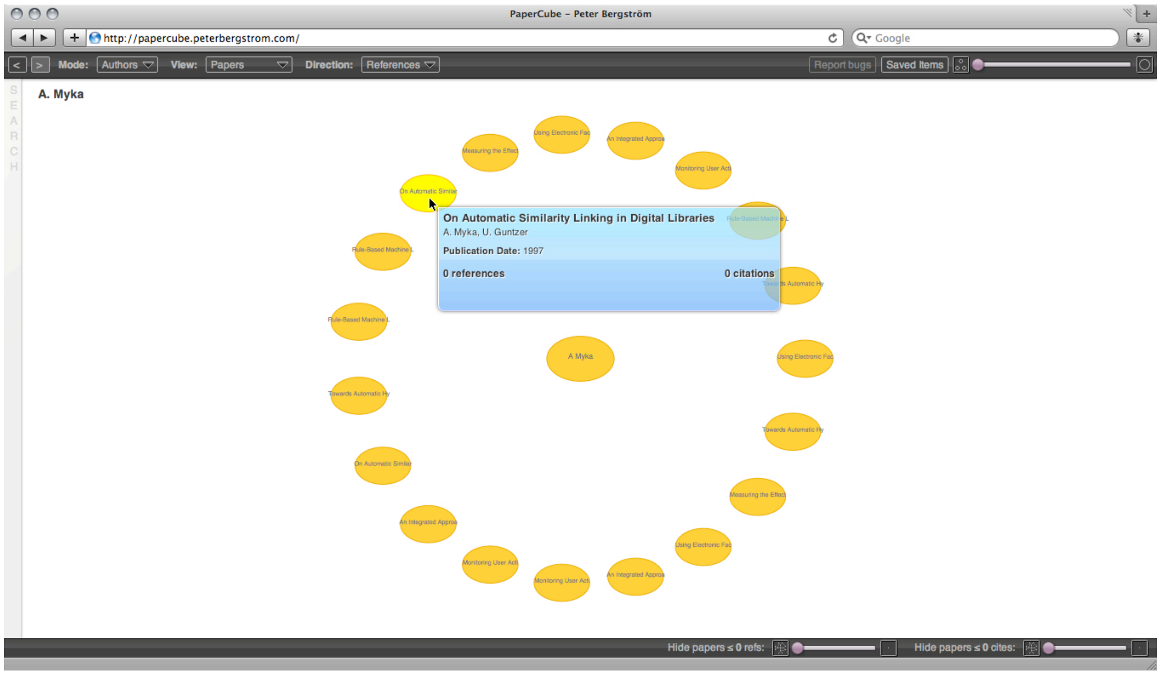
Hide authors who were referenced by focused author less than 1 times:

Figure 6.2: Author Cites view showing all the authors that the author A. Myka has referenced.

6.3 Finding an Author's Papers

The reason behind creating the Papers view was to show a very simple relationship, namely the papers that an author has published, by using a visualization. The idea was to examine if using a visualization to show bibliographic meta data that can be effectively displayed in a textual list, would be appreciated. The view was created to contrast Author Detail view which shows the papers for the focused author in a compact list.

The papers are arranged around the focused author in a circle. The display threshold parameters can be adjusted to show only papers with reference or citation counts above a given threshold. If a paper is clicked, all normal paper fan menu actions are available and the last used paper view is displayed if refocused. In Figure 6.3, all the papers that CiteSeer knows that A. Myka has written is shown. The user study showed that this view was not liked because it was too basic. If there would have been some additional thresholds to filter the rendered bibliographic meta data, perhaps by publication year, this view would have been more useful.



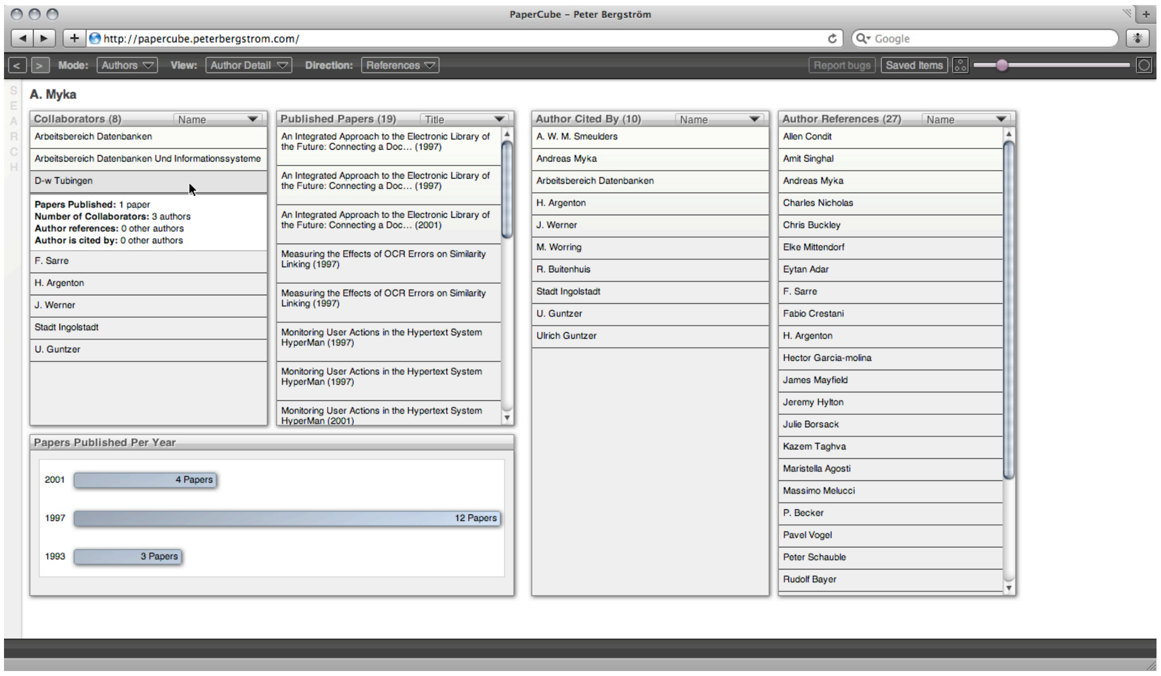
On Automatic Similarity Linking in Digital Libraries
A. Myka, U. Guntzer
Publication Date: 1997
0 references
0 citations

Figure 6.3: Papers view showing all the papers that A. Myka has written.

6.4 Bringing All of It Together in Author Detail View

As with papers, a table-based view was created to serve as a entry point into the "Author" viewing mode that shows all the available bibliographic meta data for a given author. As expected, like Paper Detail view, this view was rated highly by participants.

Author Detail view shows all the meta data that the other author views show graphically in addition to a breakdown of the years when an author published papers. The information that is shown includes collaborators, papers written, the authors that the focused author references, and the authors that cite the focused author. If any of the items in the lists are clicked, PaperCube can refocus on the item. If a paper is selected, the viewing mode will be switched to show the paper.


PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Google
Report bugs
Saved Items

Mode: Authors View: Author Detail Direction: References

A. Myka

Collaborators (8) — Name

Arbeitsbereich Datenbanken
Arbeitsbereich Datenbanken Und Informationssysteme
D-w Tubingen
Papers Published: 1 paper
Number of Collaborators: 3 authors
Author references: 0 other authors
Author is cited by: 0 other authors
F. Sarre
H. Argenton
J. Werner
Stadt Ingolstadt
U. Guntzer

Published Papers (19) — Title

An Integrated Approach to the Electronic Library of the Future: Connecting a Doc... (1997)
An Integrated Approach to the Electronic Library of the Future: Connecting a Doc... (1997)
An Integrated Approach to the Electronic Library of the Future: Connecting a Doc... (2001)
Measuring the Effects of OCR Errors on Similarity Linking (1997)
Measuring the Effects of OCR Errors on Similarity Linking (1997)
Monitoring User Actions in the Hypertext System HyperMan (1997)
Monitoring User Actions in the Hypertext System HyperMan (1997)
Monitoring User Actions in the Hypertext System HyperMan (2001)

Author Cited By (10) — Name

A. W. M. Smeulders
Andreas Myka
Arbeitsbereich Datenbanken
H. Argenton
J. Werner
M. Worring
R. Buitenhuis
Stadt Ingolstadt
U. Guntzer
Ulrich Guntzer

Author References (27) — Name

Allen Condit
Amit Singhal
Andreas Myka
Charles Nicholas
Chris Buckley
Elke Mittendorf
Eytan Adar
F. Sarre
Fabio Crestani
H. Argenton
Hector Garcia-molina
James Mayfield
Jeremy Hylton
Julie Borsack
Kazem Taghva
Maristella Agosti
Massimo Melucci
P. Becker
Pavel Vogel
Peter Schauble
Rudolf Bayer

Papers Published Per Year

2001	4 Papers
1997	12 Papers
1993	3 Papers

Figure 6.4: Author Detail view showing the details for A. Myka.

Again, as with Paper Detail view, the tables can be sorted. The list of papers published by the author can be sorted by title, number of references, number of citations, number of authors, and publication year. The list of authors can be sorted by name, number of papers written, number of collaborators, and the number of times the author has been cited.

Chapter 7

PaperCube Architectural Overview

PaperCube consist of three distinct components. First, a MySQL database that stores the bibliographic meta data for papers and authors. Second, a PHP-based database interface listens to requests for paper or author bibliographic meta data from the client and returns the result from the database in JSON format. Third, the SproutCore-based application that runs in a web browser. The majority of the business logic and complexity lies in the web browser leaving the two layers below lightweight. Figure 7.1 shows a high-level architectural diagram showing the relationship between the three components.

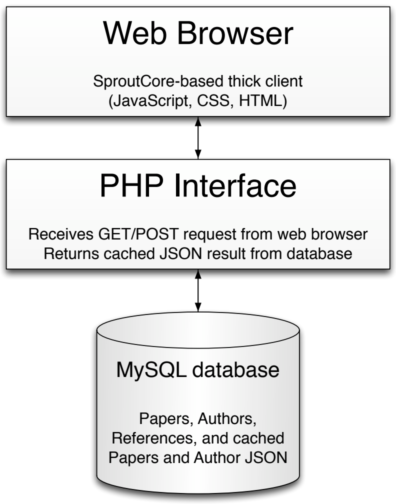


Figure 7.1: The high-level architecture showing the three distinct components of PaperCube.

Chapter 8

Server Architecture

PaperCube's server component consists of a lightweight PHP interface that translates requests from the web browser to queries to be executed by the MySQL database. This chapter will first discuss the JSON data format used to send the bibliographic meta data to the client. Then, discuss in detail how the PHP-based database interface and MySQL database are structured.

8.1 JSON Data Format

The server component sends bibliographic meta data to the web browser in JSON format. JSON is short for "JavaScript Object Notation" (http://www.json.org) and is ideal for sending information over the wire to a web browser. In contrast to XML, JSON is very compact because it is not very verbose. JSON is structured as a JavaScript object with keys and values. The values can be any data type ranging from primitive types such as integers, floats, and strings, to more complex objects such as arrays, objects, and even functions. Therefore, it is ideal to send to a JavaScript-based client application because it easily converted to useable data by the browser using built in JSON parsers or `eval()`.

The JSON data format used in PaperCube translates to the two primary content types: papers and authors. The JSON data is pulled from caching tables in the database that contains the predefined JSON strings on a per record basis. The JSON data was constructed in batch before PaperCube was deployed in order to ensure better response times when doing paper and author lookups since the relationships for papers and authors are not efficiently computed in real time. Instead, queries against the database will generate a list of unique identifiers, called *GUIDs,* that then will be retrieved from the cache tables, concatenated into an array, converted to a string, and then sent back to the client. Although the general case deals with cached JSON, Paper Detail view required additional database fields that the other view do not. Therefore, some special queries return JSON data that is not pulled from the cache, but rather fetched real time and then encoded into JSON in PHP.

Each JSON data response from the web server is in the following general format as shown in Code Snippet 8.1. The `status` string is set to `1` if it is a successful response and `0` with an error message if there is a problem. The `data` array contains the JSON record objects for response.

```
{
  "status":"1",
  "data": [
    {
       /* content object details go here */
    },
    /* more content objects */
  ]
}
```

Code Snippet 8.1: Basic JSON response structure showing a successful response.

Code Snippet 8.2 shows the JSON data hash for a paper that represents the bibliographic meta data returned from a cached GUID-based request.

```
{
  "guid":"PAPER-9772",
  "type":"Paper",
  "title":"HYDRA: A Hypertext Model for Structuring Informal Requirements
           Representations",
  "abstract":"The ultimate measurement for software quality is the degree
              to which user needs are satisfied by the system. User needs
              are an essential input for developing a requirements
              specification and, in the first place, are most often
              represented using natural language, pictures, or graphics
              (informal representations). The consideration of user needs
              as a driving force throughout the development process",
  "year":"1995",
  "fixedyear":"1",
  "references":[
    "PAPER-433312", "PAPER-359668", "PAPER-78573",
    "PAPER-76765", "PAPER-219558"
  ],
  "citations":[
    "PAPER-222727", "PAPER-459431", "PAPER-52417",
    "PAPER-439616", "PAPER-272109"
  ],
  "authors":[
    {
      "guid":"AUTHOR-21507", "name":"Peter Haumer"
    },
    {
      "guid":"AUTHOR-713", "name":"Klaus Pohl"
    }
  ]
}
```

Code Snippet 8.2: A sample paper JSON data hash.

The JSON matches up with the `Papercube.Paper` model object described in next chapter. The relationships listed in the JSON only contain GUIDs or in the case of authors, a GUID and a name. If unneeded data were returned as part of the JSON data, it would waste resources on both the server and the client. Instead, it is more efficient to query separately for additional bibliographic meta data for GUIDs when they are requested by the client application.

In the case of authors, the JSON returned is also closely related to its corresponding client model object `Papercube.Author`. Code Snippet 8.3 shows the basic cached JSON data returned for Peter Haumer, one of the authors of the paper shown in Code Snippet 8.2. Please note that due to the number of relationships returned, they have been abbreviated for this example.

```
{
  "type":"Author",
  "guid":"AUTHOR-21507",
  "name":"Peter Haumer",
  "collaborators":[
    ["AUTHOR-713", 11],
    /* more author guids */,
    ["AUTHOR-1292262", 1]
  ],
  "refAuthors":[
    ["AUTHOR-715", 9],
    /* more author guids and counts ordered by count */,
    ["AUTHOR-16171", 1]
  ],
  "citeAuthors":[
    ["AUTHOR-713", 9],
    /* more author guids and counts ordered by count */,
    ["AUTHOR-18649",1]
  ],
  "papers":["PAPER-9772", "PAPER-152388", "PAPER-162916",
            "PAPER-222727", "PAPER-223082", "PAPER-223809",
            "PAPER-224048", "PAPER-369306", "PAPER-520631",
            "PAPER-565996", "PAPER-613532"
  ]
}
```

Code Snippet 8.3: A sample author JSON data hash.

8.2 PHP-Based Database Interface

The PHP interface is a simple script that resides on the Apache web server in the `/api` directory. Since the client application is entirely static without any dynamic components, it only calls the script when it needs bibliographic meta data from the database. The client can perform a set of different queries that fall into two categories. First, search requests that perform a full-text match against a specific database column value for either a paper or author. Second, the client can request one or more papers or authors by GUID. In both cases, the PHP script will return JSON formatted data that the client easily can process. The source code for the script is included in Appendix C.

8.2.1 Search Requests

The client can search for either papers or authors. The database columns that are searchable are the same across both but the data that is returned is different. Table 8.1 shows the relationship between the human-readable string provided in the client user interface and the database columns.

Human-readable string	**field key**	**table**
Paper Title	title	papers
Paper Abstract	abstract	papers
Paper Subject	subject	papers
Paper Publication Year	pubyear	papers
Author Name	name	authors

Table 8.1: Relationship between the human-readable string in the UI and the database.

All but the "Paper Publication Year" search field perform a SQL full-text match against the database column. The full-text search allows for advanced boolean matches that can improve the quality of the search. Although advanced search was not a primary goal, the ability to do boolean searches was a fortunate byproduct of using MySQL as the database. The search returns fifty matching items at a time and allows the user to page to see more results.

The interface that the client uses for searches is quite simple. There are two search actions, `searchPapers` and `searchAuthors`. A search, regardless if it is searching for authors or papers, contains the GET parameters as shown in Table 8.2.

Key	**Description**
action	This can either be 'searchPapers' or 'searchAuthors.'
searchKey	This can either be any of the field keys listed above.
queryStart	The start of the page being requested.
queryLimit	The length of the page being requested.
sc	The sequential search increment, this is used for synchronization on the client.

Table 8.2: Available GET parameters for searching for papers and authors.

If a user searches for papers with the term "hypertext" in the title, the request can be performed against either papers or authors content. In either case it will be retuned as a JSON object containing the matching paper or author content objects. Figure 8.1 and Figure 8.2 show example requests for papers and authors. The *host* component of the URL is the domain where the web server is located.

```
http://host/api/request.php?action=searchPapers&searchKey=title
&searchValue=hypertext&queryStart=0&queryLimit=50&sc=1
```

Figure 8.1: Search request for Paper content objects.

```
http://host/api/request.php?action=searchPapers&searchKey=title
&searchValue=hypertext&queryStart=0&queryLimit=50&sc=1
```

Figure 8.2: Search request for Author content objects.

8.2.2 Requests Using GUIDs

The second category of requests for data are requests for papers or authors based on GUIDs alone. The client's views need GUIDs during rendering and request them from the database in batch form. The advantage of requesting items in batch is that in addition to processing the received data, it is relatively expensive to have multiple pending requests. Instead of having the browser send ten requests for ten papers, it is more efficient to send one request for all of them because the browser can only send a limited number of requests at a time. Therefore, any additional requests will be queued until other requests complete.

The requests for meta data by GUID are simple and require the `action` name and `guid` fields. The `guid` field can be a comma-separated list of GUIDs that is either sent as part of the GET request or as a POST variable in the form `?guid=PAPER-12312,PAPER-244212,PAPER-56211`. The advantage of using a POST is that there is no limit on the number of GUIDs that can the requested. GET requests are based on a resource URL, which can be unreliable beyond 255 characters. Table 8.3 shows a summary of the requests.

Action	**Description**
getAuthorDetails	Returns cached author JSON for all matching GUIDs.
getPaperDetails	Returns cached paper JSON for all matching GUIDs.
getAllDataForPaper	Returns additional fields for the focused paper in the detail view.

Table 8.3: The GUID-only requests and what data they return.

8.3 MySQL Database Implementation

The MySQL database that the PHP interface queries against has six tables listed below. The full database schema is available in Appendix D.

All database tables that contain content objects that may be returned to the web application contain a `type` database column that is used to map the JSON data object to an extended SproutCore `SC.Record` content object in the client data store, `SC.Store`. For example, in the case of the `authors` table, each row in the table has its `type` is set "Author."

The tables map the paper-author and paper-paper relations using intermediate tables that contain both GUIDs for the items as foreign keys which are referenced in Figure 8.3 as FK. The `guid` field in the tables are the primary keys and are tagged as PK in the figure. For example, using the reference_rel table, it is possible to find all the papers that a given paper references by matching its GUID against the `pguid` column and conversely, find all the papers that cite the given paper by matching its GUID against the `rguid` column.

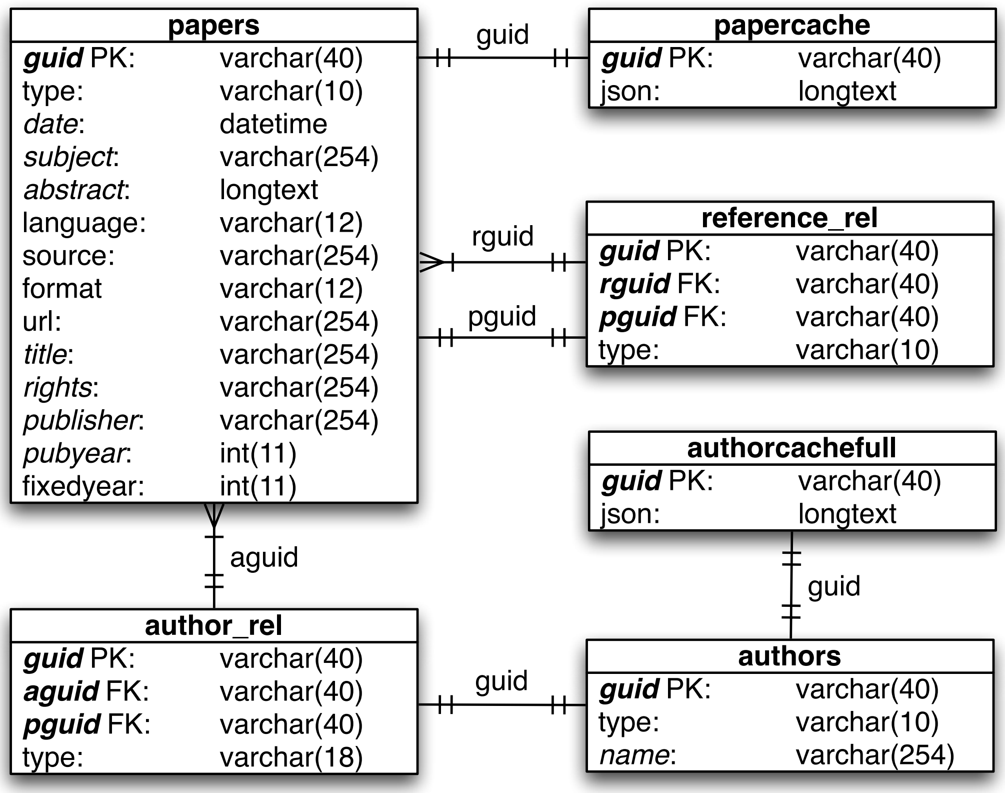


Figure 8.3: PaperCube's database schema. Bolded fields are primary keys, italicized fields are indexed. The `varchar` fields have full-text indexes applied to them.

8.3.1 Paper Tables

The papers table contains the bibliographic meta data for a paper as it exists in the CiteSeer export. Not all fields are returned to the client, but are available if the PaperCube application is extended to show that data. In order to make PaperCube's data set searchable, full-text indexes are set on the `title`, `abstract`, `subject`, `rights`, and `publisher` columns and indexes are set on the `date` and `pubyear` columns although not all are exposed to the client application.

Papers can have zero or more references and the reference_rel table maintains this relationship. A paper with the GUID `pguid` can have multiple papers with their GUID listed in the column `rguid`. Both of those columns are indexed for fast access. From this table, finding both directions of paper relationships, references and citations, is a trivial operation.

The final paper table is the papercache which contains the batch processed JSON objects that contains all the generally needed bibliographic meta data. Since the `json` column is a string, minimal processing is needed in the database or PHP interface to return paper data.

8.3.2 Author Tables

The authors table contains the name of an author. Originally, this table contained additional information including affiliation and address, but it was too difficult to disambiguate them and was unfortunately deemed out of the scope of this project. As with the papers table, a full-text index was applied to the `name` column to facilitate searching.

An author can write one or more papers. This relationship is maintained in the author_rel table that links an author's GUID, `aguid` with the guid of a paper, `pguid.` As with the reference_rel table, those columns are indexed.

There is also the authorcachefull table containing the JSON objects for each of the authors. The name of the table implies that it shows the full author meta data. The author JSON tends to be larger than the paper JSON because it shows collaborators as well as author citation and reference relationships. These relationships were computed in batch using the papers that the authors wrote.

Chapter 9

Client Architecture

This chapter focuses on the client-side architecture of PaperCube. The most complex component of the application is the JavaScript-based thick-client application that runs in the web browser. The cloud or thick-client architecture shifts the majority of the application logic to the web browser and simplifies the server. In contrast, a traditional web application is architected to contain most of the logic on the server side. This section will cover some major components within the thick-client and discuss its architecture at a relatively low level.

The client is built using the traditional "Model-View-Controller" architectural design pattern (Gamma et al., 1995) which is most often used when developing desktop applications. The basis for this pattern, most commonly referred to as *MVC,* is that business logic is separated from the UI elements which results in an application that is more manageable, modular, and testable. For instance, it should be possible to create all the business logic for an application without writing the UI elements yet still be able to simulate all the actions accessible through the GUI by using unit tests. This is particularly useful when following a test driven development model and quality assurance validation.

Figure 9.1 shows a detailed diagram of the client showing the three different layers in the MVC architecture. The model layer talks directly to the PHP-based database interface using AJAX calls. The controller layer act as an intermediary between the model and view layers. The controller layer determines what data is needed, view should be visible, and its display parameters. The display parameters include the zoom level and the thresholds used to alter the rendering of the visualizations.

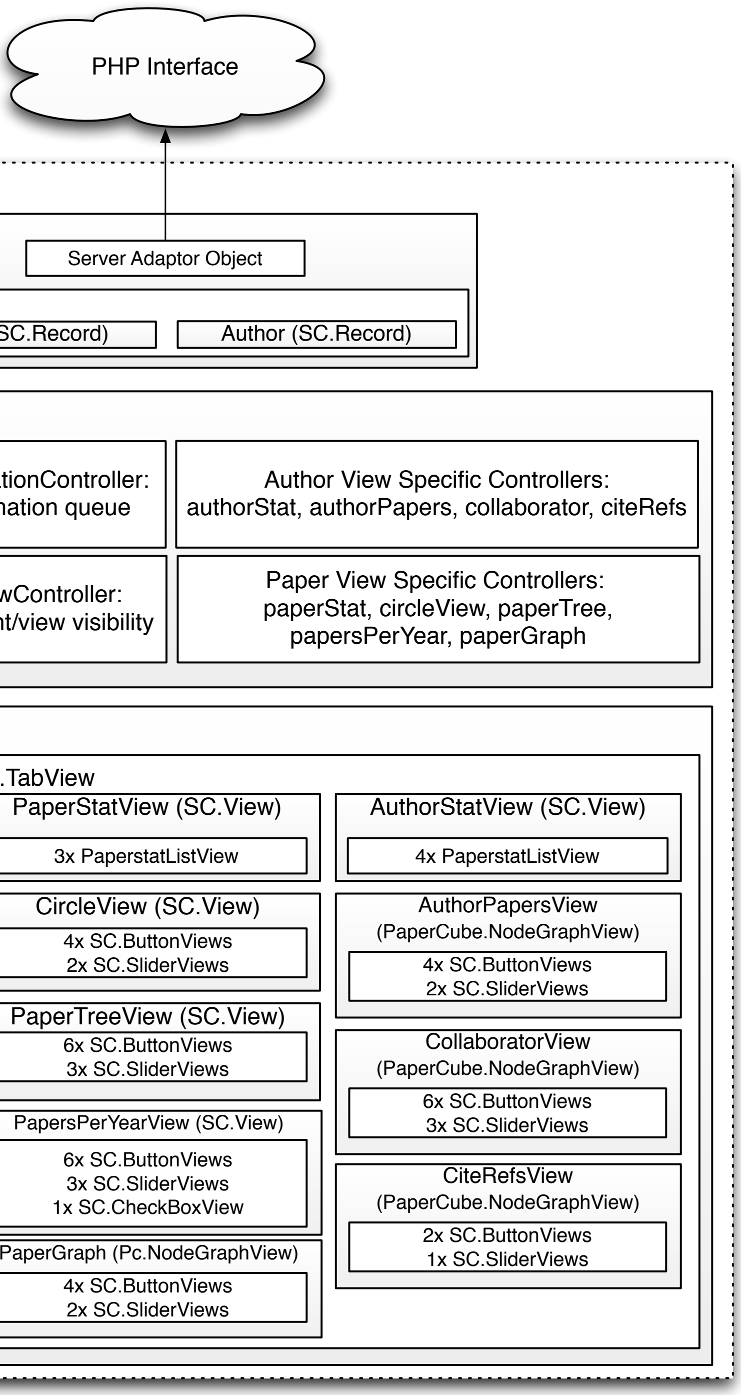

PHP Interface
Web Browser (JavaScript)
Model Layer
Server Adaptor Object
SC.Store
Paper (SC.Record)
Author (SC.Record)
Controller Layer
searchController: searching
animationController: animation queue
Author View Specific Controllers: authorStat, authorPapers, collaborator, citeRefs
canvasController: zoom, positioning
viewController: content/view visibility
Paper View Specific Controllers: paperStat, circleView, paperTree, papersPerYear, paperGraph
View Layer
Toolbar:
SC.ButtonViews/ SC.SliderView
ZoomView (SC.View)
Left View:
SC.FormView
SearchResults (SC.CollectionView)
SavedItems:
SavedItems (SC.CollectionView)
Clear/Close SC.ButtonViews
Fan Menu Container (DOM Element)
SC.TabView
PaperStatView (SC.View)
3x PaperstatListView
AuthorStatView (SC.View)
4x PaperstatListView
CircleView (SC.View)
4x SC.ButtonViews
2x SC.SliderViews
AuthorPapersView (PaperCube.NodeGraphView)
4x SC.ButtonViews
2x SC.SliderViews
PaperTreeView (SC.View)
6x SC.ButtonViews
3x SC.SliderViews
CollaboratorView (PaperCube.NodeGraphView)
6x SC.ButtonViews
3x SC.SliderViews
PapersPerYearView (SC.View)
6x SC.ButtonViews
3x SC.SliderViews
1x SC.CheckBoxView
CiteRefsView (PaperCube.NodeGraphView)
2x SC.ButtonViews
1x SC.SliderViews
PaperGraph (Pc.NodeGraphView)
4x SC.ButtonViews
2x SC.SliderViews

Figure 9.1: PaperCube's thick-client architecture following the MVC pattern.

9.1 Selection of the SproutCore Framework

Initially, the desire was to create PaperCube as a desktop application developed for Mac OS X and developed in Cocoa (http://developer.apple.com/cocoa/), but after some discussion the idea of creating a web-based solution was adopted. Therefore, the challenge was to select the best framework or plug-in to make the development of PaperCube possible. The idea of using proprietary plug-in such as Flash, Flex, or Silverlight was excluded because using standards based web technologies seemed more interesting. Furthermore, web browsers have gotten faster over the past couple of years and therefore, using JavaScript to do more has become a viable alternative to using a plug-in. The SproutCore framework was chosen because it is geared towards the development thick-client applications on the web and uses native JavaScript.

9.1.1 Why SproutCore and Not Other JavaScript Frameworks?

Prototype, Dojo, jQuery, YUI, and others are widely used by major corporations as well as individuals. However, those frameworks aim to augment existing websites and add AJAX functionality. In the general, the functionality that these frameworks provide come in the form of widgets such as form validation, search text autocompletion, basic animation, or convenience functions for manipulating the DOM.

A limited number of JavaScript frameworks exist that specifically target rich internet application development—not websites—such as ExtJS, Cappuccino, and SproutCore. All of these frameworks designed to make it easy for developers to create cloud-based applications that have desktop-class features. The reason why SproutCore stood out among the rest is that compared to ExtJS, which is partially a commercial product, SproutCore is an MIT-licensed open source framework.

Compared to Cappuccino, SproutCore allows developers to directly program in JavaScript, CSS, and HTML. Cappuccino uses a port of Objective-C called Objective-J which requires a compilation step to convert it into JavaScript which can make debugging the compiled code very difficult. SproutCore uses native JavaScript and HTML, yet still gives the developer desktop-class features. Cappuccino's goal is to allow established Objective-C desktop developers bridge the gap to web development without learning a new language. However, it abstracts away the things that the web is good for and replaces it with an unnecessary layer of abstraction.

Finally, SproutCore is a proven framework used in major commercial products including Apple's iWork.com (http://www.iwork.com) and MobileMe (http://www.me.com) services. Another company using SproutCore is Other Inbox (http://www.otherinbox.com).

9.1.2 Features of SproutCore

- **Thick-client in the Browser**

  SproutCore is based on a cloud or thick-client architecture which means that it can run solely in the browser on its own since it consists of static HTML, CSS, and JavaScript that can be loaded without any dynamic server scripting. To get data from the server, the SproutCore application requests it via an AJAX request and any subsequent interaction with the server including reading, creating, updating, or deleting content is handled asynchronously in the background without reloading the page. As the browser sends requests asynchronously, just as a desktop application, a user is free to interact with the application without being blocked and waiting. Also, as an added bonus, since more logic is done on the client, server resources are made available to serve data to additional users.

- **Bindings and Observers**

  As with Cocoa on the desktop, SproutCore uses key-value coding and key-value observing, which allows developers to use bindings and observers to maintain and update the state of an application in real time. The advantage of using a binding is that a property can be set in one place and the changed value will be propagated to all other places in the application where it is bound. Also, observers allow changes to be automatically detected automatically and execute additional code.

- **Advanced Data Model**

  One major advantage of using SproutCore is that it has an advanced data model that allows data to be stored and queried in the web browser's memory. This data can be loaded from the server or created through user-initiated actions in the browser itself. The data can be loaded from the server as JSON data objects and automatically inserted into the SproutCore data store. Also, it is easy to create, update, and delete data on the client that can be propagated to a persistent data store such as a MySQL database. Furthermore, more complex properties can be computed in the model as functions that depend on the data loaded from the server.

- **Fully-Developed Set of Views**

  Since SproutCore is a fully developed framework for the creation of desktop-like applications, many of the standard views that are available on the desktop are available on the web. SproutCore has a rich set of views that include collections views that make it possible to view lists of thousands of content objects as well as other views including buttons, tabs, segmented toolbars, menus, popups, text fields, sliders, checkboxes, advanced forms, and many more. These views allow quick development of an application using SproutCore so the developer can focus on creating any additional custom views and not worry about creating the basic UI. By using bindings and observers, UI elements can be updated automatically without having to implement complex logic to update the elements manually.

9.2 Model Layer

PaperCube's model layer is quite simple compared to most SproutCore applications because there are only two types of content objects: papers and authors. Furthermore, PaperCube only reads data from the server back end and does to write. Therefore, the needed operations are optimized for the fast reading of data and not necessarily for extensibility. Since PaperCube does not allow the creation, modification, nor deletion of any data stored in the database, a lot of potential complexity was eliminated by the design.

The model layer consists of the `SC.Store` that contains instances of author and papers objects. These authors and papers are class objects extended from the base class `SC.Record`. Each of the two content model object types, `Papercube.Author` and `Papercube.Paper`, contain a set properties that come from the server JSON form and also a set of computed properties are are programmatically calculated on the client. These computed properties are functions that use other properties to return a more complex value. For example, in the case of a contact management application, a computed property could be `fullName` which is based on a `Contact` content object's `firstName` and `lastName` properties. In most cases, the computed properties are only calculated once and then any subsequent access to those properties return a cached value. Table 9.1 describes the paper model object and Table 9.2 describes the author model object.

The final component of the model layer is the *adaptor*, `Papercube.Adaptor`, which abstracts away the details of server communication from the rest of client. This object serves are the single point of communication with the PHP interface. It receives calls from the controller and view layers and translates them to POST or GET requests that are sent asynchronously to the web server.

`Papercube.Paper` **Model Object**	
Static Properties	
Name	**Description**
guid	Globally unique identifier for the paper based on its CiteSeer ID.
type	The type of the model object, in this case it is "Paper."
abstract	Abstract for paper up to 400 characters.
authors	Array containing both the GUID and name of the paper's authors in the form [[guid, name], ...]
citations	Array containing the GUIDs of the paper's citations in the form [guid, guid,..., guid].
fixedyear	Signifies if the year returned is confirmed or not.
format	The format type of the source.
references	Array containing the GUIDs of the paper's references in the form [guid, guid,..., guid].
source	The URI pointing to the original paper. Most commonly in PostScript or PDF format.
title	Title of the paper.
year	Confirmed publication date or date added to CiteSeer.
Computed Properties	
Name	**Description**
authorCount	The number of authors for quick access.
citeCount	The number of citations for quick access.
formattedCitationCount	Human readable string for citation count.
formattedReferenceCount	Human readable string for reference count.
label	Disambiguation of title/name when reading either a Paper or Author.
refCount	The number of references for quick access.
url	Based on the GUID, create the URI to access CiteSeer.

Table 9.1: `Papercube.Paper` Model Object.

`Papercube.Author` **Model Object**	
Static Properties	
Name	**Description**
guid	Globally unique identifier for the author.
type	The type of the model object, in this case it is "Author."
citeAuthors	Array containing the GUID and the number of times the author has been cited by the listed authors in the form [[guid, numCites], ..., [guid, numCites]]
collaborators	Array containing the GUID and the number of collaborations for all the author's collaborators in the form [[guid, numCollabs], ..., [guid, numCollabs]]
refAuthors	Array containing the GUID and the number of times the author has referenced the listed authors in the form [[guid, numRefs], ..., [guid, numRefs]]
name	The full name of the author.
papers	Array containing the GUIDs for all the papers written by the author in the form [guid, ..., guid]
Computed Properties	
Name	**Description**
label	Disambiguation of title/name when reading either a Paper or Author.
paperCount	The number of papers the author has written for quick access.

Table 9.2: `Papercube.Author` Model Object.

Once a request is fulfilled, the adaptor object will take the JSON response, using `eval()`, turn it into JavaScript, and then insert the author or paper data into the `SC.Store`. After that operation is completed, the adaptor will execute a *callBack* function passed in for the particular request and notify the requesting component view or controller that the data is available.

9.3 Controller Layer

The client application contains a set of controllers that perform different roles. Every view has an associated controller that bridges between the data, the visualization, and associated buttons and slider controls. The controllers enable the manipulation of the display thresholds through the use of bindings and observers adjust the amount of information rendered without unnecessary code overhead. In addition, there are application-wide controllers that manage animation, searching, view visibility, and view positioning.

9.3.1 View-Specific Controllers

Each author and paper view has an associated controller. In all views except for Paper Detail and Author Detail, at least one or more threshold parameters can be adjusted that will alter the amount of information rendered. In PaperCube, each of the thresholds are represented by a slider control and two buttons that increment or decrement the slider's value. These elements are implemented using `SC.ButtonView`, `SC.SliderView`, and `SC.CheckBoxView.` Each of these parameters have a default value, maximum value, minimum value, current value, and increment value. These values are bound to the UI elements and can be changed either through them or programmatically. In either case, the bindings keep everything synchronized in real time. Since these elements are bound to values in the controllers, minimal work needs to be done when implementing these controls and the result and impact to PaperCube's usability is very significant. The reasons is that once the bindings change in the controllers and get propagated to the views, the changes are automatically detected through observers that trigger the view to redraw.

Also, Paper Detail view and Author Detail view have drop down menus with different sorting choices for each list. Again, these values are bindings that the views observe and update the UI elements accordingly when the options are changed.

9.3.2 Animation Controller

Some of the views require animated transitions. To easily manage animations throughout PaperCube, the `Papercube.animationController` was created. The controller object contains a queue of DOM elements that need to be animated. Any view in PaperCube, if it supports animation, can add elements to the animation queue to animate the element from one point to another. The controller allows the view to specify any number of attributes such as height, width, x position, y position, and more. In addition to providing the start and end point, the view also specifies the start time and duration of the animation for each element.

Internally to the controller, there exists a heartbeat function executes every forty milliseconds that checks if any pending animations are present in the scheduling queue. If the queue contains scheduled animations, a loop will traverse the queue and execute one frame per animation during one heartbeat. Once an animation is complete and the DOM element is positioned at the desired end point, it will be removed from the queue. If a large number of elements are scheduled to be animated, the loop may take longer to execute. In that case, instead of making all the animations take longer to finish, the animator will drop frames in order to ensure that of the animated subjects will complete their animations within the desired time. The position of an element is based on the start time, duration, and current time.

For example, if an element is being animated from x-position sx pixels to x-position ex pixels with the desired duration dt and start time st and current time ct, the current position cx pixels can be determined at any arbitrary point in time using the following equation:

$$cx = \lfloor sx + (ex - sx) * (ct - st)/dt \rfloor$$

In Code Snippet 9.1, a SVG `circle` element, `svgNode` is being animated diagonally left and down across the screen in 250 milliseconds.

```
var subject = {
  elm: svgNode,
  props:
  {
    cx: {start: 100, end: 400},
    cy: {start: 200, end: 500}
  },
  duration: 250,
  isSVG: YES
};

Papercube.animationController.addSubject(subject, svgParentNode);
```

Code Snippet 9.1: Same code showing how to queue a subject DOM node to be animated.

9.3.3 Search Controller

The `Papercube.searchController` object controls the elements in the search pane on the left-hand side of the screen. There are two major components in the search pane that are linked to the controller, the search form fields and the search results. The form elements are bound to values in the controller and when the form is submitted, the controller will look at the entered search parameters and relay the query to the web server through the adaptor object. Once the search results are returned from the server, the controller will set the data returned to the `SC.CollectionView` that renders the results in the form of a list.

The search controller also allows for paging to get additional search results that were not returned in the currently visible window that the list is showing. When the search controller detects that the viewing mode has changed from authors to papers or vice-versa, the controller will automatically resubmit the last search to get relevant results for the new viewing mode.

9.3.4 View Controller

The `PaperCube.viewController` object manages the browsing history, viewing mode, view choice, view direction, saved items, and content that is currently being viewed. The drop down menus on the toolbar are bound to the controller and depending on the selected choice, will notify any of the listening views if it is changed. Also, this controller manages the currently viewed content, which is either an author or paper object.

The set of views in PaperCube are managed in this controller. Each of the views is contained within one `SC.TabView` and there is a property called `nowShowing` is monitored by the `SC.TabView` which signifies which view should be visible. When this property changes, the `SC.TabView` will determine if the view has been used before. If view has not been initialized, it will lazily be created by the `SC.TabView`.

9.3.5 Canvas Controller

The `Papercube.canvasController` manages the viewport and zooming logic. It is used to relay positioning information between the zooming widget and the views. When the dimension values change, the various views are notified through observers that they need to redraw if they are visible. In addition to the current zoom factor, the controller also manages the width and height of the area where the view is rendered, or in other words, the *canvas.* Since a view can be zoomed in using various means, more information than just the width and height need to be managed. The window is treated as a viewport that may only show a part of the total visible area of the canvas. Therefore, the controller has a portal height and width, portal x-position and y-position offsets, canvas height and width, and zoom value. As a view is zoomed in, the canvas dimensions increase but the portal dimensions remain constant. The offset x and y positions are used to determine where to place the top left corner of the portal. In the case of HTML-based visualizations, the dimensions are applied as CSS properties on the elements. The offset positioning is accomplished by using negative top and left absolute positions of the canvas element. Figure 9.2 shows description of how the different dimension properties are related.

In the case of SVG-based views, the calculations are slightly different. Since SVG is vector-based, scaling is not an issue. Therefore, the canvas is actually rendered at the highest possible zoom value. For example, if a view that allows for 10x zoom, the canvas is rendered ten times the size of the portal and then scaled to 1/10th of the size if the currently selected zoom value is 1x. By using the `transform` attribute on the root SVG group element that contains the visualization's DOM nodes,

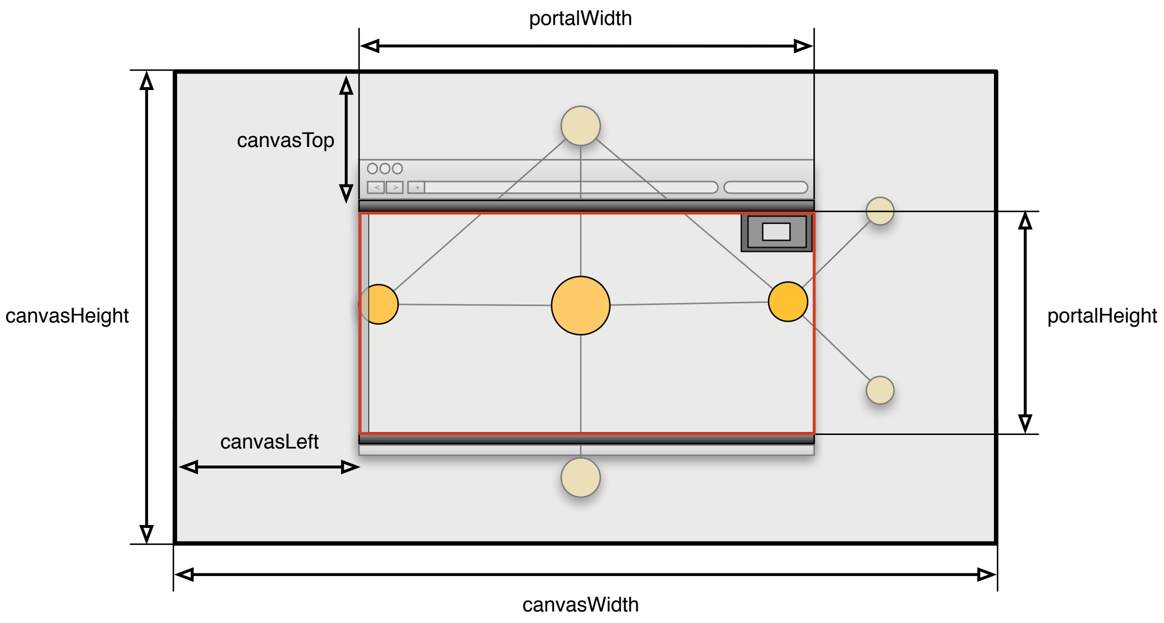


Figure 9.2: The canvas and the preview inside of the browser.

the calculation in JavaScript is performed in Code Snippet 9.2. The zoom value from the controller is inverted and saved as `zI`. In the calculation, the `scale` command component of the transform is set to `zI` then then it repositioned using the `translate` command by calculating the offset amount by taking the width and height of the canvas and dividing it by `zI`.

```
this.svgGroup.setAttributeNS(null, 'transform',
    'scale('+zI+') translate('+ (x/zI) +',' + (y/zI)+')'
);
```

Code Snippet 9.2: The code for applying the SVG `transform` attribute to scale the visualizations.

The controller also manages the fan menus used by the different views. When a view is initialized, it can register a set of menus with a set of actions that should be activated depending on what item has been clicked in the visualization. The menu is SVG-based and it is dynamically rendered within the controller and appended to the DOM in a container element. When a view tells the controller that it should display a specific menu, the controller will check if the menu has been previously used. If it has, it will be reused. Otherwise, if it has never been used before, the controller creates the menu on demand. This on the fly generation is done so that only when the menu is used, it adds weight to client.

9.4 View Layer

The view layer contains the various UI elements with which a user directly interacts. There are several major components of the view layer: the toolbar, the search pane, saved items pane, and the views themselves that are located inside a `SC.TabView`. Since they have been covered on the feature level in the previous chapter, and the previous sections in this chapter has covered some view components, this section will only discuss additional technical details of the implementation.

9.4.1 Rendering Scalability and Performance

Although the scalability and performance characteristics of the client vary quite a bit between web browsers, general steps were taken to make PaperCube as fast as possible across all browsers. According to general JavaScript benchmarking, the fastest browser is Google Chrome followed closely by Safari 4 Beta. Then, in descending order, FireFox 3.1, Firefox 3.0.6, Safari 3.2.2, and then Internet Explorer 8 far in the rear (Keizer, 2009). Since Chrome and Safari 4 Beta, are almost four times faster then Firefox 3.06 and Safari 3.2.2 and six times faster than Internet Explorer 8, optimizing JavaScript execution was key.

- **Optimized Data Fetching**

  One general strategy taken in PaperCube was to only load as much data from the server as was needed. Evaluating JSON and inserting data into the `SC.Store` are two very expensive operations in terms of JavaScript execution and memory overhead. Instead, a progressive loading scheme was developed so that even though not all the data shows up at once, the wait time to see data rendered was shorter. The views are structured in such a way that when rendering, the view explores the focused paper's citation network or focused author's network of authors to see what information has been already retrieved. Since the set of views are different visualizations on the same information, one common case is to have overlap in the needed data.

  Before a rendering pass, the views perform a recursive search of the relations up to the desired display threshold parameters. If an item met the threshold criteria, such as its depth, and it was not found in the data store, its GUID would be added to a queue for retrieval. Once the recursive search was done, the view would render the existing data while waiting for the requested data to return from the server. This way, the views can progressively load and give faster feedback instead of waiting for the entire set of meta data to the retrieved at once. Since the data retrieved is available in the data store, another view can take advantage of the already loaded items and thus only load any additional items as they are needed. If two views show exactly the same data, there would be no call to the server.

There was a case in Papers Per Year view where the depth of the data retrieval was set to infinity and close to 40,000 papers were retrieved. Although the view was still usable since it did not render all of the bibliographic meta data on the screen, the memory usage spiked so high that the machine ran out of memory. This was due to the fact that there were 40,000 `Papercube.Paper` record instances loaded in memory and the rendering algorithm was traversing the citation network to find new items to fetch for the next rendering pass. Therefore, as a performance compromise, the maximum depth is fifteen levels with a default of eight levels to prevent potential performance issues. More or less bibliographic meta data can be requested by changing the desired depth on a per view basis.

- **DOM Caching and Smart Rendering**

  The most expensive operation that one can do in JavaScript is to manipulate the DOM. When the DOM is touched in any way, the web browser has to do a lot of work in the background. Just setting an attribute on a DOM element that was already set to the same value is expensive enough to slow things down significantly. Since the visualizations in PaperCube contain many DOM elements that are positioned individually, serious performance issues can occur due to wasteful DOM operations. Therefore, all of the views maintain state regarding the dimensions, positioning, color, and more for each individually rendered item. During a rendering pass for a visualization, instead of operating directly on the DOM elements in real time, the rendering methods calculate all of the information for every item that can be rendered and check against the cached values from the last rendering pass. If the values differ, the item and its new properties are added to a queue for processing. Depending on the view and the amount of items shown on the screen, each item is either animated to its new position or positioned immediately. Another benefit of caching the values is that only the specific values that did change on an item are changed, if other values were not changed, they will not be updated.

  While caching adds complexity to the JavaScript, this added execution time is far less than the time it would take to update the DOM without any regard for the previous values. Also, the memory overhead of keeping a previously cached version of the attributes for each item is a relatively small overhead compared to the benefits.

- **Abstracting What is Rendered**

  The most data-dense paper views, Tree Map and Papers Per Year, a very large amount of bibliographic meta data can be retrieved and displayed, but performance concerns made it impractical to render the whole data set. Although the views are created in such a way to be resolution independent, certain thresholds make it technically or impractical to render the information. Since the views still have a maximum zoom value, if an item is not visible at the zoom level it is not rendered or its child relations traversed.

  In the case of Tree Map view, if a paper's child relation is less than 1 pixel wide at the current zoom level, it will be abstracted and displayed as a leaf node. However, if the view is zoomed in on that region, the paper's children will be rendered automatically. In Papers Per Year view, the maximum number of papers rendered for a year is 300 and if more exists, the whole year is abstracted away until stringent thresholds reduce the number that should be displayed. Although not directly useful in terms of the visualization's effectiveness, the abstraction prevents the rendering from being slow. Also, if additional bibliographic meta data should be rendered, it is easy to change the display thresholds parameters to show less items and as a result, reveal items in that year.

9.4.2 HTML-based Views

In addition to the relatively basic Paper Detail and Author Detail views, two other views are HTML-based. Circle View and Tree Map use HTML and they have some challenges.

Although Circle View represents all the items as circles, they are in reality `DIV` elements with CSS rounded corners. Since one cannot position them from the center, the positioning is done by looking at the diameter of the circle and subtracting its radius to get the position of the top corner of the square `DIV`. Also, another unique aspect to this view is that it uses the Canvas tag for the lines radiating from the focused paper to the first level references. The reason why the lines were drawn using the Canvas tag is that it is not possible to do so in HTML unless a browser such as WebKit is used that allows for rotation of HTML elements using CSS 3 transforms. If not, the only time the lines could be drawn would be if they are along the x- or y-axis. Although it could be done with SVG, this view was written early enough in development that none of the other SVG-based views had been created and there was no reason to re-write the visualization.

The Tree Map view's visualization is rendered using `DIVs` that contain the subsequent levels of relationships. The relations within a level are rendered using absolute positioning with a CSS `left` pixel offset from the containing `DIV` to position the papers next to each other.

One advantage of having views that are HTML-based is that it is easy to apply CSS styles to the DOM elements. Instead of setting the dimensions of the circle elements in Circle View or the rows of papers in the Tree Map view on a per element basis, PaperCube takes the approach of re-writing the CSS directly inside the style sheet. Therefore, as the browser window is resized or the amount of visible data is changed, the rendering algorithms can change all the DOM elements globally without having to do a traversal of the DOM and write to the elements individually. Although the browser will still have to relayout internally in its rendering engine, the amount of work is far less than doing it manually using JavaScript.

One technical challenge regarding the re-writing style sheet directly is that the SproutCore build tools compile and compress CSS into one single file. In order to address the styles inside of a style sheet, the index of the style needs to be known. Therefore, there was a manual process added to completing a PaperCube application build. The separate source style sheets needed to be used instead of the compiled version to ensure that the views were able to access the correct style sheet.

9.4.3 SVG-based Views

Since Scalable Vector Graphics elements are accessible through the DOM, the rendering strategies used in the HTML-based views can still be employed. However, since one of the primitive elements in SVG is a `circle`, the layout is much simpler because one can specify the element's radius and its x and y position are in the center of the circle. Also, SVG offers many other drawing primitives such as the `line` and `path` elements that only require a start and end point in order to be rendered.

One major advantage of using SVG instead of Canvas is that SVG has native text support. Since all of the visualizations show textual information inline, being able to have them be managed in SVG is very convenient since if the view is scaled, the text is scaled with it automatically. If one wants to add text to a Canvas tag-based visualization, one would either need text sprites, which do not scale infinitely, or have a separate text layer on top of the Canvas rendering, which adds additional complexity.

9.4.4 SVG Inside HTML Documents

Scalable Vector Graphics is a dialect of XML and therefore have many strict requirements that HTML does not have. Within normal HTML documents, SVG is not natively supported without using a namespace. SVG can be contained within a document with the `.svg` or `.xml` file extensions or be embedded into an HTML document using the `embed`, `object`, or `iframe` tags. However, depending on the browsers, these tags may cause issues when making the SVG inside of those tags scriptable. Microsoft Internet Explorer does not natively support SVG rendering but rather allows for inclusion of SVG through an Adobe plug-in. Since there was no desire to tailor PaperCube to work within the confines of a plug-in that may not be standards-compliant, Internet Explorer is not supported.

A better solution that is used in PaperCube is to use "XHTML 1.0 Strict" document type and include both the `XHTML` and `SVG` namespaces as shown in Code Snippet 9.3. This allowed PaperCube to dynamically create SVG DOM nodes using DOM2 JavaScript operations. This combined namespace worked correctly in all the web browsers that support native, standards-compliant SVG rendering.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

 <html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:svg="http://www.w3.org/2000/svg" xml:lang="en">

   <head>
      <meta http-equiv="Content-type" content="text/xhtml; charset=utf-8" />
   </head>

   <body>
  </body>
</html>
```

Code Snippet 9.3: The namespace declarations used to load SVG inside of an XHTML document.

When views that want to use SVG are initialized, the base needed `svg` and `g` elements are programmatically generated using DOM manipulations as shown in Code Snippet 9.4.

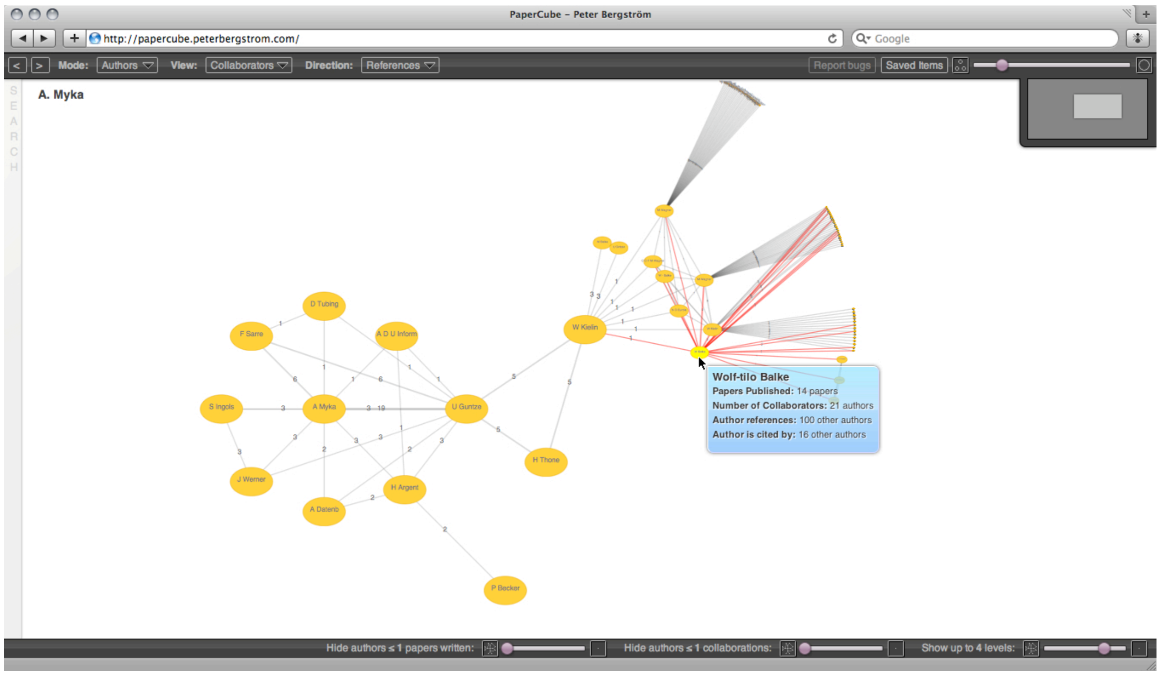
```
init: function() {
  sc_super();

  // Set up svg canvas.
  this.svgCanvas =
       document.createElementNS("http://www.w3.org/2000/svg", 'svg');
  this.rootElement.appendChild(this.svgCanvas);

  this.svgFitGroup =
       document.createElementNS("http://www.w3.org/2000/svg", 'g');
  this.svgCanvas.appendChild(this.svgFitGroup);

  this.svgGroup =
       document.createElementNS("http://www.w3.org/2000/svg", 'g');
  this.svgFitGroup.appendChild(this.svgGroup);

  this.svgEdges =
       document.createElementNS("http://www.w3.org/2000/svg", 'g');
  this.svgGroup.appendChild(this.svgEdges);
}
```

Code Snippet 9.4: The `init` function of an SVG-based view showing DOM manipulations.

9.5 `NodeGraph` class

Stemming from the need to have a graphing component for COLLABORATORS view, the `NodeGraph` class was created as a generalized solution that can be used to show any data with relationships using an undirected graph rendered using Scalable Vector Grpahics leveraging the SproutCore framework. The `NodeGraph` class is not PaperCube-specific and can be easily integrated into other SproutCore applications.

The class is used in PaperCube in four different views: PAPER GRAPH, PAPERS, COLLABORATORS, and AUTHOR CITES. The class is agnostic to the type of SproutCore content object being rendered and is able to render a mixed set of content objects. Furthermore, it is fully customizable. Views that are implemented with this class extend and customize a set methods and properties to keep the base class as generalized as possible. These methods allow the developer to abstract away the details of different content data types and provide a consistent interface to the class. When the graph is altered with the same root node, the graph is designed to smoothly transition using animations.



Wolf-tilo Balke
Papers Published: 14 papers
Number of Collaborators: 21 authors
Author references: 100 other authors
Author is cited by: 16 other authors

Figure 9.3: The `NodeGraph` class used in COLLABORATORS view.

9.5.1 Extendable Methods

Table 9.3 shows a list of all of the methods that can be overridden by the developer in order to customize the `NodeGraph` class to create the desired visualization behavior.

For example, below is a sampling of the custom functions for COLLABORATORS view. The `performCustomRequest` method, shown in Code Snippet 9.5, calls the appropriate method in the adaptor to retrieve the collected GUIDs that are required.

```
/**
  Get the details of a given item.

  @param {Array} guids The array of GUIDs.
  @param {Function} callBack The callBack function  is called when the
                    request is successful.
*/
performCustomRequest: function(guids, callBack) {
  Papercube.adaptor.getAuthorDetails(guids, callBack);
},
```

Code Snippet 9.5: Customized `performCustomRequest` method for COLLABORATORS view.

`NodeGraph` Extendable Methods	
Signature	**Description**
void setDefaultTitle (content)	Set the default title for the view.
mixed findCustomObject (guid)	Generate custom meta data for item.
Array findCustomObjectAttr (object)	Find custom object relation attribute.
String getGuidForRelation (object)	Given relation object, return the GUID for it.
String setBindingDefaults (object)	Revert bindings to default.
String findCustomObjectLabel (object)	Given an object, return its label.
Integer calcRelationWeight (object)	Given relation object, return weight for it.
void performCustomRequest (guids, callBack)	Get the details of a given item. The callBack function is called when the request is successful.
Integer relationMeetsCustomThreshold (rel)	Custom threshold calculation for complex link threshold calculations. Returns YES if the threshold is met.

Table 9.3: Extendable methods in the `NodeGraph` class.

The `NodeGraph` class require the actual content object during some steps in the rendering process. This needs to be implemented on a per content-type basis because there are different types of content. Shown in Code Snippet 9.6, the `findCustomObject` method takes a GUID string and returns the associated item. In this case it will return a `Papercube.Author` content object. If no author is found, `null` will be returned.

```
/**
  Generate custom metadata for item.

  @param {string} guid The GUID for content object that is found in
                  the SC Store.

  @returns {SC.Record} Returns the found content object.
*/
findCustomObject: function(guid) {
  return Papercube.Author.find(guid);
},
```

Code Snippet 9.6: Customized `findCustomObject` method for Collaborators view.

The relations that the graph is based on varies depending on the content type. In this case, the graph is showing collaboration relationships and this relationship is provided as an array of GUIDs. Accessing the `_attributes` array on a content object is not recommended unless the developer knows that it is not a computed property. In the case of Collaborators view, it is not computed and is accessed directly as shown in Code Snippet 9.8. If it is a computed property it does not exist the array and needs to be accessed using the general SproutCore accessor function, `get`.

```
/**
  Find Custom Object Relation Attribute.

  @param {Record} object The content object.

  @returns {Array} Returns the found relation attribute array.
*/
findCustomObjectAttr: function(object) {
  if(object) return object._attributes.collaborators;
},
```
Code Snippet 9.7: Customized `findCustomObjectAttr` method for COLLABORATORS view.

Given a relation, the `NodeGraph` needs to access its GUID. In the case of COLLABORATORS, this is quite simple because the relation object is an array containing the GUID and weight attributes as shown in Code Snippet 9.8. There are other cases where these functions can be quite complex.

```
/**
  Given relation object, return GUID for it.

  @param {Object} rel The relation object or array.

  @returns {String} Returns the GUID for the relation object.
*/
getGuidForRelation: function(rel) {
  if(rel) return rel[0];
},
```
Code Snippet 9.8: Customized `findCustomObjectAttr` method for COLLABORATORS view.

Shown in Code Snippet 9.9, the `calcRelationWeight` method looks at the importance of the relation. In this case, it is the author's number of collaborators which is located at the second index of the array.

```
/**
  Given relation object, return weight for it.

  @param {Object} rel The relation object or array.

  @returns {Integer} Returns the calculated weight for the relation object.
*/
calcRelationWeight: function(rel) {
  return rel[1];
},
```
Code Snippet 9.9: Customized `calcRelationWeight` method for COLLABORATORS view.

The relationMeetsCustomThreshold method, shown in Code Snippet 9.10, takes a relation and calculates if it should or should not be displayed. Most views have a set of sliders that determine the thresholds that an item needs to meet in order to be shown. If more than one threshold is used, a composite truth value needs to be computed. In this case, this method looks at two thresholds, the linkThreshold, which is the number of collaborators and the paperThreshold, which is the number of papers that the author has written. If the author meets both of these thresholds, it is included.

```
/**
  Custom threshold calculation for complex link threshold calculations.

  @param {Object} rel The relation's GUID.

  @returns {String} Returns if the item is accepted
                    by the threshold(s). Returns NO otherwise.
*/
relationMeetsCustomThreshold: function(rel) {
  if(rel[1] >= this._cached_linkThreshold) {
    if(this._cached_paperThreshold != 0) {
      var p = Papercube.Author.find(rel[0]);
      if(p)
        return (p._attributes.papers.length >= this._cached_paperThreshold);
    }
    return YES;
  }
  return NO;
},
```

Code Snippet 9.10: Customized relationMeetsCustomThreshold method for Collaborators view.

Lastly, the `generateCustomMetaData` method, shown in Code Snippet 9.11, needs to be customized in order to show the meta data for an item. In the case of collaborators, the bibliographic meta data is associated with `Papercube.Author` content objects.

```
/**
  Generate custom metadata for item.

  @param {string} guid The GUID for content object to
                  be shown in the meta data view.

  @returns {Boolean} Returns NO if there is an error.
*/
generateCustomMetaData: function(guid) {
  // Get the author content.
  var rec = Papercube.Author.find(guid);

  // Save the view for later.
  var view = this.metaDataView;

  if(!rec) return;

  // Set the meta data attributes.
  view.childNodes[0].innerHTML = rec.get("name");

  view.childNodes[1].innerHTML =
    "<strong>Papers Published:</strong> " +
    Papercube.pluralizeString(" paper", rec.get('paperCount'));

  view.childNodes[2].innerHTML =
    "<strong>Number of Collaborators:</strong> " +
    Papercube.pluralizeString("author",rec.get('collaborators').length);

  view.childNodes[3].innerHTML =
    "<strong>Author references:</strong> " +
     Papercube.pluralizeString(" other author",rec.get('refAuthors').length);

  view.childNodes[4].innerHTML =
    "<strong>Author is cited by:</strong> " +
     Papercube.pluralizeString(" other author",rec.get('citeAuthors').length);
},
```

Code Snippet 9.11: Customized `generateCustomMetaData` method for COLLABORATORS view.

9.5.2 Customizable Properties

Also, a rich set of properties, listed in Table 9.4, are exposed that dictate the rendering and what parts of the graph are visible. The developer can customize the colors, sizing, labeling, and meta data options. One can turn off edge rendering or show edges without weight calculations. Since the `NodeGraph` class can display any type of content, basic information about needs to be provided such as the attribute to use for the title and the human-readable name of the content type.

`NodeGraph` **Customizable Properties**	
Name	**Description**
nodeColor	Node background color. Default '#FFCB2F'
nodeColorSel	Selected node background color. Default 'yellow'
nodeBorderColor	Node border color. Default '#EAA400'
nodeBorderColorSel	Selected node border color. Default '#FFB60B'
nodeTextColor	Node text color. Default '#666'
nodeDefaultRadius	The node size ratio from the size of the screen. Default is 25 times smaller than height.
nodeBorderWidth	The default width of the node border. Default is 1 unit.
nodeXYRatio	Node radius x-y ratio. The default is x radius is 1.5 times y.
edgeColor	Edge color. Default is '#333'
edgeColorSel	Selected edge color. Default is 'red'
edgeTextColor	Edge text color. Default is '#666'
edgeMinWidth	The minimum width of an edge. Default is 1 unit
nodeTextRatio	The nodeTextRatio allows the font size for the node to be calculated. The font size is calculated as radius/nodeTextRatio. Default is 2 units
edgeTextPosOffset	The position offset for the edge label. A value of 0.1 would puts the edge label close to start node. 0.5 would put it in the middle of the edge. 0.9 would put it close to the end node. Default is 0.3
nodeOpacity	Node opacity. Default is 1
edgeOpacity	Edge opacity. Default is 0.2
edgeOpacitySel	Selected edge opacity. Default is 0.5
showEdgeLabel	If set to YES show the edge label, if NO, hide the label. Default is YES.
useEdgeWeightWidth	If set to YES calculate the edge width by looking at the weight of the item, otherwise, skip this operation. Default is YES.
showEdges	If NO, do not show edges. Default is YES.
defaultTitleKey	The key in the content object for the default title display. Default is 'title'
contentTypeViewing	The name of the content type being displayed. Default is 'Paper'
viewName	The name of the view. Default is 'none'
metaDataBoxHeightSmall	Meta data box small height. Default is 120 pixels
metaDataBoxWidthSmall	Meta data box small width. Default is 400 pixels
metaDataClassName	*Class name for meta data DIV. Default is ''*

Table 9.4: Customizable `NodeGraph` class properties.

9.5.3 Rendering Algorithm

The NodeGraph's layout algorithm positions nodes around the graph's root node in a circular layout and uses several different parameters to determine the child node positions. Several passes are made on the data to collect information needed during rendering. First, a recursive method visits all the nodes and determines if they meet the criteria to be visible and collects any additional GUIDs that need to be retrieved from the server. Second, another pass traverses all the visited nodes and positions them. Once they have been positioned, they are rendered on the screen.

The default node size and offset are calculated and saved as global variables based on the first recursive traversal of the data. The offset is the default distance between the root and its children. Inside the NodeGraph _render method, after determining the deepest visited node in the root node's network of relations, the default radius and offset can be calculated as shown in Code Snippet 9.12. The NODEGRAPH_DEFAULT_SCALE is the deepest zoom level that is supported which is defaulted to 10. In the case of PaperCube, the _h and _z properties are set from the canvasController's display properties. The nodeDefaultRadius property is determined by the developer since it is a property that can be overridden.

```
this._radius =
   this._h/this.nodeDefaultRadius/deepest*NODEGRAPH_DEFAULT_SCALE/this._z;

this._offset =
   (this._h/2-(this._h/2*.2))/deepest*NODEGRAPH_DEFAULT_SCALE/this._z;
```

Code Snippet 9.12: The code for determining the radius and offset values used in rendering the graph.

After the defaults are determined, the nodes are positioned based on these properties during the second pass that traverses all the visited nodes from the first pass. The child nodes for a given node have their radius calculated and individually positioned at the same time. By looking at all the visible child nodes, the radii for all the nodes are determined together. The more child nodes, the smaller they will appear on the screen.

For the child nodes, the radius is determined by the formula below where r is the radius of the circle, o is the global offset distance for the view, and c is the number of children of its parent. The mathematical formula translates to the code in Code Snippet 9.13.

$$r = (2 * \pi * o) * (d/360)/c/4$$

```
var radius =
      Math.min(this._radius,(2*Math.PI*offset)*(delta/360)/childCount/4);
```

Code Snippet 9.13: The radius calculation for the child nodes.

During the rendering pass, the angle of the node is passed in and its children can take up to the smallest value of 90 degrees or 1.5 times the angle of the node's parent. This is based on the fact that if the root node has a lot of children, the alloted space for the node is limited and the layout algorithm tries to reduce the possibility of overlapping nodes. No matter which angle is determined, the angle is bisected so that the nodes will fan out around the specified angle.

Since the position of the parent node is known, the position of its child nodes can be easily calculated using basic geometry. A local offset to the parent node is calculated for each child node. The more children a node has, the closer it is positioned to its parent node. The formula below determines the local offset, lo where o is the global offset distance, c is the number of children of the node, and m is the maximum number children for all the child nodes of the parent node.

$$lo = o * (1 - c/m)$$

Then, to calculate the local x and y positions for each node, a loop starts with the start angle for the children for a node. Then by looking at the number of children that should be rendered, the local angle, t, for each node can be determined. Using the local offset, the parent nodes position, x and y, the node's position, lx and ly can be determined.

$$lx = x + lo * \cos t$$

$$ly = y + lo * \sin t$$

In code, the formulas above translate into the code shown in Code Snippet 9.14. Please note that some additional constants listed are shown but they are always set and do not impact the formula. Note that the `delta` and `theta` variables are passed from the parent's executions of the recursive method. Also, additional calculations allow for animation of newly created nodes. The animation transitions the node from its parent's position to where it should end up.

```
var dl = (level == 0) ? delta/childCount : delta/(childCount-1);
var th = (level == 0 || childCount == 1) ? theta : theta-delta/2;

// For each child, calculate the position information.
for(var i=0; i<childCount; i++)
{
  pos[i] = [];
  var g = rels[i][0];
  var visitLvl = this._visited[g].level;
  if(visitLvl == nextLvl && !this._positioned[g])
  {
    var lOffset = (offset*NODEGRAPH_OFFSET_PERCENT);

     var o = (visitLvl != deepest) ?
          lOffset*(1-this._visited[g].childCount/maxChildren) : 0;

    var locOffset = (NODEGRAPH_USE_CHILDREN_OFFSET) ? (offset-o) : offset;

    var angle = th*PI180;
    var lx = x + locOffset*Math.cos(angle);
    var ly = y + locOffset*Math.sin(angle);

    if(lx < this._lBound) this._lBound = lx;
    if(lx > this._rBound) this._rBound = lx;
    if(ly < this._tBound) this._tBound = ly;
    if(ly > this._bBound) this._bBound = ly;

    this._nodes[g] = {guid: g, level: visitLvl, ex: lx,
                      ey: ly, sx: x, sy: y, radius: radius};

    this._positioned[g] = YES;
    pos[i] = [lx, ly];
    th+=dl;
  }
}
```

Code Snippet 9.14: Calculating the position for each child node of a node.

Although the rendering algorithm requires several passes to determine out what should be rendered, it is efficient because it first narrows down the data that should be displayed and then works on those nodes exclusively. Also, the actual positioning component of the algorithm is only done on nodes that need to be repositioned and not on all nodes that may be rendered on screen. The actual rendering is done in a linear fashion and only on the nodes and edges that require layout.

9.5.4 Animated Transitions

When the state of a `NodeGraph`-based visualization changes based on display thresholds, the rendering algorithm looks at the nodes that should remain visible. From the set of visible nodes, the rendering method calculates the new x- and y- positions and radius for the nodes and then adds the nodes to be animated from their previous position. In the case of PaperCube, the animator used is the `Papercube.animationController`. If the nodes change state from being visible to being hidden, they are not animated out, but rather hidden immediately. In the case of adding a set of new nodes, they are animated from their parent to their final positions.

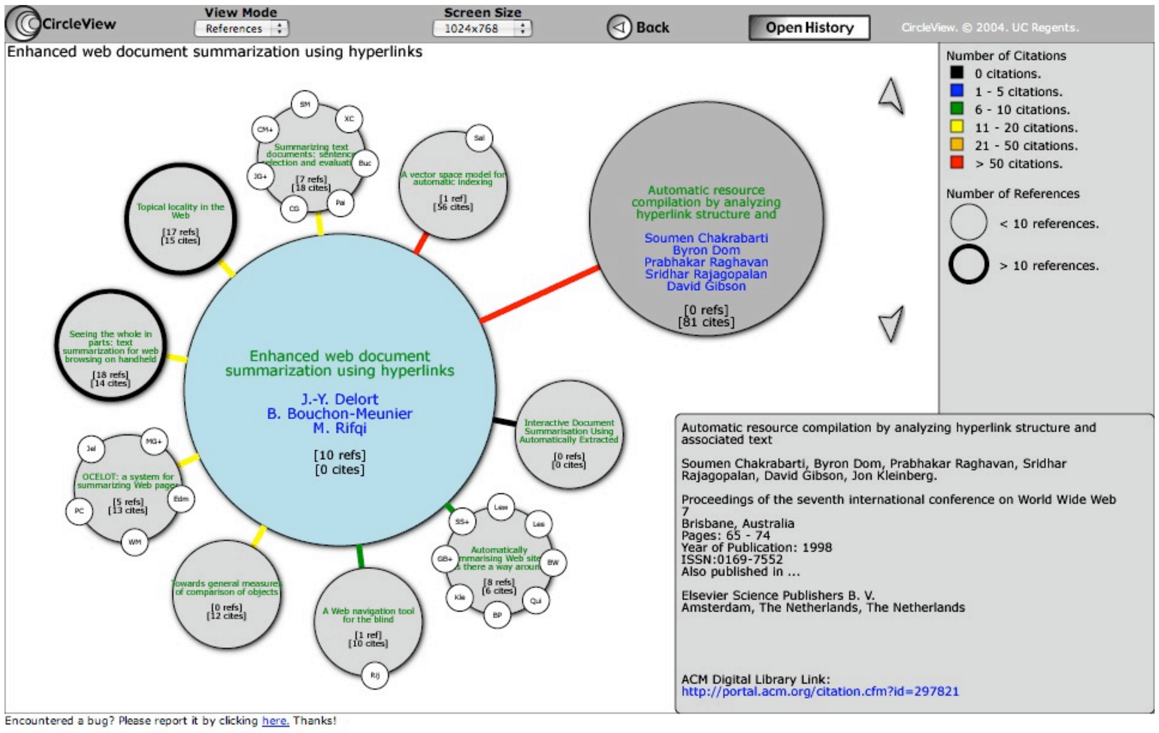
Chapter 10

Evaluation of Circle View Through the Eyes of Cognitive Psychology

This chapter is a cross-disciplinary study of the original implementation of Circle View from 2004. The goal is to improve the effectiveness of the visualization for its use as one of the views in PaperCube by evaluating the UI and visualization method in terms of cognitive psychology. More specifically, the use of color coding to show the number of citations was not as good as originally thought, and the use of the feature-present/feature-absent effect to give certain important cues is shown to be a good design choice. Furthermore, this chapter discusses the three important areas that impact UI design: resource allocation during visual search, icon concreteness and complexity, and the importance of experimentation while designing user interfaces.

10.1 Color Coding

Referring to Figure 10.1 presented earlier, a typical screenshot of Circle View in action, there are ten references to the paper "Enhanced Web Document Summarization Using Hyperlinks." The references are arranged not by the number of references the paper has, but rather by its number of citations. The number of citations can be construed to be a measure of importance. Beyond just basic render order, a clearly defined color scale shows the number of citations that a paper has received. Zero citations are shown as black; one to five citations are shown as blue; six to ten citations are shown as green; eleven to twenty citations are shown as yellow; twenty-one to fifty citations are shown as orange; and beyond fifty citations are shown as red. As one may gather from the colors, the number of citations are shown in a temperature scale. The "hotter" the color of the line to the paper, the more citations.


CircleView
View Mode
References
Screen Size
1024x768
Back
Open History
CircleView. © 2004. UC Regents.
Enhanced web document summarization using hyperlinks
Enhanced web document summarization using hyperlinks
J.-Y. Delort
B. Bouchon-Meunier
M. Rifqi
[10 refs]
[0 cites]
Automatic resource compilation by analyzing hyperlink structure and
Soumen Chakrabarti
Byron Dom
Prabhakar Raghavan
Sridhar Rajagopalan
David Gibson
[0 refs]
[81 cites]
Summarizing text documents: sentence selection and evaluation
[7 refs]
[18 cites]
A vector space model for automatic indexing
[1 ref]
[56 cites]
Topical locality in the Web
[17 refs]
[15 cites]
Seeing the whole in parts: text summarization for web browsing on handheld
[18 refs]
[14 cites]
OCELOT: a system for summarizing Web pages
[5 refs]
[13 cites]
Towards general measures of comparison of objects
[0 refs]
[12 cites]
A Web navigation tool for the blind
[1 ref]
[10 cites]
Automatically summarising Web sites is there a way around
[8 refs]
[6 cites]
Interactive Document Summarisation Using Automatically Extracted
[0 refs]
[0 cites]
SM
XC
CM+
Buc
JG+
CG
Pal
Sal
Jel
MG+
Edm
PC
WM
Rij
Lew
Lee
SS+
GB+
BW
Kle
Qui
BP
Number of Citations
0 citations.
1 - 5 citations.
6 - 10 citations.
11 - 20 citations.
21 - 50 citations.
> 50 citations.
Number of References
< 10 references.
> 10 references.
Encountered a bug? Please report it by clicking here. Thanks!

Automatic resource compilation by analyzing hyperlink structure and associated text

Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, Jon Kleinberg.

Proceedings of the seventh international conference on World Wide Web
7
Brisbane, Australia
Pages: 65 - 74
Year of Publication: 1998
ISSN:0169-7552
Also published in ...

Elsevier Science Publishers B. V.
Amsterdam, The Netherlands, The Netherlands

ACM Digital Library Link:
http://portal.acm.org/citation.cfm?id=297821

Figure 10.1: Screenshot of the original Circle View application from 2004.

10.2 Color Coding Quantity

Spence, Kutlesa, and Rose (1999) tested the effectiveness of various color schemes to code quantity in spatial displays. Several color schemes were considered. A fixed color of varying brightness (brightness), a scheme that varied saturation and brightness, but with a fixed color (H+S+B), a bipolar scale that had two complementary colors and a neutral gray center (bipolar), and a scale of varying color with constant saturation and brightness (hue-only). The authors conducted two experiments to test the effectiveness of these color schemes.

The goal of the first experiment was to evaluate performance on a simple task, a pairwise comparison of quantitative magnitudes using the four color coding schemes. The participants were undergraduate students and as an incentive to be quick, yet accurate, the top performers were rewarded for their efforts with a small amount of money. The students viewed various 3-dimensional graphs from the z-axis as to make them appear 2-dimensional and were instructed that the colors indicated height. With two crosshairs on two colors of the graph, the students indicated which color was the higher region. The results were measured across two dimensions: response time and accuracy. The response times showed that on average, the brightness coding scheme was the fastest at 1.31 seconds with H+S+B, bipolar, and hue-only with 1.44, 1.66, and 1.93 seconds, respectively.

The students were shown to be quite accurate in completing the task. The most accurate responses were made with H+S+B and brightness, with 0.7 and 0.78 errors, respectively. Most interestingly, the bipolar scheme had 1.59 errors and the hue-only scheme had 1.96 errors, which was significantly higher. The experimenters did hypothesize that the brightness and H+S+B schemes would perform the best but this experiment showed that not only are they more accurate, but also much faster. Speed is very important when viewing displays with a lot of quantitative information.

The second experiment was designed to evaluate performance on a complex task, the location and selection of a high or low region using the aforementioned color schemes used in the first experiment. Again, the participants were undergraduate students who were given a cash incentive to be fast and accurate. The students saw the same graphs as they did in the first experiment, but instead of being allowed to select the perceived highest point from two areas of the graph, they had to manually position a cursor over the color they thought was the high and low points of the image. The results were measured the same way as the first experiment: through response time and accuracy. The H+S+B scheme produced the fastest response time at 3.60 seconds followed by bipolar, brightness, and hue-only with 3.83, 3.87, and 4.04 seconds, respectively. The measure of accuracy was measured in percentiles where 100 percent is the most optimal choice. The hue-only produced the lowest percentile at 92 percent. The brightness scheme came in at 94 percent and H+S+B and bipolar at 95 percent.

The results of Spence, Kutlesa, and Rose's two experiments were in line with the their hypothesis that performance varies with perceptual linearity. Brightness and H+S+B color scales offer better performance in both speed and accuracy. The linear color scales were much better than the non-linear bipolar and hue-only scale. Therefore, if one wants to present quantitative data effectively, using a linear scale that varies brightness only or brightness and saturation is the best choice.

10.3 Color Coding in Two Dimensions

Jameson, Kaiwi, and Bamber (2001) investigated if independent dimensions of brightness and hue can be used together to code information effectively. Their hypothesis was that using two dimensions of color would be more effective than relying on only one dimension. The experimenters tested two groups of participants: Department of Defense personnel with experience in reading sonar displays and undergraduate students. Four experiments were set up to determine how the information displayed was perceived. A computer display was used in the experiment that showed vertical beam images analogous to sonar images.

The first experiment established base-line performance of signals on a monochrome display that shows only varying brightness using the DOD personnel only. The result was consistent with previous studies of static beam detection.

The purpose of the second experiment was to quantify detection performance of a brightness display by introducing random color noise into the display. This experiment used both groups of participants. The results of this experiment showed that for the DOD participants, there was no statistical difference between experiments one and two. However, there was a statistical difference between experiments one and two for the novice group. This, on the other hand, was attributed to the inexperience of the novice group.

The third experiment's purpose was to assess performance on the vertical beam detection when a red hue was used to signify a dangerous signal and a green hue was used to signify a safe signal while showing these colors with random brightness noise. The results showed that participants were able to use a meaningful color code to categorize signals. Again, the novice participants were not as good as the DOD personnel.

The fourth and final experiment's purpose was to see the effectiveness of using both brightness and hue variation together to display meaningful information. The result of this experiment was surprising; both groups significantly performed the task better. One reason that the performance may have been better is that participants may have been able to detect contrasts in the brightness-hue combination that differed less than the smallest detectable value when using a display that shows only brightness or color differences. By using two dimensions, humans can distinguish smaller differences which can be very useful in developing complex user interfaces.

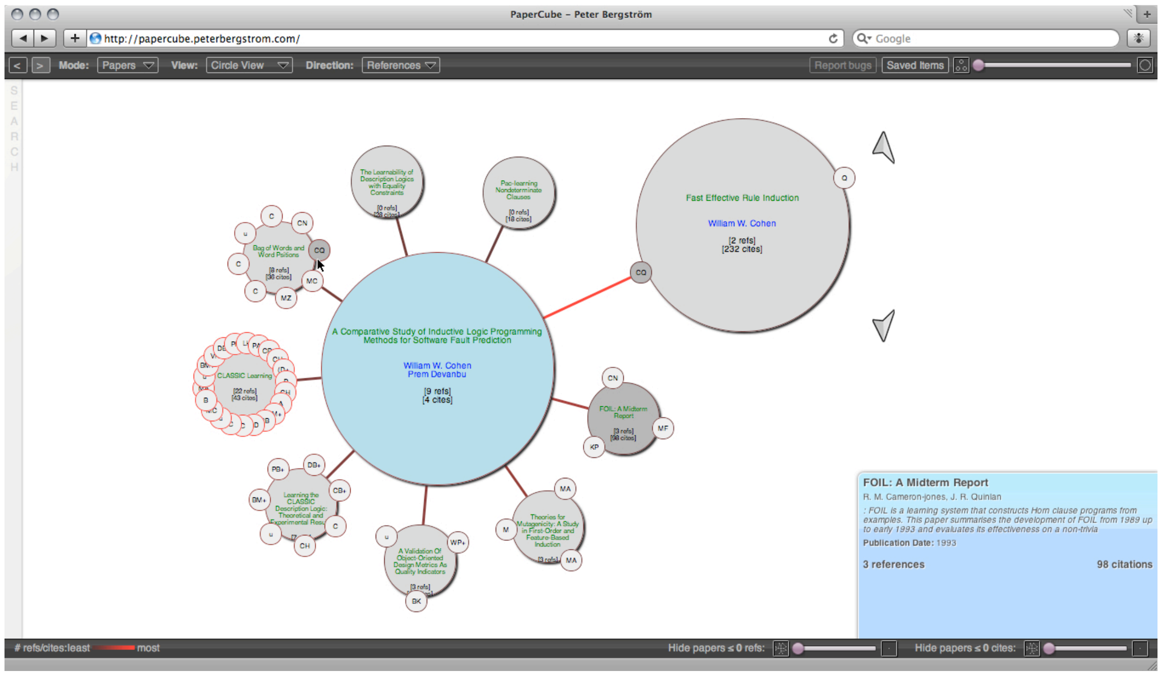
Circle View uses a visually non-linear color scale with no brightness variation, the very thing that Spence, Kutlesa, and Rose found to be the least effective. A temperature scale may seem like a good way to show quantitative information since it conceptually goes from hot to cold. However, the need for a legend on the right hand side of Circle View's UI may hint at unnecessary complexity. By changing the color scheme to use a single color with varying brightness or saturation, it may be a much more effective in showing to show the importance of a paper. Also, using such a scale would allow for a much higher resolution citation count since it is a linear scale. Just changing brightness of one color allows for a much wider spectrum than the six distinct colors used in the current temperature scale. Also, perhaps by using dimensions of brightness and saturation, one may be able to show additional information such as reference counts.

10.4 Visual Search

Circle View set out to show multiple instances of a referenced paper in a different way that most previous visualizations. More often than not, if something has multiple connections within a graph, there is one node and multiple edges or connections going to that node. These lines would cause a visual data structure to become increasingly confusing after only several of those connections. Instead, Circle View chose to display the nodes more than once to eliminate lines intersecting each other over the visualization. This also allowed for a very neat use of the feature-present/feature-absent effect described first by Treisman and Gelade's (1980) pivotal paper on the feature integration theory of attention (FIT). The feature integration theory of attention states as a first step to visual processing, a number of visual features are processed into feature maps that are then integrated into a saliency map which encodes for conspicuous features. From this, feature maps can be accessed to direct attention to areas that stand out from the rest. Matlin (2005) states that visual search is rapid when we are looking for features that are present since they seem to visually "pop out." This was used in Circle View by having a mouse over state where the mouse hovers above a paper reference and any other instances of the paper on the screen will highlight. This makes it easy to see the number of times that paper is referred to by other authors. If a paper occurs many times in a given citation network, it can be implied that it is a very important paper that should be investigated further. In Figure 10.2, CircleView uses the feature-present/feature-absent effect to make it easy to see that the selected paper is referenced three times in the immediate citation network.

Rauschenberger and Yantis (2006) explores perceptual encoding efficiency of visual search. The authors conducted ten separate experiments that challenge some of the traditional assumptions of visual searching. The first seven experiments explore FIT and attentional engagement theory (AET). The remaining experiments cover redundancy. The first seven experiments are derived from Treisman and Souther's (1985) study. In this study, participants searched for a circle with a line through it among circles without lines, and vice versa. The theory is that it is easier to find the circle with the line through it (feature-present) rather than find the circle without the line (feature-absent).

The purpose of the first experiment was to duplicate Treisman and Souther's study to establish a base-line with which to contrast the results of the subsequent experiments. The participants were all undergraduate students with normal vision. The experiment used computer displays showing circles measuring 1.4" in diameter and lines 1" in length. The results of this experiment was very close to the original study. The easy condition, searching for a circle with a line, yielded a rate of 4 milliseconds per item when the target was present and 3 milliseconds per item when the target was absent. The


PaperCube – Peter Bergström
http://papercube.peterbergstrom.com/
Google
Mode: Papers
View: Circle View
Direction: References
Report bugs
Saved Items
SEARCH
Fast Effective Rule Induction
William W. Cohen
[2 refs]
[232 cites]
A Comparative Study of Inductive Logic Programming Methods for Software Fault Prediction
William W. Cohen
Prem Devanbu
[9 refs]
[4 cites]
The Learnability of Description Logics with Equality Constraints
Pac-learning Nondeterminate Clauses
Bag of Words and Word Positions
CLASSIC Learning
FOIL: A Midterm Report
Learning the CLASSIC Description Logic: Theoretical and Experimental Results
A Validation Of Object-Oriented Design Metrics As Quality Indicators
Theories for Mutagenicity: A Study in First-Order and Feature-Based Induction
# refs/cites: least
most
Hide papers ≤ 0 refs:
Hide papers ≤ 0 cites:

FOIL: A Midterm Report
R. M. Cameron-jones, J. R. Quinlan
: FOIL is a learning system that constructs Horn clause programs from examples. This paper summarises the development of FOIL from 1989 up to early 1993 and evaluates its effectiveness on a non-trivia
Publication Date: 1993
3 references **98 citations**

Figure 10.2: Screenshot of the new Circle View view in PaperCube showing the feature-present/feature-absent effect.

difficult condition, finding a circle without a line, took 20 milliseconds and 29 milliseconds per item. These results are both consistent with FIT and AET as the original experiment found.

In the subsequent experiments, modifications of the original study were conducted. The second experiment used circles that looked like O's and Q's instead of circles, and circles with vertical lines. The hypothesis was that the results would be the same as in experiments The results of this experiment showed that rotating the stimuli by 45 degrees made the visual search about twice as fast as it was in experiment one. To test the theory that the participant recognized the O's and Q's more easily since they are more familiar, making the line stick out of the top left, the results were shown to be the same as the first experiment. The authors noted that the results of the second experiment are hard to reconcile with traditional theories.

The third experiment duplicates the first experiment, but the circles have randomly assigned directions. By moving away from homogeneous targets, the experimenters wanted to explore the effects of non-target similarity of the visual search. The target present results were similar to experiment one, but the results of a visual search with the target absent was higher at 58 milliseconds per item. The results show that having the lines in different orientations did not significantly change search efficiency. This shows that no matter what the line orientation may be, looking for an item without a line is the same. The other experiments used a variation of line orientations and line types. The results showed that the more efficiently the non-target items can be encoded, the faster the visual search is.

Circle View uses circles that are either a lighter color of gray to signify the non-target condition, and uses a darker gray (highlighted) circle as the target condition. Since the highlighted circle only changes the perceived brightness of the circle, it is easy to find during a visual search. The use of the feature-present/feature-absent effect in Circle View may be one of the most effective component of visualization method.

10.5 Using An Interface and Designing an Interface

As a visualization, Circle View exploits both top-down and bottom-up processing. Not only does it try to use well-established paradigms that users have learned through the interaction with other interfaces in the past, but also other principles, such as color coding and feature-present/feature-absent effect, that exploit bottom-up processing. Some questions about Circle View's design are how do researchers approach the interactive task of foraging for papers and how do characteristics of the rendering of the visualization make it easier or harder to identity items on the screen. Another aspect of UI design is addressing the design process from the top-down and bottom-up.

10.6 Soft Constraints Hypothesis

Gray, Sims, Fu, and Schoelles (2006) explored how individuals allocate cognitive resources while conducting an interactive task. The basis of their experiments is the so called "soft constraints hypothesis" (SCH). This hypothesis maintains that at the 1/3 to 3 second level of analysis the control system selects tasks that will minimize the cost of performance while achieving an expected benefit. This resource allocation strategy is evident while searching for information using the web, something that anyone who uses the internet can relate to at some level. The paper covers three experiments using a user interface with a resource window, workspace window, and a target window. The target window showed a pattern of blocks that a participant would duplicate in a workspace window by moving blocks from the resource window to the workspace. The contents of the resource and workspace windows were covered until the participant moved the cursor into one of the windows. The method and cost of uncovering the target window varied across the experiments. Circle View shows data depending on where the cursor is on the screen.

The first experiment had three levels of difficulty for uncovering the target window Easy: uncover it with pressing a key on the keyboard, Medium: uncover it when the mouse cursor was inside the target window, and Hard: Same as the medium difficulty level, but with a 1 second delay. The results of the experiment showed that the higher the cost of access, the more blocks were correctly placed after the first look. Since there was a higher cost of unhiding the target window, participants spent longer looking at the target once it was uncovered to encode more information per looks. For the easy condition, participants looked 6.8 times at the target window for an average of 1179 milliseconds, for the medium condition, participants looked 6.4 times at the target window for an average of 1241 milliseconds, and for the hard condition, participants looked 4.1 times at the target window for an average of 2334 milliseconds.

The second experiment also had three levels of difficulty for uncovering the target window Easy: click a large button with dimensions 260 x 260 pixels, Medium: click a smaller button with dimensions 60 x 60 pixels, and Hard: click a very small button with dimensions 8 x 8 pixels. As in the first experiment, the participants lowered the number of accesses as the difficulty became harder and looked for longer. For the easy condition, participants looked 5.1 times at the target window for an average of 1345 milliseconds, for the medium condition, participants looked 4.2 times at the target window for an average of 2182 milliseconds, and for the hard condition, participants looked 3.5 times at the target window for an average of 2669 milliseconds.

The third experiment was a variation of the first experiments' hard difficulty but with varying delays of 0-200 milliseconds, 400-800 milliseconds, and 1600-3200 milliseconds. The results of the last experiment was also as expected. Zero milliseconds yielded 5.6 views and a first look average of 1603 milliseconds, 200 milliseconds yielded 4.8 views and a first look average of 1702 milliseconds, 400 milliseconds yielded 4.5 views and a first look average of 1929 milliseconds, 800 milliseconds yielded 3.7 views and a first look average of 2392 milliseconds, 1600 milliseconds yielded 3.5 views and a first look average of 3614 milliseconds, and 3200 milliseconds yielded 2.9 views and a first look average of 4634 milliseconds. More interestingly, the blocks correctly placed with zero milliseconds was 2.00 and at 3200 milliseconds the number was 3.58.

Circle View shows the highlighted circles and associated meta data when the cursor hovers over a circle. However, it does not hide the information until the mouse hovers over another circle. The experiments in this study shows that Circle View may be aiding the researcher by not hiding the elements when the mouse leaves a circle, but waiting until the mouse enters a new one. It is easier because no strategy has to be devised to retain that information before it goes away and allows the researcher to focus on the actual bibliographic meta data rather than try to beat the user interface. It was considered to have the data disappear when the cursor leaves a circle, but it seemed like it would cause unnecessary stress while using the application.

10.7 Icon Characteristics

McDougall, de Brujin, and Curry (2000) report on a series of studies that examine the characteristics of icon usability. In a UI, it is important that icons and interface elements are easy to interpret. Good icons are better than words to communicate meaning. Circle View tries to show information that has traditionally only been shown using words in a way that uses simple iconography. Although Circle View is more complex than a single icon for your word processor, it does use some of the same principles, but on a larger scale.

One of the experiments experiment tested the effectiveness of icons in a visual search task. Icons had two independent characteristics: concreteness and complexity. Concreteness is a measure of the level of abstraction of the icon. Does it depict a concrete object or something abstract? Complexity is a measure of how complicated the icon looks. Participants, who where affiliated with the University of Wales, were shown one icon, then allowed to click on that icon in a grid of icons. The time to locate the icon in the grid was recorded. The result of the study showed that no matter if the icon was abstract or concrete, the performance was the same. However, the more complex the icon, the longer the time to locate it in the grid. Therefore, the complexity of the icon is important factor in user performance during a visual searching task.

The authors conclude that even though using concrete icons may be favorable to new users, after it is learned what the icons describe, the level of abstraction doesn't subsequently matter and does not add to usability. It can be theorized that a concrete icon would be easier due to top-down processing since it depicts something that the user already is familiar with, such as a postage stamp being the icon for an email application. Icon complexity is a lot more important because it makes the icon more difficult to identify no matter the level of concreteness. Circle View uses a simple basic building block for its visualization method, the circle. Since a circle is about as simple as possible, the user doesn't have to work hard to encode it. However, it is an abstraction to see a paper, traditionally rendered on the screen as a rectangle with horizontal lines, now be rendered as a circle. Once the fact that the circle represents a paper is learned, there should not be a problem.

10.8 Designing Interfaces Through Experimentation

Zacks and Tversky (2003) discusses using top-down and bottom-up methods for designing user interfaces. The top-down method uses well-established theories and research to create user interfaces that support those concepts. The bottom-up method looks at the goal of the task and through experimentation, derive a UI. A top-down approach works well because the concepts utilized are commonly known, however, it may be too general to effectively design certain specialized user interfaces. Circle View and our subsequent citation network visualizations and user interfaces can benefit from both looking at well established cognitive theories, but also, but taking a look at the particular task that is being performed and design something from the bottom-up. Zacks and Tversky explore the marriage of top-down and bottom-up design by looking at event cognition and media. Event cognition theorizes that procedures should be assembled hierarchically and media in the form of graphics and text can be used effectively to aid learning and memory.

In the first experiment the authors taught participants, who were undergraduates at Stanford University, to assemble a saxophone. The participants learned the steps using a computer interface that varied on two dimensions. First, the media varied: text, text with images, and text with video. Second, the layout was varied by showing the steps either in a hierarchical list or flat list. The students used the computer program to learn the steps of assembling the instrument, then were asked to assemble it from memory. The results were measured in three ways: how did the various learning methods impact the participant's performance on the task, how did the methods impact memory, and how long was required for training? Performance across the board was good and on average 94 percent of the 21 parts were assembled correctly. The richer the presentation method, the more was recalled: 16.20 parts for text, 19.33 parts for photos, and 19.67 parts for video in the hierarchal list and 18.00 parts for text, 17.83 parts for photos, and 19.67 parts for the flat list. Training time was the longest for video at 516 seconds for the hierarchal list and 452 seconds for the flat list. The lowest was the text list at 253 seconds for the hierarchal list and 236 seconds for the flat list. In the middle was the photos with 335 seconds and 422 seconds for the hierarchal list and flat list, respectively.

The second experiment had the participants assemble a model scorpion using the same user interface structure and procedure as in the first experiment but without the text-only media. Performance was not as good as the first experiment with on average 52.4 percent of the 14 parts assembled correctly. It was noted that the majority of the errors started when participants started assembly out of order. Contrary to experiment one, the richer the presentation method, the less was recalled: 12.95 parts for photos, and 10.55 parts for video in the hierarchal list and 13.75 parts for photos, and 12.86 parts for the flat list. Training time was not statistically different with a mean of 422 seconds across both learning methods.

Zacks and Tversky's experiments show that in order to arrive at an optimal UI design, one not only need approach it from the top-down with accepted cognitive design principles, but also look from the bottom-up and conduct experiments. The good thing about using an experimental approach is iterative design. In software engineering, using iteration is very popular because it permits for the quick refinement and optimization of software. Through experimental cognitive psychology, one can iterate effectively to validate and develop user interfaces. Circle View and our other planned user interfaces would greatly benefit from this approach. We have already found various places where we have made UI design decisions are less than optimal. By using an experimental approach, we would be able to refine and have scientific evidence backing our design choices.

10.9 Conclusion

Circle View incorporates UI elements that may or not be effective. This survey of cognitive psychology research has found areas where Circle View is strong and areas where it needs improvement. The use of color to signify the number of citations a paper with a temperature style scale is less than ideal. The use of a non-linear scale requires more information to be provided in the form of a detailed legend to decode the meaning of the colors. Spence, Kutlesa, and Rose suggest that using a linear scale with one color of varying brightness and saturation is the easiest method to code quantity. If Circle View changes to such as scale, we could eliminate the legend on the side. Another reason to do this is purely aesthetic: the temperature scale is ugly. Also, Jameson, Kaiwai, and Bamber show that one could perhaps show two dimensions by using brightness and color. Perhaps this is an opportunity to not only show citations but also show references in the same UI element.

Using the feature-present/feature-absent effect to aid the visual search for references is perhaps the most effective UI feature in Circle View. Treisman's, Rauschenberger and Yantis' studies vividly illustrate the effectiveness of the feature-present/feature-absent effect. In Circle View, is very easy to find references that appear more than once in the citation network when they are highlighted. Several other user interfaces have been considered could take advantage of the effect.

Designing a UI that is easy to use is very important. Not making it too complicated is key. By exploiting both top-down and bottom-up strategies, it is possible to make the interface's design to the user's advantage. First, a user can use prior knowledge of the interface's interaction behavior to optimize how the tool is used. Gray, Sims, Fu, and Schoelles used the soft constraint hypothesis to describe how users will alter their behavior if a computer based task is difficult. One example was that if something disappears after one's cursor leaves an area, users will look longer and take up more cognitive resources. In Circle View, the mouse over effect to highlight references and show meta data stays visible until the user enters another circle. This allows the user to allocate resources to interpret the meta data instead of spending cognitive resources to retain the information when it unnecessarily disappears.

The use of simple iconography in Circle View makes the UI easy to use. Although circles are abstract when it comes to describing academic papers, they are very simple. Therefore, it is easy to encode them which has its advantages where there many circles visible at the same time.

Zacks and Tversky discusses something that is very important for the further development of Circle View and other user interfaces. Originally, Circle View's interface was created without any external input and testing. This was a weak point of the original paper that caused it to be rejected to a major conference. Fortunately, the paper was accepted to a smaller conference later even though it was lacking a user study. In this paper, by looking at well known cognitive theories from the top-down, some of the design choices in Circle View's interface have been validated and invalidated. However, no bottom-up design review has been conducted and therefore the UI may have problems that will not be uncovered until some experiments are conducted.

10.10 Results Applied As PaperCube

As a result of this evaluation, Circle View as it is implemented as part of PaperCube has been enhanced to take in account some of the things learned. First, it is important to point out that the original visualization method does a lot of things right, especially through its use of the feature-absent/feature-present effect.

However, this work has improved upon Circle View by simplifying the color scheme used to show the significance of papers. Instead of using a problematic temperature-based scale, it is now a linear scale that goes from black to red where red signifies the most important paper in terms of its number of references or citations.

Also, another improvement is the dropping of absolute numbering when looking at the number of references or citations to assign color. Instead, the color intensity for a paper is determined as a relative number compared to the rest of the papers in the view. Therefore, it is much easier to determine significance without having to use a complicated legend to decode what number range that the color corresponds to in the visualization.

Furthermore, the new implementation also uses animated transitions to provide additional context to what is happening as the user interacts with Circle View. In the original implementation, the technical limitations at the time made the application reload the page during navigation, which made it easy to lose one's place.

The lessons learned in this evaluation also has been carried over across the other visualization methods used in PaperCube. For example, the highlighting of multiple instances of references is used in Tree Map view and also, the views avoid using complicated coloring schemes to convey meaning that requires a legend to decipher.

Chapter 11

User Study

The main goal of this work was to evaluate the effectiveness of PaperCube as a way to augment digital library navigation. Therefore, a user study was conducted asking past and present graduate students, researchers, and professors to participate. The survey's goal was to get their thoughts on current web-based digital libraries, the general features and interface elements used in PaperCube, as well as rating the effectiveness of the various paper and author visualizations used.

In order to provide the participants with some context, a brief video (http://vimeo.com/3323956) was created to give the participants some background of the goals of PaperCube as well as showing the available features and views.

11.1 Survey Goals

The goal for the survey was not only to evaluate PaperCube's effectiveness at augmenting digital library search and navigation, but also to as the specific UI elements that were used throughout PaperCube. Therefore, a set of HCI-centric questions are used to evaluate the various elements of PaperCube. A full sample of the survey is available in Appendix A and the full quantitative results are available in Appendix B.

- **PaperCube's Potential**

  Was PaperCube successful overall at augmenting the use of a web-based digital library and would this be useful to include in future digital library services? These questions were generally not in multiple choice format, but rather allowed the participant to respond freely.

- **View and Visualization Effectiveness**

  The views and their visualizations are at the center of this work. The desire was to find out what views were better than others in the eyes of the participants. The visualizations put forth in PaperCube may or may not be as effective as hypothesized.

- **Were Paper Or Author Views Preferred?**

  Since PaperCube shows both papers and authors, which was preferred by the participants? What were the most effective views for each of the viewing modes?

- **Slider Controls to Adjust Display Parameters**

  Was the ability of being able to adjust the display parameters of the visualizations useful to users?

- **Zooming and Resolution Independence**

  Was the ability to zoom in on the visualization useful? The idea was that this would be a very nice feature especially when showing a lot of data on the screen. Was the preview widget at the top right of the window useful? How about dragging and panning?

- **Fan Popup Menu**

  How effective and intuitive was the fan popup menu to the typical user. Since it is a very convenient way to navigate, it was used as the main method of navigation, but since it is not well-known, perhaps it was difficult to use.

- **History and Navigation Buttons**

  Was the ability to go back and forth in the history useful? Were the buttons and menus in the toolbar intuitive?

- **Saving Functionality**

  Was the feature of saving items used? If it was, was it useful to be able to return to previously viewed items?

11.2 Participants

The participant pool was made up of past and present graduate students, researchers with undergraduate degrees, and professors as shown in Figure 11.1. Forty-eight percent of participants were working professionals with a graduate degree. Their backgrounds were quite diverse, ranging from law school graduates, mechanical engineering master's students up to PhDs in computer science, computer engineering, and astronomy. The second largest pool of participants were graduate students pursing degrees in various fields including computer science, computer engineering, law, geology, and library science.

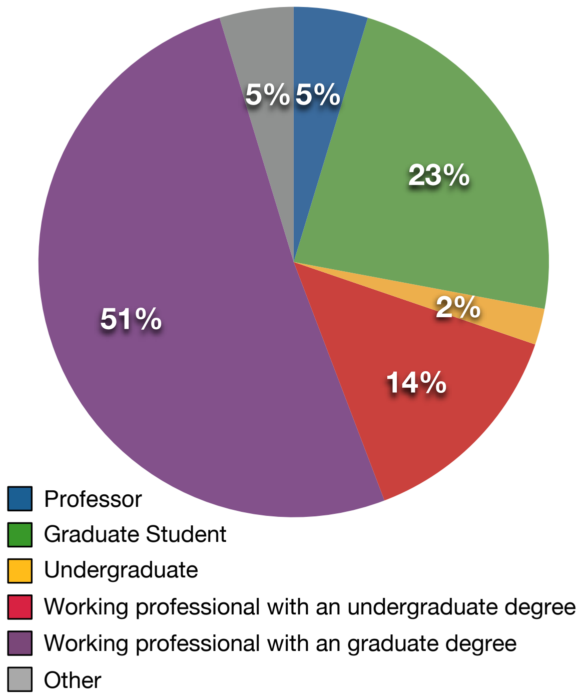


Figure 11.1: Q 3: Which describes you best? (required)

Also, several participants had no graduate degrees but have been published in major journals or conducted research as professionals into the digital libraries at major institutions. Two actively teaching professors participated in the survey.

11.2.1 "Question 9: What digital libraries do you use?"

With the diverse pool of participants used for this survey, the set of used digital library services was quite large and varied.

Figure 11.2 shows the digital libraries most commonly mentioned by the participants. Since the majority of participants were computer scientists or engineers, it was not surprising to see the ACM Digital Library and IEEExplore top the list. However, since lawyers and medical professionals also participated, services such as LexisNexus, Westlaw, and PubMed were also mentioned. Some free services were also mentioned including CiteSeer and Google Scholar.

11.2.2 "Question 10: What do you like about them?"

Many of the participants stressed the fact that they are online as their major advantage over a "physical library." Also, the online nature of digital libraries makes it easy to "download PDFs, and

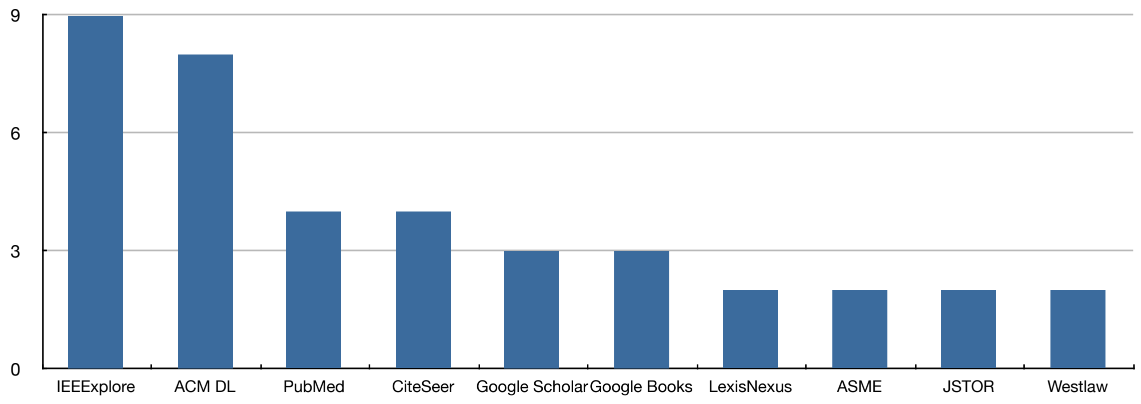


Figure 11.2: Q 9: What digital libraries do you use?

import the references in to BibTeX or EndNote" and "copy and paste text."

Also, the ease of searching across multiple online digital libraries by using services such as Google Scholar was highlighted as a benefit. For example, one participant said that it is "nice to use Google to search a wide variety of journals and databases" and that it makes Google a way to conduct an "interdisciplinary search" for papers. Another benefit is that digital libraries allow users to have "access to lots of papers" and "links to the papers themselves." Furthermore, the fact that these libraries were updated frequently, if not daily, is a great advantage over physical libraries or printed journals.

11.2.3 "Question 11: What do you dislike about them?"

The participants had several major issues with the existing services. First, one problem with a lot of digital libraries today is that they are fee-based or, as one participant put it, "behind paywalls." Some participants felt quite strongly about this issue particularly, when search engines have access to non-publicly accessible papers. "Cloaked PDFs should be banned" was the opinion of one participant. Services such as Google Scholar provide the impression that papers are accessible but then only link to the source paper that actually resides in repository that needs the user to pay for access.

Another issue was that current digital libraries do a poor job at helping the user find information that is "practical" and "deep" in the amount of information provided. Also, participants expressed the frustration that "determining relevance is hard" and it is difficult to find "references or related material." In general, participants found the user interfaces of current digital libraries "awkward," "cumbersome," and "relatively primitive." One participant summed it up succinctly: "terrible UI."

Another issue brought up was that the searching, while powerful, do not necessarily help the user to find relevant information. Participants commented that it is difficult to “find papers on a particular topic without knowing the title and/or authors” and due to this, particularly in law, “the process in which you browse in a particular field of law is disconnected from the page on which you actually type in a search.” Other participants commented that the searching capabilities and methods of digital libraries are not standardized. One participant said that the “search functions in the different databases are often unique to that database so you have to keep re-learning how to search” and another said that it “takes a while to become proficient in the myriad services each site offers.”

11.3 PaperCube’s General User Interface

One goal of the study was to validate if the UI features in PaperCube were effective. Therefore, a set of HCI-centric questions were asked so that participants could rate the effectiveness of some of the major functionality of PaperCube. These rating questions were numerically ranked from 1 to 5 where 5 was the most favorable rating.

11.3.1 “Question 12: What is your impression of the general UI of PaperCube?”

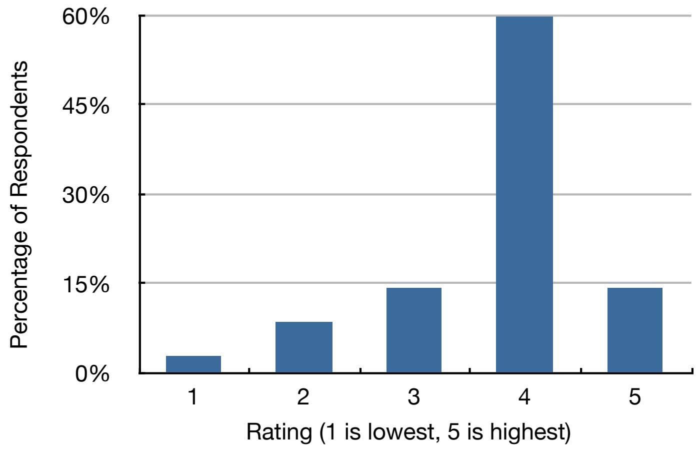


Figure 11.3: Q 12: What is your impression of the general UI of PaperCube?

Figure 11.3, shows the distribution of responses from the participants. Most participants rated PaperCube favorably with an average rating of 3.8 out of 5. Participants generally liked Paper-

Cube and thought that it has "great potential." Especially, participants commented on its dynamic nature and one participant said that the "interface is surprising rich interactive and responsive" and another said that he is "impressed by the general responsiveness and the speed of data retrieval." Furthermore, several participants liked the that PaperCube is very "fluid," "impressive," and "beautiful."

Due to PaperCube's somewhat non-standard user interface, some participants had issues with some of the UI design decisions. Some participants found the UI "confusing" at first and that "too much functionality is hidden or not readily obvious." However, participants said that PaperCube does have a "learning curve," but "easy" once a user gets used to it.

One element that was commented on by several participants was the fact that the search pane on the left of the browser window is hidden by default and it was "difficult to locate at first." Also, participants also did not like the implementation of the menus in PaperCube saying that they were a bit inconsistent in terms of user interaction.

11.3.2 "Question 13: History and navigation buttons."

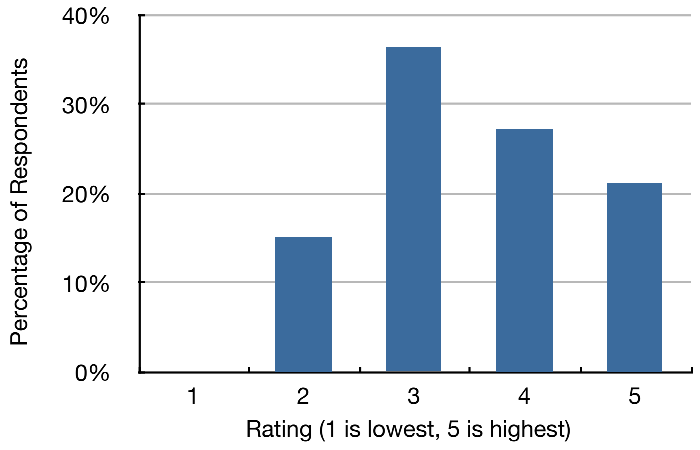


Figure 11.4: Q 13: History and navigation buttons.

The history buttons at the top of the browser window allows a user to go back and forward in the browsing history. Also, this question addresses general navigation in PaperCube.

Figure 11.4, shows that the majority of users rated this feature an average of 3.6 out of 5. Since this feature was not very discoverable, mainly due to the fact that it was disconnected from the web browser's navigation buttons, most participants did not actually use this functionality because it was hard to "discover" and the buttons could have been "larger." One participant said that he wished that the "browser's back button didn't take me out of PaperCube." The best way to get users to actually use this feature would be to make it easier to discover and coupling the functionality with the web browser's own history buttons, which is relatively simple.

Also, some participants discovered that the back button was occasionally error prone. One participant said that "going 'back' doesn't always work." However, this bugginess, while problematic, was not surprising because of the experimental nature of PaperCube.

11.3.3 "Question 14: The saving functionality."

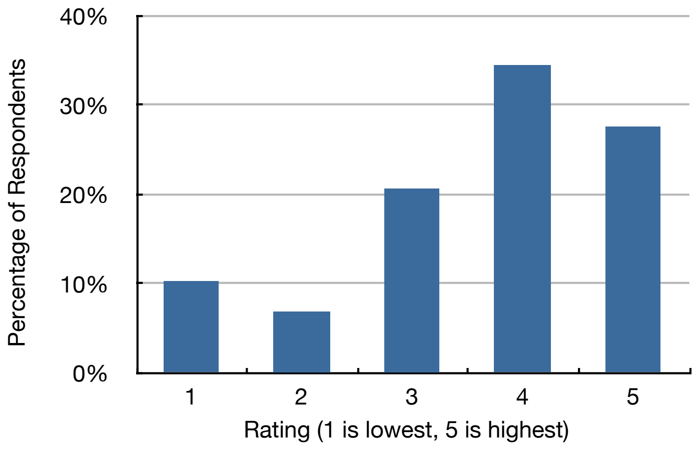


Figure 11.5: Q 14: The saving functionality.

Through the fan menu, users can save a paper or author item for later viewing. This feature is supported at the moment by using cookies and while it is useful to users, it is not a fully developed system with user management. Figure 11.5 shows that most users found the feature useful rating it on average 3.6 out of 5. However, most of the comments showed that they did not use the feature but it was nice to have.

One interesting note that was not expected was that one participant said that he liked "that you can print the list itself," something that was not planned for during development. Also, some users were confused as to how to save the paper or author item. Since the functionality was limited to the fan menu, it was not as discoverable as it should have been.

Some participants would like to have more advanced features such as e-mail to a friend, the ability to save relationships between items, or export items in some way. Some participants also had trouble using this functionality, perhaps due to limitations imposed by the their web browser's privacy settings since it is cookie-based.

11.3.4 “Question 15: The zooming and panning widget.”

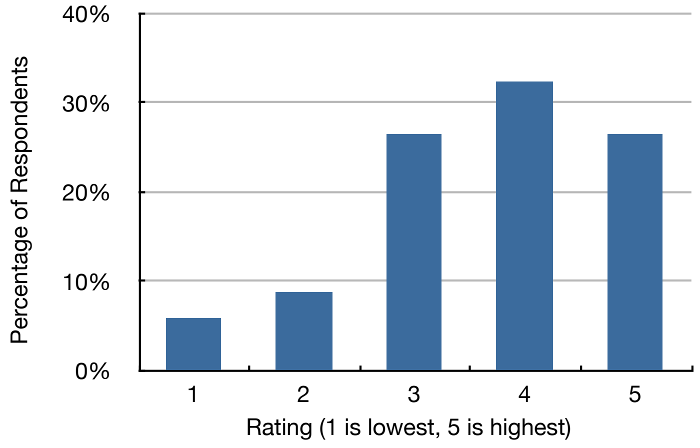


Figure 11.6: Q 15: The zooming and panning widget.

One of the major meta-features of PaperCube, resolution independence, was implemented by using the zooming and panning widget at the top right of the screen. When zoomed in, the preview window slides down to show what portion of the window was visible in the main window.

Figure 11.6 shows that the majority of users liked the zooming widget rating it an average of 3.7 out of 5. However, one of the major complaints was that when the zoom widget is panned, the actual visualization does not pan until it the user lets go of the mouse button. One participant said that it would have been “nice if the page adjusted realtime to react to the interactions within the widget” and another said, that a “live interaction between the overview and the zoomed in area would be great.” Having the main view not update when panning inside the widget was not the originally designed behavior, but rather was added later in development as a performance optimization. It is relatively expensive to pan the main window in the real time. Therefore, in order to make the experience pleasant for most users, the decision was made to optimize for performance while sacrificing instantaneous feedback.

Some of the issues brought up by the participants included the fact that the icons used in the zooming widget, were a bit “obscure” and not very “intuitive.” Several participants suggested that using + or - symbols would be more easily understood by users. The icons currently in PaperCube uses differently sized circles, which is not straight forward. Also, some participants suggested that the zooming widget include a “thumbnail” image of the visualization in the main window.

Another issue that was brought up was that when zooming in, the text inside the rendered circles also increased in size. One participant suggested that it would be nice that one is "when zoomed in, you could see the full paper title."

11.3.5 "Question 16: The general feature of resolution independence in PaperCube."

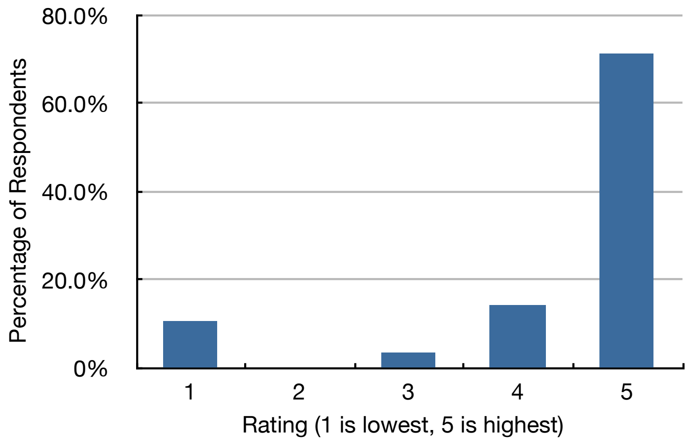


Figure 11.7: Q 16: The general feature of resolution independence in PaperCube.

Zooming was partly discussed in the previous question, but this question asks not only about the zooming widget, but also the implementation of resolution independence in the visualizations themselves. Figure 11.7 shows that a overwhelming majority of the participants liked this feature, rating it an average of 4.4 out of 5.

Participants thought that this was a very strong feature, calling it "excellent," "awesome," "the way it should be," and "fantastic!" One participant said that it was impressive but not sure if it has been fully "exploited" in PaperCube. The resolution independence used in PaperCube has a lot of potential. Furthermore, the use of "vector graphics" made it easy to scale fonts because normally it is really "messy" and this implementation keeps it "clean." An interesting anomaly for this question was that the participants who rated this feature as a 1, the lowest rating, all had favorable comments although one said that it "did not make a difference to the use product."

11.3.6 “Question 17: The UI paradigm for the pop up pie menus.”

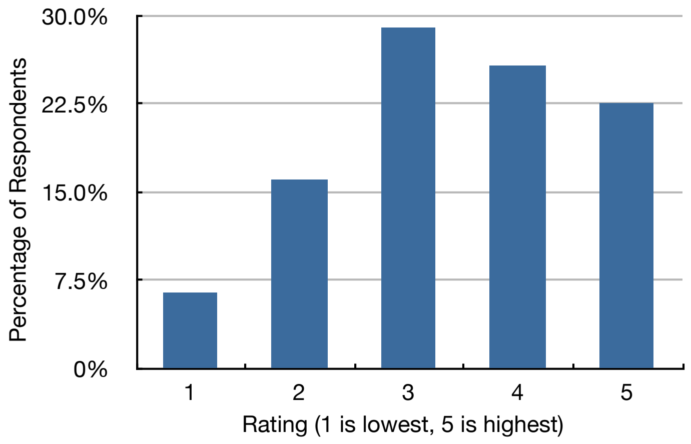


Figure 11.8: Q 17: The UI paradigm for the pop up pie menus.

Question 17 asks about the pie/fan pop up menu. One problem here is that the term “pie” menu is not fully explained and the next question uses the term fan menu. Therefore, these questions caused some confusion because of the terms being inconsistent.

As shown in Figure 11.8, the participants rated this feature an average of 3.4 out of 5. Since this is a non-standard UI element, many felt that it was “inconsistent with the rest of the UI”, “overly clever,” and one participant was not sure if it “fits” in this application. Some participants suggested that using a standard menu would have sufficed and have been less confusing to users. Having a menu of some sort was a good decision, but using this type of menu might have been pushing the envelope a bit too far. The menu is radically different and as one participant said, it is not good to “invent a nonstandard UI component” unless there is a “good reason for doing so” and in this case, a standard context menu would have done the job. However, the reasoning for using this type of menu control was also to explore the usability of the paradigm.

The participants was that they would have liked the menu to be activated on right click only and perhaps make it a full circle in order to make the text larger and not displayed on an angle.

One issue found during the course of the study was that web browsers, through browser inconsistencies, behaved differently. WebKit-based browsers hide the menu once the user exited the SVG element but Firefox kept the menu visible until something else was clicked. This modal behavior in Firefox was not intended but it was the behavior that most encountered when using the menu. However, the menu was still appreciated by the participants. One participant liked that it was nice to “simply” have access the most useful actions and not have them “buried in menu bars.”

11.3.7 “Question 18: The available options in the fan menus.”

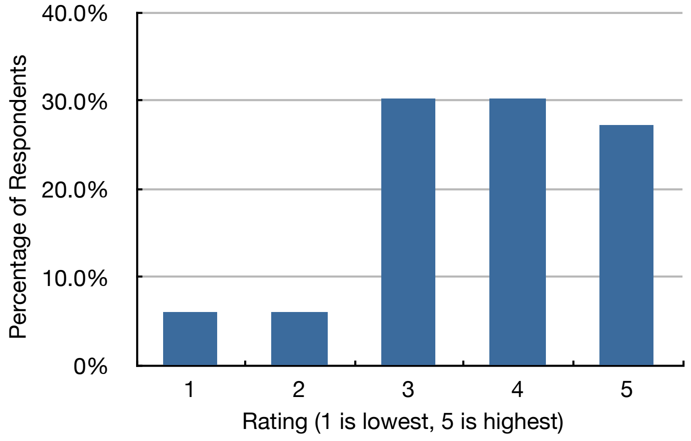


Figure 11.9: Q 18: The available options in the fan menus.

No matter the effectiveness of the pie/fan menu paradigm itself, the menu choices had to be evaluated separately. The available options were labelled using single words or short phrases which was perhaps not the best solution because some of the options were confusing or unclear. The participants liked the available options overall rating it an average of 3.7 out of 5.

The “refocus” option is perhaps the most important navigation action in PaperCube because it helps the user navigate from one paper or author to another. Without it, navigation would be very difficult. However, some participants did not find the label for the refocus action clear enough to describe the functionality. Also, the “pin lines” option was unclear in the `NodeGraph`-based views and Papers Per Year view. Therefore, the options in the menu would greatly benefit from rewording and also adding tool tip hints to add more context to the user in order to figure out what actions the options perform.

11.3.8 “Question 19: The animation transitions used in certain views.”

Figure 11.10 shows the that a lot of the participants liked animations in the visualizations. The average rating for this feature was 4 out of 5.

As expected, the experience of the participants varied depending on the speed of their web browsers and computers since the faster both are, the smoother the animation will be rendered. The animation engine in PaperCube ensures that all animations complete within the desired time span, but if an animation consists of more frames in the same time span, the smoother it will appear due to the higher frame rate.

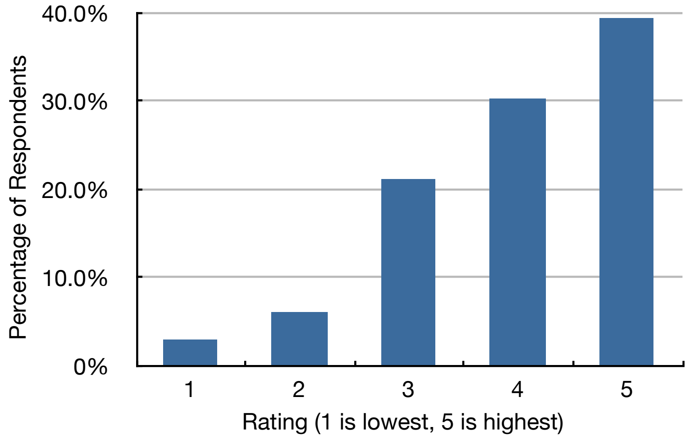


Figure 11.10: Q 19: The animation transitions used in certain views.

In general, participants liked the animations as cues as to how the data was transforming as a result to their adjustments because gave PaperCube a much more "solid feel" to see parts moving off screen instead of them hiding immediately. Another participant said that the animations "help the user understand the conceptual model" of the visualizations better. One suggestion was to add some "springiness," "elasticity," or "easing" in order to make the animated transitions feel a bit more dynamic. However, some thought that animations slow down the experience when it does not add any additional cues.

Also, one theme that was seen throughout the comments was that certain views were smoother than others. Some participants experienced that it was "choppy" or "jumpy" in certain cases. However, this varied a lot by the number of items rendered and also with the browser version used. This was expected because certain views, such as Papers Per Year view, will render more content than other views. In general, the more visible items that need to be animated, the lower the frame rate of the animation.

11.3.9 "Question 20: The ability to tweak the settings of the visualizations using slider controls."

Another top goal of PaperCube was the ability to change the display parameters in order to change the amount of information visible on the screen. In general, the parameters that could be adjusted included hiding papers or authors based on a set of thresholds or adjusting the number of levels to be retrieved from the server. The aim of this feature was to allow the user to quickly narrow down what is being shown to something that may be relevant to that particular user.

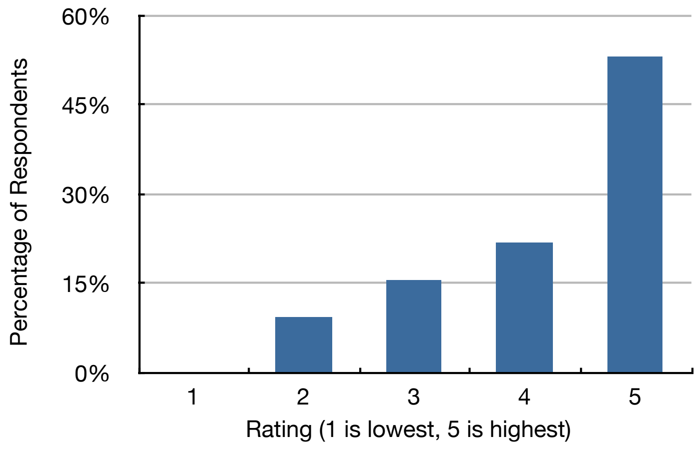


Figure 11.11: Q 20: The ability to tweak the settings of the visualizations using slider controls.

Just as with the zooming and resolution independence feature, Figure 11.10 shows that the ability to adjust the settings of a visualization with the slider controls was overwhelmingly highly rated. The average rating for this feature was 4.2 out of 5 with 53 percent rating it a 5 out of 5. The majority of participants found this feature very useful. One participant noted that this feature "lets the user narrow down results quickly" and another said that is a "great way to play around with the options."

Some participants noted that, just as with the zoom widget, the icons used for the sliders were not very descriptive, one participant calling them "hard to understand." Although there was accompanying text with each slider, the icons were not as useful as they could have been.

Also, participants noted that having multiple sliders caused some confusion because it was not always "obvious what they are targeted for." One participant brought up an idea that was actually considered earlier in development of PaperCube but was later passed on. The original idea was to have the ability to adjust multiple parameters from one slider. However, the problem of describing what such a slider would change might actually be very difficult. Therefore, the choice was made to have multiple sliders instead for simplicity.

Another issue that was commented on was the differences between "citations" and "references," which is a confusing distinction regarding the directionality of a paper relationship. Unfortunately, no other good way to describe this has been found.

11.4 PaperCube's Most Liked and Disliked Views

This section covers four questions that allowed the participants to pick their most and least liked paper and author views. The next section will cover the individual ratings and comments regarding each view in more detail. The most unexpected result of the user survey was that the least liked paper view was Tree Map view because it was expected to rate relatively high compared to the other views.

11.4.1 "Question 30: What paper view did you like the MOST?"

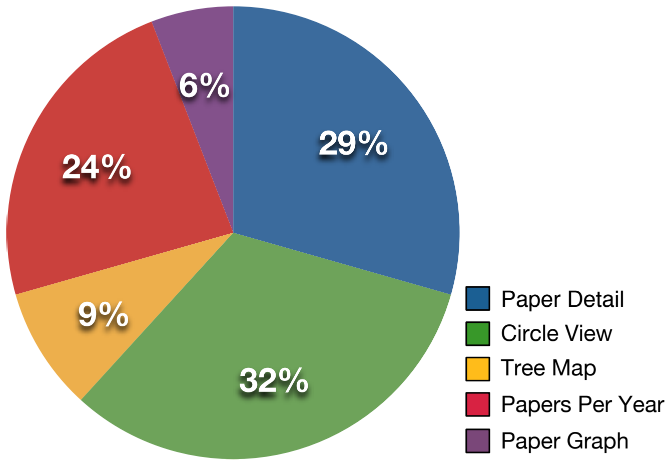


Figure 11.12: Q 30: What paper view did you like the MOST?

Not surprisingly, the largest portion of participants, 32 percent, rated Circle View as the most liked view. The surprising result was that Paper Detail view was the second most liked with 29 percent followed by Papers Per Year view with 24 percent. Tree Map view was surprisingly the second to last liked view with 9 percent of participants rating it most liked view. As expected, Paper Graph view was rated the lowest with only 6 percent.

Participants Who Liked Circle View

Participants who picked Circle View noted that the view was very effective at showing how papers are related to each other. In particular, one participant noted that the view "visualized co-relations in a pattern" and another thought that it "visually organized the information the best." Other comments included that the view made it easy to show papers "relate" to each other and that it "really helped visualize the references that were in common between papers and let me easily navigate related papers." Despite fact that Circle View organizes a paper's citation network in a very rigid manner and also only shows two levels, the visualization is very easy to interpret and quick to learn.

Participants Who Liked Paper Detail

Paper Detail view was not expected to rate as highly as it did. The reason the view was created was to give user an initial starting point as a more familiar UI with lists. The view is not a static webpage-like view, but it is not based on any type of graph visualization. Participants liked this view because it was a very "straight-forward" way to show a paper's meta data. This view also showed more information about a paper in a more information dense manner than some of the visualization-based views. Also, several participants noted that the view serves well as a "'jumping off point" to the other views.

Participants Who Liked Papers Per Year

Papers Per Year view has a lot of potential. The aim for showing how a given field of research evolves over time was central to the view and that was understood by the participants. Participants commented that it was nice to see the "progression over time" and that it was the "clearest" visualization to show relationships between papers.

Participants Who Liked Tree Map

Although not liked by as many participants as expected, the participants who liked Tree Map view noted that the view did a good job at show how "influential" a paper may be in a given field. Also, one participant said that he liked seeing the "depth" of references and another commented that it gave a lot of "insight" into a research topic.

Participants Who Liked Paper Graph

Paper Graph view was the view that the least number of people liked. Participants who liked it said that it was it was the "easiest to visualize how good" a paper was and that it was the "nicest looking" view.

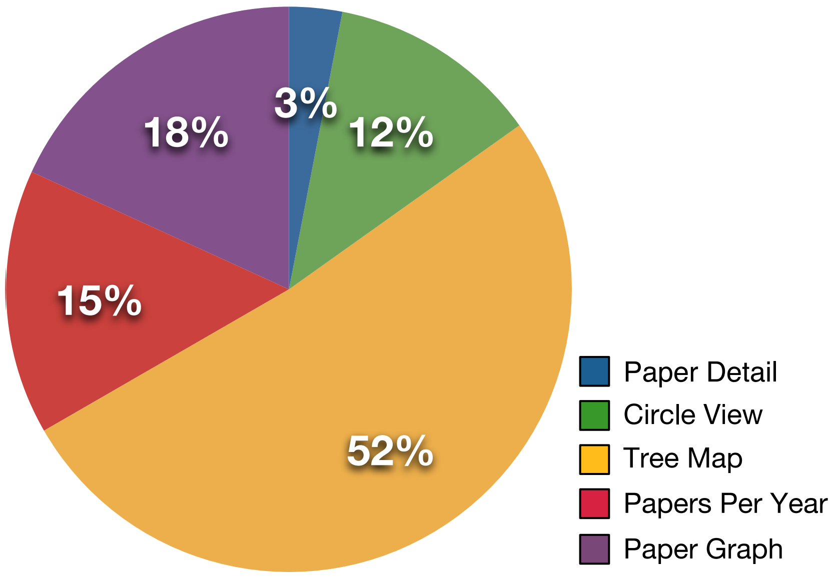


Figure 11.13: Q 31: What paper view did you like the LEAST?

11.4.2 "Question 31: What paper view did you like the LEAST?"

The results of this question are, as expected, roughly the inverse of the previous question. However, a proportionally larger percentage of participants, 52 percent, picked Tree Map view as the view that was the least liked. The other anomaly was that Circle View, the most liked view, was disliked by a larger percentage of participants than Paper Detail view. Only 3 percent of participants picked Paper Detail view as the view that they least liked.

Participants Who Disliked Tree Map

The majority of participants said that it was "hard to read," "very busy," and "cluttered." The hope for Tree Map view was to actually show more information than needed but then allow users to narrow down the data shown by adjusting the slider controls. Since the view was geared towards being very information dense and show many levels of a paper's citation network, it achieved the goal, but most participants did not find the density an advantage. Also, another trend in the comments was that participants did not like the color scheme saying that it was "too intense," not as "visually appealing," and "hard on the eyes."

Participants Who Disliked Paper Graph

The second most disliked view, with eighteen percent of the participants picking it, was Paper Graph view. Participants said that it was "hard to understand" and that it was hard to "figure it out." Also, other participants said that it was "difficult to navigate" and that it could be "helpful" when the paper relationships are very complex but not for instances when the relationships are "simple."

Participants Who Disliked Papers Per Year

Fifteen percent of participants disliked Papers Per Year view. Participants noted that it was "too intense" for the web browser and displayed "too much" data. Another user noted that it was hard to figure out the best zoom level, showing either too "many" or too "few" nodes.

One problem with this view was noted by another participant saying it was "confusing" when items were highlighted on mouse over. In early development versions of PaperCube, there was a legend showing the color relationships between the reference and citation lines that are displayed when a user mouses over a paper node. This legend was accidentally removed during development and never put back.

Participants Who Disliked Circle View

A very small percentage of participants, nine percent, disliked "Circle View." One participant said that the view was the "least flexible" of all the views. Interestingly enough, that was highlighted as a strength by participants who rated this view their most liked.

Participants Who Disliked Paper Detail

Only one participant disliked Paper Detail view. The reason given was because it had a "more standard" way of organizing the information about a paper.

11.4.3 "Question 32: What author view did you like the MOST?"

It was expected that the most liked author views would be Collaborators view followed by Author Detail view. However, the results show that Author Detail view, which was selected by forty percent of participants, edged out Collaborators view by three percent. The two other views, Author Cites and Papers were selected by seventeen and seven percent, respectively.

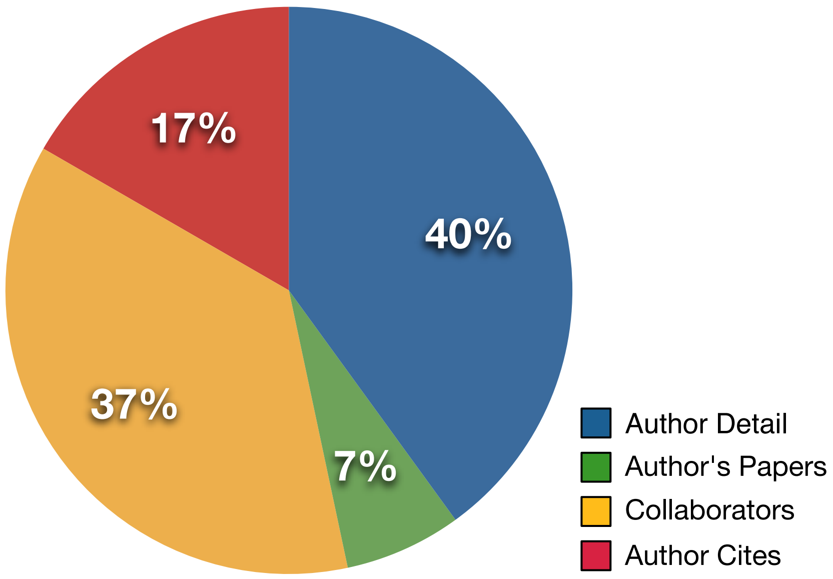


Figure 11.14: Q 32: What author view did you like the MOST?

Participants Who Liked Author Detail

AUTHOR DETAIL view was picked by a surprisingly high percentage of participants. The reasons were very much in line with the comments made by the participants who picked PAPER DETAIL as their favorite paper view. Again, the same participant noted that AUTHOR DETAIL view serves as a good "jumping off point" for further exploration. Also, as with the equivalent paper view, another participant called this view "straight-forward."

Participants Who Liked Collaborators

COLLABORATORS view was one of the views that was expected to create the most interest from participants because it is a relatively unique way to view the data. Participants noted that it was "easy to see who often worked with each other" and that it was "interesting to see who collaborates." One participant said that this visualization of the data is something "you can't get from other sites" and another said that it was "unique." Ultimately, the participants who picked this view appreciated it for the reasons namely that it was created: to easily show how authors work together, which is something that can be difficult to determine without this type of visualization.

Participants Who Liked Author Cites

Author Cites view was created as a way to show the other authors that the author has cited or have been cited by which is not a direct relationship such as collaboration. Participants who liked this view noted that this view provides information that would be "really arduous to get otherwise" and that the view is a good way to find out the "scope of networking" among authors working on the same field of research.

Participants Who Liked Papers

Papers view was expected to rate low because it did not provide as much value as the other views since it only shows the focused author's papers inside of a visualization. Participants who liked this view noted that it was a "nice visual" when there were not too many papers and that it was "nice to see" a list of papers by the focused author.

11.4.4 "Question 33: What author view did you like the LEAST?"

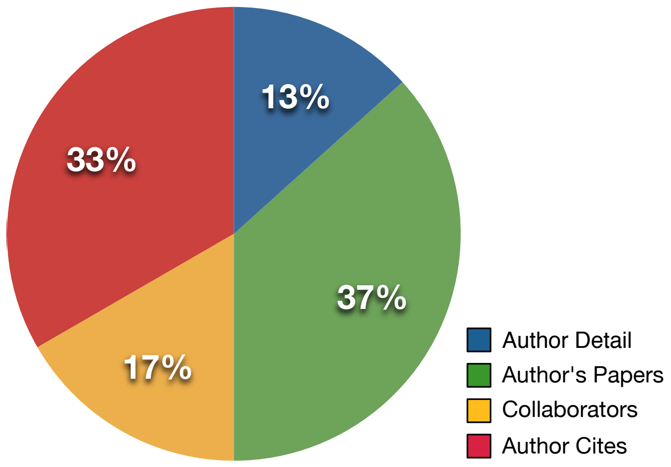


Figure 11.15: Q 33: What author view did you like the LEAST?

As with the paper views, the author views that were disliked were the inverse of the previous question, although with a different proportional distribution. The most disliked view was Papers view with 37 percent of participants followed by Author Cites view with 33 percent. Only 17 percent disliked Collaborators view and 13 percent disliked Author Detail view.

Participants Who Disliked Papers

Participants felt that this view was “useless” and that it did not “add anything” above what the other views had to offer. Participants thought that there was “no good reason” for the author’s papers to be presented as a graph and that the “paradigm” was “stretched thin.” Instead, participants thought that this information is best presented as a list.

Participants Who Disliked Author Cites

Participants disliked this view mostly because this view had “too much going on” and it produced a very “complicated web of citations.” One participant said that he wanted to click and drag around to explore, but since the visualization was so dense, he had to use the “menus and sliders” instead. Also, another participant noted that the relationships between authors made it “practically unreadable,” something that will happen when a lot of information is displayed, which this view is prone to have.

Participants Who Disliked Collaborators

Collaborators view was disliked by a very small number of participants and the only comment noted that it would “help to see number of papers that this person wrote.” However, that information is available for all the displayed authors when mousing over the node.

Participants Who Disliked Author Detail

As with “Collaborators,” this view only had a few participants who disliked it. The only comment was that the participant could not “refocus correctly” on new authors, which was a bug.

11.4.5 "Question 34: Which type of views were most useful for you?"

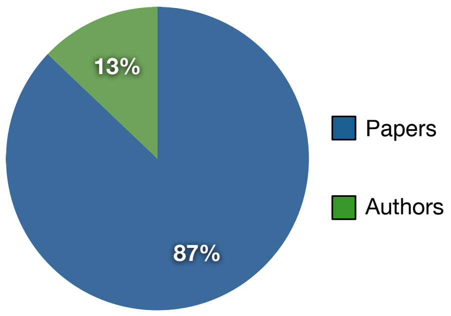


Figure 11.16: Q 34: Which type of views were most useful for you?

A large percentage of participants, 87 percent, thought that paper views were more useful than author views. This was not unexpected because of the fact that PaperCube's primary goal was to show paper relationships and as a secondary goal, show author relationships. There was a small group of participants who liked both view types and had a hard time choosing one way or another.

Participants Who Chose "Papers"

The overwhelming majority of participants chose papers as their favorite type of views for the same reason: that it was easier to find relevant material. Participants said that they are more interested in overarching paper "topics" and knowing the author is not as important. Also, many participants said that they do not necessarily know "specific details" about authors, but they are knowledgeable in the field that they are searching more.

Therefore, searching papers and looking at related papers is makes more "sense" to most and it is "semantically much more rich" when one quickly navigates through a topic of research. Also, it is easier to search against papers rather than authors. It is simple to search by subject and papers which leads to finding relevant authors.

Participants Who Chose "Authors"

Although most participants chose papers, 13 percent chose authors. One participant said that the author views were interesting because it is possible to see who has been "collaborating" with whom. Also, the majority said that they thought both authors and papers were useful and that they both "have their uses" depending on the "kind of important" one is searching for.

11.5 PaperCube's Views In More Detail

In addition to rating the views together on a high level, a series of questions were asked that gave the participants an opportunity to rate the paper and author views individually. As mentioned in the last section, the most surprising result was that Tree Map view was the lowest rated paper view.

11.5.1 "Question 21: The usability/utility of the Paper Details view."

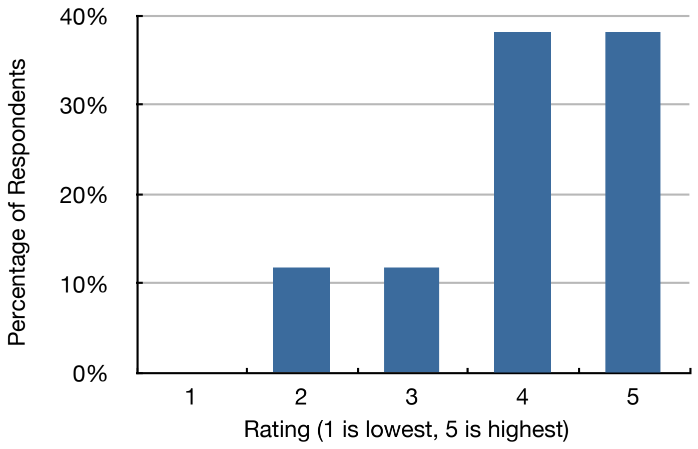


Figure 11.17: Q 21: The usability/utility of the Paper Details view.

The need for a view that serves as initial entry point for the other paper views was obvious. The best way to accomplish this was to create a view that is less visual and more list-based. Therefore this view, Paper Detail, contains lists of authors, references, and citations as well as other information about the paper. As shown in Figure 11.17, almost 80 percent of participants rated this view at least a 4 out of 5. The overall average rating for this view was 4 out of 5. Participants noted that the view is "very useful" and "easy to read." One said that it "captures all the relevant information."

Some participants thought that the interaction model in the lists was "jarring". When a user mouses over a paper or author in the list, additional information is revealed. One suggestion was to instead allow for a pop up containing meta data hovering over the item like in the other views. Also, the sorting in the lists were not clear to some participants, some saying that the options "num refs" and "num cites" were confusing. These sort options could have been spelled out more using a tool tip to alleviate potential confusing. Sorting tends to be useful when the lists are long, but with shorter lists, sorting is not as beneficial.

Also, with long lists, the screen space used is limited to a certain pixel height. Although the content will scroll inside the list, the list itself is a set height. The height chosen was made so that a typical laptop display would be able to show the whole list. However, if a larger display is used, there will be "blank space" on the page. Another issue that was noted that with papers with a lot of references or citations, the "Papers Per Year" graph on the page can become very crowded. Since it has a set height, it does not expand as there are more years to display.

11.5.2 "Question 22: The usability/utility of the Circle View."

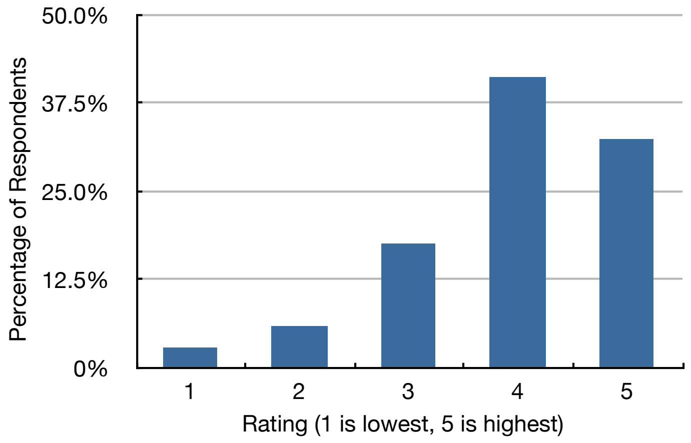


Figure 11.18: Q 22: The usability/utility of the Circle View.

Circle View was shown to be an interesting visualization method by people who tried the previous version from 2004. However, the visualization had not been validated through a study and this work took in account cognitive psychology research to improve upon the original concept. Therefore, the hope is that the view improved from the original. Figure 11.18 shows that the view was rated highly by the majority of participants with an average of 3.9 out of 5. In general, the participants found the view "very intuitive", "useful," "beautiful," and overall, aesthetically pleasing. Participants liked the "circle" metaphor for paper relationships and it gives the user a good "overall view" references and citations. In subsequent questions, many participants used this view as a point of comparison.

Some of the participants were confused with the author letter "abbreviations" used in the smaller circles that represent second level references or citations. The format used is a standard convention that displays the first letter of each author's last name or the first three letters if there is only one author. Since showing the author's name is better than nothing when screen space is limited, this display method was chosen so that the user can get some information about the paper. However, some participants were able to figure it out relatively easily and use the view without any other issues.

Also, others doubted the “usefulness” of the rotation feature. The rotation was added as a visual cue to show when a paper is put in “detail” position which allows more information to be displayed. However, the transition animation may have been unnecessary because it does not add much to the user experience beyond a context of how the visualization is being altered. In general, the use of animation cues in such a case is useful to the user but perhaps there might have been a better way to present it. The rotation speed could perhaps have been increased to make the user not wait as long for the animation to complete.

11.5.3 “Question 23: The usability/utility of the Paper Tree Map view.”

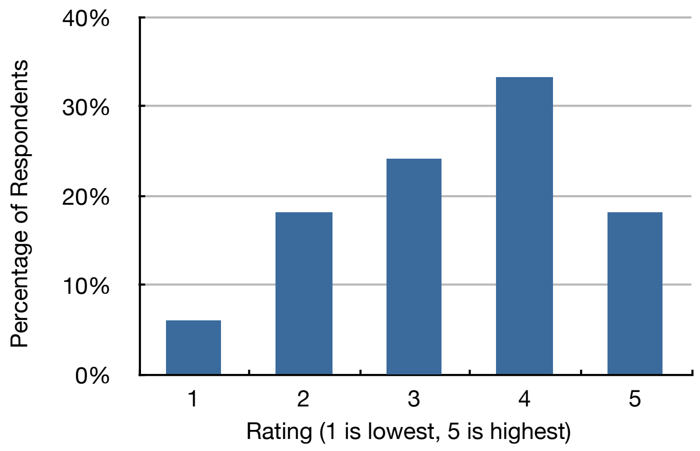


Figure 11.19: Q 23: The usability/utility of the Paper Tree Map view.

Tree Map view had the most surprising results of all the views in PaperCube. It was the hypothesis that the amount of bibliographic meta data which could be displayed would be compelling to the participants because it allows for a deep view into the citation network of a given paper. The motivation was to provide a lot of bibliographic meta data which can then be filtered down using the sliders. In Figure 11.19, it is clear that compared to the other views, this view is rated lower on average, with the majority of participants rating it on average 3.4 out of 5.

The comments are quite split between two groups. First, a group of participants thought that the information density was a good feature of the view. Participants who liked the view said that it is the “best way to show the impact of a particular paper because the amount of branching can quickly be grasped from a single glance" and that it is a "good way to see the depth of references." Also, one participant noted that the view was good to help "grasp really quickly how the topic of study has evolved and who are some of the most important researchers." Another participant liked the fact that the "route" to the focused paper is highlighted when the user mouses over a particular paper.

The second group thought that the view was too dense and was confusing as a result. Participants had a "hard time telling what the different levels" were and difficult to see "common references at the 3rd or 4th level." Also, several participants did not like the color scheme, calling it "garish" and thought it made the view "less appealing."

Also, the experience across browsers seemed to be an issue. Some browsers would "flash" and "flicker" during re-rendering, which is most evident when retrieving data from the server for additional levels of data. Although a known issue, it was not thought to be a major problem during development, and it was not expected that it would be one of the main issues that participants found. During testing, the rendering time was relatively quick for even large sets of data resulting in little to no flicker, but depending on the computer's speed and web browsers the experience may be an issue for some users.

Another thing that might have been useful would be to set the expectation of how many levels could be retrieved of a paper's citation network. For example, if a paper has no references or citations, the rendering engine will present the paper as one box taking up the whole browser window which is not very intuitive and useful, which in one case made a participant think the whole visualization was "highlighting" by mistake. Perhaps by giving the user an idea of how many levels can be retrieved would prevent potential confusion since the "depth" slider control can be set to a higher number than the actual number of levels that exist for a given paper.

11.5.4 "Question 24: The usability/utility of the Papers Per Year view."

Papers Per Year view was developed in order to allow users to trace the evolution of a domain starting from the focused paper by organizing the paper's citation network chronologically. This view was rated highly by the participants with an average rating of 4 out of 5.

Participants liked how the view was "chronological" and shows "how research progressed" in a field over the years. One participant said that it was a good way to see "time frame produced the bulk" of references or citations. Another thought of an interesting and unplanned use for this view; it could be "useful for tracking the progress of research programs and evaluate how grants are used."

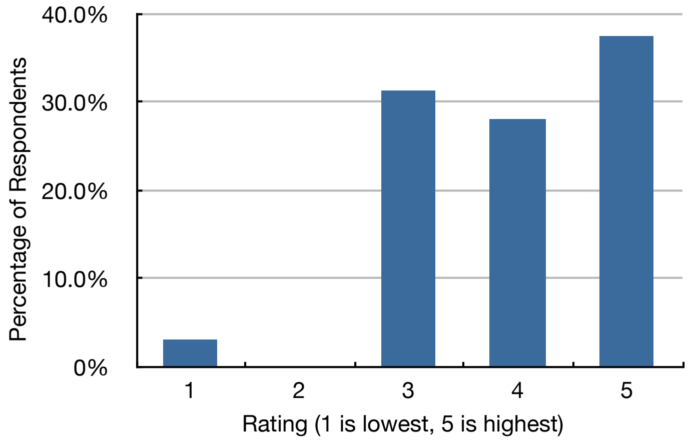


Figure 11.20: Q 24: The usability/utility of the Papers Per Year view.

One failing during the user survey was that the previously present legend or “key” explaining the color scheme for the connecting lines that references and citations for a paper were omitted as an oversight. Even though the colors were not explained, one participant said that he “loved” the layout and the lines. Therefore, some participants were rightfully in thinking that the line colors were “unclear,” but that seemed to be the only major issue with this view.

11.5.5 “Question 25: The usability/utility of the Paper Graph view.”

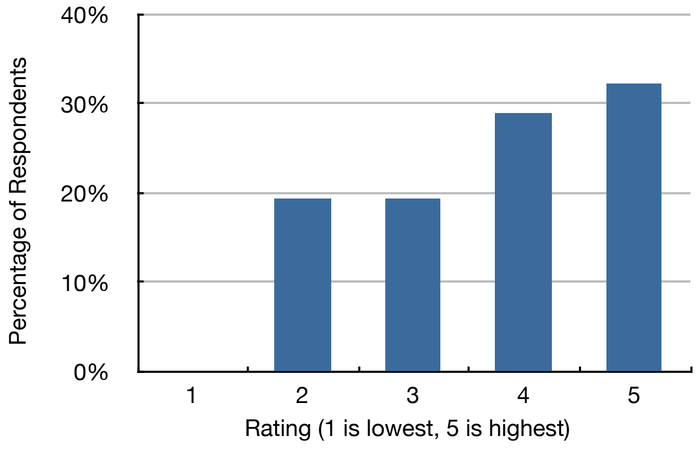


Figure 11.21: Q 25: The usability/utility of the Paper Graph view.

Paper Graph view uses the `NodeGraph` class to display a focused paper’s reference or citation relationships. Besides Tree Map view, this was rated lowest rated with an average of 3.7 out of 5.

The majority of comments by the participants seemed note that this view was "redundant" and unnecessary because the other views showed the data already. Especially, several participant wondered how this view was "different" to Circle View which was unexpected because they are very distinct in both design and functionality. As with all `NodeGraph`-based visualizations, this view can get "messy" when a lot of relationships are shown because all the edges will be drawn and may potentially cross other edges or nodes.

11.5.6 "Question 26: The usability/utility of the Author Detail view."

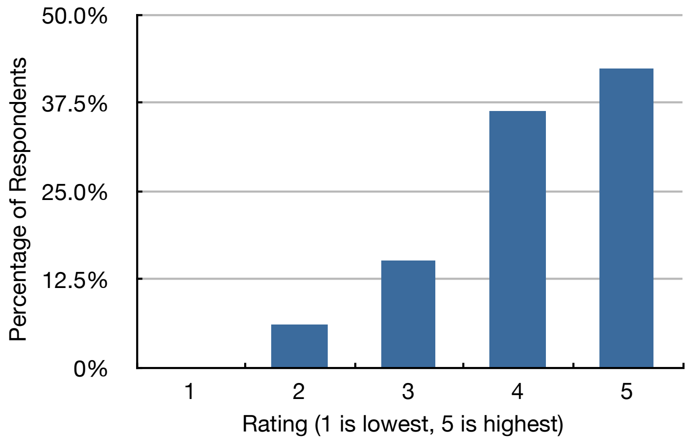


Figure 11.22: Q 26: The usability/utility of the Author Detail view.

Just as with viewing papers, having a view to serve as an initial entry point that showed author information in a more dense manner was necessary. Author Detail view was created to show bibliographic meta data about the author as well as lists of collaborators, published papers, authors referenced, and authors cited. Just as with Paper Detail view, this view was rated very high with an average rating of 4.2 out of 5.

As expected, most comments regarding this view were the same as for Paper Detail. Participants thought that it was a good way to get "started" on researching a topic and this view provided "a lot of information" on one screen. Also, this view was a good place to find work from "collaborating authors" and find out who are the "most active" researchers. Another participant thought that the graph showing the author's number of publications pear year was "really interesting."

One participant noticed a bug that occurred occasionally where not all the collaborators were retrieved. Also, one participant wanted this view have more color and "contrast" as well as show the number of papers on the top level in the lists so that user doesn't have to look at "each one."

11.5.7 "Question 27: The usability/utility of the Author's Papers view."

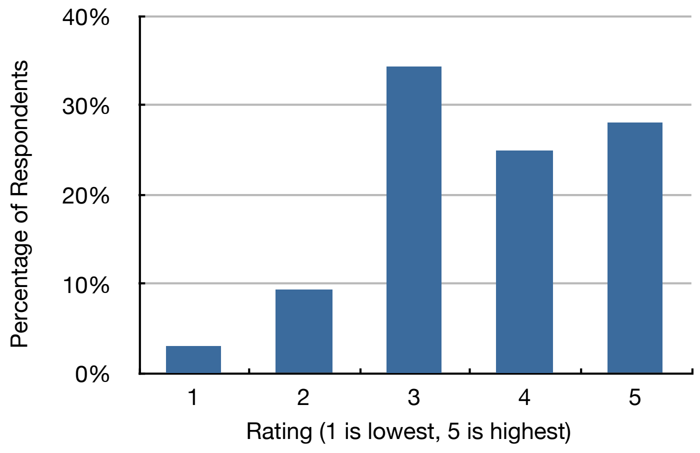


Figure 11.23: Q 27: The usability/utility of the Author Papers view.

Papers view was created as a way to show the paper's for the focused author visually. The idea was that organizing the papers around the author would give the user a way to quickly see what papers the person had written. However, the thought during development was that it did not add much to PaperCube as a whole, which was evident to the participants. The view was rated lower than most with a rating of 3.7 out of 5. However, there were some participants who thought the view was useful enough because it was a way to easily see what papers an author has published.

The general comments were that the Papers view did not add a lot of value to PaperCube and one participant commented that the view seemed "unfinished." Also, some participants found this view to be difficult to use when looking at author that has published a lot of papers because this view does not filter out any papers by default. Participants said that it is "unusable" with 300 or more papers and the smaller circles laid out around the author turns into a "huge circle."

Another suggestion made by a participant was to allow the user to reduce the amount of papers shown, perhaps using some kind of slider control filter the papers "based on time, such as only show someone's papers for the last 5 years." This method of filtering would be a very interesting addition to this view and would have made it a lot more useful.

11.5.8 "Question 28: The usability/utility of the Collaborators view."

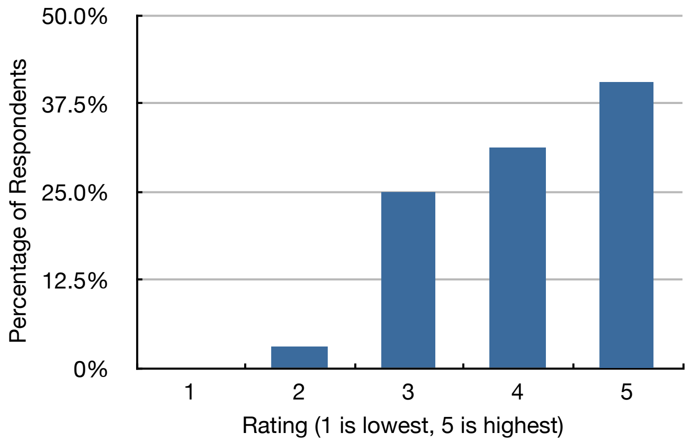


Figure 11.24: Q 28: The usability/utility of the Collaborators view.

As expected, Collaborators view was rated very highly by participants with an average rating of 4.1 out of 5. Most participants commented that this view was very useful to them. One participant said, "I really, really like these kinds of graphs" and another said "I really liked this." Participants said that it was a "good" and "quick" if not the "best" way to show relationships between authors and that the view shows the data in ways not available in "most databases."

However, again, as with most views, when there were a lot of items on the screen, it was cluttered, especially the "crossover lines" that signify relationship connections. The ability for the user to adjust the display thresholds makes this view very powerful because out of the clutter, important information can quickly be revealed. One participant wanted to be able to narrow down collaborators by "date published," something that was also touched upon in the comments of Papers view. The same participant was also confused about the sizing of the nodes in the graph saying it was unclear why their sizes varied. The reason for the varying sizes was purely for display purposes to optimize the rendering and attempt to eliminate possible overlapping nodes.

11.5.9 “Question 29: The usability/utility of the Author’s Cites view.”

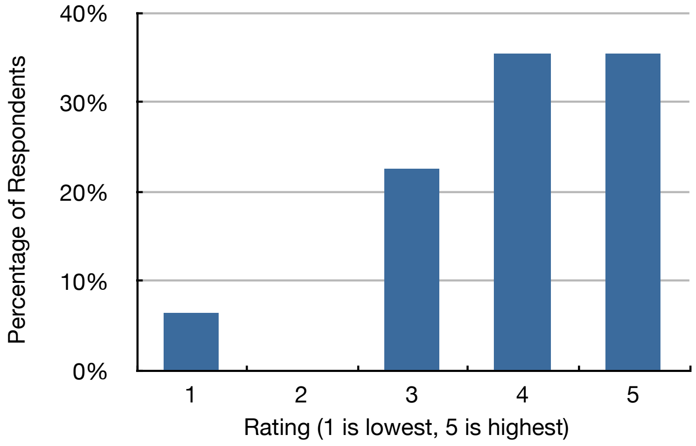


Figure 11.25: Q 29: The usability/utility of the Author’s Cites view.

Author Cites view is another `NodeGraph`-based visualization that allows the user to view the authors that he or she referenced or was cited by. This view was surprisingly rated higher than expected with an average of 3.9 out of 5. One participant said that this view was good because it allows the user to “know where the main researchers in the field are citing from.” The only major issue that participants highlighted was that it got “super hard” to read when there was a lot of items on the screen.

11.6 PaperCube’s Future Potential

The final set of questions was about the usefulness of PaperCube when it comes to augmenting paper and author search and what should be added or removed from PaperCube. Participants, regardless of the individual rating of the views and features thought that PaperCube would be a useful addition to existing web-based digital libraries. There were a lot of interesting comments regarding what features should be added or removed.

11.6.1 “Question 35: Did you find PaperCube useful when augmenting paper and author search?”

A majority of participants, 85 percent, found PaperCube useful when augmenting existing digital libraries. The remainder who answered “no” to this question, thought that it was not applicable for them because the data set was outside of their field of research.

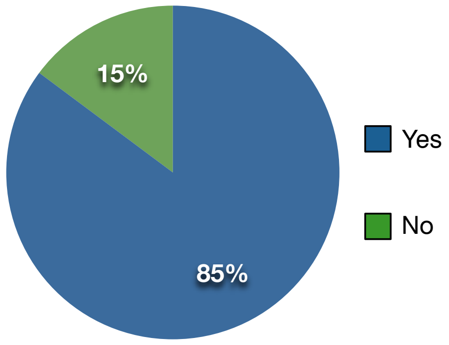


Figure 11.26: Q 35: Did you find PaperCube useful when augmenting paper and author search?

Participants thought that PaperCube makes searches "much, much easier" and it would be "useful" and "save a lot of time." One participant also thought it was good not only at showing references, which is the default mode, but also viewing the "citation tree" is "extremely helpful" because that way it is possible to know that "important papers" were not overlooked.

Also, participants appreciated that the PaperCube both lets users view papers as well as authors. Participants noted that the connections between authors and papers can be hard to "grasp" and PaperCube helps users to "discover links that usually get lost" when using user interfaces that are mostly text-based. The visualizations are "helpful" in terms to "mapping out relationships" between papers and authors. One participant said that although he starts by searching for papers by subject, PaperCube makes it "easy to find other papers by the same author and collaborating authors."

11.6.2 "Question 38: Would PaperCube's features be useful to include as part of a larger digital library service?"

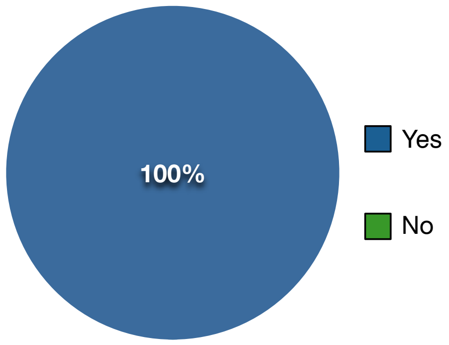


Figure 11.27: Q 38: Would PaperCube's features be useful to include as part of a larger digital library service?

All participants answered “yes” to this question. PaperCube would fit very well with a digital library service because most it helps reduce the “cognitive load” on the user by only showing relevant information. Also, showing author and paper relationships “visually” is a good way to increase the “discoverability” of new material. PaperCube is a “novel way” to view the relationships between papers in a way that is much more “understandable than the flat views” in most web-based digital libraries.

PaperCube would make “references searches much faster and complete” and as one participant said, it would be a “fantastic way to navigate and explore to find serendipitous connected papers/studies, and would make a particular library’s catalog that much more valuable.” Also, one participant especially liked the fact that the ability to filter the visible data was “much more extensive” that have other digital libraries allow.

Participants suggested that integrating with services such as CiteSeer, Nexis, or Westlaw would be “great” especially if the service gave access to the “full text” of papers. Although the requirements for digital libraries in law is different than engineering, enough overlap exists to make PaperCube be useful in both scenarios. No matter what type of papers are in the digital library, as one participant put it: “if you ever worked with a standard digital library, the two would make a perfect couple.” If the integration would result in a service that is “scalable and light weight,” PaperCube would be a good additional to a digital library. Also, due to the architecture of PaperCube, integration would be simple as long as there was a server interface layer that communicates with the client application using JSON.

11.6.3 “Question 36: What would you add?”

This question allowed the participants to comment on what they think should be added to PaperCube. The responses varied but fell into several categories, UI improvements, request for additional meta data, better search functionality, better navigation tools and saving features, and legends or keys to explain functionality.

User Interface Improvements

Quite a number of participants commented on what could be improved in terms of the user interface. First, there was a group that thought that the overall design of the UI could be improved. One participant thought that the visual design design of PaperCube was very “mathy-sciency” and that it needs to have a more “stronger, more pleasing, and cohered visual design” and another participants said that the “gray-on-gray lettering” was not very “user-friendly.” The aim was to make the controls and other elements be relatively transparent and not compete with the visualizations. Perhaps the subtleness of the UI was overdone. However, PaperCube could be improved quite easily without much effort by doing some minor visual design improvements.

Also, several participants suggested that there could be better use of color and the size of circles to indicate a varying set of importance cues to the user. For example, one participant suggested that the use of “the size of an author circle can indicate his prominence in the field, while the color indicates how old his body of work is, and a blue line shows that the focused author is citing another author, while a red line shows that the author is citing the focused author, and purple shows citing incest.” Some of this is done currently in PaperCube, but not to that extent. If such cues would be added to the visualizations, additional meaning could be implied by users. However, it would also require users to learn what the colors and sizes stand for prior to being able to take full advantage of the PaperCube application.

Another participant suggested that having a “subject” tree map view would be interesting if one could “cluster” papers into subjects and find subjects that “cite each other.” Many of the views in PaperCube already do some advanced calculations of the data in the back end. If the subject keywords are available in the data set, showing them as a visualization grouped around citations would not be an impossible task.

Additional Meta Data

As expected, several participants commented on the fact that they would like to see “additional” meta data such as the author’s “homepage” or “paper discussion blog.” Also, participants wanted additional papers from “non-peer-reviewed” sources as online technical articles or knowledge-bases.

Better Search Functionality

Although PaperCube’s goal was not to be a search solution, some participants wanted to see “advanced” search capabilities added. If PaperCube would integrate with a digital library, this search functionality would most likely be furnished by the service and not this application. However, PaperCube would greatly benefit from advanced searching functionality because the current is basic.

Improved Navigation and Saving Functionality

In earlier questions in the user survey, several participants suggested that having the history buttons be integrated with the "regular browser buttons" or have "larger" buttons. This integration would be relatively trivial and can be added easily.

Also, one participant noted the lack of a "save button" in Paper Detail view something that would be nice to add. When using this view, there is no way to save the focused paper. Also, the same participant suggested that being able to "separate out papers from different conferences" in order to organize the saved items list to fit the user's workflow instead having it be a flat list. If this was a feature, it would be easier to "catch up" on new papers before attending or submitting a paper to a conference.

Adding More Legends

Finally, some users suggested that having "key" or "a brief snippet of introductory text" on the views explaining what the visualization is representing. Also, having additional links to in-depth "help content" would be useful.

11.6.4 "Question 37: What would you remove?"

A relatively significant percentage of participants said that "nothing" needed to be removed from PaperCube. However, the participants who thought that certain features should be removed, fell into several categories. One group of participants wanted to remove certain views and another group wanted the number of views reduced. Also, there were some other participants who wanted to remove miscellaneous components.

Removing Certain Views

Five participants voiced their opinion that some views should be removed. Two participants wanted Tree Map view removed. One participant wanted Papers Per Year view removed and another suggested Papers view.

Furthermore, one participant felt that some views felt redundant and either "axe" one or the other or if they show different information, they should be "more distinct." The participant suggested, in order to remove redundancy, to remove either Circle View or Paper Graph view, as well as Author Cites view or Collaborators view. In PaperCube, all the views are distinct and offer different views into the bibliographic meta data although the views may look similar. This is especially true with the `NodeGraph`-based views, which all have the same color scheme. Perhaps using a different color scheme for each view would have made them feel more distinct.

Reducing The Number Of Views

A set of participants felt that PaperCube offer “too many options and views” which may “confuse the user” and let the user “concentrate” of a few ways to look at the information. One participant said that by consolidating the views and make them sub-modes instead, it would PaperCube “make it less about giving all the options at any time, but expose the ‘best’ view(s) depending on the current results and focus.”

Other Comments

Some other comments that participants had regarding what to remove included removing the “separation” or “distinction” between the reference or citation directions. Instead, the participant suggested that using “color coding” in order to “convey” both sets of meta data at the same time. Although tried in Papers Per Year view, many of the views would have become too complex if both directions would have been included. Therefore, most of the views use the distinction between the two modes in order to reduce complexity.

Also, another participant did not like the use of SVG. Instead, having a “HTML structure” with “standard contextual menus” would be better. Participants also suggested that removing the “animation” or “rotation” in some of the views would be good.

11.6.5 Additional Comments

Finally, participants were able to put down any additional comments regarding PaperCube. Participants really liked PaperCube and thought it was an “impressive use of SVG,” “extremely useful tool that’s both visually exciting and intuitive,” and “an amazing example of the apps that we’ll be seeing on the web in a couple of years.”

Furthermore, participants suggested additional elements to improve in PaperCube. One participants suggested modifying the “top toolbar” show the dependency between the “Mode” and “View” drop down menus as well as the distinction between the “Paper” and “Author” modes in a less subtle way. Another participant suggested that there needed to be some “refinement” regarding the user feedback on “hover” and “click.”

However, the majority of the comments were hinting on hopes for the future of PaperCube. As one participant put it, “wish I had this tool when I wrote my thesis” and another said that he could “spend a lot of time with this, if it were widely available!”

More focused commentary regarding the future of PaperCube were thoughts on integration with existing services, which was one of the initial goals of the work. One participant wanted to see a “more lightweight” version of PaperCube that could be “easily be transposed into existing citation management or digital library software.” Also, another participants would like to see PaperCube “be able to access as many databases as possible” because it would allow it to be a “one-stop literature search engine.”

11.7 Discussion

The user study showed that PaperCube has a lot of potential and the participants’ impression of it overall was excellent. The participants rated PaperCube highly when it comes to augmenting web-based digital libraries and all of the participants said that they would use it as part of a larger digital library service. Although not explicitly mentioned, the participants recognized that the goal for PaperCube was to be a potential companion to a full-featured digital library service.

In general, the survey validated top-level design choices of PaperCube. The decision to show both papers and authors views was well-liked by the participants and as expected, the paper views were found more useful than the author views. Also, the ability to adjust the display parameters of the views in real time was rated very highly. Furthermore, the zooming and overall feature of resolution independence was very well liked by the participants.

However, the fan menu control was rated lower than expected. The UI element was thought to be a good addition to PaperCube because it allows for quick navigation. Several participants thought that including a non-standard UI element was unnecessary.

The view choices for both papers and authors were well-liked with some exceptions. The Tree Map view was rated much lower than hypothesized. The view was thought to be one of the best views, but most participants thought that it showed too much information to be useful. The views that stood out as being the most well-liked views were Collaborators and Circle View along with the two landing page views, Paper Detail and Author Detail.

11.7.1 Possible Refinements

Based on the survey results, several possibilities to improve PaperCube are evident. First, improving the UI design to make it more intuitive would be very useful. Most of the improvements would be to make the color scheme a bit more pleasing and some of the UI elements stick out more. The thought was to make the UI fade away and allow for minimal distraction while the user is interacting with the various views. However, the subtleness of the interface made PaperCube have a higher learning curve to the novice user than expected. The participants noted that once the learning curve was overcome, PaperCube was easy to use.

One such example is the fact that the search pane was hidden on the left side of the screen. In order to lower the learning curve, having the search pane open by default would be a simple improvement and would result in an enhanced user experience. Many participants reported that it was difficult to figure out where to start once PaperCube loaded.

Another area on improvement in terms of user interaction is the fan menu. Although the fan menu was thought to be a bit difficult to use due to its non-standard layout, the majority of the issues stemmed from having appear on left click instead of right click. One improvement would be to use left click to select an element, instead of hover, and show the menu on right click like a contextual menu in desktop applications.

Furthermore, participants thought that instead of having more views, PaperCube might benefit from having less views but with more features. By having more targeted views, PaperCube could be a lot more focused and easier to use. Not all of the views were as useful as they could have been. One prime example is the PAPERS view showing all the papers that an author has published. That view could be removed and improve PaperCube as a whole by making it more focused.

11.7.2 Potential Extensions

The PaperCube as validated in the survey is a self-contained application using a static data set. The participants noted that they would like to see PaperCube interact with a live data set. Since PaperCube is web-based and communicates using JSON, it would not be difficult to integrate directly with an publicly available digital library such as CiteSeerX. The interaction model with a live digital library could manifest itself it two ways.

First, we could give PaperCube more search capabilities to search the digital library directly, and have it exist as a stand-alone service. This would have the benefit of being self-contained and only the interface to the digital library would have to be developed. However, the rich capabilities of an existing web-based digital library would have to be duplicated in PaperCube which might distract from its central goal, namely focus on the visualization of bibliographic meta data. Adding additional features to make it a full interface to a digital library would be a distraction.

Second, one could make PaperCube even more lightweight and incorporate it into a web-based digital library directly. The idea is that when a user views a paper in a digital library a link would be present to show that paper in PaperCube. From PaperCube, the user would be able to navigate and view papers and authors like in the existing application. The second interaction model with a digital library might be the most desirable and effective. PaperCube was designed to augment existing digital libraries through the use of visualizations, not replace them.

Another possible extension to PaperCube would be to expand beyond viewing papers and authors and explore the visualization of other information with relationship links. At a demo of PaperCube, the suggestion to use the visualization framework to show the relationships between knowledge base articles was brought up. Also, other information could be visualized quite easily, especially using the `NodeGraph` class developed as part of PaperCube.

11.7.3 Incorporating Feedback

Based on the valuable feedback from the user study there are several things that can be incorporated to improve the user experience of PaperCube and increase its effectiveness. The following is a list of the feedback that is planned to be incorporated:

- Reveal search pane on page load.
- Vary the color scheme of visualizations to make views more distinct.
- Alter the fan menu to appear on right click, and left click selects an item in a visualization.
- Add tool tips on icons and buttons to give users more information about functionality.
- Change iconography to be more descriptive.
- Incorporate PaperCube's custom history management to use the web browser's built-in history object.
- Make the Paper view more useful by adding the ability to filter per year.
- Show introductory video on first access to PaperCube to introduce new users to it.

There are other items of feedback that could be incorporated but would most likely be too difficult to do with a lot of additional effort. One feature that would be interesting to add is the ability to change a set display threshold parameters using only one slider control. However, this would be quite difficult because it would involve an in-depth investigation regarding how to combine the various thresholds into a single threshold, yet have them all retain their original meaning.

Chapter 12

Related Work

This chapter covers some of the past research into the visualization and navigation of digital libraries. The selection of research covered are mainly the most influential to the author of this work and may not be exhaustive. The scope of this work may be more narrow than some of the other research covered, but they still have served as an inspiration. In the past 20 years, computers have become fast enough to efficiently be able to render large data sets and the advent of the Internet has made it possible to access vast quantities of information quickly.

12.1 The Document Lens

Robertson and Mackinlay (1993) describe a specialized visualization method showing papers. Papers are laid out on a virtual table and the user uses a rectangular fisheye lens to focus on a certain document while remaining aware of where on the table the user is focusing on. The presentation outside of the lens is stretched but still readable enough to know where on the table one is spatially located.

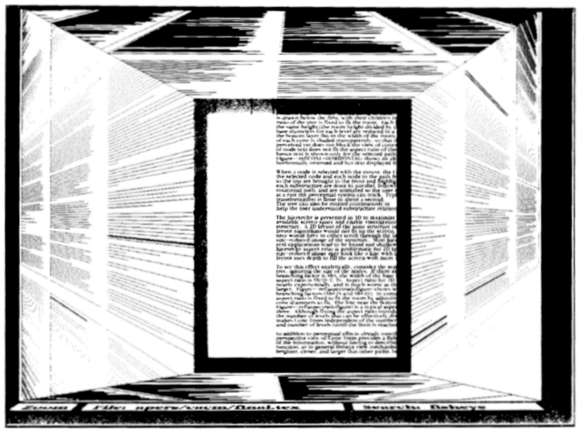


Figure 12.1: Screenshot of the Document Lens concept.

Within the context of this work, Circle View and to a limited extent, the `NodeGraph`-based views is also a kind of fisheye view but rather than using a rectangular viewing context, it uses circles to accomplish something similar. However, one significant difference is that the Document Lens does not show relationships, but rather allows the user to browse a large amount of unstructured information.

12.2 An Organic User Interface For Searching Citation Links

Mackinlay, Rao, and Card (1995) created an application named Butterfly to browse digital libraries. Although created right after the inception of the world wide web, no widely used web-based digital libraries existed at the time. Therefore, Butterfly was a desktop-based application that provided access to DIALOG Science Citation databases available through the Internet at the time. Since Butterfly was meant to serve as the primary interface to a digital library, it had rich search capabilities and focused a lot of the query interface to DIALOG-based databases and displaying the results of searches.

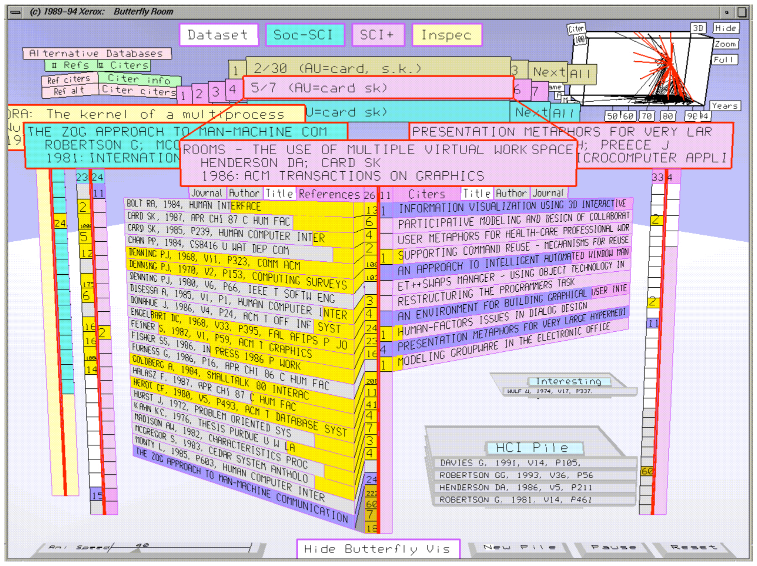

(c) 1989–94 Xerox: Butterfly Room
Dataset
Soc-SCI
SCI+
Inspec
Alternative Databases
# Refs
# Citers
Ref citers
Citer info
Ref alt
Citer citers
2/30 (AU=card, s.k.)
5/7 (AU=card sk)
Next
All
3D
Hide
Zoom
Full
Years
THE ZOG APPROACH TO MAN-MACHINE COM
ROOMS - THE USE OF MULTIPLE VIRTUAL WORK SPACE
HENDERSON DA; CARD SK
1986: ACM TRANSACTIONS ON GRAPHICS
PRESENTATION METAPHORS FOR VERY LAR
Journal
Author
Title
References
Citers
Interesting
HCI Pile
DAVIES G. 1991, V14, P105
ROBERTSON GG. 1993, V36, P56
HENDERSON DA. 1986, V5, P211
ROBERTSON G. 1981, V14, P461
Hide Butterfly Vis
New Pile
Pause
Reset

Figure 12.2: Screenshot of the Butterfly UI showing the "wings" and "veins" of bibliographic meta data.

At the core of Buttefly's UI, there is the "butterfly," which is a three dimensional representation of a paper's bibliographic meta data. One "wing" was used to list an articles references and the other "wing" was used to list the article's citers. The three dimensionality of the application was used to support rapid navigation between articles. Although the Butterfly application was 3D-based, the UI only supported the listing of 22 "veins" that represent references or citations for legibility reasons. Color coding is used to provide additional information to the user; darker red if the article has already been visited, purple if directly visited, yellow indicates the number of citers. Butterfly uses the idea that articles with many citers is probably a good article to highlight to the user, a concept that is used in PaperCube's visualizations.

12.3 BIVTECI: A Bibliographic Visualization Tool

The following year Modjeska, Tzerpos, Faloutsos, and Faloutsos (1996) created BIVTECI to address the need for an interactive bibliographic visualization tool (BVT) that allows users to access the complete bibliographic data for an article, filter by title, author, and keyword, ordered citation links, information views at several levels of detail, multiple synchronized views, and the visualization of large data sets. PaperCube shares some of goals of BIVTECI very closely but aims to be entirely web-based, something that was not possible at the time that BIVTECI was conceived.

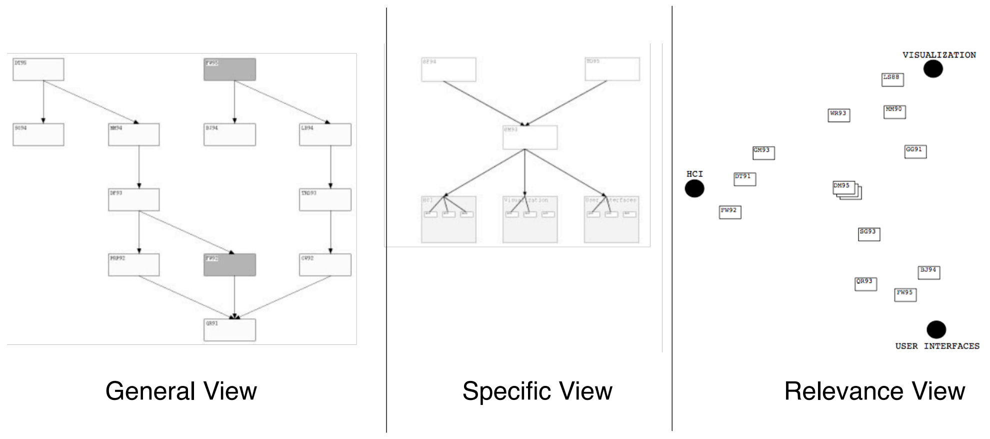


Figure 12.3: Screenshot of BIVTECI from the paper.

BIVTECI provided three distinct views of articles. First, there was the "General View" that allowed the user to visualize a large portion of a database based on search criteria. Second, there was the "Specific View" that presented the relations for a specific article showing its citers and citees. Lastly, there was the "Relevance View" that provided a view of various bibliographic attributes in a non-hierarchical manner where items were laid out spatially based on relevance.

12.4 Visualizing the Evolution of a Subject Domain: A Case Study

Chen and Carr (1999) explored using Pathfinder-based visualization techniques to enhance contemporary methodologies, such as author co-citations, that serve as ways to analyze a research domain. In particular, the goal was to show the evolution of the particular research field, in this case hypertext, by visualizing citation and author co-citation relationships. The findings of their case study showed that expected authors who were very significance in the hypertext field were clearly highlighted and clustered around authors including Mackinlay and Robertson.

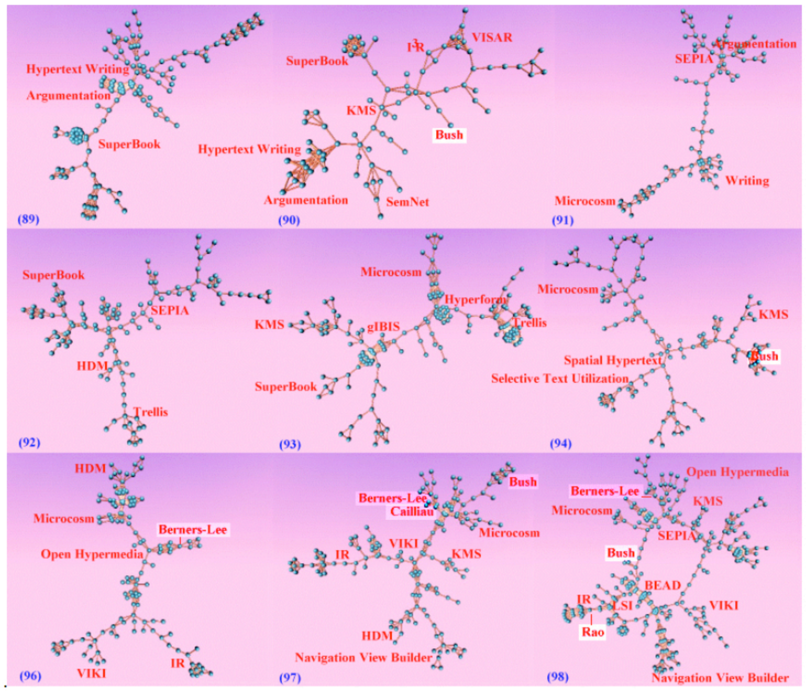
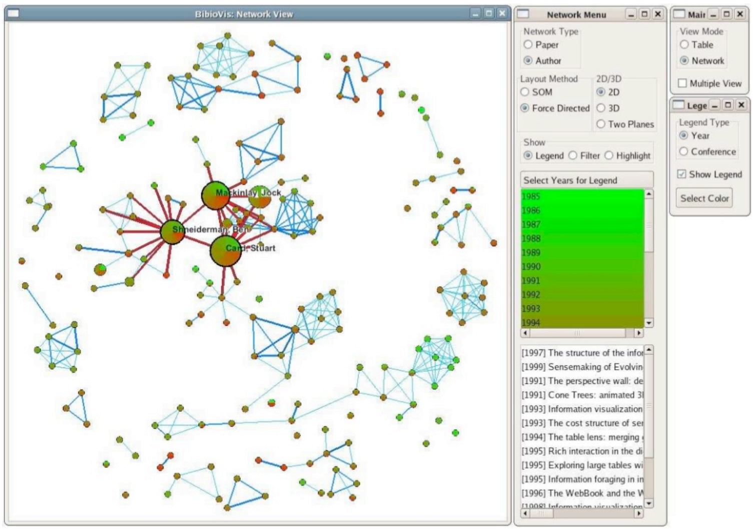


Figure 12.4: Screenshot of an author co-citation network from Chen and Carr's paper.

Although PaperCube does not visualize author co-citation relationships, it takes the approach of looking at papers chronologically and making it very easy to view the collaboration networks of authors to figure out who the significant authors are when looking at meta data. Therefore, it is possible to find some of the same relationships through different means. However, PaperCube would greatly benefit from being able to visualize co-citation networks, something that was not part of the initial goals of application.

12.5 BiblioViz: A System for Visualizing Bibliography Information

BiblioViz is a desktop application that was created by Shen, Ogawa, Teoh, and Ma (2006) that took the InfoVis 2004 contest winner's solutions and combined them into a larger tool. The resulting application uses an array of different visualization methods to show bibliographic meta data. BiblioViz consists of two highly customizable methods of laying out the data: a Table View and a Network View. The Table View allows the user to choose what meta data should be used for filtering then the information that should be shown on the x- and y-axis. Using the focus+context technique, when a cell in the table is selected, it is enlarged to reveal more information. The Network View on the other hand is a graph visualization suite that supports many different graph layout methods as well as 2D and 3D rendering modes to allow the user the flexibility to view the bibliographic meta data in as much ways possible.



Network Type: Paper [✓ Author]
Layout Method: SOM [✓ Force Directed]
2D/3D: [✓ 2D] 3D Two Planes
Show: [✓ Legend] Filter Highlight
Select Years for Legend
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994

[1997] The structure of the infor
[1999] Sensemaking of Evolvin
[1991] The perspective wall: de
[1991] Cone Trees: animated 3
[1993] Information visualization
[1993] The cost structure of se
[1994] The table lens: merging
[1995] Rich interaction in the di
[1995] Exploring large tables wi
[1995] Information foraging in in
[1996] The WebBook and the W

View Mode: Table [✓ Network]
Multiple View
Legend Type: [✓ Year] Conference
[✓ Show Legend]

Figure 12.5: Screenshot from BiblioViz showing that authors Card and Mackinlay are closely related to Shneiderman.

One of the primary goals of PaperCube is to give the user the flexibility to determine the amount of information that is visible at any time much like BiblioViz allows. Although PaperCube only supports two dimensional graph rendering due to its web-based nature, it still uses various graph layout methods to show the same bibliographic meta data in multiple ways.

12.6 CiteWiz: A Tool for the Visualization of Scientic Citation Networks

Elmqvist and Tsigas (2004) developed CiteWiz, an application that allows for the extensive visualization of citation networks. Instead of using a more basic node-link visualization method, CiteWiz uses "Growing Polygons" technique to present the citation network and show chronology and influences.

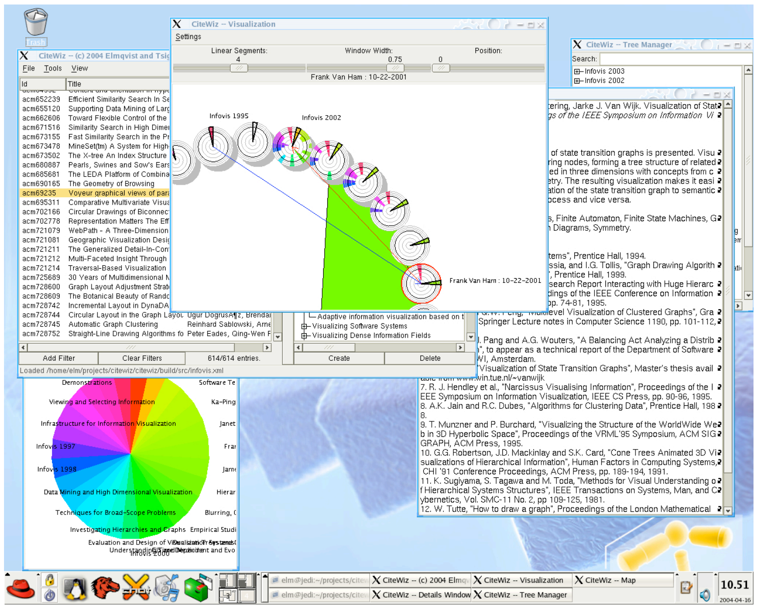


Figure 12.6: Screenshot of the CiteWiz application.

While only one visualization method was developed, the goal for CiteWiz was also to create a framework in which additional visualization techniques could be employed to show additional dimension on the data, something that is closely aligned with the goals of PaperCube itself.

12.7 Understanding Research Trends in Conferences using PaperLens

Lee, Czerwinski, Robertson, and Bederson (2005) developed PaperLens, a digital library visualization application that allowed a user to easily see citation and collaboration connections throughout a community of researchers. The UI showed statistical information in order to reveal connections that would not be obvious in a traditional digital library. The application enabled the user to view bibliographic meta data in several different views at once. The most interesting views included were the "popularity by topic" view, "year by year top 10 cited" view, and "degrees of separation" view.

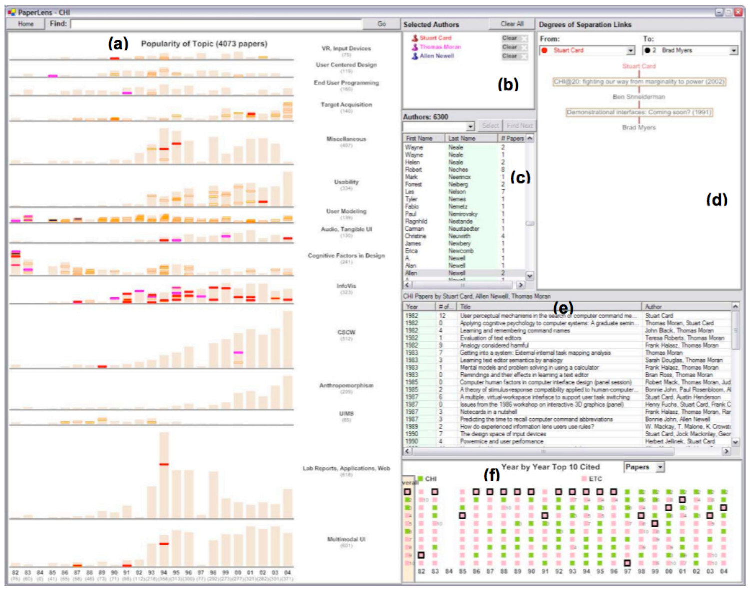

PaperLens - CHI
(a)
Popularity of Topic (4073 papers)
(b)
Selected Authors
(c)
Authors: 6300
(d)
Degrees of Separation Links
(e)
(f)
Year by Year Top 10 Cited

Figure 12.7: Screenshot of the PaperLens application.

The work also included a user study that showed two key things. First, that keeping the user interface simple is key to keep the user from being overwhelmed and that the meta data, namely trends, were not suited for using complex graph visualizations. Second, the study showed that the visualization method that used tiled squares to represent papers and with a fish eye lens skew did not scale very well with large data sets because the individual papers were too small on the screen.

The some of the goals of PaperLens was similar to PaperCube but is mainly focused on showing trends is a research conference ecosystem while PaperCube looks directly at the citation relationships between papers. The most similar aspect of PaperLens is that it allows the user to see collaboration relationships in the "degrees of separation" view. PaperCube's Collaborators view focuses on one author and allows the user to see direct collaborations with other authors. In contrast, PaperLens allows the user to select a set of authors that may be separated by many hops and visualizes the relationships between them.

Chapter 13

Conclusion

This work set out to develop and test the effectiveness of an application that allows a scholar to interact with a digital library and explore bibliographic meta data using a defined set of visualizations. This application, PaperCube, was designed to augment—not replace—existing digital library services. PaperCube uses a set of visualizations to allow a scholar to be focused on a paper yet see where it belongs in the greater context of a publication space. Current digital libraries are limited by the webpage-based paradigm accepted as the norm. PaperCube set out to push the limits of web browsers and see if it was possible to break the mold and use a new paradigm to navigate bibliographic meta data. This new paradigm went beyond and created a dynamic, desktop-like experience that incorporated rich, interactive visualizations.

Borrowing its rich interaction model from desktop applications, PaperCube allows the user to interact with a fluid user interface and not be confined by any of the traditional limitations of the web browser. This interaction model was well received by the study participants. Written using the SproutCore JavaScript framework and leveraging only standards-based rendering technologies such as Scalable Vector Graphics, Canvas tag, HTML and CSS, PaperCube showed both paper and author relationships with a set of views. Paper Detail, Author Detail, Circle View, Tree Map are HTML-based. Circle View does use the Canvas tag to a limited extent. The remaining views, Papers Per Year, Collaborators, Papers, and Author Cites are purely SVG-based.

Leveraging the powerful features in SproutCore including bindings and observers, PaperCube is able to provide a high level of interactivity not commonly available in a web browser. Bindings and observers enabled PaperCube to easily implement resolution independence. By adjusting a slider control in the UI, a visualization can be zoomed in using SVG transforms or in the case of HTML-based views, using CSS without having to redraw. Furthermore, through the use of bindings, slider controls dynamically adjust the display threshold parameters of the views, permitting the depth of a paper citation network to be changed dynamically. Also, customizable thresholds are used to determine if a paper or author should be rendered in the visualizations.

Stemming from the need for a flexible graph API for various views in PaperCube, the SVG-based `NodeGraph` class was created as a generalized solution that can display any type of relational data as an undirected graph. The class is not PaperCube-specific, and there has been some interest in integrating it into other SproutCore-based web applications.

The user study validated that PaperCube's set of visualizations achieve the goal of augmenting digital libraries effectively. Participants unanimously said that PaperCube would fit well within the framework of an existing digital library service and that if available, they would use it. Although the data set used focused on papers in the engineering and scientific fields, participants from other disciplines such as law found that with the right data set, PaperCube would greatly improve their daily work.

The effectiveness of PaperCube stems from the views of the bibliographic meta data that are exposed to the user. Although showing only direct relationships between papers and authors, the types of visualizations used were in general thought to be effective. Users could focus on a paper or author at the low level or explore the networks of relationships up to fifteen levels deep.

The user survey showed that participants liked to see the details of a paper or author first, then branch out using the other views starting with the ones showing the most immediate relationships then going beyond. Therefore, Circle View and Collaborators view, visualizations that were more constrained in scope, were rated the highest by participants. Views such as Papers Per Year, Tree Map, and Author Cites were not rated as highly because participants thought that the potential data density was overwhelming at first.

However, participants found that PaperCube as a whole helped reduce the cognitive load of a researcher by making it easy to focus and show relevant information. All the views gave the user the control to adjust the amount of data shown at any time by a set of parameters. Especially in data dense views, this control made it easy to determine what papers or authors are important by altering significance factors. Therefore, instead of trying to determine what is relevant to a user, PaperCube trusts the user to determine relevance by himself or herself.

Furthermore, the study showed that PaperCube made it possible to find references, citations, and authors that are not directly connected with the focused paper or author through the ability of seeing many levels of relationships at once. Therefore, a researcher can stumble upon important papers and authors that may be seem unrelated to the direct search at hand, yet indirectly be at the heart of the researcher's search.

Based on the feedback from the user study, several areas of improvement were identified that relate to PaperCube's usability. Participants said that PaperCube was difficult to use at first until they learned how the application worked. The changes identified include changing the history management, icons, colors, interaction model with the fan menu, revealing the search pane by default, and adding the possibility to filter by year in the author PAPERS view. By making these minor adjustments to PaperCube, it will be come more intuitive to novice users.

This work showed that augmenting the foraging for research material in digital library discover through visualizations on the web is a possibility. PaperCube was shown to be to reduce the "cognitive load" put on a scholar and aid the "discoverability" of new research material. Furthermore, it was shown that participants thought that it was "visually exciting and intuitive" application and an "amazing example of the apps that well be seeing on the web in a couple of years."

Appendix A

User Survey Questions

A.1 Page 1. PaperCube User Survey - Introduction

Thank you for taking the time to take our user experience survey!

We want to know what past or present professors and graduate students think about PaperCube so that we can assess its effectiveness at allowing users browse and explore digital libraries. Also, we want to find out what we can do to make it better. If possible, please complete the survey before March 9th.

Before you begin, please view this brief introduction video that describes how to use PaperCube: http://www.vimeo.com/3323956

You can access the PaperCube demo application here: http://papercube.peterbergstrom.com

Demo mirror: http://concerto.engr.scu.edu/ pbergstr

Also, if you want more in depth information about the application and its general goals, please check out http://www.peterbergstrom.com

Let's get started with the survey!

A.2 Page 2. Some basic information about yourself

Question 1: Name

Open ended answer box.

Question 2: Affiliation

Open ended answer box.

Question 3: Which describes you best? (required)

Required multiple choice question with choices:

( ) Professor
( ) Graduate Student
( ) Undergraduate
( ) Working professional with an undergraduate degree
( ) Working professional with an graduate degree
( ) Other

A.3 Page 3. Your research

Please tell us a little about your research background and your research habits.

Question 4: What is your field of study?

Open ended answer box.

Question 5: On a scale from 1 to 5, 5 being the most, how familiar are you with the literature of your field of research or study?

Multiple choice question with choices:

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 6: Do you read journals and periodicals in your field on a regular basis?

Yes/No question.

( ) Yes
( ) No

Question 7: Do you use the web to find papers on a regular basis?

Yes/No question with optional box for comments.

( ) Yes
( ) No

Question 8: Do you find it easy to find related work and new research?

Yes/No question.

( ) Yes
( ) No

Question 9: What digital libraries do you use?

Open ended answer box.

Question 10: What do you like about them?

Open ended answer box.

Question 11: What do you dislike about them?

Open ended answer box.

A.4 Page 4. PaperCube Feature Impressions

Please rate the following next section of questions from 1 to 5, where 5 is the most favorable/good. There is some space for additional comments but this is not required or expected.

Question 12: What is your impression of the general UI of PaperCube?

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 13: History and navigation buttons.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 14: The saving functionality.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 15: The zooming and panning widget.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 16: The general feature of resolution independence in PaperCube.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 17: The UI paradigm for the pop up pie menus.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 18: The available options in the fan menus.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 19: The animation transitions used in certain views.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 20: The ability to tweak the settings of the visualizations using slider controls.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

A.5 Page 5. Rate PaperCube's views

Please rate the following next section of questions from 1 to 5, where 5 is the most favorable/good. There is some space for additional comments but this is not required or expected.

Question 21: The usability/utility of the Paper Details view.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 22: The usability/utility of the Circle View.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 23: The usability/utility of the Paper Tree Map view.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 24: The usability/utility of the Papers Per Year view.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 25: The usability/utility of the Paper Graph view.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 26: The usability/utility of the Author Detail view.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 27: The usability/utility of the Author Papers view.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 28: The usability/utility of the Author Papers view.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

Question 29: The usability/utility of the Author's Cites view.

Multiple choice question with optional box for comments.

( ) 1
( ) 2
( ) 3
( ) 4
( ) 5

A.6 Page 6. Your ideas about PaperCube

Please answer these questions to give us an impressions of your ideas regarding PaperCube.

Question 30: What paper view did you like the MOST?

Multiple choice question with optional box for comments.

( ) Paper Detail
( ) Circle View
( ) Tree Map
( ) Papers Per Year
( ) Paper Graph

Question 31: What paper view did you like the LEAST?

Multiple choice question with optional box for comments.

( ) Paper Detail
( ) Circle View
( ) Tree Map
( ) Papers Per Year
( ) Paper Graph

Question 32: What paper view did you like the MOST?

Multiple choice question with optional box for comments.

( ) Author Detail
( ) Author Papers
( ) Collaborators
( ) Author's Cites

Question 33: What paper view did you like the LEAST?

Multiple choice question with optional box for comments.

( ) Author Detail
( ) Author Papers
( ) Collaborators
( ) Author's Cites

Question 34: Which type of views were most useful for you?

Multiple choice question with optional box for comments.

( ) Papers
( ) Authors

Question 35: Did you find PaperCube useful when augmenting paper and author search?

Yes/No question with optional box for comments.

( ) Yes
( ) No

Question 36: What would you add to PaperCube?

Open ended answer box.

Question 37: What would you remove from PaperCube?

Open ended answer box.

Question 38: Would PaperCube's features be useful to include as part of a larger digital library service?

Yes/No question with optional box for comments.

( ) Yes
( ) No

Question 39: Any other comments?

Open ended comment box.

Appendix B

Quantitative User Survey Results

B.1 Page 2. Some basic information about yourself

Q1. Name

Answer Options	*Response Count*
	39
answered question	39
skipped question	4

Q2. Affiliation

Answer Options	*Response Count*
	32
answered question	32
skipped question	11

Q3. Which describes you best? (required)

Answer Options	*Response Percent*	*Response Count*
Professor	4.70%	2
Graduate Student	23.30%	10
Undergraduate	2.30%	1
Working professional with an undergraduate degree.	14.00%	6
Working professional with a graduate degree.	51.20%	22
Other	4.70%	2
	answered question	43
	skipped question	0

Q4. What is your field of study?

Answer Options	*Response Count*
	39
answered question	39
skipped question	4

B.2 Page 3. Your research

Q5. On a scale from 1 to 5, 5 being the most, how familiar are you with the literature of your field of research or study?

Answer Options	*Response Percent*	*Response Count*
1	12.80%	5
2	20.50%	8
3	33.30%	13
4	25.60%	10
5	7.70%	3
	answered question	39
	skipped question	4
	average	2.94871794871795

Q6. Do you read journals and periodicals in your field on a regular basis?

Answer Options	*Response Percent*	*Response Count*
Yes	43.60%	17
No	56.40%	22
	answered question	39
	skipped question	4

Q7. Do you use the web to find papers on a regular basis?

Answer Options	*Response Percent*	*Response Count*
Yes	66.70%	26
No	33.30%	13
Comments		12
	answered question	39
	skipped question	4

Q8. Do you find it easy to find related work and new research?

Answer Options	*Response Percent*	*Response Count*
Yes	51.30%	20
No	48.70%	19
	answered question	39
	skipped question	4

Q9. What digital libraries do you use?

Answer Options	*Response Count*
	35
answered question	35
skipped question	8

Q10. What do you like about them

Answer Options	*Response Count*
	34
answered question	34
skipped question	9

Q11. What do you dislike about them?

Answer Options	*Response Count*
	34
answered question	34
skipped question	9

B.3 Page 4. PaperCube Feature Impressions

Q12. What is your impression of the general UI of PaperCube?

Answer Options	*Response Percent*	*Response Count*
1	2.90%	1
2	8.60%	3
3	14.30%	5
4	60.00%	21
5	14.30%	5
Comments		26
	answered question	35
	skipped question	8
	average	3.74285714285714

Q13. History and navigation buttons.

Answer Options	*Response Percent*	*Response Count*
1	0.00%	0
2	15.20%	5
3	36.40%	12
4	27.30%	9
5	21.20%	7
Comments		23
	answered question	33
	skipped question	10
	average	3.54545454545455

Q14. The saving functionality.

Answer Options	*Response Percent*	*Response Count*
1	10.30%	3
2	6.90%	2
3	20.70%	6
4	34.50%	10
5	27.60%	8
Comments		22
	answered question	29
	skipped question	14
	average	3.62068965517241

Q15. The zooming and panning widget.

Answer Options	*Response Percent*	*Response Count*
1	5.90%	2
2	8.80%	3
3	26.50%	9
4	32.40%	11
5	26.50%	9
Comments		19
	answered question	34
	skipped question	9
	average	3.64705882352941

Q16. The general feature of resolution independence in PaperCube.

Answer Options	*Response Percent*	*Response Count*
1	10.70%	3
2	0.00%	0
3	3.60%	1
4	14.30%	4
5	71.40%	20
Comments		13
	answered question	28
	skipped question	15
	average	4.35714285714286

Q17. The UI paradigm for the pop up pie menus.

Answer Options	*Response Percent*	*Response Count*
1	6.50%	2
2	16.10%	5
3	29.00%	9
4	25.80%	8
5	22.60%	7
Comments		24

	answered question	31
	skipped question	12
	average	3.41935483870968

Q18. The available options in the fan menus.

Answer Options	*Response Percent*	*Response Count*
1	6.10%	2
2	6.10%	2
3	30.30%	10
4	30.30%	10
5	27.30%	9
Comments		18

	answered question	33
	skipped question	10
	average	3.66666666666667

Q19. The animation transitions used in certain views.

Answer Options	*Response Percent*	*Response Count*
1	3.00%	1
2	6.10%	2
3	21.20%	7
4	30.30%	10
5	39.40%	13
Comments		18

	answered question	33
	skipped question	10
	average	3.96969696969697

Q20. The ability to tweak the settings of the visualizations using slider controls.

Answer Options	*Response Percent*	*Response Count*
1	0.00%	0
2	9.40%	3
3	15.60%	5
4	21.90%	7
5	53.10%	17
Comments		18
	answered question	32
	skipped question	11
	average	4.1875

B.4 Page 5. Rate PaperCube's views

Q21. The usability/utility of the Paper Details view.

Answer Options	*Response Percent*	*Response Count*
1	0.00%	0
2	11.80%	4
3	11.80%	4
4	38.20%	13
5	38.20%	13
Comments		22
	answered question	34
	skipped question	9
	average	4.02941176470588

Q22. The usability/utility of the Circle View.

Answer Options	*Response Percent*	*Response Count*
1	2.90%	1
2	5.90%	2
3	17.60%	6
4	41.20%	14
5	32.40%	11
Comments		17
	answered question	34
	skipped question	9
	average	3.94117647058824

Q23. The usability/utility of the Paper Tree Map view.

Answer Options	*Response Percent*	*Response Count*
1	6.10%	2
2	18.20%	6
3	24.20%	8
4	33.30%	11
5	18.20%	6
Comments		22

	answered question	33
	skipped question	10
	average	3.39393939393939

Q24. The usability/utility of the Papers Per Year view.

Answer Options	*Response Percent*	*Response Count*
1	3.10%	1
2	0.00%	0
3	31.30%	10
4	28.10%	9
5	37.50%	12
Comments		13

	answered question	32
	skipped question	11
	average	3.96875

Q25. The usability/utility of the Paper Graph view.

Answer Options	*Response Percent*	*Response Count*
1	0.00%	0
2	19.40%	6
3	19.40%	6
4	29.00%	9
5	32.30%	10
Comments		11

	answered question	31
	skipped question	12
	average	3.74193548387097

Q26. The usability/utility of the Author Detail view.

Answer Options	*Response Percent*	*Response Count*
1	0.00%	0
2	6.10%	2
3	15.20%	5
4	36.40%	12
5	42.40%	14
Comments		10
	answered question	33
	skipped question	10
	average	4.15151515151515

Q27. The usability/utility of the Author Papers view.

Answer Options	*Response Percent*	*Response Count*
1	3.10%	1
2	9.40%	3
3	34.40%	11
4	25.00%	8
5	28.10%	9
Comments		12
	answered question	32
	skipped question	11
	average	3.65625

Q28. The usability/utility of the Author Collaborator view.

Answer Options	*Response Percent*	*Response Count*
1	0.00%	0
2	3.10%	1
3	25.00%	8
4	31.30%	10
5	40.60%	13
Comments		13
	answered question	32
	skipped question	11
	average	4.09375

Q29. The usability/utility of the Author's Cites view.

Answer Options	*Response Percent*	*Response Count*
1	6.50%	2
2	0.00%	0
3	22.60%	7
4	35.50%	11
5	35.50%	11
Comments		11
	answered question	31
	skipped question	12
	average	3.93548387096774

B.5 Page 6. Your ideas about PaperCube

Q30. What paper view did you like the MOST?

Answer Options	*Response Percent*	*Response Count*
Paper Detail	29.40%	10
Circle View	32.40%	11
Tree Map	8.80%	3
Papers Per Year	23.50%	8
Paper Graph	5.90%	2
Comments		26
	answered question	34
	skipped question	9

Q31. What paper view did you like the LEAST?

Answer Options	*Response Percent*	*Response Count*
Paper Detail	3.00%	1
Circle View	12.10%	4
Tree Map	51.50%	17
Papers Per Year	15.20%	5
Paper Graph	18.20%	6
Comments		24
	answered question	33
	skipped question	10

Q32. What author view did you like the MOST?

Answer Options	*Response Percent*	*Response Count*
Author Detail	40.00%	12
Author Papers	6.70%	2
Collaborators	36.70%	11
Author's Cites	16.70%	5
Comments		20
	answered question	30
	skipped question	13

Q33. What author view did you like the LEAST?

Answer Options	*Response Percent*	*Response Count*
Author Detail	13.30%	4
Author Papers	36.70%	11
Collaborators	16.70%	5
Author's Cites	33.30%	10
Comments		19
	answered question	30
	skipped question	13

Q34. Which type of views were most useful for you?

Answer Options	*Response Percent*	*Response Count*
Papers	87.10%	27
Authors	12.90%	4
Comments		19
	answered question	31
	skipped question	12

Q35. Did you find PaperCube useful when augmenting paper and author search?

Answer Options	*Response Percent*	*Response Count*
Yes	85.20%	23
No	14.80%	4
Comments		18
	answered question	27
	skipped question	16

Q36. What would you add to PaperCube?

Answer Options	*Response Count*
	21
answered question	21
skipped question	22

Q37. What would you remove from PaperCube?

Answer Options	*Response Count*
	18
answered question	18
skipped question	25

Q38. Would PaperCube's features be useful to include as part of a larger digital library service?

Answer Options	*Response Percent*	*Response Count*
Yes	100.00%	34
No	0.00%	0
Comments		23
	answered question	34
	skipped question	9

Q39. Any other comments?

Answer Options	*Response Count*
	18
answered question	18
skipped question	25

Appendix C

PaperCube PHP Interface

This appendix contains the complete source code for the PHP interface.

```php
<?php
/*
  This PHP script interfaces the PaperCube client application with the MySQL
  database. It constructs the JSON to be returned to the client.

  PAPERS
  searchPapers // summary info only
  getPaperDetails // returns {Array} paper details with authors
  getAllDataForPaper // reutnrs {Array} full paper details.

  AUTHORS
  searchAuthors // summary info only
  getAuthorDetails // returns {Array} author details

  All but the 'getAllDataForPaper' action queries the database for GUIDs then
  looks up cached JSON to return to the client.

  @author Peter Bergstrom
  @copyright 2008-2009 Peter Bergstr m, Santa Clara University.
*/

// If not debug, then return JSON header.
if(!isset($_GET["debug"])) {
  header("Content-Type: text/x-json, X-JSON: X-JSON");
}

// Get the action.
$action = (isset($_GET['action'])) ? $_GET['action'] : '';

// Grab the GUID(s)
$guid = '0';
if(isset($_GET['guid'])) {
  $guid = $_GET['guid'];
}
else if(isset($_POST['guid'])) {
  $guid = $_POST['guid'];
}
```

```
// Determine the action to execute.
switch($action) {
  /****************************
    Paper actions.
  ****************************/

  // Search for papers.
  case "searchPapers":
    searchPapers();
    break;

  // Get the CACHED details for papers.
  case "getPaperDetails":
    $guids = explode(',', $guid);
    getPapers($guids);
    break;

  // Get all the details for papers. NOT cached.
  case "getAllDataForPaper":
    $guids = explode(',', $guid);
    getAllDataForPaper($guids);
    break;
  /****************************
    Author actions.
  ****************************/

  // Search for authors.
  case "searchAuthors":
    searchAuthors();
    break;

  // Get the CACHED details for authors.
  case "getAuthorDetails":
    $guids = explode(',', $guid);
    getAuthors($guids);
    break;

  // Return error if no matching action is found.
  default:
    $result = array("status" => 0, "error" => array("1",
                "No action specified!"));
    packageResponse($result);
    break;
}

/*
  Connect to the database.
*/
function openDB() {
  $link = mysql_connect('localhost', 'root', '');
  if (!$link) {
    $result = array("status" => 0, "error" => array("5",
                "Unable to connect to database."));
    packageResponse($result);
    exit();
  }
  mysql_select_db("citeseer", $link);
  return $link;
}
```

```
/*
  Disconnect form the database.
*/
function closeDB($link) {
  mysql_close($link);
}

/*
  Based on the search parameters, find papers.
*/
function searchPapers() {
  $result = array();

  // Get the search parameters.
  $key = $_GET["searchKey"];
  $value = $_GET['searchValue'];
  $start = (isset($_GET['queryStart'])) ? $_GET['queryStart'] : '0';
  $limit = (isset($_GET['queryLimit'])) ? $_GET['queryLimit'] : '5';

  // If no key or value is specified, return error.
  if(!isset($key) || !isset($value)) {
    $result = array("status" => 0,  "error" => array("2",
                    "Key or value NOT specified!"));
    packageResponse($result);
    return;
  }

  // Construct the query based on the key and value.
  if($key == 'name') {
    // Get papers with a non-exact match of an author.
    $query = "SELECT ar.pguid AS guid FROM author_rel ar JOIN authors a ON".
             "a.guid=ar.aguid WHERE MATCH(a.name) AGAINST('".
              mysql_escape_string($value)."') LIMIT ".$start.", ".$limit;
  }
  else if($key == 'date') {
    // Get papers with an exact match of pubyear.
    $query = "SELECT guid FROM `papers` WHERE pubyear='".
             mysql_escape_string($value)."' LIMIT ".$start.", ".$limit;
  }
  else {
    // Get papers with a non-exact match of any key.
    $query = "SELECT guid FROM `papers` WHERE MATCH(".
             mysql_escape_string($key).") AGAINST ('".
             mysql_escape_string($value)."' IN BOOLEAN MODE) LIMIT ".
             $start.", ".$limit;
  }

  // Query for the GUIDs.
  $papers = performQuery($query);

  // Compile the guids.
  $guids = array();
  while ($row = mysql_fetch_assoc($papers)) {
    array_push($guids, $row['guid']);
  }

  // Get the paper JSON.
  getPapers($guids);
}
```

```
/*
  Based on an array of GUIDs, return the cached JSON for papers to the client.
*/
function getPapers($guids) {
  // Construct the SQL query string.
  $whereClause = "guid='" . implode( "' OR guid='" , $guids) ."'";

  $query = "
  SELECT json
  FROM papercache
  WHERE ".$whereClause;

  // Perform the query.
  $res = performQuery($query);

  // Construct the JSON.
  $jsonResult = array();
  while ($row = mysql_fetch_assoc($res)) {
    $jsonResult[] = $row['json'];
  }

  // Return the JSON to the client.
  packageJSONResponse(array("data" => $jsonResult, "status" => 1));
}


/*
  Given an array of GUIDs, return non-cached data.
*/
function getAllDataForPaper($guids) {
  // Construct the query.
  $whereClause = "guid='" . implode( "' OR guid='" , $guids) ."'";

  $query = "SELECT title, type, abstract, language, source, format, ".
           "pubyear as year, guid, fixedyear FROM papers WHERE ".$whereClause;

// Perform the query.
  $res = performQuery($query);

  // Construct the JSON.
  $result = array();
  while ($row = mysql_fetch_assoc($res)) {
    $row['allDataRetrieved'] = true;
    $result[] = $row;
  }

  // Encode the arrays into JSON.
  echo json_encode(array("data" => $result, "status" => 1));
}
```

```
/*
  Based on the search parameters, find authors.
*/
function searchAuthors() {
  $result = array();

  // Get the search parameters.
  $key = $_GET["searchKey"];
  $value = $_GET['searchValue'];
  $start = (isset($_GET['queryStart'])) ? $_GET['queryStart'] : '0';
  $limit = (isset($_GET['queryLimit'])) ? $_GET['queryLimit'] : '5';

  // If no key or value is specified, return error.
  if(!isset($key) || !isset($value)) {
    $result = array("status" => 0,  "error" => array("2",
                "Key or value NOT specified!"));
    packageResponse($result); return;
  }

  // Construct the query based on the key and value.
  if($key == 'title') {
    // Get authors with a non-exact match of a title.
    $query = "SELECT DISTINCT ar.aguid AS guid FROM author_rel ar JOIN ".
             "papers p ON p.guid=ar.pguid WHERE MATCH(p.title) AGAINST('".
             mysql_escape_string($value)."' IN BOOLEAN MODE) LIMIT ".
             $start.", ".$limit;
  }
  else if($key == 'date') {
    // Get authors with an exact match of a pubyear.
    $query = "SELECT DISTINCT ar.aguid AS guid FROM author_rel ar JOIN ".
             "papers p ON p.guid=ar.pguid WHERE p.pubyear = '".
             mysql_escape_string($value)."' LIMIT ".$start.", ".$limit;
  }
  else if($key == 'abstract' || $key == 'subject') {
    // Get author with a non-exact match of an abstract or subject.
    $query = "SELECT DISTINCT ar.aguid AS guid FROM author_rel ar JOIN ".
             "papers p ON p.guid=ar.pguid WHERE MATCH(p.".$key.") AGAINST('".
             mysql_escape_string($value)."') LIMIT ".$start.", ".$limit;
  }
  else {
    // Get authors bassed on the author name.
    $query = "SELECT guid FROM `authors` WHERE MATCH(".
             mysql_escape_string($key).") AGAINST ('".
             mysql_escape_string($value)."' IN BOOLEAN MODE) LIMIT ".
             $start.", ".$limit;
  }

  // Get the authors.
  $authors = performQuery($query);

  // Get the GUIDs.
  $guids = array();
  while ($row = mysql_fetch_assoc($authors)) {
    array_push($guids, $row['guid']);
  }

  // Compile the author JSON.
  getAuthors($guids);
}
```

```
/*
  Based on an array of GUIDs, return the cached JSON for authors to the
  client.
*/
function getAuthors($guids) {
  // Construct the query.
  $whereClause = "guid='" . implode( "' OR guid='" , $guids) ."'";
  $query = "
  SELECT *
  FROM authorcachefull
  WHERE ".$whereClause;

  // Get the JSON based on the GUIDs.
  $res = performQuery($query);

  // Compile the JSON.
  $jsonResult = array();
  while ($row = mysql_fetch_assoc($res)) {
    $jsonResult[] = $row['json'];
  }

  // Return the JSON result to the client.
  packageJSONResponse(array("data" => $jsonResult, "status" => 1));
}

/*
  Given a string query, retrieve data from the database.
*/
function performQuery($query) {
  // Open the database.
  $db = openDB();

  // Query the database.
  $queryResult = mysql_query($query, $db);

  // If there is an error, return the error.
  if(!$queryResult) {
    $result = array("status" => 0,  "error" => array("6",
                    "Database query error: " . mysql_error()));
    packageResponse($result);
    exit();
  }

  // Close the database and return the result.
  closeDB($db);
  return $queryResult;
}
```

```
/*
  Given a result array, build the cached JSON object and return it.
*/
function packageJSONResponse($result) {

  // Display error.
  if($result['status'] == 0) {
    $statString = '"status": "'.$result['status'].'", "error":[\''.
                  $result['error'][0]."', '".$result['error'][1]."']";
  }
  // Display success.
  else {
    $statString = '"status":"'.$result['status'].'"';
  }

  // Echo the result.
  echo '{'.$statString.', "data":['.implode(',', $result['data'])."]}";
}


/*
  Given non-cached JSON, encode the data and return it.
*/
function packageResponse($result) {
  echo json_encode($result);
}


?>
```

Appendix D

MySQL Schema

```
- MySQL dump 10.11
--
-- Host: localhost    Database: citeseer
-- ------------------------------------------------------
-- Server version 5.0.45

--
-- Table structure for table ‘author_rel‘
--

CREATE TABLE ‘author_rel‘ (
  ‘guid‘ varchar(40) NOT NULL,
  ‘aguid‘ varchar(40) NOT NULL,
  ‘pguid‘ varchar(40) NOT NULL,
  ‘type‘ varchar(18) NOT NULL,
  PRIMARY KEY  (‘guid‘),
  KEY ‘guid‘ (‘guid‘),
  KEY ‘pguid‘ (‘pguid‘),
  KEY ‘aguid‘ (‘aguid‘),
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Table structure for table ‘authorcachefull‘
--

CREATE TABLE ‘authorcachefull‘ (
  ‘guid‘ varchar(40) NOT NULL,
  ‘json‘ longtext NOT NULL,
  PRIMARY KEY  (‘guid‘)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
--
-- Table structure for table ‘authors‘
--

CREATE TABLE ‘authors‘ (
  ‘guid‘ varchar(40) NOT NULL,
  ‘type‘ varchar(10) NOT NULL,
  ‘name‘ varchar(254) NOT NULL,
  ‘step1‘ varchar(10) default NULL,
  PRIMARY KEY  (‘guid‘),
  KEY ‘name‘ (‘name‘),
  KEY ‘guid‘ (‘guid‘),
  FULLTEXT KEY ‘name_2‘ (‘name‘),
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Table structure for table ‘papercache‘
--

CREATE TABLE ‘papercache‘ (
  ‘guid‘ varchar(40) NOT NULL,
  ‘json‘ longtext NOT NULL,
  PRIMARY KEY  (‘guid‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Table structure for table ‘papers‘
--
CREATE TABLE ‘papers‘ (
  ‘guid‘ varchar(40) NOT NULL,
  ‘type‘ varchar(10) NOT NULL,
  ‘date‘ datetime NOT NULL,
  ‘subject‘ varchar(254) NOT NULL,
  ‘abstract‘ longtext NOT NULL,
  ‘language‘ varchar(12) NOT NULL,
  ‘source‘ varchar(254) NOT NULL,
  ‘format‘ varchar(12) NOT NULL,
  ‘url‘ varchar(254) NOT NULL,
  ‘title‘ varchar(254) NOT NULL,
  ‘rights‘ varchar(254) NOT NULL,
  ‘publisher‘ varchar(254) NOT NULL,
  ‘pubyear‘ int(11) default NULL,
  ‘fixedyear‘ int(11) default ’0’,
  PRIMARY KEY  (‘guid‘),
  KEY ‘guid‘ (‘guid‘),
  KEY ‘pubyear‘ (‘pubyear‘),
  FULLTEXT KEY ‘title‘ (‘title‘),
  FULLTEXT KEY ‘subject‘ (‘subject‘),
  FULLTEXT KEY ‘abstract‘ (‘abstract‘),
  FULLTEXT KEY ‘publisher‘ (‘publisher‘),
  FULLTEXT KEY ‘rights‘ (‘rights‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
--
-- Table structure for table ‘reference_rel‘
--

CREATE TABLE ‘reference_rel‘ (
  ‘guid‘ varchar(40) NOT NULL,
  ‘rguid‘ varchar(40) NOT NULL,
  ‘pguid‘ varchar(40) NOT NULL,
  ‘type‘ varchar(10) NOT NULL,
  PRIMARY KEY  (‘guid‘),
  KEY ‘guid‘ (‘guid‘),
  KEY ‘pguid‘ (‘pguid‘),
  KEY ‘rguid‘ (‘rguid‘),
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Bibliography

B. B. Bederson. PhotoMesa: A zoomable image browser using quantum treemaps and bubblemaps. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 71–80, New York, NY, 2001. ACM. ISBN 1-58113-438-X. doi: http://doi.acm.org/10.1145/502348.502359.

P. Bergström and E. J. Whitehead. CircleView: Scalable visualization and navigation of citation networks. In *Proceedings of the 2006 Symposium on Interactive Visual Information Collections and Activity (IVICA 2006)*, College Station, Texas, 2006.

T. BernersLee. Information management: A proposal, 1989. URL http://www.w3.org/History/1989/proposal.html.

V. Bush. As we may think. *The Atlantic Monthly*, 1945. URL http://www.theatlantic.com/doc/194507/bush.

S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in information visualization: Using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1999. ISBN 1558605339. URL http://portal.acm.org/citation.cfm?id=300679.

S. Champeon. JavaScript: How did we get here?, September 2001. URL http://www.oreillynet.com/pub/a/javascript/2001/04/06/js/w_history.html.

C. Chen and L. Carr. Visualizing the evolution of a subject domain: A case study. In *VIS '99: Proceedings of the conference on Visualization '99*, pages 449–452, Los Alamitos, CA, 1999. IEEE Computer Society Press. ISBN 0-7803-5897.

ECMA International. *ECMA-262: ECMAScript Language Specification*. ECMA (European Association for Standardizing Information and Communication Systems), Geneva, Switzerland, third edition, December 1999. URL http://www.ecma-international.org/publications/standards/Ecma-327.htm.

N. Elmqvist and P. Tsigas. CiteWiz: A tool for the visualization of scientific citation networks. Technical Report CS:2004-05, Chalmers University of Technology, 2004.

D. C. Engelbart and W. K. English. A research center for augmenting human intellect. In *AFIPS '68 (Fall, part I): Proceedings of the December 911, 1968, fall joint computer conference, part I*, pages 395–410, New York, NY, 1968. ACM.

J. Ferraiolo, F. Jun, and D. Jackson. Scalable Vector Graphics (SVG) 1.1 specification, W3C recommendation. Technical report, World Wide Web Consortium, 2003. URL http://www.w3.org/TR/SVG11/.

G. W. Furnas. Generalized fisheye views. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 17(4):16–23, 1986. doi: http://doi.acm.org/10.1145/22339.22342.

E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley, Boston, MA, January 1995. ISBN 0201633612.

C. L. Giles, K. D. Bollacker, and S. Lawrence. CiteSeer: An automatic citation indexing system. In *DL '98: Proceedings of the Third ACM Conference on Digital Libraries*, pages 89–98, New York, NY, 1998. ACM. ISBN 0-89791-965-3. doi: http://doi.acm.org/10.1145/276675.276685.

W. D. Gray, C. R. Sims, W.-T. Fu, and M. J. Schoelles. The soft constraints hypothesis: A rational analysis approach to resource allocation for interactive behavior. *Psychological Review*, 113:461–482, 2006.

I. Hickson. HTML5 - Draft recommendation. Technical report, Web Hypertext Application Technology Working Group, 2009. URL http://whatwg.org/html5.

http://calendar.yahoo.com. Yahoo Calendar. http://calendar.yahoo.com, 2009.

http://citeseer.ist.psu.edu. CiteSeer. http://citeseer.ist.psu.edu, 2009.

http://citeseerx.ist.psu.edu. CiteSeerX. http://citeseerx.ist.psu.edu, 2009.

http://developer.apple.com/cocoa/. Cocoa. http://developer.apple.com/cocoa/, 2009.

http://developer.yahoo.com/yui. The Yahoo! user interface library (YUI). http://developer.yahoo.com/yui, 2009.

http://ieeexplore.ieee.org. IEEEXplore. http://ieeexplore.ieee.org, 2009.

http://portal.acm.org. The ACM Digital Library. http://portal.acm.org, 2009.

http://scholar.google.com. About Google Scholar. http://scholar.google.com, 2009.

http://silverlight.net. The official Microsoft Silverlight site. http://silverlight.net, 2009.

http://vimeo.com/3323956. Introduction to PaperCube. http://vimeo.com/3323956, 2009.

http://www.adobe.com/products/flex. Adobe Flash Flex 3. http://www.adobe.com/products/flex, 2009.

http://www.apa.org/psycinfo. PyschINFO. http://www.apa.org/psycinfo, 2009.

http://www.apple.com/safari. What is Safari? http://www.apple.com/safari, 2009.

http://www.cappuccino.org. Cappuccino web framework - Build desktop class applications in Objective-J and JavaScript. http://www.cappuccino.org, 2009.

http://www.derlien.com. Disk Inventory X. http://www.derlien.com/, 2009.

http://www.dojotoolkit.org. The Dojo Toolkit. http://www.dojotoolkit.org, 2009.

http://www.extjs.com. Ext - A foundation you can build on. http://www.extjs.com, 2009.

http://www.getfirefox.com. Firefox web browser — Faster, more secure, & customizable. http://www.getfirefox.com, 2009.

http://www.google.com/chrome/. Google Chrome. http://www.google.com/chrome/, 2009.

http://www.iwork.com. iWork.com. http://www.iwork.com, 2009.

http://www.jquery.com. jQuery: The write less, do more, JavaScript library. http://www.jquery.com, 2009.

http://www.json.org. JSON (JavaScript Object Notation). http://www.json.org, 2009.

http://www.macromedia.com/software/flash. Adobe flash player. http://www.macromedia.com/software/flash/about, 2009.

http://www.me.com. Apple MobileMe. http://www.me.com, 2009.

http://www.ncbi.nlm.nih.gov/pubmed. PubMed. http://www.ncbi.nlm.nih.gov/pubmed, 2009.

http://www.otherinbox.com. Other Inbox. http://www.otherinbox.com, 2009.

http://www.prototypejs.org. Prototype JavaScript framework: Easy ajax and DOM manipulation for dynamic web applications. http://www.prototypejs.org, 2009.

http://www.sproutcore.com. SproutCore—a JavaScript framework. http://www.sproutcore.com, 2009.

http://www.w3.org. World Wide Web Consortium - Web Standards. http://www.w3.org, 2009.

http://www.webkit.org. The WebKit open source project. http://www.webkit.org, 2009.

http://www.westlaw.com. Westlaw. http://www.westlaw.com, 2009.

K. A. Jameson, J. L. Kaiwi, and D. Bamber. Color coding information: Assessing alternative coding systems using independent brightness and hue dimensions. *Journal of Experimental Psychology: Applied*, 7:112–128, 2001.

B. Johnson and B. Shneiderman. TreeMaps: A spacefilling approach to the visualization of hierarchical information structures. In *VIS '91: Proceedings of the 2nd Conference on Visualization '91*, pages 284–291, Los Alamitos, CA, 1991. IEEE Computer Society Press. ISBN 0-8186-2245-8.

G. Keizer. Safari 4 rivals Google Chrome in JavaScript race, February 2009. URL http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9128638.

C. Lagoze, H. Van de Sompel, M. Nelson, and S. Warner. The open archives initiative protocol for metadata harvesting. Technical report, Open Archives Initiative, 2002. URL http://www.openarchives.org/OAI/openarchivesprotocol.html.

J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. doi: http://doi.acm.org/10.1145/223904.223956.

B. Lee, M. Czerwinski, G. Robertson, and B. B. Bederson. Understanding research trends in conferences using PaperLens. In *CHI '05: CHI '05 Extended Abstracts on Human Factors in Computing Systems*, pages 1969–1972, New York, NY, 2005. ACM. ISBN 1-59593-002-7. doi: http://doi.acm.org/10.1145/1056808.1057069.

C. Lynch. Where do we go from here? The next decade for digital libraries. *D-Lib Magazine*, 11 (78), 2005.

J. D. Mackinlay, R. Rao, and S. K. Card. An organic user interface for searching citation links. In *CHI '95: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 67–73, New York, NY, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. doi: http://doi.acm.org/10.1145/223904.223913.

M. Matlin. *Congnition 6th ed.* Hoboken, NJ: John Wiley & Sons, Inc, 6 edition, 2005.

S. J. P. McDougall and O. de Brujin. Exploring the effects of icon characteristics on user performance: The role of icon concreteness, complexity, and distinctiveness. *Journal of Experimental Psychology: Applied*, 6:291–306, 2000.

D. Modjeska, V. Tzerpos, P. Faloutsos, and M. Faloutsos. BIVTECI: A bibliographic visualization tool. In *CASCON '96: Proceedings of the 1996 Conference of the Centre for Advanced Studies on Collaborative Research*, page 28. IBM Press, 1996.

MySQL Boolean Full-Text Searches. MySQL 5.1 reference manual :: 11.8.2 Boolean fulltext searches. http://dev.mysql.com/doc/refman/5.1/en/fulltext-boolean.html, 2009.

National Science Board and United States. *Science and Engineering Indicators 2008*, volume 1. Washington, U.S. Govt. Print. Off., 2008.

T. H. Nelson. *Literary Machines*. Mindful Press, 1990.

T.-H. Ong, H. Chen, W.-k. Sung, and B. Zhu. Newsmap: A knowledge map for online news. *Decis. Support Syst.*, 39(4):583–597, 2005. ISSN 0167-9236. doi: http://dx.doi.org/10.1016/j.dss.2004.03.008.

K. Quely and D. L. McClain. A year of heavy losses, 2008. URL http://www.nytimes.com/interactive/2008/09/15/business/20080916-treemap-graphic.html.

R. Rauschenberger and S. Yantis. Perceptual encoding efficiency in visual search. *Journal of Experimental Psychology: General*, 135:116–131, 2006.

J. Resig. Extreme JavaScript performance, October 2008. URL http://arstechnica.com/open-source/news/2008/10/extreme-javascript-performance.ars.

G. G. Robertson and J. D. Mackinlay. The Document Lens. In *UIST '93: Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, pages 101–108, New York, NY, 1993. ACM. ISBN 0-89791-628-X. doi: http://doi.acm.org/10.1145/168642.168652.

Z. Shen, M. Ogawa, S. T. Teoh, and K.-L. Ma. BiblioViz: A system for visualizing bibliography information. In *APVis '06: Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation*, pages 93–102, Darlinghurst, Australia, 2006. Australian Computer Society, Inc. ISBN 1-920682-41-4.

I. Spence, N. Kutlesa, and D. L. Rose. Using color to code quantity in spatial displays. *Journal of Experimental Psychology: Applied*, 5:393–412, 1999.

A. M. Treisman and G. Gelade. A featureintegration theory of attention. *Cognitive Psychology*, 12: 97–136, 1980.

A. M. Treisman and J. Souther. Search asymmetry: A diagnostic for preattentive processing of separable features. *Cognitive Psychology*, 114:285–310, 1985.

A. Wright. *Glut: Mastering Information Through the Ages*. Cornell University Press, September 2008.