Interaction Techniques for Automating Collecting and Organizing Personal Web Content

Lubomira A. Dontcheva

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2008

Program Authorized to Offer Degree: Department of Computer Science & Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Lubomira A. Dontcheva

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Co-Chairs of the Supervisory Committee:

_____

Michael F. Cohen

_____

David H. Salesin

Reading Committee:

_____

Michael F. Cohen

_____

David H. Salesin

_____

Steven M. Drucker

Date: _____

Signature_____

Date_____

University of Washington

Abstract

Interaction Techniques for Automating Collecting and Organizing Personal Web Content

Lubomira A. Dontcheva

Co-Chairs of the Supervisory Committee:
Affiliate Faculty Michael F. Cohen
Department of Computer Science & Engineering

Professor David H. Salesin
Department of Computer Science & Engineering

The growth of the World Wide Web has led to a dramatic increase in accessible information. Today, people use the Web for a large variety of activities including travel planning, comparison shopping, entertainment, and research. However, the tools available for collecting, organizing, and sharing Web content have not kept pace with the rapid growth in information. Today people continue to use bookmarks, email, and printers for managing Web content. In this thesis, I present a set of semi-automatic interaction techniques for retrieving content from the Web using the structure of webpages, presentation principles based on layout templates for user-guided organization of content from any number of Web sources, and a new template-based search paradigm for the Web that transforms keyword search into a goal-oriented rich visual experience. To demonstrate the efficacy of these ideas I combined them into a working system and evaluated them through a three-month longitudinal user study. Finally, the ideas that I present in this thesis when popularized by an online Web community would allow average Web users to build a kind of machine-readable "Semantic Web" piece by piece as they go about accomplishing their personal tasks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisors for supporting my research ideas and helping me find a thesis topic I am truly passionate about. It is thanks to Michael Cohen and his endless patience, compassion, and guidance that I decided to stay in graduate school and finish my doctorate. David Salesin, on the other hand, has been my driving force and has always urged me to aim high and never consider something to be impossible. Finally, Steven Drucker, although not officially my advisor, has participated in every step along the way to completing this thesis work and is in many ways my advisor as well. Steve has shown me the human side of doing research. For over two years the four of us met each week and discussed progress, new directions, and trendy technology. Often the conversation would turn to fine dining, kids, or the latest social networking website. But, I suppose all good things must come to an end. Despite my overwhelming happiness over finishing my thesis work, I cannot help but feel a bit sad that our time together, at least in this form, has come to an end.

I would like to thank Sharon Lin and Cheyne Mathey-Owens for helping me with the longitudinal user study. Without their help, the study would not have been possible. I would also like to thank a number of colleagues, who have helped along the way. Geraldine Wade helped with the initial designs of the layout templates. Blaine Bell helped me integrate maps. Ty Lettau and Gregg Wilensky helped me visualize and clarify the concepts behind template-based search.

I would like to thank my family for their support throughout twenty years of schooling. They have been instrumental in my success thus far. I would also like to thank all of my close friends for making graduate school a wonderful experience. I would like to give special thanks to Wilmot Li for joining me on this journey and helping me through many deadlines and stressful moments.

I would like to thank all of the people in the Graphics Lab (GRAIL) for supporting me and my research and for continuing to inspire me. Karen Liu, Adrien Treuille, Keith Grochow, and Dan Goldman have become some of my closest friends. Aseem Agarwala, Gary Yngve, and Pravin

Bhat have been excellent colleagues. Brian Curless, Steve Seitz, and Zoran Popović have always been generous with their time and have helped make GRAIL a vibrant, collaborative environment. I thank Barbara Mones for her incredible support over the last seven years. She has always been my cheerleader, and I truly value her opinion.

I would like to thank James Landay and the DUB group for introducing me to the field of human-computer interaction and inspiring me to do great user interface work. I would like to thank Lindsay Michimoto, David Notkin, Hank Levy, and David Wetherall for helping me through these graduate school years by showing me the true spirit of UW CSE. And finally, to the denizens of room 324 in the Allen Center, thanks for sharing an office with me and letting me ask lots of questions.

CHAPTER 1

INTRODUCTION

As the amount of content delivered over the World Wide Web grows, so does the consumption of information. Today people collect and organize information in ways different from before: in order to plan a vacation, they no longer go to a travel agent; when making purchases, they no longer go to a neighborhood specialty store; and when learning about a new topic, they no longer go to the library. Instead, they browse the Web. One study [34] of Web usage in 2005 shows that users visit thousands of webpages per week and in any thirty minute browsing session may visit hundreds of individual pages. Although advancements in search technologies make it much easier to find information, users often browse the Web with a particular task in mind and are concerned not only with finding but also with collecting, organizing, and sharing content. These types of browsing sessions, which this works calls *exploratory Web research* sessions, typically last a long time, span several sessions, involve gathering large amounts of heterogeneous content, and can be difficult to organize ahead of time, as the categories emerge through the tasks themselves [79]. Research in current practices for collecting and organizing Web content shows that users save bookmarks or tabs, collect content in documents, store pages locally, and print them out [43]. These methods require a great deal of overhead as pages must be saved manually and organized into folders, which distracts from the real task of analyzing the content and making decisions.

Previous research, including systems like WebBook [14] and TopicShop [3], has focused on providing better tools for managing Web content by representing bookmarks visually as webpage thumbnails. This approach is useful for returning to previously visited webpages but is less appropriate for exploratory Web research tasks due to the large number of webpages visited by users. Kaasten and Greenberg [46] study webpage representations at different sizes and find that for recognizing a specific webpage, webpage thumbnails must be at least 208x208 pixels. When users

want to not only recognize a webpage but also use the content inside of a webpage for analysis or decision making, the webpage thumbnails must be scaled back to their original size, as webpages are designed for access on desktop and laptop displays.

In my work I break out of the webpage paradigm and consider the individual pieces of content inside of the webpage to be the basic unit that must be collected, as it is the information inside the webpage that is of most importance. If the goal is to let people more easily accomplish their information tasks, then tools must support the manipulation of information, not webpages. There are a few examples of systems that give users access over the content inside of a webpage, such as HunterGatherer [76], Internet Scrapbook [85], and C3W [28], but it is the Semantic Web that promises to truly transform the way people manipulate information. The Semantic Web was first proposed in 1999 by Tim Berners-Lee when he envisioned a machine-readable Web [18]. Today, the World Wide Web remains readable only by humans. Computers still struggle with even the most basic tasks, such as extracting specific information from a webpage like an address, phone number, or price. In a machine-readable World Wide Web every webpage will include additional semantic information describing its content. Semantic understanding of webpages will enable a structured World Wide Web with a vast new set of capabilities. With a Semantic Web user decision-making and problem-solving will become much easier as automated processes will extract, aggregate, and analyze information from any number of sources. For example, booking travel will no longer be a painful nightmare. All users will have to do is list their preferences, and automatic algorithms will find the best hotel, the most appropriate flight, and the rental car that fits the entire family. The Semantic Web will also enable retrieval and disambiguation of accurate and trusted information from the sea of webpages that currently exist on the Web. The Semantic Web brings even greater benefits to scientists as it will provide a platform for sharing data through a standard structured interface thereby enabling easy comparison between findings and hypotheses. Unfortunately, the Semantic Web remains unrealized largely because it requires a large collaborative effort in defining an appropriate data representation and adopting that representation. Content providers are not yet willing to invest in embedding semantic information into the existing Web and coordinating their efforts with others.

In my work, I take advantage of three trends in the World Wide Web: the growing number of structured webpages, the vast rise in online collaboration, and the pervasive use of search technolo-

gies. First, as websites have become data driven, their appearance has become structured. Many webpages today are composed on the fly by combining databases and layout templates. Gibson et al. [30] study the evolution of Web content over the last decade and report on the amount of template material on the Web. Their findings show that 40-50% of content on the Web in 2005 was template content, which they define as content or formatting that is common in multiple webpages in a website, such as navigation sidebars, corporate logos, and background colors and styles. Their analysis also shows that the proportion of template material increases at a rate of approximately 6% per year. This growth is not surprising given the large amount of data that is delivered over the Web. Managing a website is much easier when the content and layout are separated as changes to either layer become independent. In addition to becoming more uniform through the use of presentation templates, the Web is also becoming collaborative. Websites such as `del.icio.us` [19] and `flickr.com` [92] have grown content categorizations organically — also known as folksonomies — by allowing users to annotate content with tags. Amazon [2], on the other hand, uses cumulative user shopping patterns to offer shopping suggestions to customers. Meanwhile, Wikipedia [87], a free Web-based encyclopedia, continues to grow through the contributions of thousands of people from all over the world. This massive online collaboration means that many can benefit from the efforts of just of few or, alternatively, that a little bit of individual effort can lead to a significant growth in overall progress. Finally, search engines have become the primary interface for accessing the Web. Studies show that people no longer save information [10]. They simply hope they can find it again through their favorite Web search engine. This behavior is not surprising given the rapid growth of the amount of information that is available on the Web. Furthermore, search engines are growing increasingly sophisticated in tracking user behavior and preferences and thus can bring forth information from a wide variety of appropriate sources including social networking websites, personal blogs, and professional journals.

In this thesis work I present a new approach to collecting and organizing Web content in a set of semi-automatic interaction techniques and algorithms that allow people to not only collect and organize Web content more quickly and easily but also enable them to build a form of the Semantic Web as they accomplish their own tasks. The interaction techniques I present enable users to transform the Web into their own personal Web that provides personalized access to the information they care about in the form of their choosing. The contributions of my thesis described

in this dissertation work in concert. First, I take advantage of the growth in template material on the Web and design semi-automatic interactive extraction of content from similar webpages using structural and content **extraction patterns**. Second, I employ **layout templates** and user labeling to create rich displays of heterogenous Web content collections. Third, I use search technology for proactive retrieval of content from different websites through user-defined **relations**. Fourth, I let users define their own personalized and aesthetic views of heterogenous content from any number of websites through **cards**. And finally, I introduce a new **template-based search** paradigm that combines the user-defined relations and cards into a search template. Search templates present a goal-driven search mechanism that creates visual personalized summaries of the content users need to accomplish their task. As users accomplish their goals with these tools, they also produce artifacts, such as the extraction patterns and relations, that are persistent and sharable. These artifacts can be stored in a public repository and reused by others. Furthermore, they can be added to the websites where they originated, thereby enhancing the existing Web with semantic information, such as relationships between different websites or the types of content available in a particular webpage. If popularized by an online community, the ideas presented in this thesis can help realize a machine-readable World Wide Web.

To demonstrate that automation can successfully be combined with interaction techniques to help users more effectively collect and organize Web content, I combined and implemented these ideas into a system that takes the form of a Web browser extension. I also conducted an ten-week longitudinal study with 24 participants in order to evaluate the effectiveness and usability of my approach. The longitudinal study included a shared repository where participants shared the artifacts they created with their tasks. This preliminary evaluation demonstrates the possible realization of this user-defined form of the Semantic Web.

This dissertation is organized as follows. First, I describe related work in this area including ethnographic research on collecting and organizing Web content and existing systems that attempt to solve all or part of this problem. To motivate automated content extraction from webpages, I describe a study of webpage structure within a website and over time in Chapter 3. In Chapters 4 and 5, I present the Web Summaries framework and its associated interaction techniques. Chapter 6 describes a longitudinal study evaluating Web Summaries and the potential for collaborative online repositories. Finally, I conclude and discuss future research in Chapter 7.

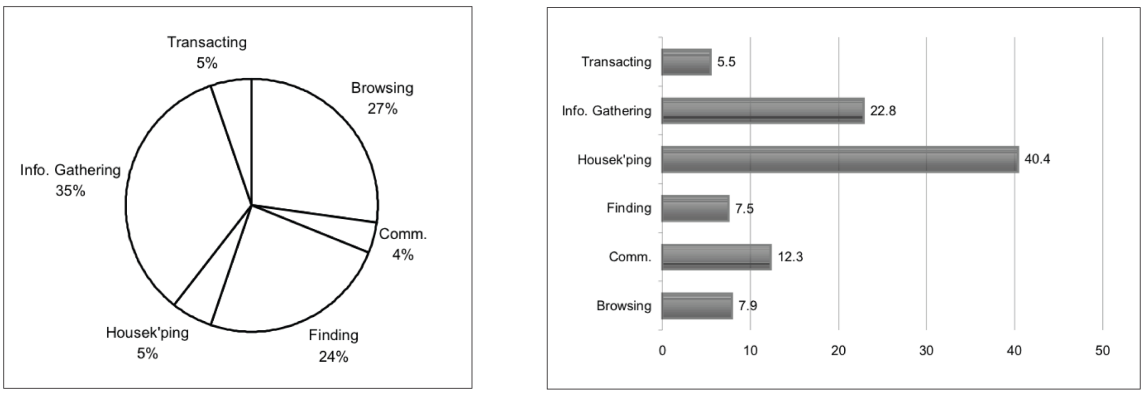
CHAPTER 2

RELATED WORK

My thesis work is the first to combine interaction techniques and automatic content extraction retrieval for exploratory Web research activities. However, I am not the first to have thought about the problems surrounding collecting and organizing Web content, and thus I first survey related work in this area. First, I discuss ethnographic research on existing user practices and derive a set of requirements for tools for collecting and organizing Web content. Then, I describe information workspaces, which aid users in not only finding information but in the subsequent content organizing and analysis. I also explore systems in related domains, such as digital media and notetaking, and discuss their possible application into the Web domain. Since my goal is to automate the collecting and organizing process as much as possible, I also survey a variety of approaches to automation on the Web, including automatic content extraction, Web databases, and end-user Web programming.

2.1 Ethnographic studies

Sellen et al. [79] study how knowledge workers use the Web, where a *knowledge worker* is someone "whose paid work involves significant time: gathering, finding, analyzing, creating, producing, or archiving information." Although the target audience of my work is the average Web user, the practices of this specific demographic group are more widely applicable because transforming knowledge often involves ad-hoc processes rather than pre-defined or routine procedures. Professionals might be able to find relevant information more quickly, but the way they save and organize that information is much like the average Web user. Sellen et al. [79] report on a two-day diary study of 24 knowledge workers and their Web activities. The study excludes email use because email is used for communicating not accessing information. Their findings divide all recorded Web activities into six categories: finding, information gathering, browsing, transacting, communicating, and housekeeping. Figure 2.1 shows the frequency and average duration of each category of activity. The *information gathering* activities, which are the activities of concern for this paper, are most frequent,



(a) (b)

Figure 2.1: Figure (a) shows the frequency of the Web activities of knowledge workers during two working days. Figure (b) shows the average duration for each category in minutes.

occurring 35% of the time. These activities are also second longest in duration, lasting on average 23 minutes. This category includes research tasks, such as answering a set of questions, comparing and contrasting information from multiple sources, and acquiring general information about a topic. The other categories include finding a specific fact (phone number, product name, or bus times), browsing the news or following an interesting link, making bank transactions or ordering a product, communicating in chat rooms and discussion groups, and performing housekeeping tasks such as maintaining the accuracy of saved links. The information gathering tasks are often project-driven, time-consuming, and complex. They require scanning and skimming large amounts of information to assess relevance and comparing and contrasting information from different sources. Of the recorded information-gathering activities only 60% were completed in a single session, either because the rest required a long time to finish or were interrupted by other tasks. For those tasks that were not completed, the study participants reported saving their findings into folders (on the desktop or in bookmark collections) or printing them on paper. Those who print said that they had trouble managing their folders and were concerned with saving intermediate results that would help them return to the task at a later time.

2.1.1 Saving Web content

Today, the most universal mechanism for saving Web content is making bookmarks, as they are provided by every browser. Because bookmarks are easy to create, they can serve as an easy mechanism for marking information that may be useful at a later time. Many bookmarks, however, go unused for months and most people do not organize their bookmarks, as discussed by Abrams et al. [1]. More recently Bruce et al. [10] study how users keep track of any information they find on the Web. Somewhat surprisingly, they find that Web content is most frequently not saved at all. Users simply rely on search engines to re-find the information they have already seen. Bruce et al. conjecture that this reliance on search engines is because searching is easier than managing and organizing information. Participants in their study mentioned:

> *"Since the advent of Google, I rarely, if ever, add any links to my Favorites."*

> *"Sometimes, this [search] is easier when topic phrases via Google are known to me to produce what I want on the first page of hits."*

When information is saved, Jones et al. [43] report that saving techniques vary according to work roles. Researchers (similar to Sellen et al.'s [79] knowledge workers) most typically email themselves links, paste URLs into documents, print out webpages, and bookmark relevant pages. On the other hand, information specialists such as librarians rely heavily on bookmarks, as their job involves supplying information to others. Lastly, managers primarily use their email to keep track of content, because they are usually the recipients of collected and organized information. So it seems that bookmarks can be very useful for certain types of activities, such as returning to a favorite site, but they are less useful for exploratory Web research. For example, bookmarks must be manually organized into hierarchical folders, but the organization of content in exploratory research often emerges only through the task itself. Additionally, bookmarks bear little resemblance to the content they mark and are typically stored locally on the machine and browser where they are created. Users email themselves links and paste URLs into documents so that they can access the information from a variety of places and easily share the content with others. Using a text document to keep track of relevant links also enables users to add additional content or write notes reminding themselves why a link is important. According to Jones et al., users who paste URLs into a document said:

"I do this when creating reports that will be shared with others so that they can access the site as well."

"I use this sometimes when I'm doing research on something specific. I will cut and paste the URL and then write some notes about the site."

"This is how I save links in the context of other information (e.g., a list of graduate schools I'm considering attending, along with links to their sites)."

Finally, users print webpages so that they can take the content with them when they are not in front of a computer and keep a persistent copy, as the content may change in the future.

2.1.2 Paper vs. digital technologies

Both Sellen et al. [79] and Jones et al. [43] report that printing webpages is a common occurrence, and thus it is important to consider why people prefer paper in certain situations and not in others. O'Hara and Sellen [66] compare the use of paper and digital technologies in the process of summarizing a document. They recruited 10 participants and asked half of them to write a summary of a scientific article using only paper and pencil. The rest of the participants performed the same task using a computer. Their study finds that computer tools offer certain advantages such as fast keyboard entry, information reuse, easy editing of text, and spell checking. However, for the purposes of writing a summary, the paper medium is far more effective than the digital medium, because it supports the synthesis process required for writing. Paper supports annotation without disrupting the reading process. It allows for easy movement between the summary and the original content, which helps the user in understanding and integrating the content into the summary. Finally, paper can easily be spatially rearranged, helping users understand the overall structure of documents and interleave reading and writing. Although O'Hara and Sellen [66] focus specifically on reading an article and summarizing its contents, their findings are also applicable to tasks that may be less reading intensive. Techniques for moving between documents quickly and spatially arranging content for analysis should also be addressed by tools for information gathering. Multi-scale representations must be provided so that the user can focus on a detail or examine its context. Annotation and

note-taking are also very useful, especially when the information gathering is interrupted and must be resumed at a later time.

Other studies [9, 57] show that paper is more effective for organizing active "hot" and "warm" documents. "Hot" documents are those currently used in a task, and "cold" documents are those that are no longer used on a day-to-day basis and have been archived. Paper can easily be piled into stacks, and stacks can be regrouped and restructured with minimal cognitive effort, allowing the user to focus on the content instead of the artifacts that contain the content. Paper, however, also has its drawbacks. It cannot be easily revised, reformatted, and incorporated in other documents. It cannot be easily replicated (without the help of photocopiers and scanners) and shared with others. It also requires physical space for its use and storage [78]. Current practices for organizing paper suggest functionalities that must be supported by an effective system for exploratory Web research. Such a system must allow the user to organize information spatially, provide multi-scale representations of content, allow for annotation, and provide mechanisms for categorizing the content quickly and easily so that it can be reorganized and restructured.

2.1.3 Organization and analysis

Marshall and Shipman [58] study the process of *information triage*, which they define as "the process of sorting through (the possibly numerous) relevant materials, and organizing them to met the needs of the task at hand." The task they study is summarizing a set of documents in order to make a recommendation to a manager. They find that during the information triage process, users categorize the content, take notes on a separate piece of paper, and annotate the documents. The categorization is often fluid, and categories are adapted and refined as more information is added. They report that users form three types of categories: semantic, pragmatic, and rhetorical. *Semantic categories* are derived directly from the content; *pragmatic categories* are based on the expected relevance of content; and *rhetorical groups* are used for the purpose of communicating the gathered content to others. O'Day and Jeffries [65] categorize analysis techniques during information gathering and the subsequent content analysis and find that 80% of analysis tasks include:

- looking for trends and correlations and experimenting with different aggregates,
- making comparisons and identifying a critical subset of relevant and unique content, and

- making assessments and interpreting data.

Fully automating the analysis process for a specific task is extremely challenging because it requires that the system make task-specific decisions. However, automatic algorithms can assist users in certain aspects of this process. For example, semantic categories can be created automatically if semantic content can be extracted directly from the data. Some pragmatic categories can also be proposed to users by exploiting prior information about topics they find important. Finally, the presentation of content can be automated using layout templates so that users can easily share their findings. For analysis tasks such as looking for trends or making comparisons, it is important to provide an intuitive interface and a variety of visualization techniques so that users can easily group content and view it in different contexts.

Russell et al. [72] define a structure for analyzing the cost of *sensemaking*, which is "the process of searching for a representation and encoding data in that representation to answer task-specific questions." Sensemaking, as they describe it, is an iteration between searching for a good representation of a body of content and instantiating that representation. As it is difficult to fully determine a representation ahead of time, the representation becomes more complete as additional content is collected. What Russell et al. [72] call representation is both a schema relating the collected content so it can be clustered and analyzed algorithmically and a visual representation so that users can reflect on the content and devise appropriate organization schemes. Their work calls for fast and intuitive mechanisms for categorizing information so that users can easily cycle through different representations and visualization techniques, making the instantiation of representations and their analysis more efficient. Sensemaking is yet another way to think about exploratory Web research. Users start with an idea and refine it as they find more content. The representation of that content evolves with additional pieces of content. Russell et al. [72] report that most of the time in sensemaking is spent in the data extraction phase, which includes finding the relevant content, selecting the information, and transforming it into a canonical form. Over a decade has passed since this definition of the cost structure of sensemaking, and today the process may be different. Search engines can now help users find relevant content more quickly. Although not yet widely used, there are automatic algorithms that can extract content from webpages. However, there is still little support for sensemaking on the Web. Even if automatic extraction tools are available, users do not have

access to the underlying schemas that represent the content and have few tools for changing the visual representation of that content. Most users do not have access to automation in their gathering and analysis tasks, and thus they rely on lists of URLs in email, documents, and bookmarks.

The studies I survey lead to the conclusion that collecting, organizing, and sharing Web content is a common and important activity, and that existing digital tools inadequately meet the user's needs for these tasks. Thus, users often rely on paper, even when paper is not the best medium for keeping track of this content. To be effective in supporting exploratory Web research tasks, techniques must be developed to fulfill the following set of requirements:

- **Integration.** There must be fluid interaction between the interfaces for finding Web content, i.e. the Web browser, and those for collecting and organizing that content, so users can easily move between finding, analyzing, and organizing content.

- **Intermediate states.** There must be support for saving Web browsing findings, so users can return to them in the future and share them with others.

- **Organization.** Content must be easily grouped and categorized, so users can refine the structure as they collect new information.

- **Views.** Multiple views and visualizations of the content must be available, so users can view the information at different scales and in the most appropriate context.

- **Annotation.** There must be mechanisms for annotation and note-taking, so users can record ideas and augment collected content.

These requirements are similar to the considerations for information environments described by Furnas and Rauch [29]. They propose the NaviQue system, which provides users with an integrated workspace that allows them to organize content spatially and search for new information directly in the workspace. My approach is the opposite. Since users are already familiar with the Web browser, my goal is to provide them with tools and techniques that enhance their browsing experience and assist the process of exploratory Web research. In the next section, I discuss existing systems for collecting and organizing Web content and their approaches to the requirements I have described.

2.2 Information workspaces

Card et al. [13] were the first to propose the concept of an *information workspace* suggesting that information workers need more than an information retrieval system, such as a search engine, but also a space to collect, organize, and analyze content. In following work, they combine this concept with Information Foraging Theory [67] to build the WebBook and Web Forager [14]. In the WebBook a collection of webpages is represented as a 3D interactive book, and these books are stored in the Web Forager, a 3D information workspace. Kaasten and Greenberg [45] use the notion of an information workspace to integrate bookmarks and history into a new revisitation system, more closely resembling the user's mental model of browsing. They also study webpage representations and the usability of thumbnails and titles in an information workspace [46]. They found that people are much better at recognizing thumbnails of webpages than titles, because a visual representation is more effective at triggering memory. However, they also found that thumbnails are only useful above a certain size. Robertson et al. [70] designed Data Mountain, which employs a 3D space metaphor for managing bookmarks. In Data Mountain users arrange thumbnail images of webpages on an inclined plane in a 3D desktop. In a study evaluating their aproach [17], the authors found that users' spatial memory is so effective in remembering where a bookmark is placed that even if the thumbnail images are removed, users could still retrieve the pages successfully. These systems were designed prior to the common use of search engines today, and thus their focus is on using different metaphors to assist the user in managing important information. It is unclear whether these paradigms are still useful today, as users' browsing patterns have changed drastically. For example, one feature of the WebBook is that all the saved webpages are loaded simultaneously, and the user can just ruffle through the book to access all the pages.

Amento et al. [3] propose a different approach to browsing, somewhat similar to the NaviQue system [29]. In their system, TopicShop, users organize webpages spatially in a workspace according to topics. The user need not browse the Web to find content, as the system automatically delivers relevant webpages given a set of seed pages. Relevant webpages are found by crawling from the seed webpages and building a graph of the seed websites by comparing textual and hyperlink content. The TopicShop system allows users to organize the webpages as thumbnails in a workspace using the paper metaphor of "piles." Users can also color code and annotate different piles of thumbnails.

Jhaveri and Räihä [42] propose presenting the information workspace as part of the browser, so that users can easily save noteworthy URLs and view them as thumbnails. The thumbnails assist users in returning to a browsing session, because they serve as visual reminders of the pages. Evaluations of all these systems show that users are more efficient at returning to webpages and forming collections using visual metaphors than using traditional bookmarks and history. However, all these systems use a webpage as the entity of value, not the information contained within the webpage. Kaasten and Greenberg [46] found that as the number of webpages grows, the visualizations shrink in size and become less effective. Furthermore, the organization and possible views of the content are limited. Organization is typically supported through spatial manipulation. In the case of TopicShop and Web Forager, the user can also create hierarchical collections. These approaches do not support categorizing pieces of webpages or assigning metadata to group webpages in different ways. Viewing these collections is only possible by interactively exploring each webpage, and there is little support for viewing content at multiple scales or in different contexts. Perhaps these limitations are present because these systems were built for quickly returning to a browsing session, with less emphasis on the organization, analysis, and sharing aspects of the tasks themselves. One strength of these systems is that their use of webpage thumbnails makes for fluid transitions between gathering content and organizing that content, as the user is always dealing with familiar entities.

Hunter Gatherer [76] was the first system to collect both full webpages and also fragments of webpages. The user organizes the fragments into collections much as one might organize bookmarks. The collections of fragments can be viewed as a list of links or displayed together in one webpage. In Spatial Hypertext [81], which predates the Web, a document is represented as a collection of data objects, such as text or hyperlinks. These objects can be manipulated and organized spatially within the document. Spatial Hypertext systems, such as VIKI [82] and the Visual Knowledge Builder [80], have been shown as effective platforms for organizing information, as users can freely group, color-code, and annotate content. Although these systems have not yet been extended to include Web content, such an extension would be not be difficult. The disadvantages of these approaches are that visualizations of the content are limited to lists, and collecting and organizing content requires lots of user effort.

Figure 2.2: MediaFinder [48] employs semantic regions to help the user organize and present personal media using content semantics. In this example an image collection is organized using three regions, according to category (i.e. individual friends), time, and geographic location. Regions can be aggregated into a parent region as shown by the grouping of images of friends into categories for high school, college, and graduate school.

Exploratory visualization systems, such as Visage [71] and Polaris [83], provide graphical primitives so the user can easily aggregate data and look for patterns. These systems focus on quantitative information and require structured content as input. Web content, on the other hand, includes a variety of information types including graphics and text and is not well suited to quantitative visualization systems. Furthermore, the structure required by such systems is not present at the onset of exploratory Web research. Such tasks require representations that can evolve as the structure emerges. Kang and Shneiderman [49] propose a mechanism for visualizing semi-structured data with the notion of a *semantic region*. Semantic regions are rectangular regions in a 2D workspace that can be associated with semantic information, such as time, geographic location, or a particular attribute of the data, such as author or topic. Semantic regions provide users with the ability to view content in different contexts, for example on a map, as a tree, or on a timeline. Content can be placed in multiple regions and automatically organized. The workspace includes a zoomable interface so that the user can focus on one region at a time. Figure 2.2 shows a sample workspace employing semantic regions. The content is simultaneously organized on a map, in a calendar, and grouped by

category. Kang and Shneiderman use the semantic regions paradigm in MediaFinder [48], which is a system for managing personal media collections. This paradigm may also be useful for organizing Web content as well, but it will need to be augmented to deal with the more textual nature of the Web and extended to include lightweight semantic metadata authoring.

2.2.1 Adaptive display of content

Effective layout of the heterogeneous content available on the Web is a whole research area in and of itself. As the Web has become template driven, the seperation of content and style has become an important area of research. Several World-Wide-Web Consortium (W3C) standards support the separation of style and content including the Extensible Stylesheet Language (XSL) [91] and Cascading Style Sheets (CSS) [90]. Badros et al. [5] proposed a constraints-based extension to Cascading Style Sheets (CCSS), allowing authoring of more adaptive designs. Jacobs et al. [40] combine constraints with pre-defined templates to create adaptive documents that respond to changes in screen size and adapt content for the desired viewing dimensions. In subsequent work, Schrier et al. [77] propose higher level template primitives, which enable a more flexible layout specification for a variety of content, including dynamically aggregated documents, such as newspapers and RSS feeds.

Other approaches for adapting the layout of Web content for different uses rely on user interaction. One such system, Collapse-to-Zoom [6], displays webpages on small-screen devices, such as a PDA. The user can interactively collapse and expand regions of the webpage simplifying navigation of the content. As regions collapse and expand, the content automatically fills the empty screen space. Woodruff et al. [89] propose highlighting and enlarging keywords in a webpage to aid users in Web search. This type of visualization can help users assess whether a page is relevant and can be useful for displaying content for different viewing dimensions. Although these approaches are effective for adapting webpages for small-screen devices, it is not clear if they are effective mechanisms for exploratory Web research. For example, instead of specifying irrelevant content by collapsing webpage regions, the user will likely be interested in specifying the content that is relevant and should be saved.

Schilit et al. [75] present a system, XLibris, for reading and annotating in a digital medium. XLibris provides the user with different views of an annotated document using a “Reader’s Notebook” that displays only the annotations or notes made during the reading process. The user can in-

teractively switch views between the notebook and the annotated document. Their system also suggests relationships between annotations by looking for similarities in the annotated content. Fluid Documents [15] present secondary content, such as annotations, in the context of the document without requiring shifts in the user's attention. Their approach uses physical space, animation, and different visual representations, such as changes in font, to provide a fluid shift in attention between the primary and secondary content. Fluid display of content may be applicable in the presentation and display of Web content. It could be one mechanism for providing details, such as notes, in the context of an overview.

In the following section, I survey systems that employ automation for a variety of Web tasks. This area is still in its infancy so the approaches I discuss vary in both their purposes and the amount of automation. I start with fully automatic systems and then discuss approaches that combine user interaction and automation.

2.3 Automation on the Web

With the growth of the Web, there has been extensive research on automatically extracting information from webpages for storage in databases, summarization, or reformatting for display on small-screen devices. Approaches include content-based text summarization, structural and visual webpage analysis, and extraction wrapper construction. Content-based text summarization approaches [94, 11, 52] analyze the content of a webpage — typically its text — and build models to represent the most common concepts within a single or multiple documents. Structure-based approaches [62, 32, 95] evaluate the structure of a webpage to build a hierarchy of webpage elements. This hierarchy can then be used to find common patterns present in the structure, such as the items in a list or the cells in a table. Structural patterns can be combined with content-based analysis [61] to identify types of content, such as price, title, or address. Additionally, structural patterns can be combined with visual webpage analysis [16, 59, 12] to help group visually adjacent webpage elements. Finally, there is a large number of papers on constructing extraction wrappers for specific types of content (see Laender et al. [51] for a survey) and these approaches include NLP algorithms, constructing models, and ontologies. As discussed by Mukherjee and Ramakrishnan [61], these algorithms can be brittle and are specific to the type of content. Although there has been significant effort in automatically reformatting webpages and filling databases with webpage content, there has

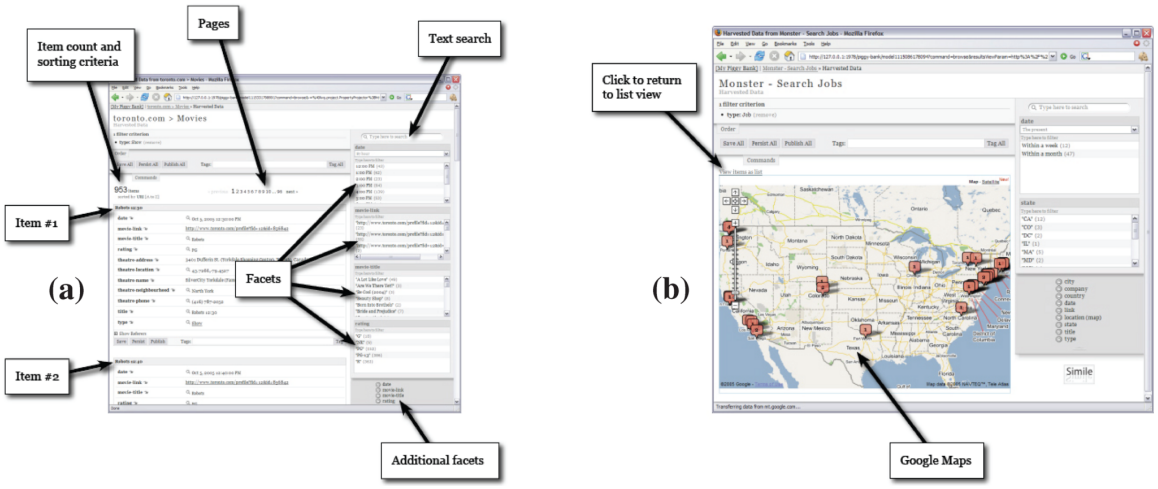


Figure 2.3: Figure (a) shows a list view of the PiggyBank database and (b) shows a map view of the same content. The user filters the content by interactively selecting facets describing the content, such as date and rating. Faceted metadata search was shown to be an effective technique for filtering content by English et al. [25]

been little work on using automation to aid in the information gathering tasks performed by a user and his Web browser.

2.3.1 Database approaches

Two systems that let users control Web extraction are Piggy Bank [37] and Thresher [35]. They are part of the Haystack [69] project, which aims to develop an information management system for personal information, such as documents, email, appointments, and tasks. Haystack's goal is to provide the user with maximum flexibility in describing personal information so that it can be visualized, manipulated, and shared. Piggy Bank is the component of Haystack that lets users collect information from the Web and store it in the Haystack framework. The user collects Web content using either semantic structures provided by the webpage author or by writing scripts, or Web scrapers, that parse the webpage to collect the desired content and assign it semantic meaning. Although writing a Web scraper may not be accessible to the average Web user, once such scrapers are in place they can be shared with less technical users. The gathered content is stored in a collaborative database where it can be searched and shared. Piggy Bank allows users to store Web content in a uniform manner, but its interface to collect, visualize, and organize that content is lim-

ited (see Figure 2.3). Subsequent work on Thresher provides a more intuitive interface by allowing users to interactively create such Web scrapers by highlighting text and assigning semantic meaning. Although Thresher provides a much better interface to gathering content than Piggy Bank, it still requires the user to shift focus from the Web browser to a semantic wrapper creation phase, which is less accessible to the average Web user and is disruptive to the actual task of collecting information. As Figure 2.3 shows, the interface for viewing and organizing the collected content is much like a database interface allowing the user to make queries to filter the content and assign metadata for organizing. Piggy Bank does offer different views on the data, such as a map. However, due to the database style of the user interface the views are difficult to manipulate and understand. The Piggy Bank framework, although a great step towards automatically extracting semantic content from the Web, does not fulfill the requirements of a system for exploratory Web research. Although Piggy Bank does integrate directly into the browser, it does not allow for fluid movement between gathering and organizing content, as the user must make new queries into the database to view the newly found content. Visual or spatial organization of the content is not provided and little attention is paid to creating multi-scale representations for an overview of the content. Aside from tagging the data, Piggy Bank does not support taking notes or annotating the data in any way.

2.3.2 End-user programming

Chickenfoot [8] is a system that is not designed for collecting information from the Web but rather for automating Web tasks and customizing webpage appearance. I include it in this discussion because its approach to automation is relevant for information gathering. Chickenfoot provides a platform for writing scripts with respect to higher-level webpage constructs such as buttons and links instead of low-level HTML. It uses keyword pattern-matching to provide the user with an interface for expressing commands such as `click("Google Search")`, `pick("California")`, and `find("Directions form")`. The Chickenfoot interface is useful for automating repetitive tasks such as filling in forms or comparing prices; however, it requires that the user know all of the source webpages and how they will be used ahead of time. This is not typically the case during exploratory research tasks. Furthermore, it requires scripting, which may be distracting from the task itself. Marmite [88] and Yahoo Pipes [93] are recent graphical tools that use a data-flow architecture for creating Web-based mashups. In my work I also strive for a simple graphical interface for

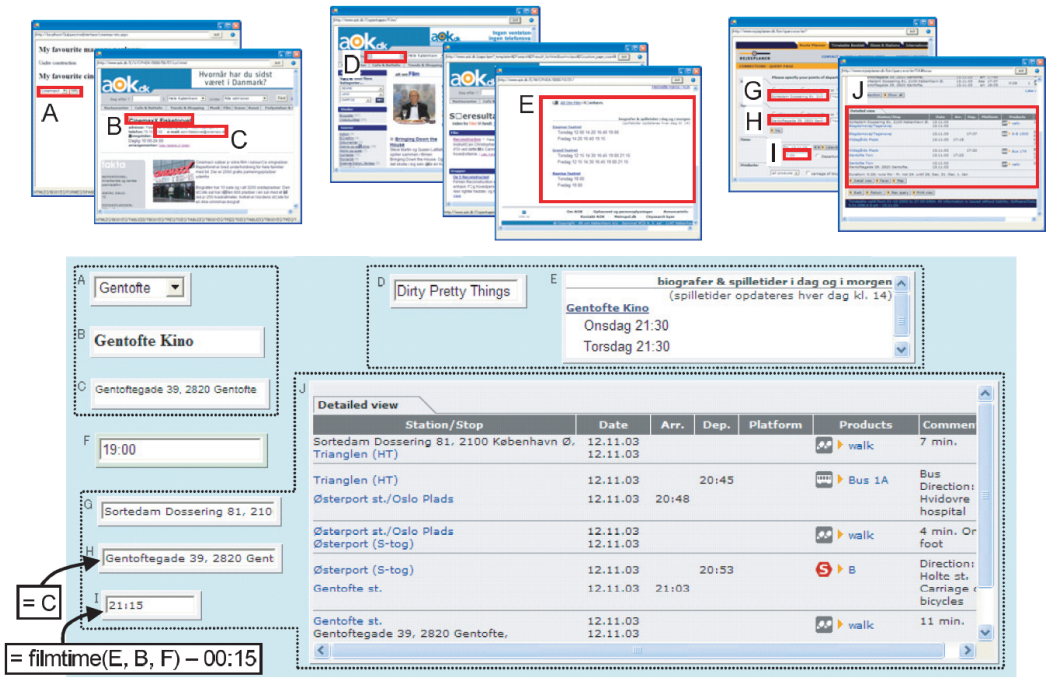


Figure 2.4: The C3W system allows the user to select pieces of webpages and assemble them into a personal user interface. In this example, the user is planning a trip to a theater and must find an appropriate bus, which arrives at the theater at the appropriate time. With the elements named A, B, and C the user selects the theater. D and E point to the selected film and the hours that film is playing. The G, H, and I elements point to the bus stop near the theater, and J shows the busses that arrive at the theater at the appropriate time. The system tracks the user webpage visitations and selections to create a personal user interface. With this interface the user can access the same information about other theaters and films automatically.

mixing content from different sources together; however, my intended audience includes essentially all Web users and not just programmers. While previous systems let users graphically create scripts, my goal is to let users more easily perform the task of collecting and organizing Web content. As a result, I do not expose the underlying programming model and allow the user to manipulate the content directly.

Internet Scrapbook [85] and C3W [28] are two systems that, like Chickenfoot, automate repetitive Web tasks. With Internet Scrapbook, the user can copy parts of a webpage and paste them into a document that dynamically updates these elements as the content on the source webpage changes. C3W provides a mechanism for building personalized interfaces by assembling interface

components from different webpages. The system tracks the user's actions while browsing and can reproduce them to collect new information. This system can be very useful when the task at hand requires that the user visit several webpages in sequence, such as when planning a trip to the theater. Figure 2.4 shows one example of a personalized interface, which assembles a bus schedule, theater schedule, and two addresses. Once such an interface is assembled, the user can check bus schedules for different films at different times. Both Internet Scrapbook and C3W focus on the dynamic aspects of the Web and use automation to enable easier user interaction with Web content for tasks that occur frequently. However, the focus of information gathering tasks is not on frequently visited information but rather on collecting and assessing new information. Nonetheless, automatic algorithms for dynamically updating content could be very useful, especially if the user is collecting content that changes, such as prices or schedules.

2.4 Discussion

While information workspaces provide better visual paradigms than are currently present in Web browsers, they still use manual techniques for managing Web content, which require a large investment of user effort. On the other hand, automatic Web techniques aid Web programmers in extracting large amounts of content for filling databases. Few systems aid the average Web user in managing Web content through automation. My thesis research fills this gap and provides semi-automatic approaches for collecting Web content and template-based layout principles for organizing and displaying the collected content as a summary.

Ethnographic research on existing practices for collecting and organizing Web content points to five requirements for tools that address exploratory Web research tasks, including:

- **fluid integration** between finding Web content and collecting and organizing that content,
- storage of **intermediate states,**
- **easy grouping and categorization** as new content is discovered,
- the availability of **multiple views** at a variety of scales,
- and easy **annotation** and note-taking.

Ethnographic research can not only describe existing practices with popular tools, but it can also inform future directions. Sellen et al. [79] show that even professional knowledge workers struggle to keep track of their Web research, despite the fact that they spend a large proportion of their time performing such tasks. In their conclusions, the authors suggest technologies for facilitating information-gathering activities. *Webscrapbooks* can allow the user to save more than just a list of URLs. Instead, users can record the content within the webpages, such as graphics, text, or search results. *Better history functions* can use webscrapbooks to allow the user to save intermediate results for later use. *Better tagging of information* can help knowledge workers more easily assess the quality of information. *Better search tools* can use tagging to provide quality information more easily. My thesis research proposes solutions to all of these proposals. In Chapter 4, I describe Web Summaries, which allows users to clip pieces of webpages, such as text, images, or tables, and organize their clippings into a persistent Web summary that can be retrieved at any time and shared with others. Whenever users clip content, they also tag that content with a semantic label. Such labels allow for rich display of the clipped content on maps and calendars, as well as uniform organization of content from any number of websites. But, such labels can also be shared and added back into the webpages where they were created. This sharing of semantic information can enable better assessment of the quality of information on those webpages. Frequent clipping of the same content by different people also gives indications about the content quality. And as the size and versatility of the Web continues to grow, approaches for evaluating the quality of a given webpage or website will become even more prevalent. I address the request for better search tools in Chapter 5 where I describe a new template-based search paradigm that retrieves quality information for the user and presents it in a personalized way.

CHAPTER 3

THE STRUCTURE OF WEBPAGES

To inform the design and implementation of tools for collecting and organizing Web content, it is important to consider the current state of the Web. The Web changes constantly, and these changes can be highly relevant to tools and algorithms for managing Web content. Gibson et al. [30] find that 40-50% of Web content consists of webpage templates and that the proportion of templatization is growing approximately 6-8% per year. Despite this growth in uniformity the task of reliably collecting structured content from the Web remains difficult. In this chapter I explore two phenomena. First, I study the differences in the structure of webpages within one website through an analysis of 225 webpages from nine different websites. Second, I study the changes in webpage structure over time through a five-month study of some 12,000 webpages from 20 different websites. My results show that websites use just a few templates to display their content, and these templates do not change very often. The structure of webpages from low-volume sites changes very little, while webpages from high-volume sites change in mostly minor ways. Some of the sites do go through drastic structural changes, but only on the order of once every couple of months. I discuss the implications of webpage structure for the design of structure-based algorithms and how they can evolve to accommodate structural webpage changes. My analysis leads me to the conclusion that structural extraction algorithms can play an important role in applications for aggregating and summarizing Web content. [1]

First, I give some background on webpage structure and representation. The content of every webpage is accessible through the Document Object Model (DOM), which is an interface for dynamically accessing and updating the content, structure, and style of Web documents. With the DOM, every element in a webpage, represented with tags such as `<body>`, `<table>`, or `<p>`, becomes a node in the DOM hierarchy, with the `<html>` tag at the root of the hierarchy. Elements in the webpage can be accessed through XPATH queries, which specify the path from the root to the

[1] The findings in this chapter were published as a technical report [21].

corresponding node in the DOM. I refer to XPATH queries as *extraction rules* and to collections of extraction rules as *extraction patterns*. Extraction patterns can be specified interactively or through code, and they can extract any number of webpage elements.

The Internet Archive [47] has been collecting snapshots of the Web for the last ten years, and when I first started this study I considered using this archive for the analysis. However, their data set is sparse and does not include all of the types of webpages I wanted to analyze. For instance, the Internet Archive does not include websites that request not to be archived, such as `amazon.com` or `hotels.com`. Further, the Internet Archive does not always record pages daily and as a result does not include a fine granularity of data.

3.1 Structural differences within a website

To measure the structural differences of webpages within one website I analyzed 225 webpages from nine different websites. I selected popular websites for shopping, travel planning, and academic reference from `www.alexa.com`, which tracks Web traffic patterns. For the academic reference category, I chose the three most popular reference websites in computer science. For each website, I visited 25 randomly selected webpages. On the first visited page for each website, I created an extraction pattern of 3-6 webpage elements. I then visited the remaining 24 webpages and collected the corresponding content with the extraction pattern. I measured how much of the content was collected from the 24 webpages and computed accuracy and recall for each extraction pattern. I define *accuracy* as the ratio of the number of page elements collected using the extraction patterns to the number of page elements that were present in all the visited webpages. I define *recall* as one minus the ratio of the number of erroneous webpage elements that were collected to the number of page elements that were present in all the visited webpages. Table 3.1 shows the accuracy of the patterns on the selected websites. I also measured how many extra page element selections would be necessary to collect 100% of the desired information and report that number in the fifth column. Remember that each selected element corresponds to a new path through the DOM tree, so additional element selections correspond to variations in the templates for the 25 randomly selected webpages. The recall of all extraction patterns was 100% due to the nature of structural extraction. The DOM trees of data-driven websites tend to be fairly complex, and thus it is highly unlikely to collect erroneous webpage elements using structural extraction.

Table 3.1: I measure the accuracy of structural extraction patterns on nine popular websites. Websites that use more than one template, such as eBay and Expedia, require more user assistance. The fifth column shows the number of additional specifications necessary to collect 100% of the content.

domain	website	accuracy (%)	initial elements	additional elements
Shopping	Amazon (books)	100	3	0
	Amazon (household)	90	3	4
	eBay (electronics)	82	5	11
	Netflix (films)	100	4	0
Travel	Expedia (hotels)	59	5	17
	Orbitz (hotels)	100	5	0
	Travelocity (hotels)	99	5	1
Academic	ACM Dig. Lib. (papers)	100	6	0
	CiteSeer (papers)	100	4	0
	IEEEXplore (papers)	100	5	0

These results show that popular websites tend to use templates to display content, and structural extraction patterns can robustly extract content from such websites. Two websites that include more variation are eBay and Expedia. Each product webpage on eBay can be customized by the seller, and thus there are a variety of possible presentations. Although not shown in the table, I did measure pages created by the same seller, and one pattern was sufficient for collecting all of the content. Similarly, Expedia uses several different layouts for hotels. Subsequent discussion with Expedia employees uncovered that Expedia has a business model around layout templates; thus, it is to be expected that more expensive hotels will have better visual layouts.

Since my dataset was relatively small, these are only preliminary results but it would not be unusual for a website to employ only a few templates, one for each type of content. When websites use fewer templates, they can make changes to the layout more quickly. Next, I explore webpage changes over time and whether structural extraction patterns can adapt to those changes.

3.2 Structural changes over time

To study webpage changes over time, I conducted a five-month study of the evolution of 12,000 webpages from twenty websites that cover a wide spectrum in both types of content and volume of traffic. Since my goal is to design robust extration algorithms, I analyze the changes in the structure of webpages not the content.

Previous analyses [64, 27] of changes on the Web were motivated by search, caching, and indexing applications and as such focused on changes of the content of webpages rather than their structure. Ntoulas et al. [64] found that most webpages change very little, and that those that do change do so frequently. Their results indicate that creation of new webpages is a much more significant source of change on the Web than changes to existing pages. Fetterly et al. [27] found that large pages tend to undergo many more changes than smaller ones and that the number of past changes is a good predictor of the number of future changes. In addition to degree of change, Lim et al. [54] analyze the clustering of changes in webpages to find that document changes are typically small and clustered. Gibson et al. [30] used the structural content of the HTML tags to help identify template regions of a webpage, but they did not analyze how the structure of the templates changes over time.

3.2.1 Experimental setup

I selected a set of 100 webpages from 24 popular websites. To categorize the types of changes that arise in different types of websites, I chose a wide variety of websites. Since the goal of this study is to inform applications for collecting and organizing Web content and often this type of activity involves commerce, I selected a number of catalog websites. I used `alexa.com`, which tracks website traffic, to select websites that vary in amount of traffic volume. I selected some high-traffic-volume websites such as `amazon.com` and `citysearch.com` and some low-traffic-volume, specialized websites such as `giantrobot.com` and `borders.com`. I also selected websites with dynamic content, such as `nytimes.com` and `flickr.com`. Table 3.2 shows the list of websites I logged and their traffic rankings in November 2006.

From each website I selected one to five webpages. I used up to five pages to reaffirm observed changing patterns for a website. However, some sites, such as blog websites, have only one domi-

Table 3.2: I logged webpages from this set of websites for 5 months (June-Nov. 2006) and collected over 15,000 webpages.

website	ranking	website	ranking	website	ranking
yahoo.com	1	tripadvisor.com	459	tennis.com	27,072
amazon.com	19	travelocity.com	468	nyc.com	28,692
imdb.com	35	citysearch.com	519	photoblogs.com	30,670
flickr.com	38	hotels.com	1,011	stevemadden.com	31,648
nytimes.com	95	epicurious.com	2,910	giantrobot.com	143,718
weather.com	124	bbc.com	4,245	theparamount.com	317,540
target.com	278	concierge.com	8,308	borders.com	358,950
nih.gov	309	michaels.com	9,247	buymusic.com	n/a

nant webpage, which I used. Since this study is about changes in the layout templates, or structure, of webpages, not in the number of layout templates employed by different websites, a few webpages from each website were sufficient for the analysis. Prior work on the evolution of the Web has used a much larger set of webpages. I chose to use a smaller set, but still one of considerable size, so that I could delve deeper with my analysis. This smaller data set allowed me to categorize the types of structural changes that occur on common websites. Prior analysis on the number of layout templates used within one website [64] points to the use of only a few templates; thus, similar types of changes can be expected in all of the similar pages of a website — e.g. product webpages will change in similar ways to one another but not necessarily in a similar way to the front page.

I chose to log only English language websites, but I expect similar patterns of change to occur in foreign language sites as well. While language differences might not cause any structural differences, there might be cultural differences that could affect how often the overall structure of a site changes. That analysis is beyond the scope of this thesis.

Collecting the data

I downloaded the 100 webpages every day for a period of five months from June 1st to November 15th, 2006. As is to be expected, all of the targeted webpages were not always available; however, the pages failed to download fairly infrequently. The only website that was unavailable more then

50% of the time was `imdb.com`, and so I omit those results from my analysis. For two weeks in August I was unable to collect data for any websites due to technical difficulties.

To collect the data I used the GNU Wget software package [63] and registered a process using the `cron` utility to collect data at the same time every day. In the end I collected over 15,000 webpages, which when compressed take 1.5GB of space. Of those I was unable to use 3,000 of the downloads due to bad or missing data. The data set I analyze includes 12,000 webpages from 20 different websites.

Processing the data

I analyze the structural changes in webpages by examining the Document Object Model (DOM). I use Python for my analysis and employ the standard Python DOM implementation. In order to create the DOM, a webpage must be transformed from HTML to XHTML, because HTML is generally not well-formed. For example the `<p>` tag and `<img>` tag need not be closed in order to display an HTML document in a browser. To clean the HTML and convert it to well-formed XML, I use the TidyHTMLTreeBuilder Python library, which uses a library version of the HTML Tidy utility.

To track changes in webpage structure over time, I examine the types of tags present in the HTML webpages. HTML includes over 50 types of tags, some of which are mainly concerned with style, such as the size and look of text (`<b>` or `<h1>`), and others with structure, for example to specify divisions or tables (`<div>` or `<table>`). To analyze the structure of one webpage I could ignore all tags but a small set, but as there are a number of different ways to create webpages and form their layout, I instead chose to remove only elements that are definitely not structural and perform the analysis over the remaining hierarchy. I removed all script tags, which include `<script>` and `<noscript>`. I removed all text and image nodes, as I am not interested in how the webpage content is changing. And finally I removed style tags, because the style tags are closely related to the content and would affect the results for webpages that are highly dynamic and change content frequently. The HTML tags I considered stylistic are `<font>`, `<basefont>`, `<b>`, `<i>`,`<h1>`,...,`<h6>`, `<style>`,`<em>`, `<strong>`,`<u>`, and `<s>`.

Note that today many websites are turning to AJAX as an approach for building rich user experiences. As a result, there is a large amount of Javascript embedded in the webpages. Scripts can often add or remove content from the webpage to dynamically alter the structure. For this analysis

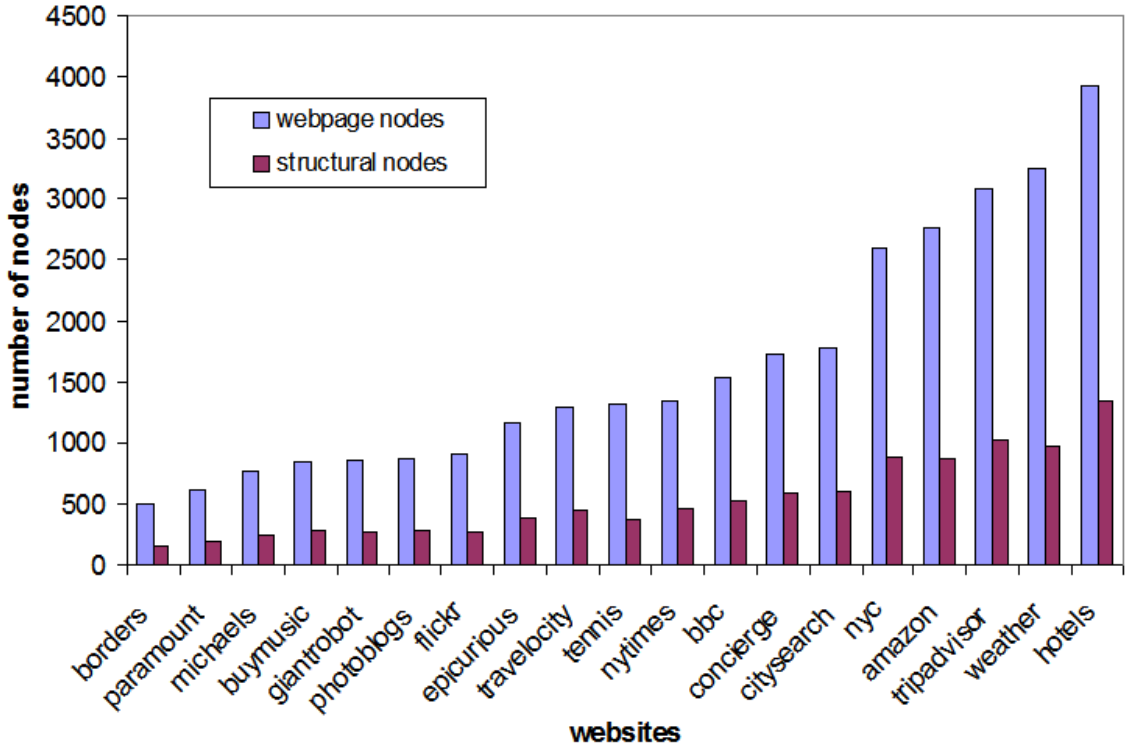


Figure 3.1: The webpages of the websites I logged vary greatly in size. The average number of nodes is 1600, and structural nodes make up approximately 33% of the total number of nodes.

I ignore this behavior as these type of effects tend to be local to a part of the page and do not drastically alter the whole structure. For the rest of this paper I will use the term "structural nodes" to refer to the remaining nodes after the removal of the script, style, and content nodes.

Measuring change

I measure change in the structure of the webpage using the number of structural nodes in the tree. I track if the number of nodes changes from day to day, and compare the trees from consecutive days to find changes in the tree structure. The comparison algorithm takes as input two trees to be compared and recursively traverses the trees. At each node in the tree the algorithm compares the node's tag name and number of children. If the node name and number of children are the same, the algorithm continues; otherwise, I employ one of two strategies. Whenever there is a mismatch in the number of children, the *strict strategy* gives up and counts all of the children and their subtrees as

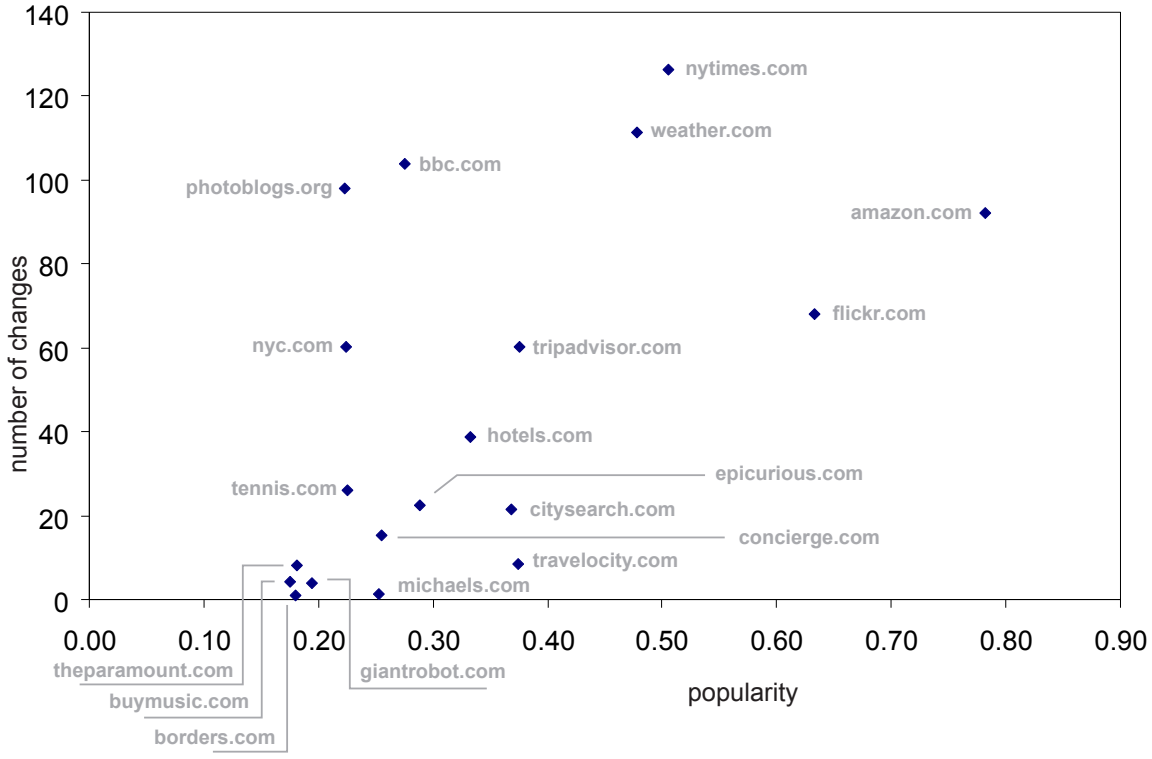


Figure 3.2: This figure shows the relationship between website popularity and the average number of changes. To compute popularity I use traffic volume data from `alexa.com`. In general more popular websites have a higher number of changes. Also, dynamic webpages, such as `photoblogs.org` and `bbc.com` also tend to exhibit more changes, which is in part due to the dynamic nature of the content.

changed. The *flexible strategy* attempts to find a mapping between the different number of children, follows the subtrees as long as it can, and then also gives up and counts all of the unmatched children and subtrees as changed. I use a simplified version of tree alignment, which is employed by some structure-based extraction approaches [35, 95]. In my version, I only account for node insertions at the end of the list of children. Note that this comparison approach gives higher weight to structural changes closer to the root of the tree, because these types of changes tend to be more significant.

I track the location of the changes in the tree with two metrics: the depth of the change from the root of the tree, and the maximum distance of the change from the leaves of the tree. I also perform some qualitative analysis of the type of changes that are occurring and discuss their impact on extraction algorithms in the next section.

3.2.2 Results

First, I describe the general patterns of the data and how changes vary between low-volume and high-volume websites and between websites with relatively static and relatively dynamic content. I then describe some examples of the patterns in more detail and finally give some anecdotal evidence for the types of changes that are occurring.

Figures 3.1-3.3 show a high-level summary of the data. Figure 3.1 shows the distribution of the total number of nodes across the different websites. Structural nodes make up approximately 30% of the nodes for a particular webpage. My findings on changes in the structure of webpages agree with Ntoulas et al.'s conclusions that many webpages do not change very much. Figure 3.2 shows the relationship between popularity and structural changes. I compute popularity using traffic information from `alexa.com`. A low traffic ranking means that the website is very popular. In this figure popularity is equivalent to the inverse of the logarithm of the website's traffic ranking. The figure shows that popular websites tend to change more frequently than less popular websites. There are a few exceptions to this general trend. The `bbc.co.uk` and `photoblogs.com` experience a high number of changes as compared to other websites of similar popularity. This is due to the dynamic nature of their content. Although `nyc.com` has fairly static content for the hotel webpages I logged, during this period the website periodically updated the type of options available to the user, which accounts for a high number of changes. Also, note that traffic ranking is an absolute metric that ignores regional effects. The BBC website is more popular in some parts of the world than others. One can expect similar regional effects for `nyc.com`. Lim et al. [54] studied the proximity of content changes in webpages and found that content changes tend to be clustered. Similarly, I find that most changes occur at the leaves of the DOM tree. And finally, Figure 3.3 shows how the amount of structural change varies with the number of nodes in a webpage. Unlike Fetterly et al., who found a strong correlation between the size of a webpage and the frequency and degree of change, my results show little correlation between the number of nodes and the amount of change.

As a general trend, the number of structural changes increases both with the amount of dynamic content and with the traffic volume of the website. This is the case because many of the changes are due to advertisements and dynamic content, which is much more common on high-volume websites. Most changes to the structure happen very close to the leaves of the webpage, which implies either

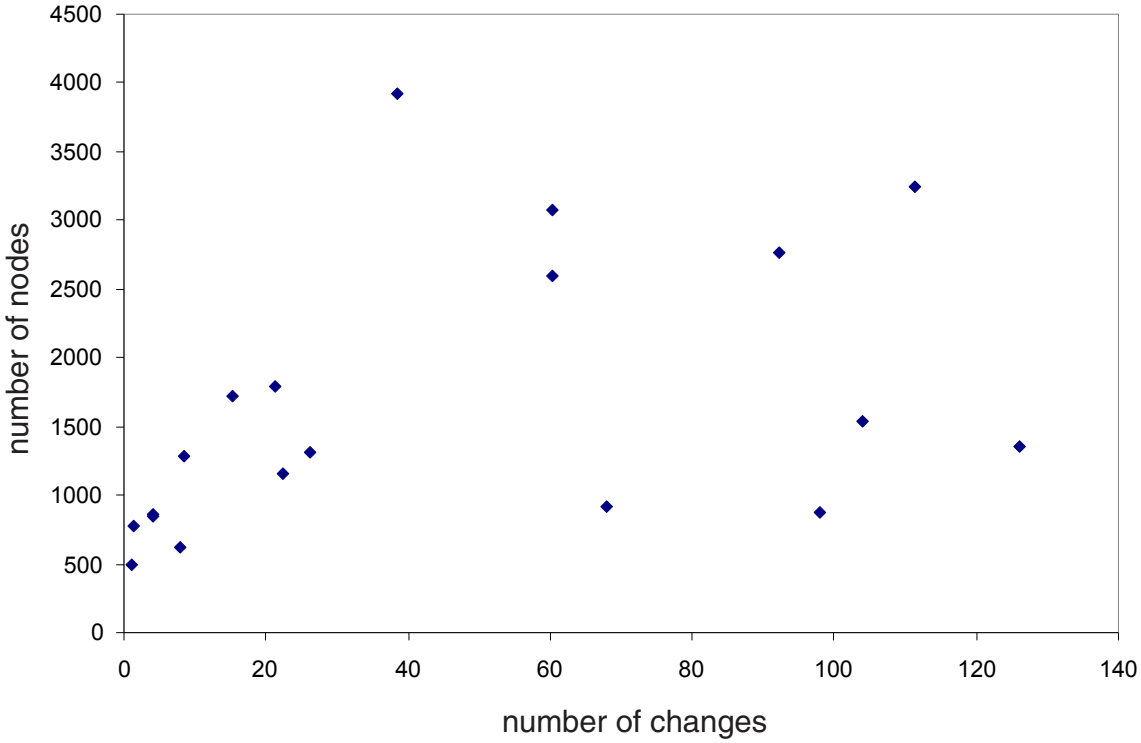


Figure 3.3: This plot shows the relationship between webpage tree size and the number of changes. Previous research finds that large webpages go through more changes than small ones. This correlation is not present in this data set.

content changes or minor adjustments of the layout. Changes higher up in the tree are responsible for larger mismatches in the comparisons and are detrimental to automated extraction algorithms that rely on a consistent structure. These types of large-scale changes are not very frequent — on the order of every couple of months. The flexible comparison strategy matches a superset of webpages relative to the strict comparison strategy. On average, the flexible strategy matched 94% of the tree structure, while the strict comparison strategy matched 91% of the tree structure.

For a more thorough analysis of the changes, I discuss four examples.

A low-volume static website

An example of a low-volume static website is the Easy Street Records website, `buymusic.com`. Figure 3.4 shows four plots of the data for one product webpage. Plot (a) shows five curves for the

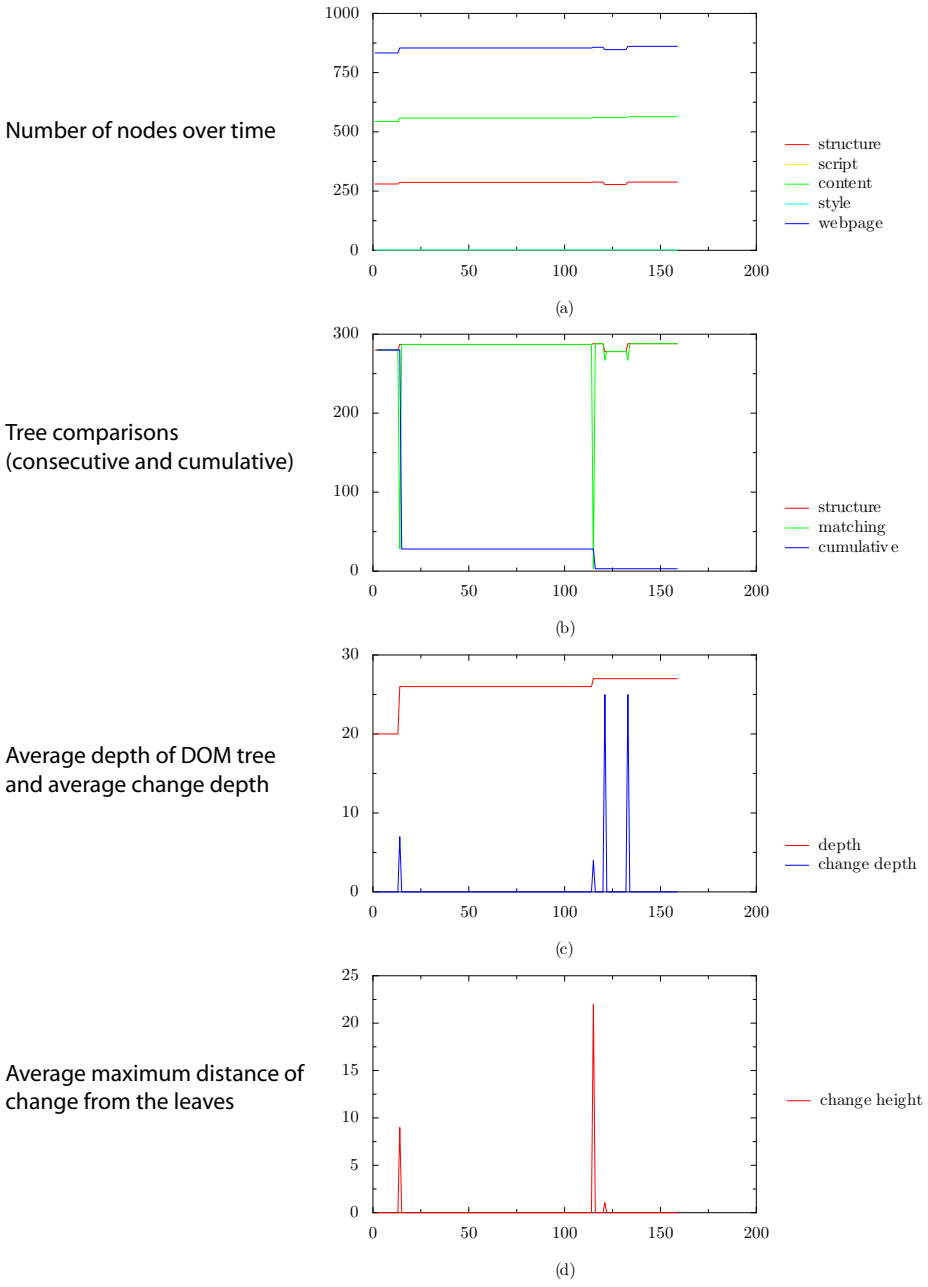


Figure 3.4: Changes in the structure of `buymusic.com`, a low-volume static website. Plot (a) shows the number of structural, style, script, content, and total webpage DOM nodes over time. Plot (b) graphs the number of structural nodes, and the results of the consecutive and cumulative comparisons. Plot (c) shows the average depth of the tree and the average change depth from the root of the tree. A change depth of zero implies that there are no changes. Plot (d) graphs the average maximum distance of the changes from a leaf in the tree. Note that all the plots are aligned to relate the location, magnitude, and type of change. Together these plots show that `buymusic.com` changes infrequently, but when it does the change can be significant.

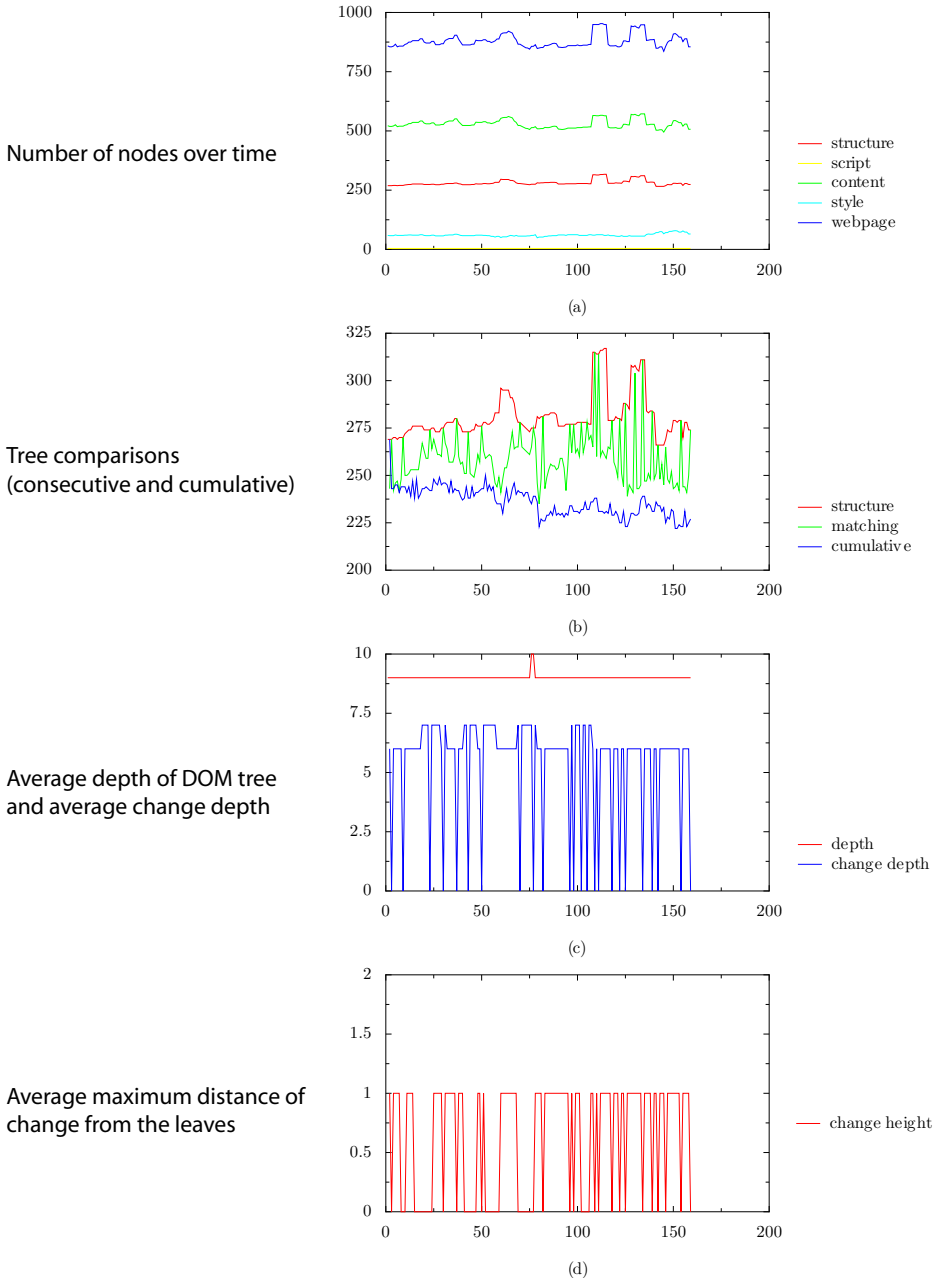


Figure 3.5: Changes in the structure of `photoblogs.com`, a low-volume dynamic website. Please see Figure 3.4 for a description of the characteristics of the four plots. Together these plots show that `photoblogs.com` changes frequently due to the dynamic nature of its content. Approximately 200 of the structural nodes remain static over the five-month period showing no drastic structural changes. The structural changes due to content changes are at approximately the same depth of the tree and are always one link away from a leaf node.

number of nodes in four categories of nodes — structural, content, style, and script — as a function of time. It also shows the total number of nodes on the webpage. The script and style curves are close to zero because the layout for this website is fairly basic. This plot shows that the number of nodes for this particular webpage has not changed significantly over the last five months. Plot (b) shows the number of structural nodes and the number of nodes that matched using the flexible comparison strategy. Remember that only adjacent samples are compared. So, for samples from day 1, 2, 3, and 4, I compare day 1 and day 2, day 2 and day 3, and day 3 and day 4. The graph labeled "matching" shows how many of the structural nodes could be matched between adjacent days using the flexible comparison strategy. This plot shows that there are a very small number of changes, four in total over a period of five months. Of the four changes, two are small and two are significant, as shown by the sharp spikes for days 15 and 115. In this plot, I also show a "cumulative" comparison for the flexible strategy. So for samples from day 1, 2, 3, and 4, I compare how much of the tree remains the same between day 1 and day 2, day 1 and day 3, and day 1 and day 4. Plot (b) shows that the layout stays mostly static and that any changes to the structure are fairly permanent. Plot (c) and (d) show the location of the changes with respect to depth from the root and distance from the leaves. Two of the changes are close to the leaves, which is why they do not affect the total number of matched nodes shown in (b) significantly. The remaining two differences are caused by changes high in the tree.

The structural change patterns at Easy Street Records are very similar to the other low-volume static websites. These websites change the structure of their webpages infrequently, but when they do, they tend to change the structure significantly. Extraction algorithms can adapt to this without too much trouble because the content for the most part remains static. When there is a major change, the most commonly used tree matching algorithms may be successful; however, a more efficient approach might be to first re-find the content of interest by searching through the webpage and then rebuilding the structural extraction patterns. Other websites that exhibit similar change patterns include `borders.com`, `theparamount.com`, `giantrobot.com`, and `michaels.com`.

A low-volume dynamic website

I discuss changes in low-volume dynamic webpages in the context of `photoblogs.com`. Figure 3.5 shows how its structure changes. To illustrate the differences between the different cate-

gories I use the same plot characteristics as in the previous example. Plot (a) shows that the number of nodes in the whole website changes frequently as is to be expected from a blog with frequent new posts. There is a correlation between the content and structure changes, which implies that some structure is defined by the content; however, the magnitude of changes in the structure is much smaller. Plot (b) shows how the structure is changing and how well the flexible matching strategy performs. Since some of the structural changes are due to content changes, fewer of the total structural nodes on the webpage remain static over time. The cumulative comparison graph reveals that up to 225 nodes remain static, and the total number of structural nodes varies by up to 50 nodes. This shows that while overall the structure is changing, it is only changing near the leaves. Plot (d) confirms this behavior as it shows all changes occurring one link away from a leaf. Plot (c) shows that most changes occur at depth 6.

The structural patterns at `photoblogs.com` are similar to other blog websites I logged, and the frequency of change at these low-volume dynamic websites is similar to high-volume dynamic websites. Extraction algorithms for these types of websites must be aware of and adapt to the dynamic nature of the content; however, they can rely on a fairly static webpage layout. The performance of these algorithms on such websites will largely depend on the type of data they are collecting. If, for example, the algorithm is extracting the most recent blog post or the main headline, even a simple extraction algorithm will be appropriate. If, however, the algorithm is extracting details within posts, a more complex algorithm that employs more than structure will be necessary. Other websites that exhibit similar change patterns include `flickr.com` and and `bbc.com`.

A high-volume static website

Figure 3.6 shows the evolution of the structure of one restaurant webpage from `citysearch.com`. Plot (a) shows how the webpage changes overall. This type of website changes more frequently than the low-volume static websites but much less frequently than the dynamic websites. Similar to the `photoblogs.com` graphs, plot (a) shows a correlation between the overall changes and the structural changes, which implies that new content is added and removed from the webpage. This type of content is likely to be new reviews, promotions, or advertisements. On day 93, the total number of nodes decreased dramatically from around 2300 nodes to about 1300 nodes. The total number of structural and content nodes decreased proportionally. This signals a redesign of the

layout of the webpage. Plot (b) shows the effects of the changes. Most changes have small effects on the overall matching because they are close to the leaves of the tree. The webpage redesign on day 93 causes a complete mismatch. Plots (c) and (d) show the location of the changes in the depth of the tree. The changes at day 68, 93, 122, and 136 are closer to the root of the tree; however, only changes on days 68 and 93 cause significant mismatches.

Extraction patterns for these types of websites must adapt to periodic addition or removal of content. Most of these modifications can be accommodated by a tree alignment algorithm that adapts to predictable changes in the content, such as the addition of reviews or promotions. Thresher [35] uses this type of algorithm to find data records in one webpage, and this algorithm could be extended to analyze the same webpage over time. For significant changes, such as re-factoring of websites, tree matching algorithms will not be effective. To adjust to such changes algorithms could employ domain knowledge. Other websites that exhibit similar patterns of change include `nyc.com`, `hotels.com`, `tripadvisor.com`, and `tennis.com`.

A high-volume dynamic website

Figure 3.7 shows how the front page of the "Style" section of the New York Times website evolved over five months. Similar to the low-volume dynamic website, the webpage tree fluctuates in size, and the structural changes correlate with the content changes. Plot (b) shows daily changes in the structure. The effects of these changes remains fairly constant with the flexible matching algorithm matching approximately 60% of the structural nodes. The cumulative matching graph shows that unlike low-volume dynamic websites, high-volume dynamic websites experience structural changes in addition to those due to content changes. Initially the graph drops to 100 nodes, but for about 30 days it hovers around 400 nodes. This pattern is due to the fact that the New York Times website uses several levels of templating. The top level layout template is expressed with 100 nodes including the navigation bars. Each section uses separate layout templates. Plot (c) and (d) show that while most of the daily changes are fairly close to the leaves of the tree, corresponding to news stories, there are a significant number of changes, 9-10 in plot (d), closer to the root.

High-volume dynamic websites go through the highest amount of structural changes, and structural extraction algorithms may not be appropriate for extracting content. As with the low-volume dynamic websites, there is a website layout template, but for high-volume websites that template

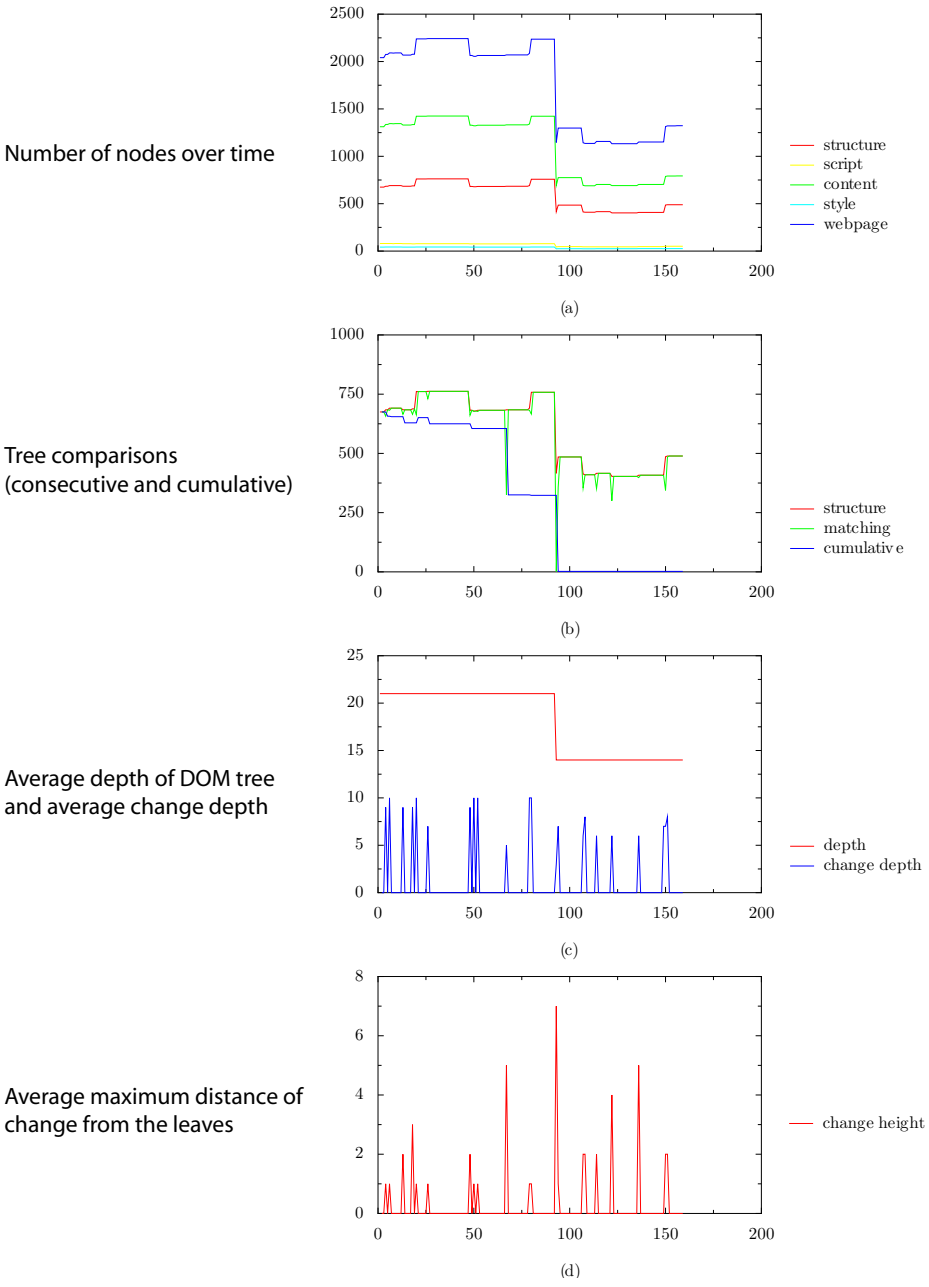


Figure 3.6: Changes in the structure of `citysearch.com`, a high-volume static website. Please see Figure 3.4 for a description of the characteristics of the four plots. Together these plots show that `citysearch.com` changes regularly, on the order of once every 15 days. Most of the changes are close to the leaves and affect the webpage structure in minor ways; however, there are two drastic structural changes at days 68 and 93. Structural extraction algorithms can accommodate the minor changes, but they may have difficulties with the drastic changes.

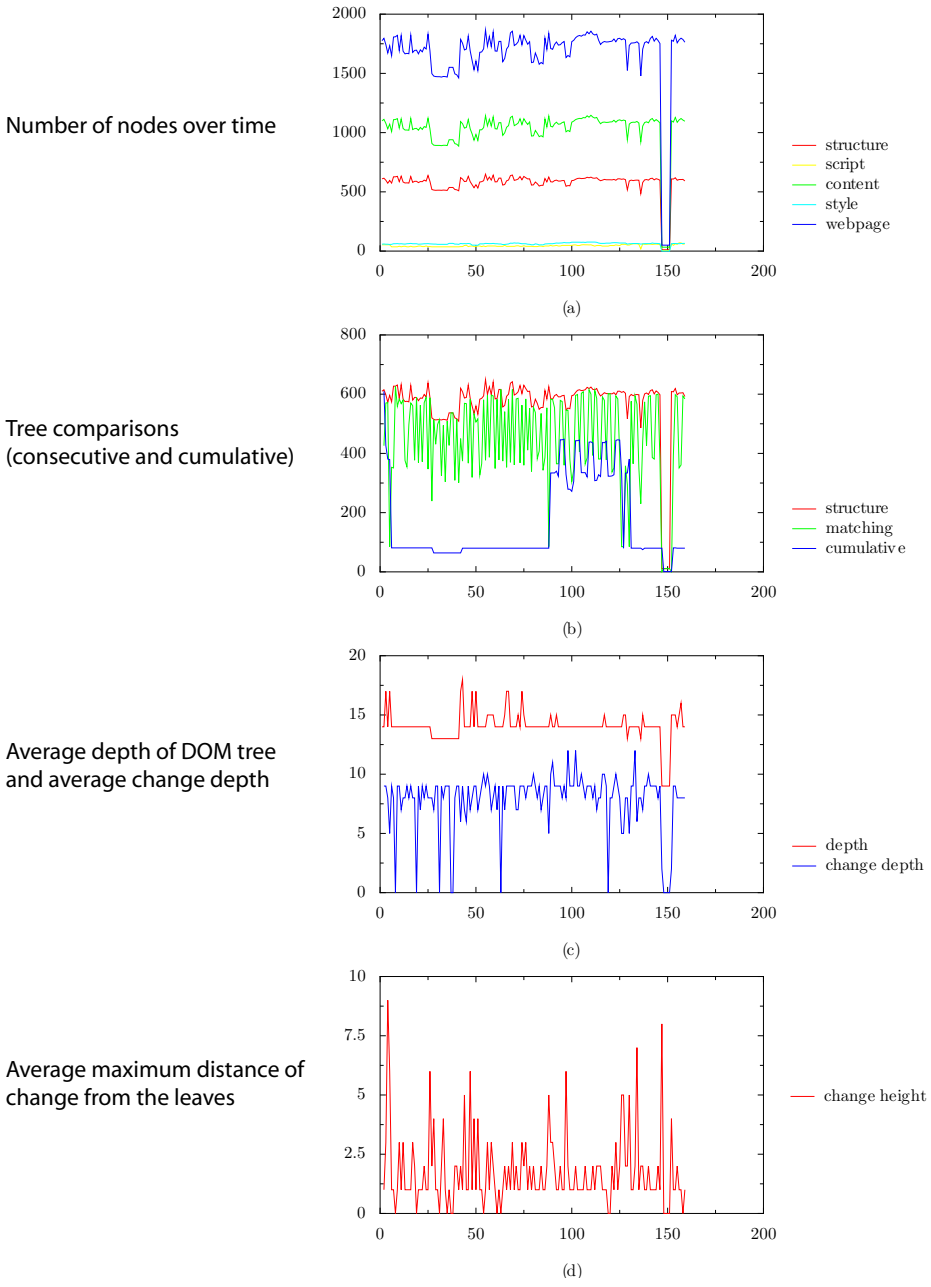


Figure 3.7: Changes in the structure of `nytimes.com`, a high-volume dynamic website. Please see Figure 3.4 for a description of the characteristics of the four plots. Together these plots show that `nytimes.com` changes on a daily basis due to its dynamic content. The magnitude of these changes is approximately 200 nodes and as these changes are primarily at the leaves, they are due to content changes. As Plot (d) shows, however, there are periodic drastic changes. This is due to a hierarchy of layout templates corresponding to different sections for different types of content.

is limited to just the navigation bars. Layout templates for the rest of the webpage often change with respect to the content. Other websites that exhibit similar patterns include `weather.com` and `amazon.com`. Additionally, these are websites that use a lot of Javascript to dynamically modify the webpages. As a result, the analysis may not be as valid, since I remove all script information.

High-volume dynamic websites can be well served by a collaborative effort for extracting content. Since they are visited by many people, many stand to benefit from the efforts of just of a few. Even if the structure of these types of webpages changes frequently, as soon as one person corrects the extraction pattern the rest of the community can continue retrieving content.

3.2.3 Categorizing changes

I also performed a qualitative analysis of the types of changes in webpages in order to get a better understanding for how extraction algorithms can adapt to those changes. There are two types of structural changes, minor changes and major changes. The minor changes are usually close to the leaves of the tree, while major changes usually occur closer to the root and affect a large percentage of the tree.

Minor changes

Most of the changes I observed in all of the webpages are minor changes. I define *minor structural changes* as changes that are either close to the leaves of the tree or involve inserting or deleting a small number of nodes. In such situations, tree matching algorithms can usually determine a mapping between the existing nodes and the new ones. There are several flavors of tree-matching algorithms. The simplest one performs tree matching by using the Levenshtein string matching algorithm [53] to align the children of every two nodes that are compared. More complex ones compare the tree structures and compute mappings between them [86].

From the qualitative analysis, I hypothesize that a simple tree alignment algorithm will be effective for most minor changes. For example, `amazon.com` changes throughout the year to accommodate new events. Figure 3.8 shows the difference in layout for two consecutive days. The only change in the structure is the addition of the Father's Day promotion. A different type of minor change appears at `buymusic.com`. On day 115, the website changed its structure by adding another level in the hierarchy very high in the tree. Because my comparison algorithms do not account

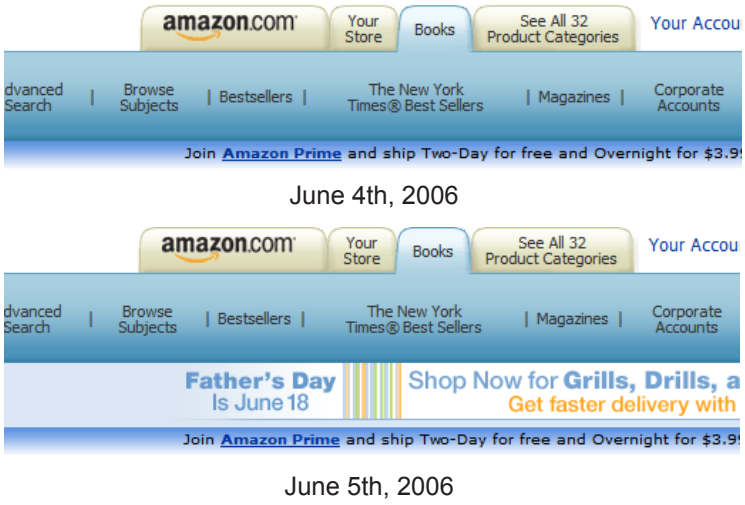


Figure 3.8: On June 5th the structure of the Amazon webpages I logged changed because of a promotion for the upcoming holiday. This type of change is fairly common and can be overcome by structural extraction algorithms because the change is localized and does not affect the rest of the webpage. Furthermore, since it will eventually be removed, algorithms can learn to adapt to these types of modifications over time.

for this, I see a large mismatch in the comparison in Figure 3.4. Structural matching algorithms should be able to account for such changes, and in this situation a full tree matching algorithm will be necessary. Figure 3.9 shows the types of changes common in high-volume websites such as `tripadvisor.com`. New reviews appear in the body of the webpage, but the overall structure does not change.

Major changes

While most structural changes are minor, 74% of the websites I logged exhibited at least one major change. I define *major structural changes* as those that alter the structure of the webpage by modifications close to the root of the tree. Figure 3.6 shows a complete refactoring of the CitySearch website. This type of change is not frequent, as it requires significant human effort. What is more frequent is the modification of a part of the page. For example, in mid-July `borders.com` moved its navigation bar outside of the table that includes the main content. Although this modification was not visible to people visiting the site, the underlying webpage structure changed significantly.

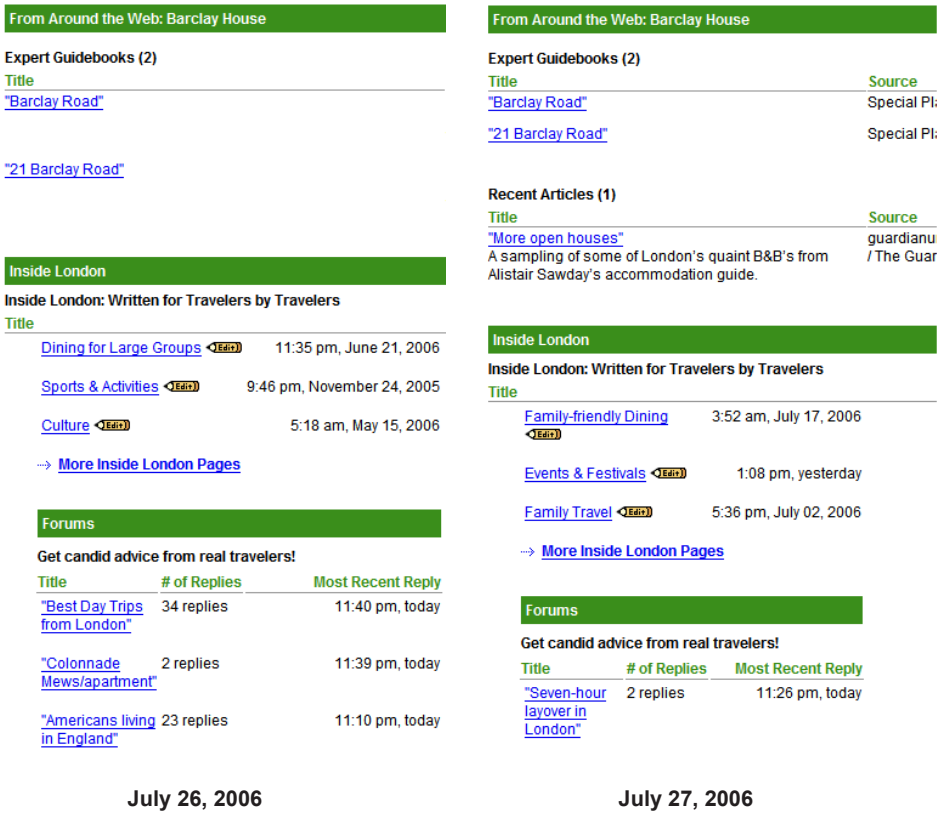
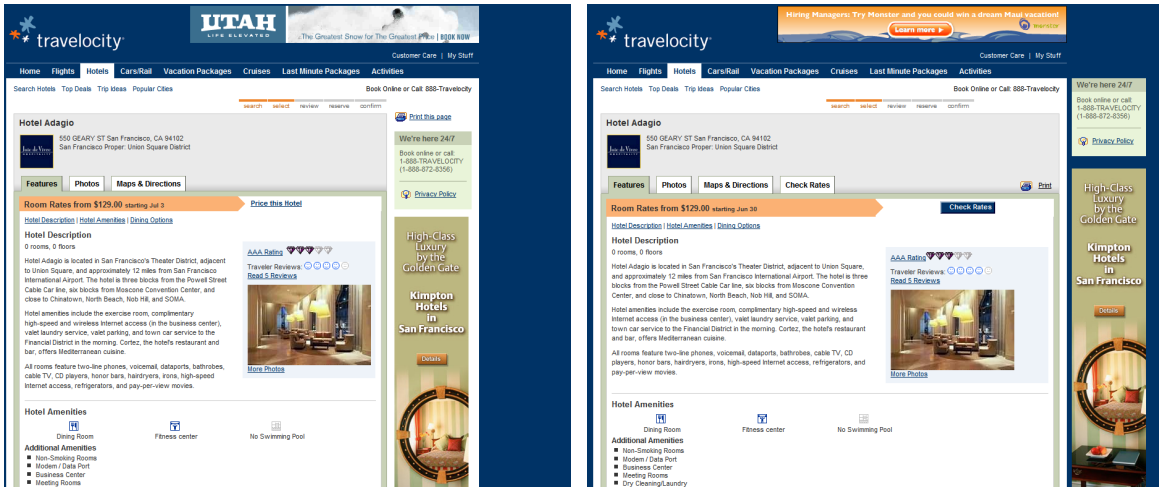


Figure 3.9: A common type of change for all websites is the addition of new reviews. On July 27th, Trip-Advisor updated the reviews and advice they display for the Barclay House Hotel. The addition or removal of content does not significantly affect the structure of the webpage, as the changes happen at the leaves of the tree.

Figure 3.10 demonstrates a similar pattern. Prior to June 29th, the advertisements on the Travelocity website were inside of the table that contains content. Following June 29th, they appear in a column adjacent to the table that contains the main content. Structurally, there is little similarity between the Travelocity website on June 28th and on June 29th.

Structural algorithms have little hope of adapting to major changes using tree matching algorithms. In such situations, the algorithms can rely on domain knowledge to extract content through a wrapper induction algorithm. Alternatively, they can be assisted by a user. Prior research [35, 39, 23], has shown that users can easily assist algorithms in computing extraction wrappers. Users can share their expertise in a community, helping algorithms to adapt to major structural



June 28th, 2006 June 29th, 2006

Figure 3.10: Periodically, although not very often, websites will redesign their layout templates and drastically affect the structure of the webpages. Travelocity changed the structure of their webpages drastically on June 29th. They moved advertisements outside of the main region of content, added new functionality for printing, and created a new tab for checking rates. Structure-based extraction algorithms are sensitive to these types of drastic structural changes.

changes. As soon as one person goes to a website and identifies the content, all subsequent extractions will be successful.

3.3 Discussion

I have presented an in-depth analysis of the structural evolution of webpages. Although the webpages in the data set do not represent all websites, I have sampled a specific space that is representative of the types of websites targeted by extraction algorithms. My analysis leads me to the firm conclusion that structural extraction algorithms have a role in future applications for aggregating and summarizing Web content. While they are not appropriate for all types of webpage changes, they can be effective for most. In the next chapter, I describe Web Summaries, which uses structural extraction algorithms to summarize personal Web browsing sessions.

CHAPTER 4

WEB SUMMARIES

Web Summaries is a semi-automatic system for collecting and organizing Web content. It allows a user to specify webpage **extraction patterns** by interactively clipping webpage elements. The user can then apply these extraction patterns to automatically collect similar content. Web Summaries also includes a technique for creating visual summaries of a collection of content by combining user labeling with predefined **layout templates**. These summaries are interactive in nature: depending on the behaviors encoded in their templates, they may respond to mouse events, in addition to providing a visual summary. Finally, the summaries can be saved or sent to others to continue the research at another place or time.

The Web Summaries system addresses two of the requirements derived in Chapter 2: fluid integration and organization. It is integrated directly into the browser as an extension and uses layout templates to provide rich content organization. Its biggest contribution, however, is the semi-automatic gathering of content through extraction patterns. To describe this tool, I first present the user experience and then discuss the technical details. [1]

4.1 User experience

I designed the interface of Web summaries with the goal of making the process of collecting information as unobtrusive as possible and allowing the user to focus on the task at hand rather than worry about organizing and keeping track of content. The system is implemented as an extension to the Firefox browser and is presented to the user through a toolbar (see Figure 4.1). The toolbar includes four buttons and a checkbox. The "Start" button opens the "summary" window that displays a visual summary of the content gathered thus far. The "Select" button initiates the selection mode and enables the user to select and label pieces of Web content. The "Add Page" button allows the user to automatically add the content found by an extraction pattern to the summary. This button is enabled

[1] The work presented in this chapter was published as a paper [23] at UIST 2006.

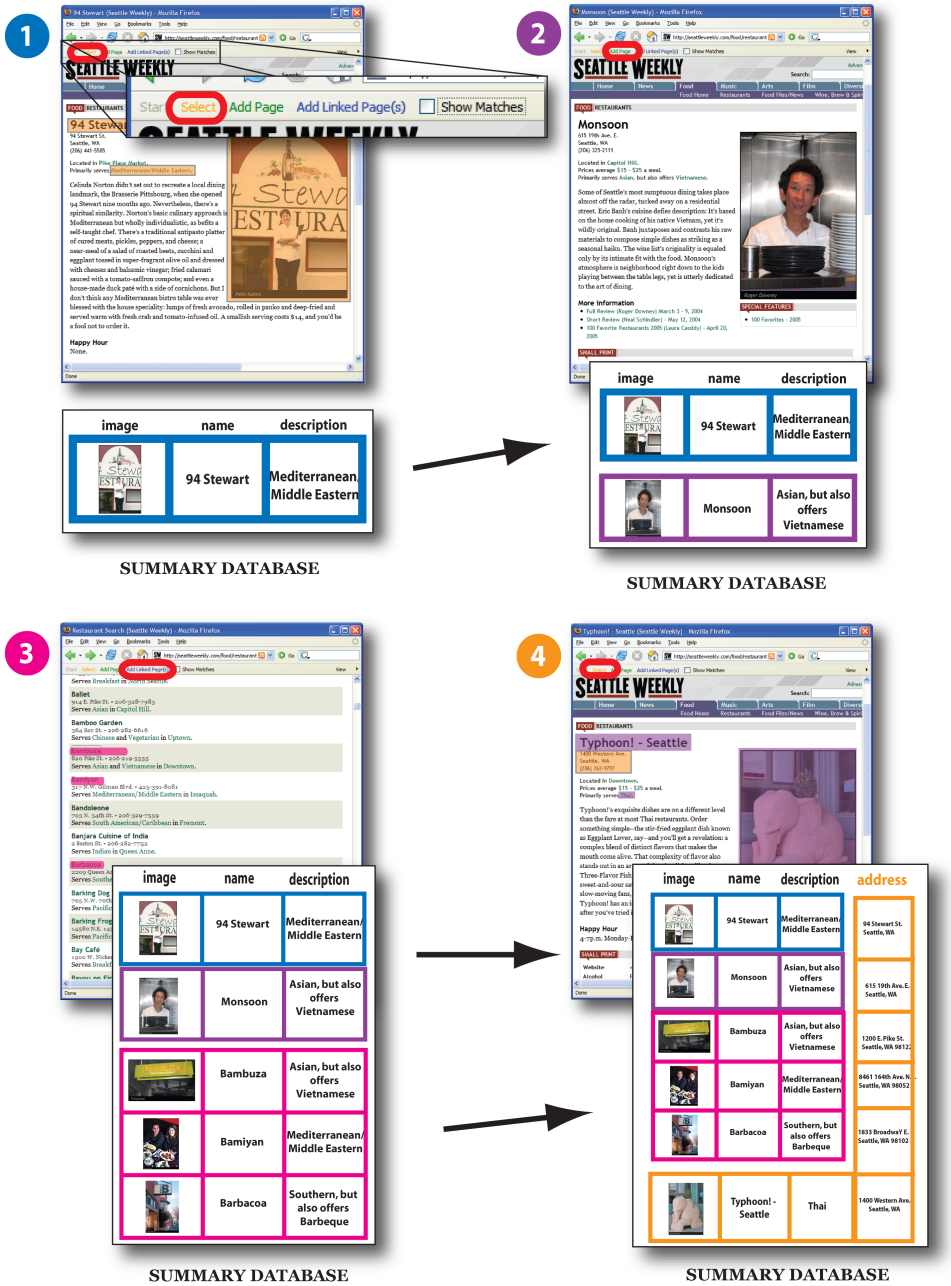


Figure 4.1: In step 1, the user selects the picture, name, and type of cuisine for a restaurant; labels them with "image," "name," and "description"; and adds them to the summary database, shown below the restaurant webpage. When the user finds a new restaurant, as shown in step 2, he can add the same content to the database automatically by pressing the "Add Page" button. In step 3, the user finds a list of restaurants at that website and adds the content for three of the restaurants to the database simultaneously by selecting the linked restaurant names. In step 4, the user selects a new page element, the address, and the system automatically finds all the address elements for the restaurants previously gathered and adds them to the summary database.

only when there is a matching pattern that can be used to extract content automatically. The "Add Linked Page(s)" button initiates a different selection mode in which the user can select any number of hyperlinks to add the content from the linked pages to the summary directly and simultaneously. A checkbox specifies whether to visually outline the elements found by an extraction pattern on a new webpage. If it is checked, the elements found are outlined in purple. The summary window (shown in Figure 4.2a) has buttons for opening and saving summaries and a menu for changing the layout template used to compose the summary.

4.1.1 Sample user scenario

To help explain the interface, I describe an example user scenario and show the steps taken by the user to create a summary. In particular, I describe the steps a user takes in planning a night out in the city of Seattle.

The user starts the browsing session by visiting `www.seattleweekly.com` and looking through restaurant listings. When he finds a restaurant he wants to save to his summary, he presses the "Select" buton, which enables the selection mode. This mode disables the webpage's default functionality to allow the user to select page elements, such as a paragraph or an image. As the user moves the cursor over the webpage, the page element directly underneath the cursor is outlined in red. To select an item, the user clicks with the left mouse button. The outline becomes purple, and the item remains highlighted while he selects other elements. The user also assigns a label to each selected element with a right-button context menu. This label assignment step is necessary because the summary layout templates are defined with respect to these labels. Figure 4.1, step 1, shows that the user has selected three elements and has assigned them the "name," "image," and "description" labels. To finish the selection mode and save the selected elements to the summary, the user turns off the "Select" button. The system stores the selected elements locally and builds an extraction pattern for the selected elements in that page. The user can view the collected content as a summary at any time by going back to the summary window and selecting a layout template, such as a calendar, map, or grid.

When the user finds a new restaurant, all he has to do is press the "Add Page" button and all of the same content is automatically added to the summary (see Figure 4.1, step 2). He can also add several restaurants to the summary simultaneously by selecting hyperlinks (Figure 4.1, step 3).

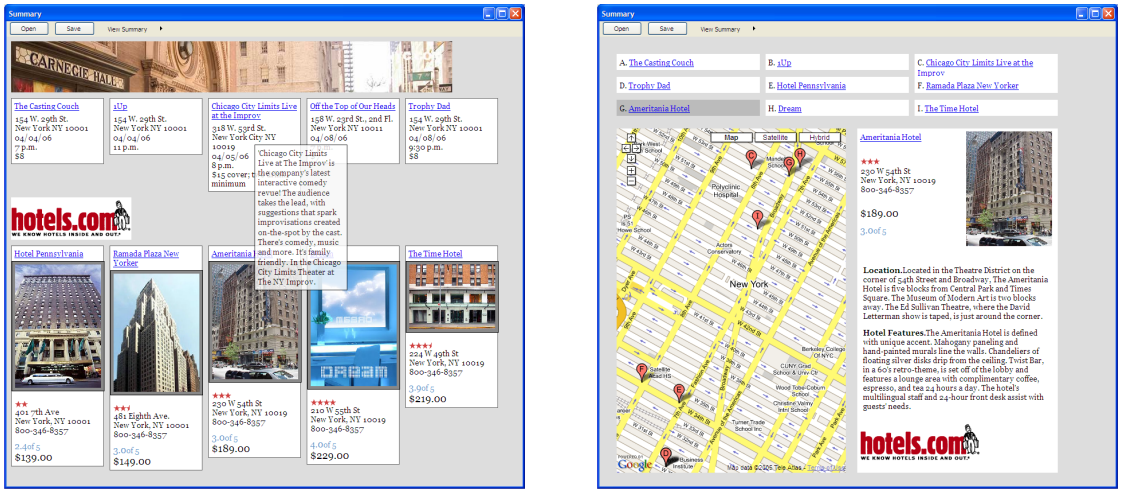


Figure 4.2: **(a)** The grid layout template places all the content in the database on a grid in groups, according to extraction pattern. **(b)** With the map layout template, the user can view the content with respect to a geographical map. The template filters the database and presents only content that includes an address.

To add content simultaneously, the user presses the "Add Linked Page(s)" button, which enables the hyperlink selection mode. To finish selecting hyperlinks, the user turns off the "Add Linked Page(s)" button, and the content from all the linked pages is simultaneously added to the summary.

The user can select new page elements for pages he has already visited, without returning to the page where he first selected the elements. For example, to add the address to all the already gathered restaurants, he can select the address on any restaurant page. He again initiates the selection mode with the "Select" button and selects the address (see Figure 4.1, step 4). The elements corresponding to previously selected page elements are highlighted in purple. When he is finished, the summary is automatically updated to include the addresses of all the events he has already gathered. Because the saved restaurants now include an address, the user can see where they are located using the map layout template (Figure 4.2b).

In addition to gathering information about restaurants, the user can collect any other type of information, such as movies or current events in the area. When he goes to a new website and wants to add content, he must first specify which content for that website is relevant. This is necessary because this system gathers content using the structure of a given webpage, as I describe in the next

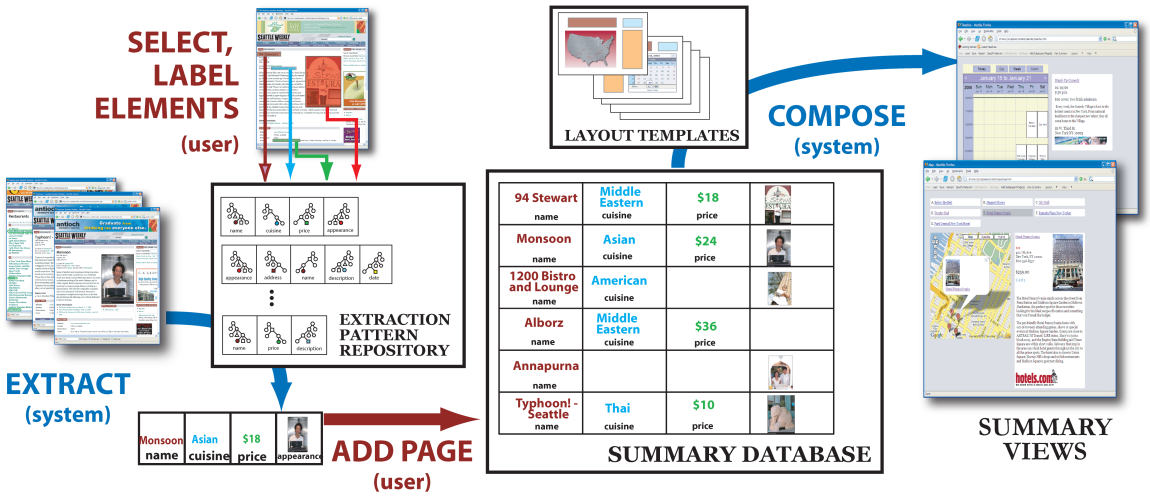


Figure 4.3: The user-selected page elements define extraction patterns that are stored in the extraction pattern repository. Every new page the user visits is compared with the patterns, and if matching page elements are found, the user can add them to the summary database. The layout templates filter the database and compose summary views. The user can view the database contents at any time with any view.

section, and different websites have different structures. If, however, he has been to this website in a previous browsing session and has already specified important elements, he can use the same specification and automatically collect new content. Since the labels assigned to data from different websites are a constrained set, all the content can be presented uniformly in one summary. The user can add content from any number of sites and go back and forth updating the summary. All summary elements are linked to the original webpages, and the user can navigate directly from the summary and return to those webpages.

The Web Summaries experience separates the content presented on the Web from its structure, presentation, and source, and allows the user to create personal summaries of content. This type of interface can lower the overhead of gathering and managing data and allow the user to focus on the task at hand.

4.2 System description

I first describe the overall system design and then explain the technical details for each part. The system includes three components: an extraction pattern repository, a database, and a set of layout

templates. The *extraction pattern repository* contains *patterns* defined from the user-selected page elements. When the user visits a new page, the patterns in the extraction repository are compared with the page. If there are matches, the user can add the matching content to the database. The *database* stores the content according to the user-specified labels and source webpages. The *layout templates* filter the database to create summary views. Please refer to Figure 4.3 for a visual description of the summaries system.

4.3 Gathering content

The page-element selection interface uses the Document Object Model (DOM) structure to provide a mechanism for selecting content. When the user initiates the selection mode, the typical behavior of the webpage is frozen, and the browser mouse event handlers are extended to allow the user to select pieces of the DOM hierarchy. As the user moves the cursor and clicks, the DOM nodes directly underneath the cursor are highlighted. Once the user has selected a set of nodes, the system generates an extraction rule for each selected node. The *extraction rule* consists of the selected node, the path from the root of the document to the selected node, and the user-assigned label. The path enables finding analogous elements in documents with similar structure. Structural extraction rules are also known as XPATH queries. The extraction rules rely on consistent structure. Thus, if the structure changes, the user will have to go back and re-specify extraction rules. Gibson et al. show that template material changes every 2 to 3 months; however, they give few details on the types of changes. To evaluate the performance of structural extraction patterns over time, I conducted a five-month study of webpage changes. I describe those results in detail in Chapter 3.

In addition to the *structural* extraction rules just described, the system also provides content-based rules. *Content-based extraction rules* collect content from new webpages by matching text patterns instead of structural patterns. To specify a content-based rule, the user selects an element and labels it not with a keyword, as he does with the structural rules, but with text from the selected element that should be matched in analogous elements. For example, to only collect articles by author "John" the user selects an article and its author, chooses "semantic rule" from the right-button context menu, and types "John." A content-based rule first tries to find matching content using the structural path, but if it is not successful, the rule searches the entire document. It finds matching content only if the node types match. For example, if the rule finds the word "John" in a `<table>`

node and the selected node defining the rule is in a <p> node, the rule will fail. This limits the number of spurious matches and ensures consistency in the gathered content. The effectiveness and power of content-based rules, when possible, was shown by Bolin et al. [8] in Chickenfoot.

The user may specify any number of extraction rules for a given page. As those rules should always be applied together, the system collects the extraction rules into *extraction patterns* and then stores them in the extraction pattern repository. An extraction pattern can include any number of extraction rules and can be edited by the user to add or remove rules. For example, the user might care about the name, hours, and address of a restaurant. The system creates an extraction rule for each of these elements and then groups them together so that for any new restaurant page, it searches for all three page elements in concert.

When the user visits a webpage, each available extraction pattern for that Web domain is compared with the DOM hierarchy, and the pattern with the highest number of matching rules is selected as the matching pattern. If the user chooses to store the matched content, the webpage is stored locally, and the elements found by the matching pattern are added to the summary database. When the user selects hyperlinks to collect content from multiple pages simultaneously, the system loads the linked pages in a browser not visible to the user, compares the pages with the extraction patterns, and adds all matching elements to the database. If an extraction pattern does not match fully, it may be because some of the content is absent from the page, or because the structure of the page is slightly different. In these cases, the user can augment the extraction pattern by selecting additional elements.

Although the growing use of webpage layout templates for formatting and organizing content on the Web makes it possible to automate collecting information from the Web, this automation comes at a cost. The automatic extraction is sensitive to the structure of HTML documents and depends on a sensible organization to enable the selection of elements of interest. If the structure does not include nodes for individual elements, the user is forced to select and include more content than necessary. On the other hand, if the structure is too fine, the user must select multiple elements, adding overhead to the selection process. Most websites that do use templates tend to use templates with good structure, because good structure makes it easier to automate webpage authoring.

4.4 Summary composition

The database organizes the webpage elements according to the user-assigned label, the page where it was found, and the extraction pattern used to collect it. Since the same set of labels applies to all webpages, layout templates that filter the database to create summaries use the labels rather than the specific HTML content.

A layout template consists of placement and formatting constraints. The *placement constraints* specify how the data should be organized in the summary. For example, a placement constraint can specify that all content with the label "name" be placed in a list at the top of the document. The position of each element can be specified in absolute pixel coordinates or be relative to previously placed elements. For relative placement, the browser layout manager computes the final content position; for absolute placement, the template specifies all final element positions. Placement constraints can be hierarchical. For example, the template designer can specify that content collected from the same webpage be grouped into one visual element and that such groupings be organized in a list. Although the hierarchy can have any depth, in practice I have found that most layout templates include placement constraint hierarchies no deeper than two or three levels. *Formatting constraints* specify the visual appearance of the elements, such as size, spacing, or borders.

Each layout template can also specify mouse and keyboard interactions for the summary. For example, when the user moves the cursor over an item in the calendar, a short description of the event is presented. When the user clicks on the item, a detailed view of that item is displayed in a panel next to the calendar.

The layout templates are implemented with Javascript and Cascading Style Sheet (CSS) style rules. To create the summary, the system loads an empty HTML document and dynamically populates the document with the DOM nodes in the database using the placement constraints of the current layout template. Each node is wrapped in a `<div>` container, and the container's class attribute is set to the label stored with the node. This allows us to specify some of the formatting constraints with CSS style sheets.

I provide "save" and "load" functionality, which makes it possible to share a summary and any specified extraction patterns. The user can share the database without sharing all the locally stored HTML documents, making summaries like those shown in this paper no bigger than 500KB. Since

the system is implemented as a browser extension, a collaborator can install the extension, load an existing summary and its corresponding extraction patterns, and continue the research session.

4.5 Layout templates

To demonstrate different possibilities for summarizing Web content I implemented a set of layout templates. I present two types of layouts: table-based and anchor-based. The table-based layouts organize the content in a grid, and the anchor-based layouts relate the content through a graphical element.

The *grid layout template* (Figure 4.2a) places webpage elements found on the same page into one visual element, a box, and arranges these elements according to an extraction pattern. If there is only one pattern specified for a website, as in Figure 4.2a, the content appears to be arranged according to website. Webpage elements labeled as "description" are available to the user through mouse rollovers.

The *PDA layout template* (Figure 4.4a) separates the content into tabs to make it more accessible on a small-screen device. Each tab contains a list of the elements collected from the same website. The webpage elements labeled as "name" or "image" are displayed in a list on each tab. When the user clicks on a name or image, the elements found on the same webpage as the clicked item appear. For example, on the PDA on the right in Figure 4.4a the user clicked on the "Starlight Review" and is now viewing details about this event.

The *print layout template* (Figure 4.4b) organizes content such that it can be placed on paper. It resizes images and condenses the content to minimize the number of printed pages.

The *text layout template* (Figure 4.5a) is designed for text-intensive tasks, such as literature surveys. The template organizes the names — or, in the case of Figure 4.5a, the paper titles — in a list. The user can click on each title to see more information, such as the paper authors or abstract. If present, a link to the document is placed with the title.

I present two anchor-based layouts, a map and a calendar. An anchor-based layout allows layout of items with respect to a common element. To create this relationship I analyze the collected content. Any element with the label "address" is parsed to find the actual address, while any element with the label "time" is parsed to find the date and time. As in previous work [84], I use heuristics to find the address, date, and time within the selected nodes. While these heuristics are not capable

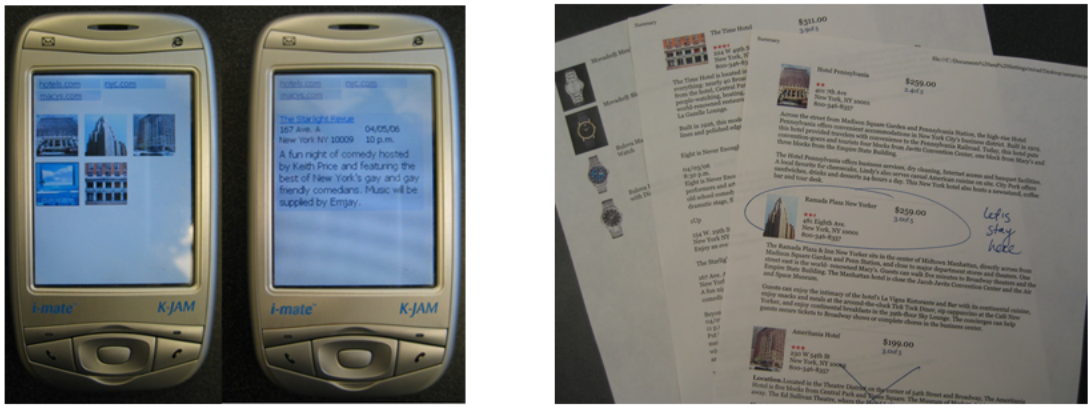


Figure 4.4: **(a)** The PDA layout template is designed for a small-screen device so that the user can take the summary anywhere. The layout separates the content into tabs according to website and provides detailed views when the user clicks on an item. **(b)** The print layout template formats the content so that it can be printed.

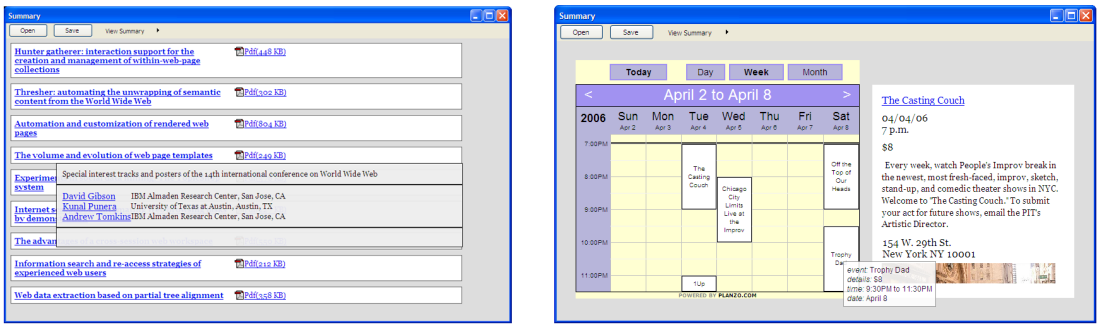


Figure 4.5: **(a)** The text layout template is designed for collecting text content, as is common in a literature search. The paper titles are placed in a list, and further details for each paper, such as the authors and abstract, are available through mouse interactions. **(b)** With the calendar layout template, the user can view content with respect to time. When the user clicks on an event, details about the event are displayed in a panel next to the calendar.

of finding an address in a whole document, they are sufficient for finding the address in the nodes typically selected.

The *map layout template* (Figure 4.2b) has three parts, a list of names at the top, a map in the bottom left, and a detail-view panel in the bottom right. The user can click on a name on the list to see its details in the detail-view panel and its location on the map. The user can also interact with the map and navigate the content geographically. Unlike the other templates, the map layout displays only the content that includes an address. To create a map I use the Google Maps API [31].

The *calendar layout template* (Figure 4.5b) displays a calendar on the left and a detail-view panel on the right. Content that includes a date and time is placed in the calendar, and the user can navigate between the day, week, and month view and click on events in the calendar to see more details about each event. Similar to the map layout template, the calendar layout template filters the database and presents only content that includes a date and time. The implementation of the calendar is provided by the Planzo API [68].

4.6 Exploratory user study

I conducted a first exploratory evaluation of Web Summaries with three questions in mind. First, since Web Summaries exposes the underlying webpage technology, I was curious whether people could effectively specify extraction patterns by selecting from a webpage's underlying Document Object Model. Second, I wanted to know whether users would be happy with the workflow of specifying patterns and subsequently adding extracted items to a summary. Finally, I wanted to find out more about the kinds of operations users felt should be supported in interacting with the summary. I interviewed nine participants, five women and four men. Seven of the participants were graduate students, and the remaining two were staff in the university. All participants were highly active Web researchers, both personally and professionally. Of the nine participants, four had extensive digital collections of content and used a variety of tools for managing their collections including bookmarks, documents, and the file system. One user reported using JabRef, which is a specialized database tool for organizing academic references. The remaining five participants had minimal organization schemes for their professional activities and almost none for personal activities. Similar to the conclusions of the KTFT study [10], those that did not save Web content simply hoped they would find it again.

I performed the study on a WindowsXP desktop machine with 512MB RAM and 1Ghz processor. The machine was connected to the Internet via a well provisioned campus network. The extension was installed on Firefox v1.5. The participants had individual sessions, and each session lasted approximately one hour. Each session included a background questionnaire (10 minutes), a tutorial of the system (10 minutes), three tasks (30 minutes), and a debriefing session (10 minutes). During the tutorial the participants were shown how to create and use an extraction pattern on the website `www.giantrobot.com`. The three tasks were framed as part of the same scenario, which was "a family visit to Seattle." In the first task, the participants were directed to `www.hotels.com` and asked to find five potential hotels. At the end of the first task, the users were asked if certain hotel attributes (name, address, price, image, and review) were included in their summary. If they were not, the users were asked to go back and add those elements to the summary. In the second task, the users were asked to go to `www.seattleweekly.com` and find five suitable restaurants. For the last task, the participants were asked to go to `www.seattleartmuseum.org` and collect any information of their choosing.

4.6.1 Observations and feedback

The amount of aid offered to the participants decreased with each task. For the first task, I made sure the participants understood the pattern specification mode and how to add information using the patterns. In the second task I only guided the users if they asked for help, and by the third task, all participants were using the system independently.

Overall, the participants were very positive about using the tool. Several users asked when the system would be available. One user mentioned, "I would be willing to use this even if it has some bugs, because it would save me so much time. I don't need it to be perfect."

Specifying extraction patterns.

Most of the participants found the process of specifying extraction patterns straightforward. One user noted, "What I like about this is that it's really easy and quick." However, three participants had trouble with the selection interface. Upon further discussion with the participants, I found that this was in part due to the immediate highlighting response to the cursor. In my design, I aimed to make the selection interface quick and responsive, and perhaps I went too far. It would be easy to include

a delay and only highlight items when the cursor hovers over them for some time. Alternatively, the interface could be extended to a click-based interaction model with the user cycling through the available regions by repeatedly clicking with the mouse. Saund et al. [74] showed this type of interface can be effective in their drawing system, ScanScribe.

Four of the participants attempted to select regions of the webpage that were not available through the DOM by highlighting text. While the system could easily be extended to include such cut-and-paste functionality, it is not clear how to make general extraction rules out of text selections. There are two possible scenarios: a text selection that includes DOM nodes fully, and a text selection that includes DOM nodes only partially. In the first case, I can use my current approach and just group the selected DOM nodes together. For the second case, I can use my approach to find matching nodes, but I need an alternative mechanism for finding the corresponding text in the matching node. For example, the address on a page can be embedded in the same node as the name and telephone number. The user might select just the address and expect that I match the address from other pages. For this, I might consider using regular expressions or devising a set of heuristics.

Somewhat surprisingly, four of the participants did not remember to label the elements they selected. I suspect that, for some of the users, requiring labeling at the beginning of the gathering process may not be best. Currently, if the user does not label an element, it is placed in the summary according to the order of selection. One way to solve this problem is to allow label assignment in the visual summary.

Model for collecting information.

Although I designed the interface with the goal of minimizing user effort, at the onset of the user evaluation I was concerned about requiring manual specification of extraction patterns. I was pleasantly surprised to find that all nine participants were happy to specify their own patterns, as it considerably accelerated the rest of the task. One user mentioned, “I’m willing to do that [manually select pieces of content], because to me it’s a method for narrowing down the space of what I’m looking at. By waiting for me to specify what I care about, it allows me to eliminate what I don’t care about.” Another user noted, “No . . . it’s not too much work [to manually select elements] . . . it’s fast. And it’s way better than saving the pages because then I can email the summary instead of emailing the 10 links to the pages.” I was also surprised to observe that all participants used the

"Add Linked Pages" functionality significantly more than the "Add Page" functionality. One user noted, "This way I get a double save. I don't actually visit the other pages and I don't get distracted by all those other things that are on those pages." Another user remarked that once she realized the system worked, she trusted it to collect content independently, and this is what really made the system useful.

Three of the participants requested multi-page extraction patterns. Extending the framework to include content from multiple pages could be done with an approach similar to C3W [28] by recording user actions between pages or allowing the user to interactively specify how the pages are related (through a hyperlink, form, etc). Two of the participants requested sub-page patterns for pages that included lists of items. For example, the `www.hotels.com` page shows not just the name of the hotel, but also a picture and price. The participants wanted to start the summary session by selecting only the name and price of a hotel in that list and then applying that pattern to other items in the list. Web Summaries can be extended to provide such sub-page patterns. Zhai and Liu [95] present an algorithm that automatically extracts all the fields in such a list. I could combine my interface and their partial tree alignment algorithm.

Summary representation

During the three tasks, the participants primarily used the grid and map layout. Six of the participants explicitly commented on the pleasing aesthetics of the summary layout templates. One user mentioned, "This [the summary] is exactly what I make in my word file." As expected, the participants wanted to further organize the summary contents. I received requests for annotating the summary, using metadata to filter content, spatially organizing the elements, and grouping content in pages and folders.

Five of the participants requested mechanisms for configuring both the label choices and the summary layouts. I expected that the set of labels currently defined by the templates might not be meaningful for all users or in all situations. For such cases, it would be best to allow the user to create they own personally meaningful labels. In the next chapter, I discuss a graphical authoring tool for layout templates. I expect that there will be two types of users of Web Summaries. There will be those who use the defaults that I present here and those who would want to create their own layouts and schemas for collecting and presenting information.

Feedback

Overall, the participants were very positive about using Web Summaries. Six of them are willing to use it immediately. Two of the users actually wanted to try their own task immediately following the session. One of them wanted to see how well it would work on `www.ieeexplore.org`, and the other was curious how well it works on `www.amazon.com`.

4.7 Discussion

Web Summaries allows users to transform unstructured Web content into structured data. This structure enables rich organization and interaction through layout templates that can be designed for any size or purpose. In the next chapter, I discuss how users can alter the layout templates and design their own representations. However, it is the automatic retrieval that makes Web Summaries more than just another system for managing Web clippings. Users can automatically collect content by only pointing at hyperlinks that seem interesting. Next, I describe how I extend the automatic retrieval algorithm to collect related content from different websites. Finally, the summaries generated by Web Summaries are just webpages that one can return to at any time, publish on the Web, or share with others.

CHAPTER 5

RELATIONS, CARDS, AND SEARCH TEMPLATES

Extraction patterns help users automatically collect content within one website, but during exploratory Web research tasks users may go to many different websites and collect related information. To aid them with this process I created relations. **Relations** are directional links between data contained in websites. They provide mappings between data from different sources and can be used to automatically retrieve data from those sources. Layout templates organize the content collected by users into rich visualizations but allow little customization of the final summary, which may lead to redundant information or poor use of space. I introduce a design tool for creating personal representations, or **cards**, as a medium for user customization and personalization.

Adding relations and cards to Web Summaries enables easier gathering of content and richer organization, but Web Summaries still remain separate from the initial search process and from the goals of the user. Faaborg and Lieberman [26] envision a goal-oriented Web browser that can help users accomplish their goals. Let's look at the recipe domain as an example. If a Web browser could understand any recipe webpage, then with just the press of a button it could let the user buy the ingredients from a favorite online grocery, compute the total number of calories, or provide a list of pesticides used for any one of the fruits or vegetables. So, what would a goal-oriented tool look like for collecting and organizing Web content? I propose a goal-oriented search interface that lets the user specify the domain, such as finding a good restaurant, and then given a simple keyword query, returns a rich visual summary that allows easy comparison of the results and fast retrieval of additional information. Up until this point Web search personalization has only really addressed reordering the list of hyperlinks typically returned by search engines. My work on **search templates** is the first to propose a general approach to personalizing search results into a rich goal-oriented visual summary. [1]

[1] This chapter was published as a paper [22] at UIST 2007.

5.1 User experience

I describe relations, cards, and search templates in the context of the same example, which shows the steps taken by a user as he looks for a restaurant for a night out in Seattle. Figure 5.1 outlines the process graphically.

5.1.1 Relations

The user begins by visiting `nwsource.com` and collecting information about the restaurant Brasa. He collects the name, address, price range, and neighborhood for Brasa. He then visits `yelp.com`, a favorite review website, finds reviews about Brasa and adds them to his collection. Since the content extracted from `nwsource.com` and `yelp.com` refers to the same restaurant, the user can relate them together. To create a relation, he draws a line from the name, "Brasa," collected from `nwsource.com` to the name, "Brasa Restaurant," collected from `yelp.com` (Figure 5.1a). The system responds by visually joining the extracted content and displaying "Connecting nwsource.com to yelp.com." Now, when the user adds a new restaurant from `nwsource.com` to his collection, the corresponding review from `yelp.com` is automatically retrieved and added to the collection. He can also collect hyperlinks pointing to potentially interesting restaurants at `nwsource.com`, and the reviews for those restaurants will be automatically collected from `yelp.com` (Figure 5.1b).

Upon inspecting his collection of restaurants, the user decides that he might need to take the bus to his dinner destination. He visits the website `metrokc.gov` and finds bus information for the downtown area. He adds the bus information to his collection and interactively connects the neighborhood of Brasa — "downtown" — to the bus route information (Figure 5.1a). Now when he collects a new restaurant, the system will automatically collect reviews for the restaurant from `yelp.com` and bus schedules from `metrokc.gov`. To retrieve the bus schedules for all the restaurants he has already collected, the user clicks on the "Update All" button, and the corresponding bus schedules are retrieved automatically.

If the retrieved results are not the correct ones, the user can click on the icon in the bottom right corner and see any other content that was retrieved. For example, if he clicks on the icon for the "Brasa Restaurant" from `yelp.com`, he will see a lit of other "Brasa" restaurants that were retrieved from `yelp.com`, such as "Pollo Brasa y Sazon," "Pollo a La Brasa Vermont," "Fogo E

Brasa," or "Pollos A La Brasa Marion." The user can at any time pick the best content from this list or return to a website to collect new information.

5.1.2 Cards

As the user collects more content, his collection space quickly fills up. The user can address the growing clutter by creating a specialized card that displays only the information of interest. To create a card, the user clicks on the "New Card" button and opens a canvas for card design. He first draws the outline of the card and then draws containers inside of the card. He places content from his collection directly into the containers. The user can resize and reposition the containers until he is satisfied. He clicks "Done," names the card, and can now view all the related collected content as personalized cards (see Figure 5.1c). Cards can be designed for any size or purpose and can contain information from any number of source webpages. They also include hyperlinks to the original webpages, so the user can at any time return to the original content.

5.1.3 Search templates

In addition to collecting content by visiting actual websites, the user can also collect content through keyword queries and a search template. Thus, a user can go directly from a query term, such as "seafood," to a series of cards filled with content from multiple sources (see Figure 5.1d). The system collects seafood restaurants and any related content. Search results are considered only temporary and are not part of the user's collection. Thus, they are displayed separately, below the existing collection. The user can promote any search result to the actual collection by clicking on the "+" button in the upper left corner. He can also delete a card with the "x" button. This way the user can quickly scan through many restaurants and identify good options.

A search template is defined implicitly by a user-defined card. To create a search template, the user clicks on the "New Search Template" button, selects a card, and can optionally add additional websites or relations to be part of the template. A search template allows the user to package all of the work he has already done in collecting and associating content and use it to find new content more efficiently. Figure 5.2 shows a query for "chocolate cake" with the "recipes" search template, which retrieves and reformats recipes from `allrecipes.com` and `cooking.com`. Figure 5.3

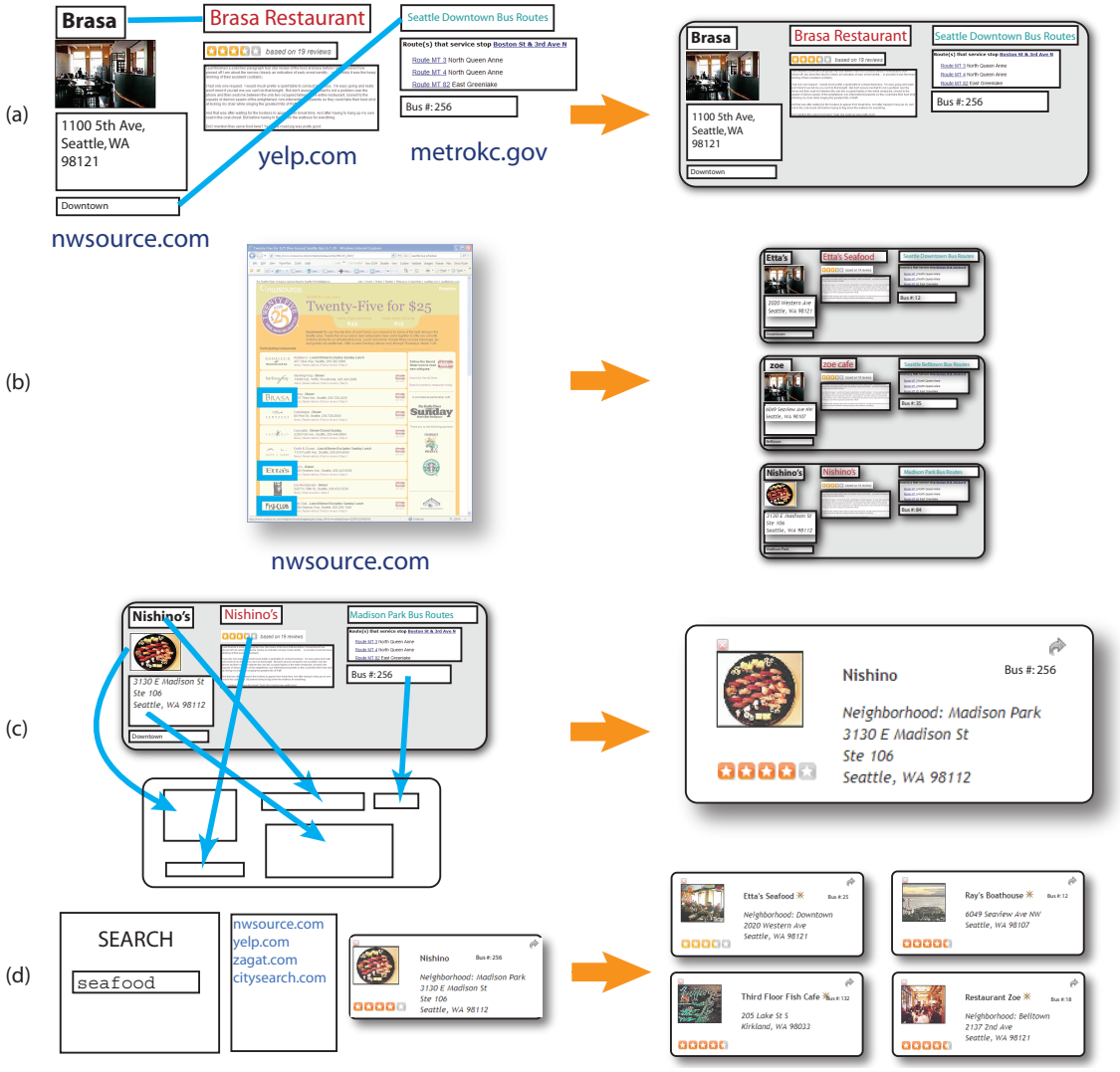


Figure 5.1: **(a)** The user draws lines between webpage elements, thereby creating directional relations from `nwsource.com` to `yelp.com` and `metrokc.gov`. **(b)** The user can collect more restaurants by selecting hyperlinks that point to interesting restaurants. Each new restaurant triggers the system to automatically collect related content through the user-defined relations. **(c)** To display the restaurants more concisely and uniformly, the user designs a card and assigns content to each card container. **(d)** Search templates are defined using a card. They include a set of websites and relations. With a search template, the user can type a keyword and retrieve a series of restaurant cards.

shows several queries for different types of cars. Each card includes car specifications and reviews from `autos.msn.com` and `edmunds.com`.

5.2 System overview

To include relations, cards, and search templates in Web Summaries, I added several new components to the system design presented in Chapter 4. In addition to a data repository, a collection of extraction patterns, and a set of layout templates, the system includes a set of user-defined cards and a set of search templates. The *data repository* includes relation trees and the summary database that holds all of the content collected by the user according to the source webpage and semantic tags for the webpage elements. I refer to each piece of content collected by the user as a *webpage element*. Each webpage element is associated with a semantic tag, such as name, address, date, time, etc. All webpage elements collected from the same webpage form a record in the data repository. The data repository also holds *relations*, which specify relationships between tags in different records. Records that are related in this way form a *relation tree*.

The user can view collected Web content through cards. A *card* defines which webpage elements within a relation tree should be displayed and their visual arrangement.

The user can collect data by visiting webpages or through search templates. A *search template* includes a set of websites and possibly relations for those websites. When the user types a keyword query, the search template queries each of the websites with the keyword, extracts content from the search results with extraction patterns, triggers the collection of related content, and displays them as a series of cards.

5.3 Retrieval using relationships

All of the content collected by the user is associated with semantic metadata or tags. Web Summaries enforces that users tag the content they clip, but any other extraction algorithm would also provide this semantic information. When the user connects content collected from different webpages, he creates a relation. I define a *relation* as a directed connection from tag_i from $website_A$ to tag_j from $website_B$. For example, when the user draws a line between the names of the restaurants, he creates a relation from the "name" tag on Northwest Source to the "name" tag on Yelp. When the user connects restaurants and buses, he creates a relation from the "area" tag to the "route" tag.

All relations are stored in the data repository and are available to the user at any time. Webpage elements that are associated through relations form a relation tree.

When the user collects content from a new webpage, the system checks for relations that connect any of the collected webpage elements to other websites. When such relations exist, the system uses them to generate new search queries and limits the search results to the website specified in the relation. For example, when the user collects information about the restaurant "Nell's," the system generates two queries. To collect restaurant reviews it generates a query using the "name" tag, i.e. "Nell's," and limits the search results to `yelp.com`. To collect bus schedules the system generates a query using the "area" tag, i.e. "Green Lake," and limits the search results to the bus website, `metrokc.gov`.

To define this process more formally, the execution of a relation can be expressed as a database query. For a given relation r, where $r = website_A.tag_i \rightarrow website_B.tag_j$, one can express the process of automatically collecting content for any new data record from $website_A$ for tag_i as a `JOIN` operation or the following SQL pseudo-query:

`SELECT` $\star$ `FROM` $website_B$ `WHERE` $website_B.tag_j = website_A.tag_i$

Since the Web is not made up of a set of uniform databases, I use a number of different techniques to make this query feasible. I use the Google Search AJAX API to find webpages within $website_B$ that are relevant. To extract content from each of the search results, I employ the user-defined extraction patterns I discussed in Chapter 4. Finally, I designed a number of heuristics to compute a similarity metric and rank the extracted search results. The system displays only the highest ranked extracted search result to the user but makes the remaining search results available.

In the current implementation, the system extracts content from only eight search results because the Google AJAX Search API limits the search results to a maximum of eight. For all of the examples in this chapter, eight search results are sufficient and limit the delay in collecting information. For very common keywords, however, collecting eight search results is not sufficient. For example, searching for "Chili's" will yield many instances of the restaurant chain. For those types of queries narrowing the search through one of the approaches mentioned above would be necessary.

My approach for collecting related content is limited to websites that are indexed by general search engines. There are many websites, such as many travel websites, that are not indexed by search engines because they create webpages dynamically in response to user input. To handle these

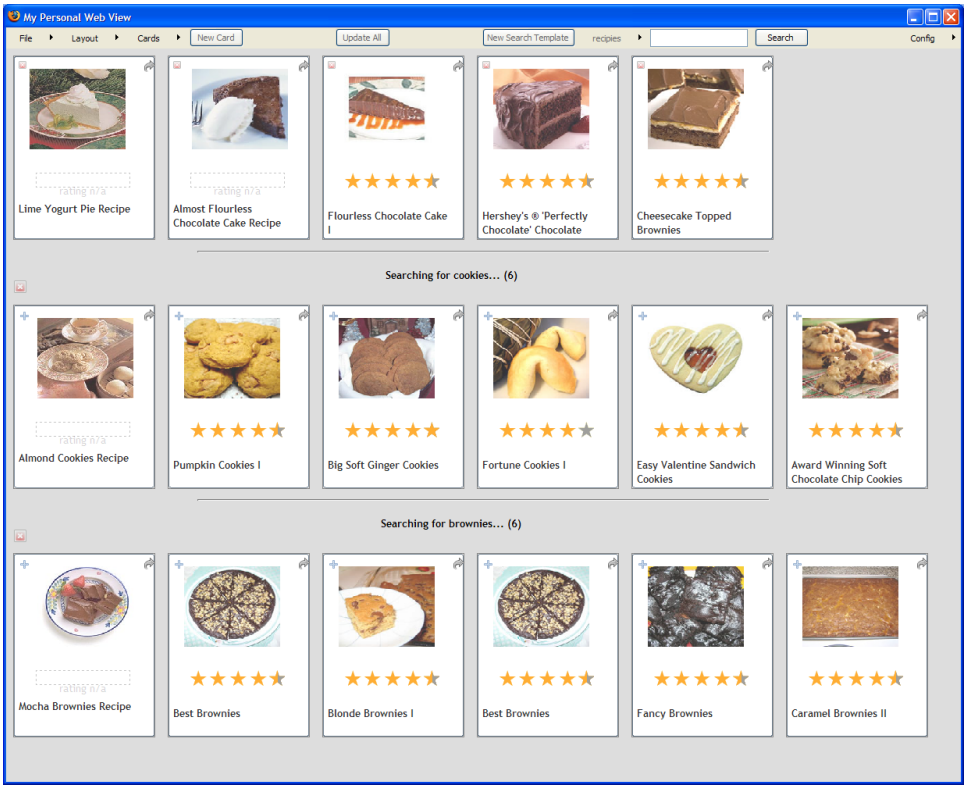


Figure 5.2: With the recipe search template, the user collects recipes from `cooking.com` and `allrecipes.com`. Here the user has a collection of five cards and has made two queries, one for "cookies" and and another for "brownies". The system automatically collects, extracts, and displays relevant recipes.

dynamic webpages, in subsequent work I hope to leverage research into macro recording systems such as WebVCR [4], Turquoise [60], Web Macros [73], TrIAs [7], PLOW [44], and Creo [26]. These systems allow users to record a series of interactions, store them as scripts, and replay them at any time to retrieve dynamic pages. Recent research on retroactive macro recording could be even more applicable for Web Summaries [36]. Madhavan et al. [56] are developing information retrieval approaches to this problem that do not require user intervention.

The search process introduces ambiguity at two levels, the query level and the search result level. The system must be able to formulate a good query so that it can retrieve the relevant content. It must also be able to find the correct result among potentially many that may all appear similar. Both of these forms of ambiguity pose considerable challenges and are active areas of research. Liu et al. [55] pose the query formulation problem as a graph partitioning problem. Dong et al. [20]

propose propagating information across relations to better inform similarity computation. Next, I describe how I address these two types of ambiguity.

5.3.1 Query formulation

I formulate two keyword queries in parallel. One query includes only the extracted text content, and the other includes the extracted content and the tag associated with the content. These two queries are usually sufficient to collect the appropriate result within the top eight search results. Sometimes, however, queries may include too many keywords, and the search results are irrelevant or cannot be extracted. In such cases, I employ heuristics to reformulate the query. If characters such as '/','-','+', or ':' appear in the text, I split the string whenever they appear and issue several queries using the partial strings. I found this approach particularly effective for situations in which something is described in multiple ways or is part of multiple categories. For example, a yoga pose has a Sanskrit name and an English name. Querying for either name returns results, but querying for both does not, as the query becomes too specific. Other approaches for reformulating queries include using the semantic tag associated with the webpage element or using additional webpage elements, such as the address, to make the query more or less specific. With an interactive system, processing a large number of queries can be prohibitive due to the delay caused by the search and extraction process. I focus on finding good heuristics that quickly retrieve results that are close to the desired content. If the system fails to find a good search result, the user can always go to the website and collect the content interactively.

5.3.2 Search result comparison

For each query I extract the first eight search results and rank the extracted content according to similarity to the webpage content that triggered the query. To compute similarity I compare the text of the extracted webpage elements using the correspondence specified in the relation that triggered the search. For example when collecting content for the "Ambrosia" restaurant from `nwsource.com`, the system issues the query "Ambrosia" limiting the results to the `yelp.com` domain. The search results include reviews for the following establishments: "Ambrosia Bakery" (in San Francisco), "Cafe Ambrosia" (in Long Beach), "Cafe Ambrosia" (in Evanston), "Ambrosia Cafe" (in Chicago), "Ambrosia on Huntington" (in Boston), "Ambrosia Cafe" (in Seattle), and "Caffe Ambrosia" (in

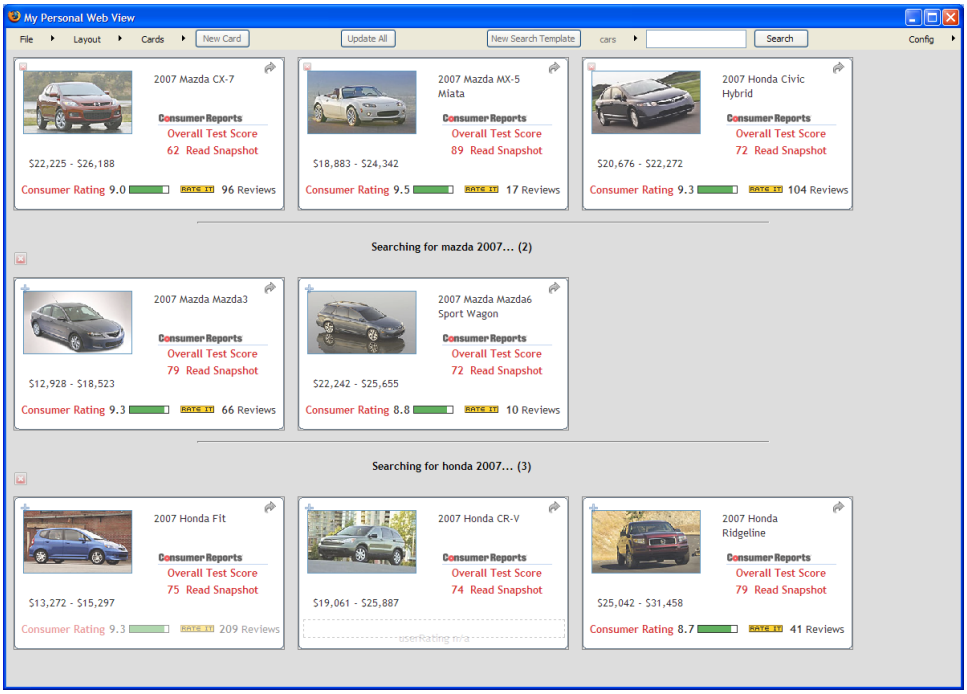


Figure 5.3: The user is shopping for cars and using a car search template to find new cars. He has three cards in his collection and has made two queries: “mazda 2007” and “honda 2007.” Each card includes car reviews from `autos.msn.com` and `edmunds.com`.

San Francisco). Because the relation between `nwsource.com` and `yelp.com` links the names of the restaurants, I compare the name “Ambrosia” to all the names of the extracted restaurants. I compare the strings by calculating the longest common substring. I give more weight to any strings that match exactly. For all seven restaurants in this example, the longest common substring is of length eight; thus, they receive equal weight. Next, I compare any additional extracted elements. I again compute the longest common substring for corresponding webpage elements. In this example, I compare the addresses of the extracted restaurants and compute the longest common substring for each pair of addresses, resulting in a ranking that places the Seattle restaurant “Ambrosia Cafe” as the best match to the original content. I display the highest ranked extracted content but provide all of the extracted content to the user so that he can correct any errors. The highest ranked content is marked as confident when multiple webpage elements match between websites.

The problem of retrieving related information from multiple different sources is described in the database community as data integration [33]. Data integration is the problem of combining

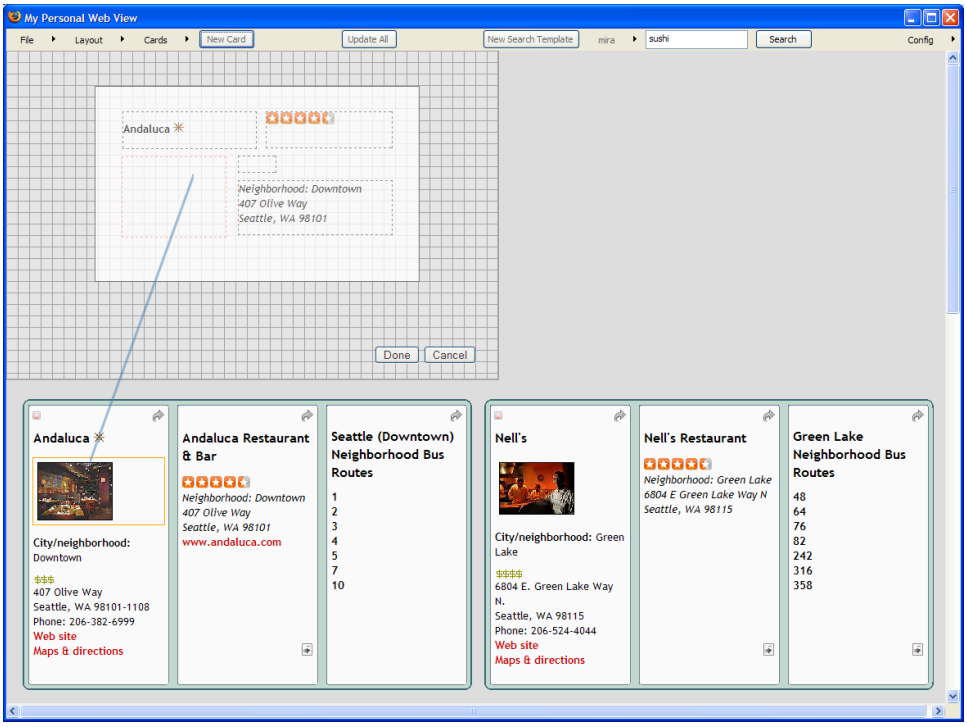


Figure 5.4: To design a new card, the user opens a canvas and begins drawing. He first draws the outline of the card and then draws containers. To place content in the containers, the user draws a line from the webpage element to the container. Here, the user adds an image to the card.

data residing in different sources and providing the user with a unified view of these data. The difficulty in data integration lies in forming mappings between heterogeneous data sources that may include different types of data and defining one single query interface for all of the sources. This problem emerges in a variety of situations both commercial (when two similar companies need to merge their databases) and scientific (combining research results from different bioinformatics repositories). With the growth of the Web, database researchers have shifted their focus towards data integration of unstructured Web content and its applications to Web search [56]. My work is complementary to database research in that it offers interactive techniques for data integration on a personal scale. I provide an interface that allows users to specify mappings between different data sources, i.e. websites, and then use these mappings to automatically extract content from the Web.

Finally, the current implementation allows the user to specify only one-to-one relations. In some situations a one-to-many relation is more appropriate; for example, if the user is interested in

collecting academic papers and wants to collect all of the papers written by each of the authors for any given publication. The system can actually query for all of the papers by a given author, but it is not designed to let the user view all elements of the collection as relevant. In future work, I plan to explore other types of relations and also introduce transformations into the relations.

5.4 Authoring cards

The user can view his collection of Web content through cards. A card imposes a uniform design on content that may come from many different websites and may initially appear very different. It defines which content should be displayed and how the content should be organized visually. Recall that the user's content collection is stored in the data repository and is accessible through relation trees. It is the relation trees that specify which records in the data repository are related. In database terminology, a card can also be described as defining a view on the relation trees in the data repository — i.e., it lists the tags for the data that should be displayed in the card. For example, a card can include the name and address of a restaurant. Or it can also include pricing, rating, and images. The system includes a default card, which displays all records and webpage elements in the relation tree irrespective of the tags associated with the webpage elements. The user can use an interactive editing tool to create new cards at any time. Cards are persistent, can be reused, and shared with others.

To create a new card the user clicks on the "New Card" button, which opens a canvas directly in the summary (see Figure 5.4). The card designer is tightly integrated with the collection space so that the user can quickly and easily merge content from different websites without switching windows or views. The user first draws the outline of the card and then proceeds to create containers that hold the webpage elements. To assign data to a container, the user clicks on a webpage element and drags a line to the container. The data is automatically resized to fit the size of the container. Each container is associated with the tag of the webpage element it contains and the element website. If the element was not marked as confident during the automatic ranking process, it is rendered semi-transparent to alert users that they may want to confirm the information by visiting the source webpage. When the user is finished creating containers and assigning data, he clicks on the "Done" button, and the system transforms his drawing into a template for organizing and displaying Web content, a card. The user can at any time edit the card and add or remove content. Currently,

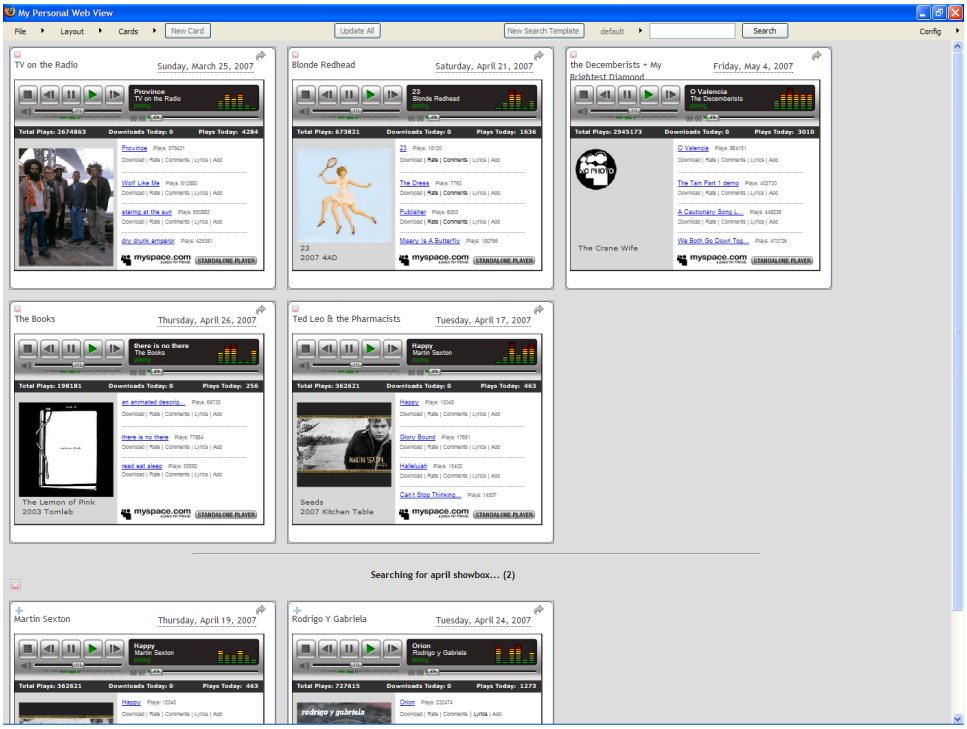


Figure 5.5: The user relates `upcoming.org` to `myspace.com` to automatically collect music samples for upcoming shows.

the cards are presented in a grid, but they could also be manually organized into piles or arranged visually on a map or calendar. The authoring principles behind the card designer can also extend to authoring layout templates that organize the cards.

Figure 5.5 shows a collection of upcoming shows. In this example the user has related concerts from `upcoming.org` with band webpages at `myspace.com`. Whenever the user adds a new concert to his collection, the system automatically collects music samples. The music samples are embedded in a music player, and because the player is just another HTML object, a Flash object, one can extract it just like any other webpage element. The music player retains full functionality, and the user can press the "play" button on the control to listen to the music.

Cards can be interactive in nature, encoding interactions specific to the type of data that is collected. The card authoring tool does not currently provide capabilities for specifying interactions. I could expose a scripting interface and allow the user to specify any type of card interactions, but since Web Summaries is designed for the novice I chose to remove all scripting from the interactions.

5.5 Template-based search

Search templates combine the user designed cards and relations as a basis for a new type of search interface that targets the keyword query to appropriate domains and organizes the search results in a visual summary. The user can thus bypass visiting webpages directly and collect content through a search template. Figure 5.6 shows a visual description of the gathering process. For example, if the user wants to find vegetarian restaurants but does not know where to start, he can simply query with the word "vegetarian" directed towards the data underlying the restaurant card. More formally, a *search template* includes a set of websites and any associated relations. When a user types a query to the search template (Figure 5.6a), the system sends the query to a general search engine (Figure 5.6b), in this case through the Google Search AJAX API, limiting the search results to the list of websites defined in the template. For each search result, the system extracts content using predefined extraction patterns (Figure 5.6c) and triggers any relations that are in the template to collect additional content (Figure 5.6d-e). Due to limitations on the number of search results provided by the Google Search AJAX API, for each query/website pair, the system processes only eight search results. The user can also modify the search template by adding additional websites to be queried and relations to be triggered. Extracted search results are presented to the user as a set of cards (Figure 5.6f). These are initially considered temporary, indicated by being displayed below the main collection of cards. The user can promote a search result to the actual collection or he can delete all of the search results for a given query.

The success of template-based search lies in the ability to extract semantic information from webpages. Although semantic extraction is only in its infancy, I believe that it will only grow in the coming years, and template-based search is an example of the powerful new applications that will take advantage of machine-readable webpages.

5.6 Exploratory user study

I conducted a laboratory exploratory study to solicit feedback on the extensions to Web Summaries. I interviewed six participants, three women and three men. Four of the participants were graduate students, and the remaining two were staff in the university. All of the participants were active Web researchers and reported using the Web for both personal and professional research often, from a few

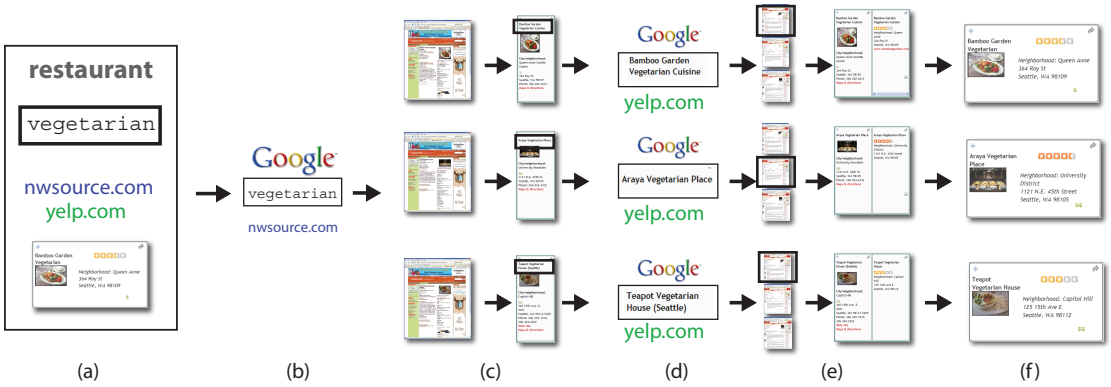


Figure 5.6: **(a)** The user types the query "vegetarian" into the restaurant search template to collect vegetarian restaurants. This search template includes two websites and a relation. The `nwsource.com` website is considered primary because information from `yelp.com` is collected through a relation. **(b)** The system issues the query "vegetarian" to a general search engine and limits the results to `nwsource.com`. **(c)** Each search result is processed with an extraction pattern to extract relevant content. **(d)** The extracted content triggers relations, which issue further queries. **(e)** The subsequent search results are extracted, ranked, and added to the content that triggered the additional retrieval. **(f)** Finally, the user sees the extracted content as a series of cards.

times a week to several times a month. Two of the participants had extensive bookmark collections, and the remaining four participants reported saving and organizing information infrequently. The primary organization techniques were using email and adding information to documents.

I performed the study on a WindowsXP laptop with 2GB RAM and 2 Ghz processor. The laptop was connected to the Internet via a wireless connection. The extension was installed on Firefox v1.5. The participants had individual sessions, and each session lasted approximately one hour. Each session included a background questionnaire (10 minutes), a tutorial of the system (15 minutes), three tasks (20 minutes), and a debriefing session (15 minutes). During the tutorial the participants were shown how to use the system to collect and organize information related to purchasing a car. Content was collected from two websites, `autos.msn.com` and `edmunds.com`. The participants were shown how to create a relation between the two websites, how to create a new card for cars, and how to create a search template and use it to find new cars.

The three tasks were framed as part of the same scenario, which was "finding a restaurant for a night out." In the first task, the participants were directed to `nwsource.com` and asked to pick a restaurant and add it to their collection. Then, they were asked to go to `yelp.com`, find a review

for the restaurant, and add it to their collection. To add Web content to their collection, users had to only click a button. The participants were then asked to use the relation interface to associate the content together and then collect several more restaurants. In the second task, the users were instructed to design a card for the restaurants in their collection. For the last task, the participants created a restaurant search template and made several queries.

5.6.1 Observations and feedback

Overall, the participants were very positive about the tool. They expressed that it would make Web research a more efficient process and let them easily return to tasks over time. One user mentioned, "I think you would save me a ton of time," while another said, "There is some startup cost but I think this [tool] is easy to learn."

Although overall impressions were positive, all of the participants suggested improvements to the interface for specifying relations and placing content in cards. Instead of drawing lines between webpage elements, the participants wanted to drag and drop pieces of content. Web Summaries can easily be extended to include drag-and-drop interactions. It is only through iterations with potential users that one can uncover such user preferences and refine the interface.

Relations

All participants were able to create relations. One of the participants said, "It's pretty obvious what the links should be. I can't think of a situation when the UI would not be adequate for what I want." In the interest of time and keeping the tasks simple, I had the participants collect content from only two websites, thus requiring only one relation instantiation. I believe that this will be representative of many tasks, which will require information from a few websites. However, a more complex scenario may uncover additional usability problems with the interface for creating relations. I am now exploring exposing possible relations to the user as he collects new content.

Cards

Four of the six participants listed the ability to create their own card as one of their favorite aspects of the system. And when asked which card they preferred, the default or the one they had just created, five of the six participants said they preferred their own. The one participant who did not

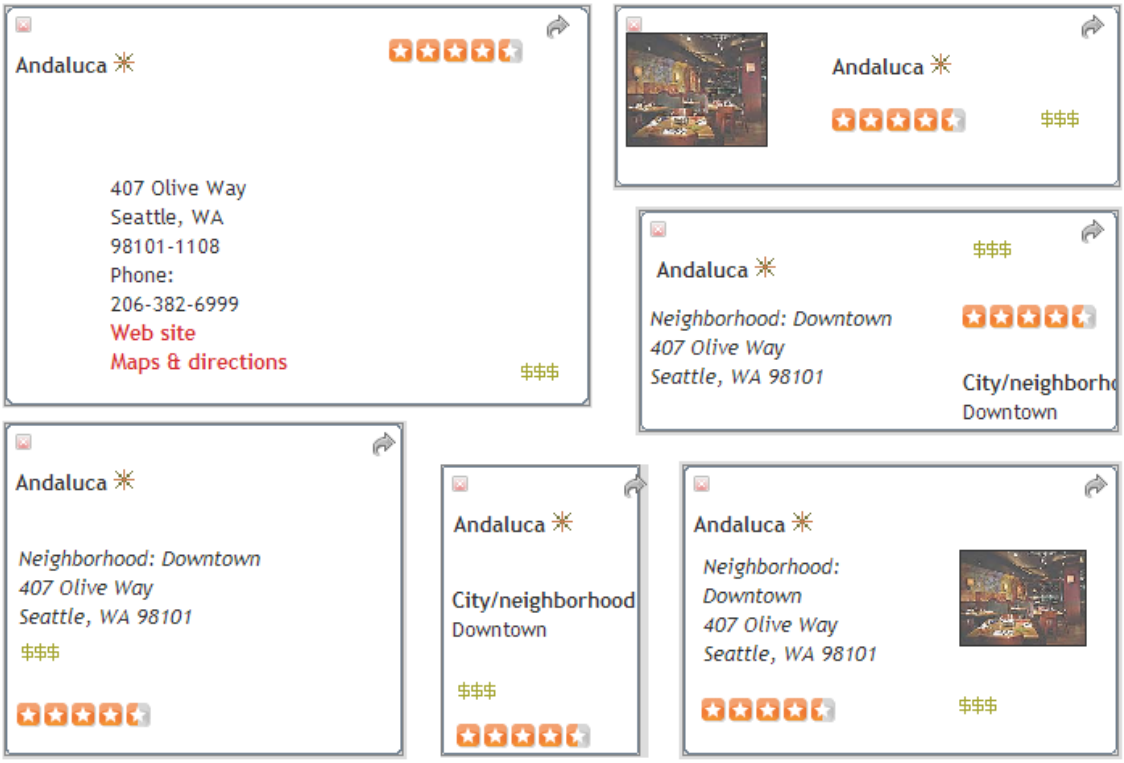


Figure 5.7: Each participant designed a personalized card that presented the information of interest.

prefer her own card thought it was not as pretty as the default, as she did not consider herself able to create artistic cards. One user mentioned, "I really liked the ability to synthesize a new page or card from multiple sources. In comparison shopping activities I frequently look at multiple pages about a good service and it would be great to organize them quickly." The participants quickly learned how to use the card designer and were all able to create their own cards. Figure 5.7 shows cards created by the participants. While most users were happy to draw containers, several users requested a more lightweight authoring tool through default templates. One participant mentioned, "The layout design tool is a bit too fine grained for me. I kind of want to just throw a few items onto the card and maybe shuffle their arrangement, but I don't necessarily need to draw out their exact layout." The card designer was inspired by design tools such as Adobe Illustrator, but this interaction paradigm may not be appropriate for novices or users who are interested in completing their task as quickly as possible. A good card designer should make it possible to create cards quickly but also give the user

control. This could be accomplished through good defaults and more automated layout that adapts the containers as the content changes, as was shown by Jacobs et al. [41].

Search templates

When asked about their favorite aspect of the system, two of the six participants picked template-based search. One user mentioned, "Google is good when I am searching for one thing. If searching for a bunch of stuff, I would prefer to see the actual content and be able to manipulate it by saving and deleting." When asked about the lag in retrieving search results when using a search template, another participant said, "A little lag time is a much better use of my time than going through zillions of search results. I can check stuff I have already gotten back while other things are coming in. After all, it's getting me what I asked for."

While most of the participants were able to use the search templates effectively and made queries such as "Italian" and "Chinese" to retrieve new restaurants, two of the six participants were confused about the source of the search results and the types of queries that would yield restaurant cards. For example, one user made the query "Tom Douglas," a famous chef and restaurant owner, thinking that it would return cards for each restaurant owned by Tom Douglas. Unfortunately, this query did not return any search results because a query for "Tom Douglas" at `nwsource.com` returns articles about Tom Douglas and not listings of his restaurants in the top eight search results. Because I use a general search engine to retrieve relevant search results, there will be situations in which the results are not appropriate for the search template. In such situations, users could quickly try out new queries, or the system could help them by suggesting possible ways to augment the search query. Currently, the search templates do not return results if they are unable to find appropriate content. This gives the user little intuition as to how to modify the query to achieve better results. The system could be extended to give the user feedback about the available search results, even if they do not fit the expected data representation.

5.7 Discussion

The Web Summaries system addresses the requirements of tools for exploratory Web research. It integrates directly with the Web browser. It stores intermediate state and allows users to return to their findings at any time. Furthermore, it provides an automated mechanism for organization through

layout templates and multi-scale views of the collected content through user-defined cards. Web Summaries also provides capabilities not envisioned previously, such as automatically collecting content within one website with extraction patterns and among several websites through relations. Finally, Web Summaries is the first system to reinvent the Web search interface through personalized, visual summaries.

Web Summaries is capable of not only helping people more quickly and easily accomplish their exploratory Web research tasks, but it is also a mechanism for realizing the Semantic Web. As people use the tools to collect and organize Web content, they create extraction patterns and relations that include semantic information about the content inside of webpages. This semantic information can be shared in a public repository and reused by others, who are attempting to accomplish the same task. But, it can also be embedded back into the websites as semantic information for use by other applications and services. Web Summaries bring new promise to the Semantic Web by giving users tools for building it piece by piece as they accomplish their own tasks.

CHAPTER 6

EVALUATION

I conducted an ten-week longitudinal user study to evaluate Web Summaries and solicit user feedback for further improvements. In order to release Web Summaries in the field, several modification of the original implementation were necessary. I extended Web Summaries to include a collaborative online community and more flexible layout templates, which allow users to sort and filter their content collections. In the summer of 2007, I deployed Web Summaries to 24 participants with several goals in mind. First, I wanted to evaluate the utility of automatic extraction and understand when and how often users are in situations where automatic Web content extraction is useful. Second, I wanted to gain a better understanding for how people collect and organize information. Although there have been a number of ethnographic studies [79, 10, 50] exploring user behavior patterns when dealing with information, they do not reveal the granularity or type of information users collect during exploratory Web research. Finally, I wanted to explore how a community of users can benefit one another through an online repository of shared semantic content.

I this chapter I first describe the design of the extended system and the methodology for the longitudinal study. I then discuss the study data and analysis. [1]

6.1 The system

Since many users requested filtering and sorting capabilities for their collections of content during the laboratory studies, I added these operations to Web Summaries by creating layout templates using Exhibit [38]. Exhibit is a lightweight Javascript framework for publishing structured data. It combines structured JSON files that describe data with presentation parameters to produce webpages that allow users to organize and filter data interactively. I implemented four layout templates – a thumbnail view (Figure 6.1), a detail (Figure 6.14) view, a table (Figure 6.15), and a map (Figure 6.10). The blue pane on the right of the display (see Figure 6.1) allows users to tag their

[1] The study findings were submitted for publication as a paper [24] and are currently under review.

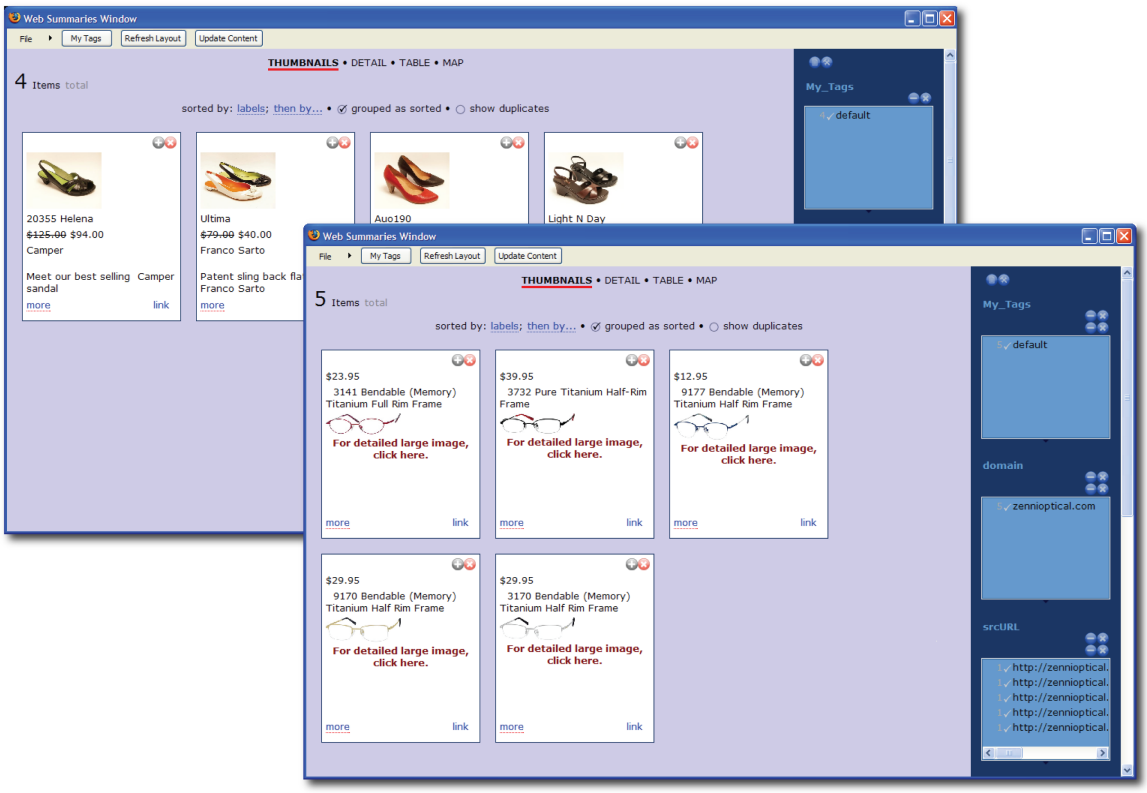


Figure 6.1: The participants used automatic content gathering for a variety of shopping tasks, including searching for shoes and glasses.

collection items or filter the collection using a facet, such as domain, review, or price. Users can also sort the collection using the facets. For example, sorting according to name sorts the items in alphabetical order. Users can also at any time press the "update content" button at the top and dynamically update all of the content in the summary. This allows them to retrieve the most recent content from any webpage or alternatively add new content from previously visited pages.

In addition to creating layout templates that include filtering and sorting, I also added a collaborative component to Web Summaries through a *community pattern repository*. The community pattern repository stores all patterns created by the study participants, thereby allowing them to share patterns with one another and among devices. As described in Chapter 4, a pattern can tag and extract content from a webpage. Thus, this community pattern repository holds a semantic description for all of the webpages and websites visited by the subjects. Each user has a *personal pattern*

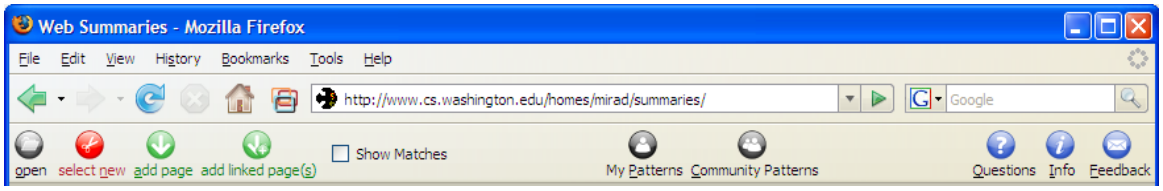


Figure 6.2: Web Summaries is available to the user through a toolbar interface.

repository that includes all patterns that he has used. The union of all personal pattern repositories creates the community pattern repository. The community pattern repository was implemented with a PostgreSQL database and an Apache Web server. All communication between the extension and the server used the Javascript HTTPRequest object. All server scripts for accessing the database were written in PHP.

6.1.1 The interface

Figure 6.2 show the toolbar interface for the application that I deployed in the field study. The "open" button starts Web Summaries, opens the summaries window, and enables the toolbar. Since only one summary window can be open for each browser window, subsequent presses of the open button bring the summary window in front of any other windows. Clicking on the modal "select new" button enables selection and tagging of webpage elements. Figure 6.3 shows the selection and tagging of an image. The user can tag webpage elements with the default tags or he can create his own tags and categories of tags. In Figure 6.3 the user has created a NYTIMES category and is adding a 'photo' tag. The tags determine how the element is displayed in the summary. For example, an "address" tag tells the system that the clipped content can be displayed on a map. To turn off selection and tagging users click the "select new" button again. The extension adds the selected content to the summary and creates an extraction pattern for that page using the selected elements. The "select more" button allows users to add more elements to existing extraction patterns. The "select new" button becomes a "select more" button if the user has created an extraction pattern for the type of webpage currently viewed. Thus as the user browses the Web this button changes from to as needed. The "add page" button allows users to automaticaly add content to their summary with an extraction pattern. If there isn't a matching extraction pattern, this button is disabled. If an extraction pattern for this page exists in the community pattern repository, the add

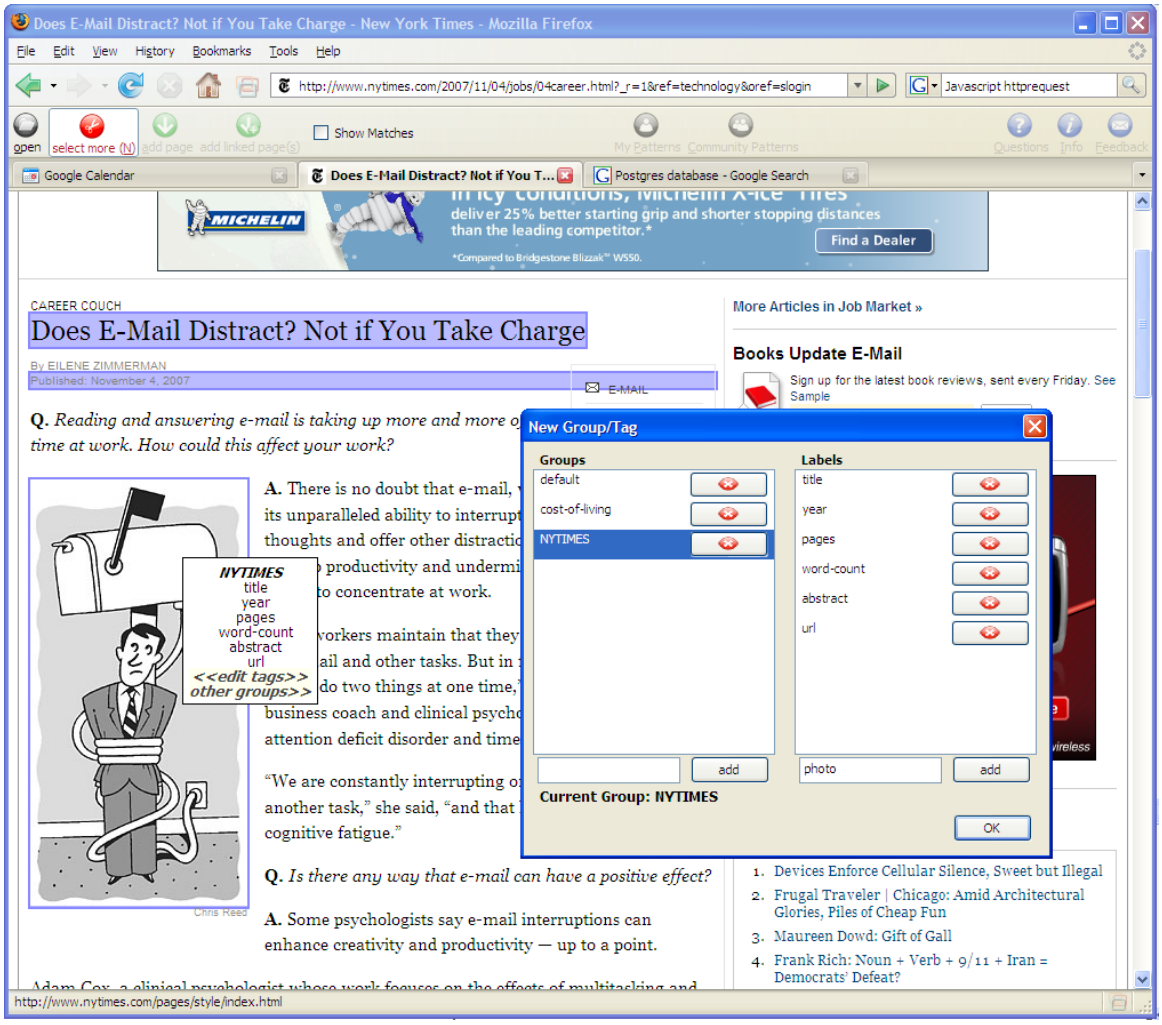


Figure 6.3: The user clips and tags pieces of a webpage. He can create new tags and categories at any time.

page button changes color and icon and becomes enabled. The user can click on the button to see a list of matching patterns. Figure 6.4 shows the interface for viewing the shared database of patterns. I discuss this interface in more detail in the next section. The "add linked pages" button allows users to select hyperlinks to pages they want to add to their summary. Web Summaries follows each hyperlink, extracts the content from the hyperlinked page and adds the extracted content to the summary. The "show matches" checkbox lets users decide if they want to see what is matched on any webpage. When this option is checked, all elements in a loaded webpage that match an extraction pattern are outlined in red. The "my patterns" button opens a window that displays

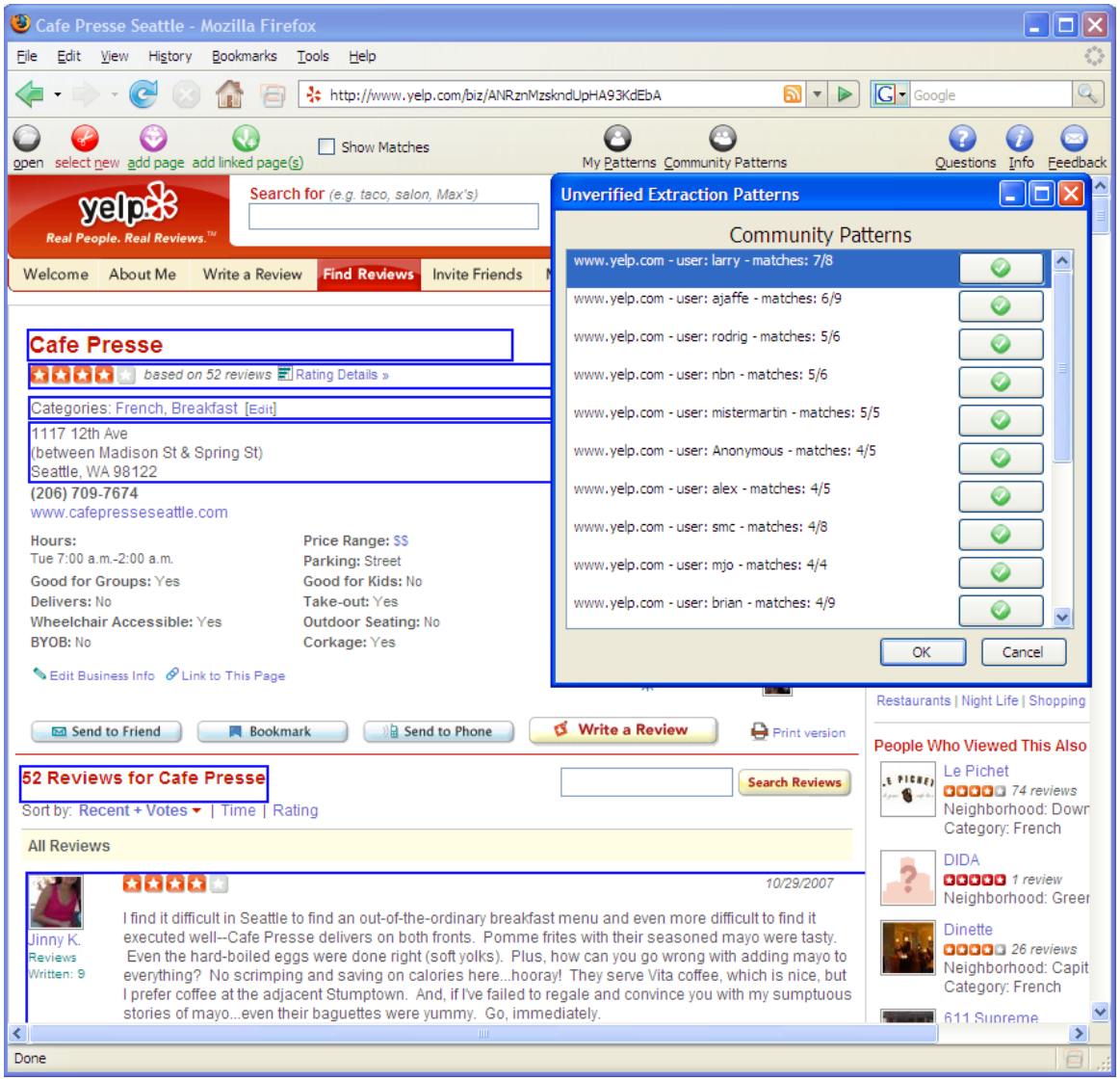


Figure 6.4: When the "add page" button is purple it opens the interface to the community pattern repository. The user can view all patterns that apply to the current webpage and download them. To see which parts of the page are extracted by a pattern, the user moves the cursor over the list of patterns. The elements that can be extracted by the pattern under the cursor are highlighted in blue.

a list of extraction patterns that the user has used in the past (Figure 6.5). These patterns may be ones created by the user or downloaded from the community pattern repository. The user can delete or download new patterns through this interface. The "community patterns" button displays a list of extraction patterns that all users have created. Users can filter the list according to username

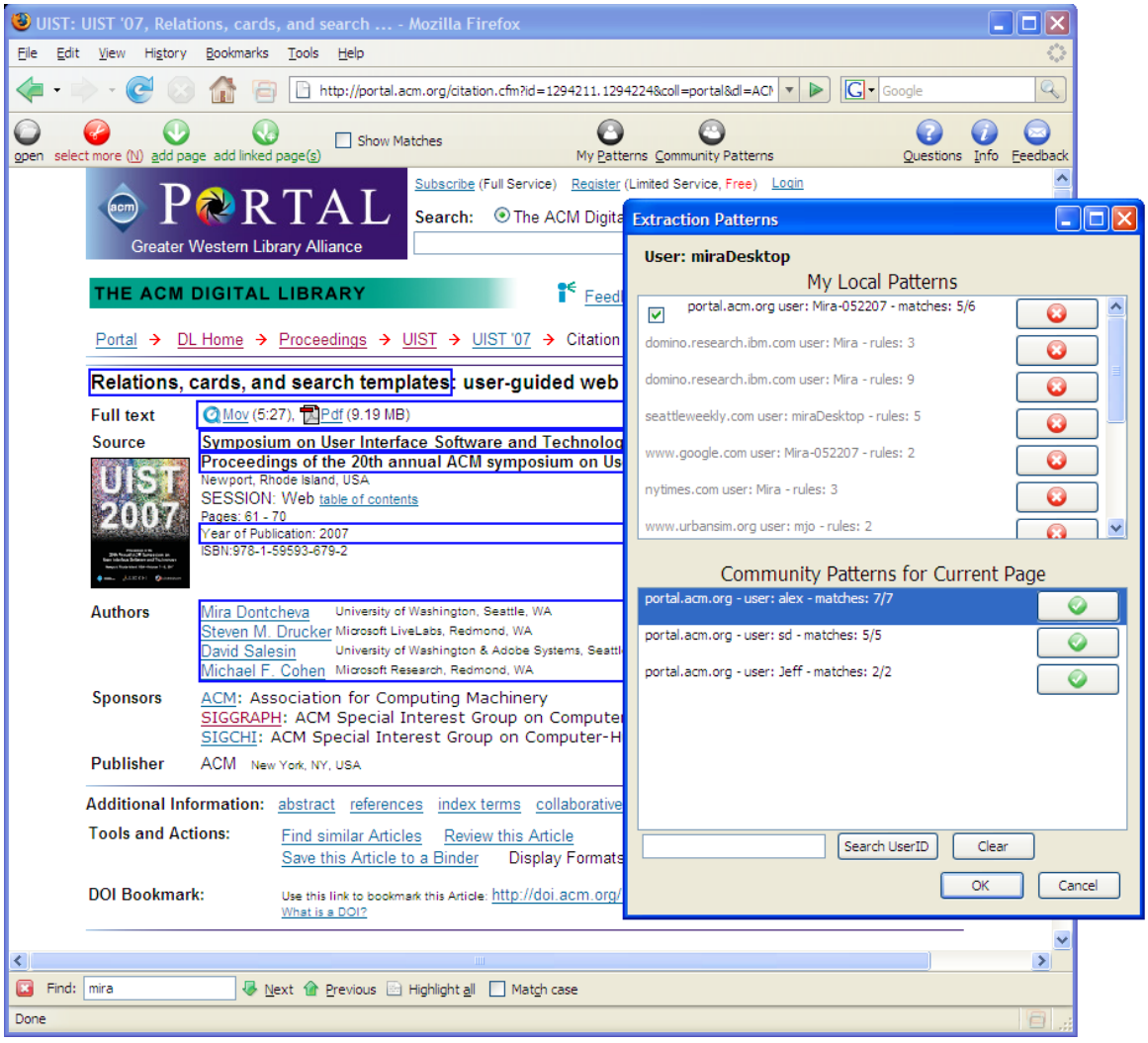


Figure 6.5: The user can press the 'my patterns' button to view all personal patterns and the community patterns that apply to the current page.

and look for all patterns created by a specific person. The "questions" button opens the user's mail client and allows users to submit questions at any time. The "info" button takes the user to the study website. And, the "feedback" button opens a survey that allows the user to submit feedback at any time.

6.1.2 Community pattern repository

Every time a user creates an extraction pattern, it is added to the community pattern repository. This repository is in turn visible to all participants. Figure 6.4 shows the interface for viewing all patterns that are applicable to the current page. Each pattern in the list is displayed according to its domain, its author, and the number of webpage elements it can collect on the current page. The display also lists the total number of webpage elements the pattern can extract. Thus, the first pattern listed in Figure 6.4 is for the `www.yelp.com` website, was created by a user named *larry*, and can collect seven webpage elements from this webpage for the "Caffe Presse" restaurant. It can, however, extract a total of up to eight webpage elements, which means that there is one more item that it can collect that either does not exist on this webpage or is in a different location. It is likely that *larry* created the extraction pattern for `yelp.com` on a different webpage, not the one for "Caffe Presse", which is why it only matches partially. Patterns that only match partially may be more versatile and account for layout variations. Alternatively, webpage structure changes may make it difficult to extract all originally selected elements. The patterns are listed according to the highest number of extractable webpage elements. To view the webpage elements that are extracted by a pattern, the user moves the cursor over the list of patterns, and the elements that can be extracted by the pattern underneath the cursor are highlighted in blue. In Figure 6.4, the cursor is over the first extraction pattern and seven webpage elements are highlighted, including the name, rating, restaurant categories, address, and reviews. To download a pattern the user clicks on the green check mark button and the pattern becomes part of his personal pattern repository.

Users can view patterns they have used in the past by clicking on the "my patterns" button, which opens a view of their personal pattern repository. Figure 6.5 shows the interface for the personal pattern repository. It includes two lists. The top list shows the patterns that the subject has used before. To delete a pattern the user can press on the delete button. All patterns that do not apply to the current webpage appear grey. The checkbox allows a user to store many patterns for a particular webpage and change the one that is considered default. The second list shows patterns that are available for the current webpage in the community repository. As the user moves the cursor over the list of patterns, the webpage elements that can be extracted by the pattern under the cursor are highlighted.

6.2 Study methodology

I recruited 24 participants through emails to university distribution lists. The subjects were offered a gift certificate to the university bookstore for their participation. The gift certificate was prorated based on their participation. Ten of the participants were female and fourteen were male. The participants ranged in age from 19 to 67, with a median age of 28 and a mean age of 31. Fourteen of the participants were graduate students in technical departments; four were undergraduates in technical departments; and six were staff members at the university. Nineteen of the participants stated that they perform some form of Web research every day, while the remaining participants said they perform such tasks several times a week. When asked about which tools they typically use, twenty of the participants mentioned emailing themselves and using bookmarks. Sixteen said that they copy and paste information into a document. Thirteen said they print webpages. Ten said they send email to other people. Nine mentioned more advanced tools such as specialized toolbars or online tools such as Google Notebook and del.icio.us. When asked about how often they organize their Web content collections, half of the participants said they either never organize Web content or only do so as needed and when time allows. A third of the participants said they organize things when they collect new content. Only two participants mentioned organizing content several times a month, and one stated organizing content several times a week. Thus, my participant population showed behavior similar to user behaviors described in previous studies [10] [43] – frequent Web use for research with infrequent organization.

The study included two in-person interviews, one at the beginning of the study and one at the end. During the first interview, the participants filled out a demographic questionnaire and took part in a tutorial. I installed the Web Summaries extension on their computer, showed them how to use the tool, and then gave them a specific task to test whether they understood the tool. All of the participants used either a personal laptop or a work desktop computer for the study. The subjects were instructed to use the Web Summaries tool for their own personal tasks. During the closing interview, the participants filled out a closing survey while I uninstalled Web Summaries. In addition to the two interviews, the study included weekly surveys, which asked about their experiences with the tool. The participants were also sent weekly structured tasks, which asked them to visit a specific website and collect information. In total the users received seven tasks, including:

- **Shopping** – visit `amazon.com` and collect 10 books of your choice.
- **Reviews** – visit `yelp.com` and collect 10 items of your choice.
- **Travel** – visit `tripadvisor.com` and collect information about 10 different hotels.
- **Events** - visit `upcoming.org` and collect 10 different events for the city of your choice.
- **Entertainment** – visit `imdb.com` and collect information about 10 movies and/or actors.
- **Cooking** – visit `epicurious.com` and collect 10 recipes.
- **Reference** – visit the ACM Digital Library at `portal.acm.org` and collect references to 10 articles.

The goal in assigning tasks was to give the participants examples of tasks for which they might consider using Web Summaries. Additionally, the weekly tasks were useful in simulating a large number of participants by encouraging all subjects to visit the same website with the same goal in mind, thereby allowing me to evaluate the community pattern repository interface. In addition to soliciting user feedback through surveys, Web Summaries also logged all user interaction with the extension such as each click of the "add page" button, the number of links selected with the "add linked pages" tool, switching of summary views, or deletion of a summary item. The log files were uploaded directly to the server without any user intervention.

6.3 Results

Of the 24 initial participants, only 15 were active contributors and completed the study. Thirteen participants took part in the closing interviews. An additional two participants were frequent users of the tool but did not participate in the closing interviews. I received 60 survey responses – 36 over email, and 24 through the tool. Figure 6.6 shows a sorted list of the participants according to the number of days each one used the Web Summaries tool. This data does not impose a minimum usage time, thus even if the subject only opened Web Summaries briefly, I record that activity as a day of usage. Figure 6.7 shows a temporal plot of the tool usage. The stacked bar plot shows the duration

Table 6.1: This table shows the frequency and types of events logged during the study.

event	**number logged**	**type**
loaded new webpage	15009	browsing
switched tab	1854	
created new pattern	257	toolbar
modified pattern	138	
pressed add page button	425	
pressed add linked pages button	254	
pressed my patterns button	95	
pressed community patterns button	66	
downloaded community pattern	54	
pressed information button	21	
pressed feedback button	9	
pressed questions button	8	
added new tag	298	tags
added new category of tags	48	and
deleted tag	29	categories
deleted category of tags	15	
added new item	1033	summary
changed view	318	
deleted item	235	
filtered using a facet	96	
clicked on dynamic update button	77	
clicked on link in summary	72	
assigned user tag	14	
removed filter	9	
created summary tag	3	
deleted summary tag	1	

in minutes of each subject's interaction with the tool for a given day. Each subject is represented by a different color. I compute the usage duration by ignoring browsing events and removing gaps in usage greater than 20 minutes. The grey vertical lines correspond to dates on which the participants received a new task. The graph shows active initial usage that decreased over time. I expect that this is due to the initial novelty of the tool and an initial period of exploration. Although some participants only used the tool for the assigned tasks, many participants found personal tasks for which it was useful.

During the ten-week study, Web Summaries logged 26244 events. The tool only logged user activities when it was active (i.e., when the Web Summaries window was open). It logged user interactions with the summary, pattern specification and editing, and all toolbar button presses. The tool also logged the URL of every webpage the user visited. Table 6.1 shows the frequency and types of events that were logged. The events are grouped into four types: browsing, toolbar, tags and categories, and summary. *Browsing events* include switching tabs and loading a new webpage. The majority of the logged events were of this type. *Toolbar events* include pattern specification events and any toolbar button presses, such as pressing the "add page" button or "add linked pages" button. The *tags and categories events* include creating and deleting tags and categories. In the interface categories were listed as groups. *Summary events* include all user actions on the actual summary such as changing views or deleting an item. Some events were logged for debugging purposes. For example, the tool logged 4134 pattern specification events that show where the user clicked on the page while creating patterns.

This data shows that the participants created many patterns and used them repeatedly to collect over 1000 items of content from different websites. They also created their own tags for the content that were personally meaningful to them. The participants changed summary views often and deleted content they did not find useful. However, they did very little organization of the summaries. The majority of the log events were browsing events because many times users left the summary window open and continued browsing for some other purpose.

6.3.1 Qualitative feedback

During the closing interviews participants were asked about their favorite features of Web Summaries. Half of the participants said that the automatic gathering of content through the "add linked

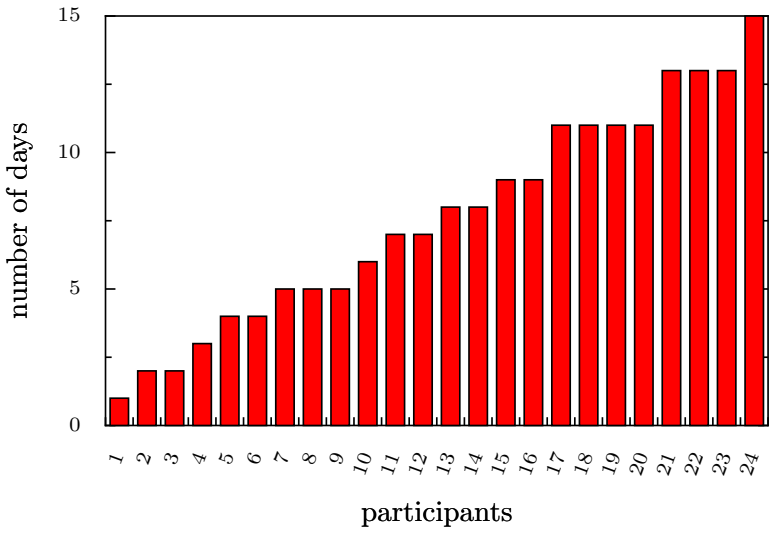


Figure 6.6: This figure shows the number of days each participant used the Web Summaries tool.

pages" button was their favorite feature. Other favorite aspects included the ability to save persistent copies of Web information that is often dynamic, the ability to collect text and images and make rich views of the content, the variety of views offered by Web Summaries, the tool's integration with the browser, and the fact that only one tool was necessary for accomplishing a variety of tasks. Users wrote:

> *"I liked constructing 'captures' - the interaction was easy and fun, and I liked that I could customize what I wanted to capture. I liked the way the 'captures' are displayed in the thumbnail view."*
>
> *"I liked its intuitive interface and close integration with the existing page."*
>
> *"I love my 'add linked' pages option! It's such a time-saving functionality. It makes the process so easy once you've decided what info you want to gather."*
>
> *"Add linked pages tool = HUGE time saver."*

When I tried to uninstall the tool, two of the participants asked to continue to use it so that they could access the data they had collected. One of the participants used the tool for her own work and said that it helped her conduct her own research. She said,

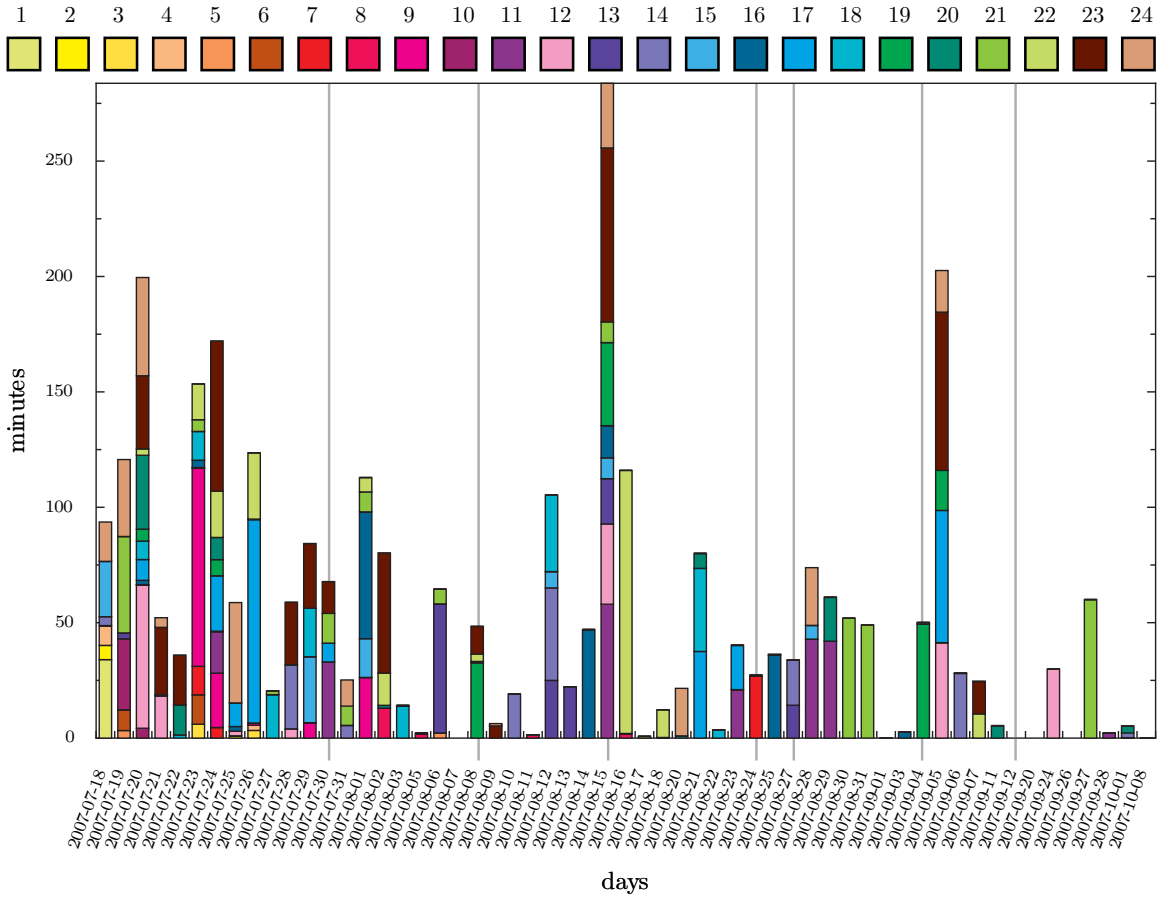


Figure 6.7: This figure shows the usage of the Web Summaries tool over time. The grey lines correspond to dates on which the participants received a task. Each color corresponds to a study participant.

> *"I found that Web Summaries is great at saving my time of looking up, revisiting, and documenting WebSVN and bug databases. The summary that I created is now a part of my research document."*

Although users were very positive about the automatic gathering functionalities, they also had many suggestions for improvements. The participants requested more feedback during the automatic extraction process, a more flexible interface for specifying and modifying extraction patterns, a smaller toolbar, and the ability to edit the summary views. During the closing interviews some of the participants remarked that they had used the tool much less than they had originally expected.

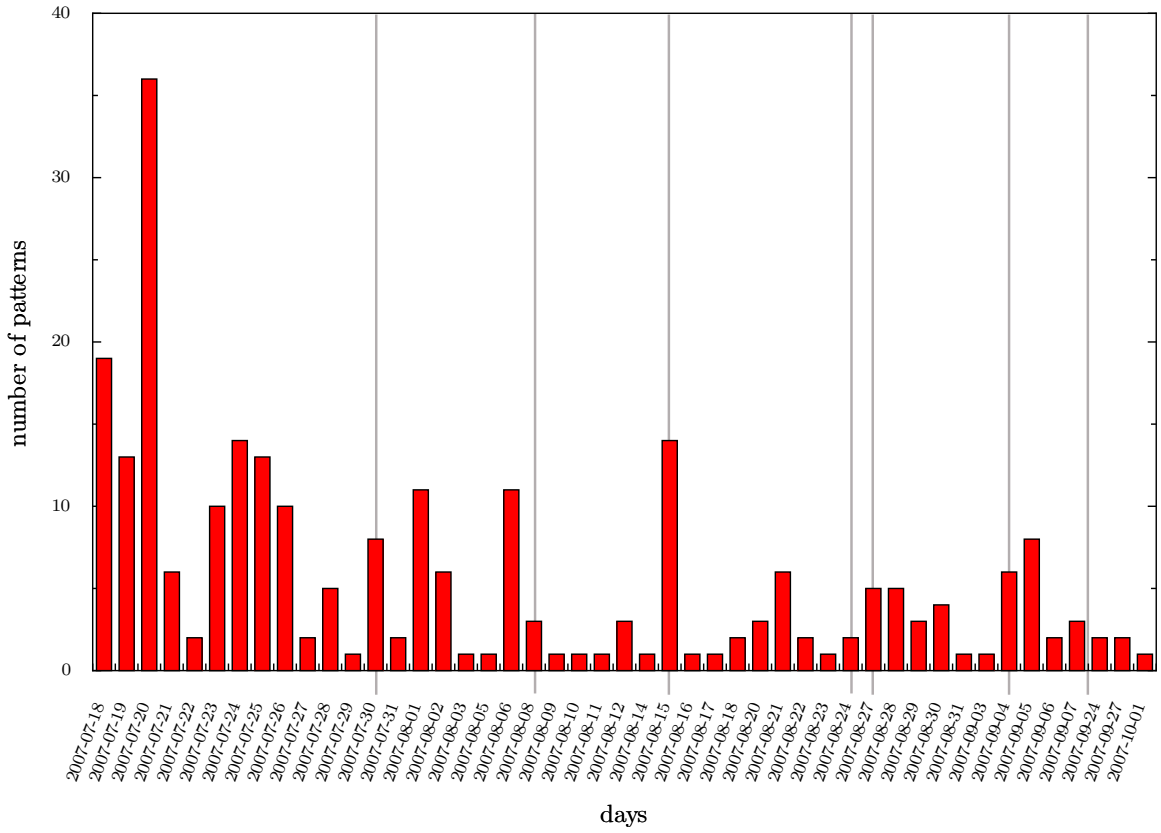


Figure 6.8: This figure shows the number of extraction patterns created during the study. The grey lines correspond to dates on which the participants received a task.

Since many of the subjects used laptops, the size of the summary toolbar limited their browser screen real estate. As a result they hid the Web Summaries toolbar and often forgot about the availability of the tool. Since the users had to look at the summary window to see if the content they wanted to store had been added to their summary, many reported spending too much time switching between the browser window and the summary window. I expect that users with multi-monitor displays will have a much easier time with the summary window, but in order to support laptop users I plan to explore different approaches for providing feedback. Despite an imperfect interface, many of the users completed the assigned tasks and used the tool for personal tasks.

6.3.2 Collecting Web content with extraction patterns

During the ten-week study, the participants created 257 extraction patterns, used the "add page" button 425 times and collected information from 987 hyperlinks by applying the "add linked pages" functionality 254 times. They created a total of 447 unique summaries. Figure 6.8 shows the number of patterns created over time. The grey lines correspond to dates on which the participants received a new task. The participants also modified extraction patterns 138 times. Of the 257 patterns, 114 were used to collect content automatically. The remaining 147 extraction patterns were not applied to automatically collect content, because their authors were interested in clipping content rather than collecting a number of comparable items. Patterns were used on average 9 times with a standard deviation of 10. Figure 6.9 shows the utility of each pattern. Most of the patterns that were used more than 10 times were shared among the participants. Interestingly, the pattern that was used the most, over 80 times, was used by only one participant. It was created for the ACM Digital Library website and used to collect over 80 references.

Despite the fact that the subjects created and used many patterns, they encountered some problems when creating patterns. The modal interface for specifying extraction patterns was sometimes confusing. Several users requested that the tool automatically save their selection when they navigate away from the page or close the tab. Several users requested a more flexible selection mechanism.

> *"It works pretty well, but there are times when I would rather have more control over what's selected, i.e, combining elements into one or selecting just the part of an element that's after certain punctuation."*

One participant, a Web programmer, requested an interface for specifying regular expressions as extraction patterns. This would have enabled him to collect content from the website `mybus.org`. Structural extraction patterns are not appropriate for this website as the location of the buses changes according to arrival time. Another participant requested retrieval from a list of items, as many website do not have detail webpages for particular events or locations. Other participants requested multi-page extraction patterns, as content is often split among several tabs or screens.

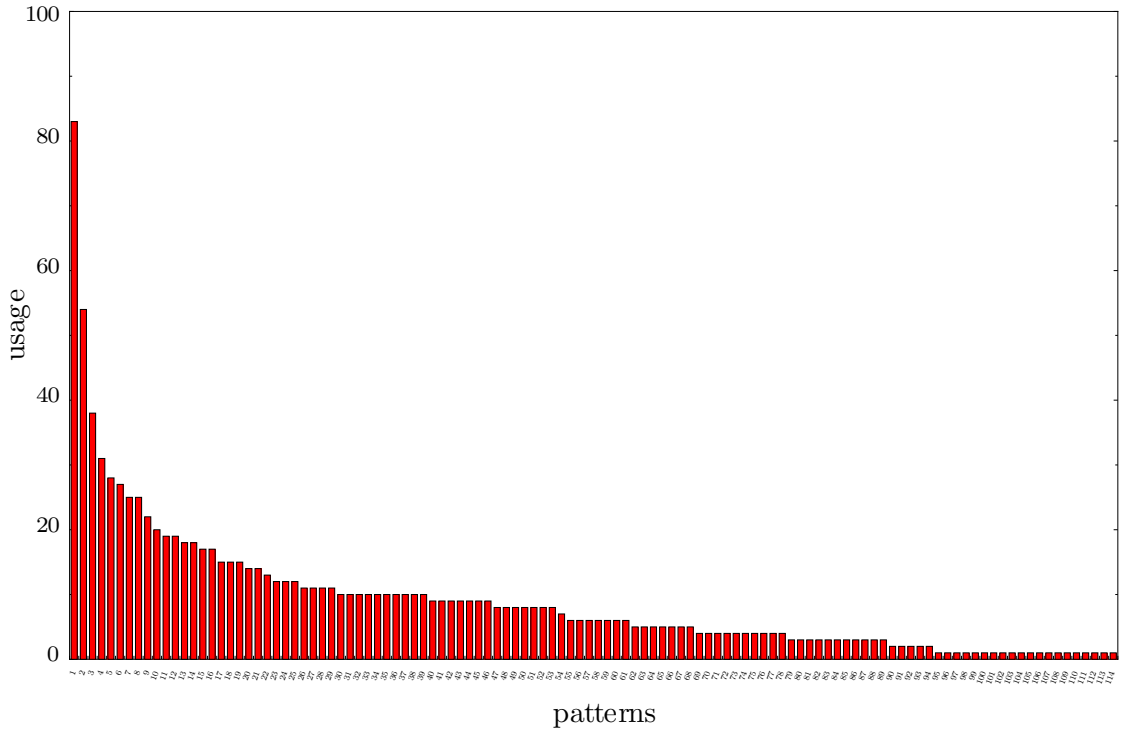


Figure 6.9: This figure shows the utility of each pattern.

6.3.3 Accomplishing tasks

Most of the active study subjects completed the assigned tasks. Table 6.2 shows the number of subjects that visited the assigned task websites. Some of the users visited the assigned task websites more than once. The last column shows the discrete visits to those websites. The low visit rate for the reference task is due to the fact that some participants did not use the ACM Digital Library for academic references. They chose to use other reference libraries, such as Inspec or IEEE Xplore. Also, since this was the last assigned task, the participants did not have as much time to complete the task.

In addition to successfully accomplishing many of the assigned tasks, the subjects also used Web Summaries for a number of personal tasks. The participants created extraction patterns on 88 distinct domains and used automatic extraction on 44 of those domains. They collected a variety of data for many purposes. I group task data into three categories - life, work, and fun. Most of the participants engaged in some type of life information task, such as comparison shopping (Figure 6.1), searching

Table 6.2: This table shows the subject participation in the assigned tasks. The events column shows that some participants visited each website more than once.

type	website	participants	visits
shopping	amazon.com	12	14
reviews	yelp.com	14	14
travel	tripadvisor.com	14	16
events	upcoming.org	11	11
entertainment	imdb.com	12	14
cooking	epicurious.com	8	10
reference	portal.acm.org	4	5

through rentals (Figure 6.10) or job listings (Figure 6.11), looking for local car washes or farms. Work personal tasks included collecting information about conference courses, collecting articles (Figure 6.14), and gathering technical data (Figure 6.12). The participants also used Web Summaries to collect information for hobbies. One participant collected comics (Figure 6.13) and used the "update content" button to automatically retrieve the most recent comic strip. Another participant collected information about local hiking trails, while yet another collected historical weather data (Figure 6.15). These sample summaries show that users collected a large variety of data. Some collected highly structured content, while others stored entire articles with many paragraphs of text.

6.3.4 Organizing summaries

In addition to automatically gathering content, users had the opportunity to explore a richer organization metaphor than is currently possible in the browser through layout templates. During the ten-week study, the participants created 447 unique summary collections and added over 1000 items to those collections. Figure 6.16 shows the number of summaries created by each participant. Most of the summaries were not accessed more than once, which is in line with the participants' survey feedback. Although the participants used Web Summaries for a variety of tasks, they did not return to their summaries and continue with the tasks at a later time. Some of the subjects did not encounter a content intensive task that required returning to a summary during the study period. Also, since

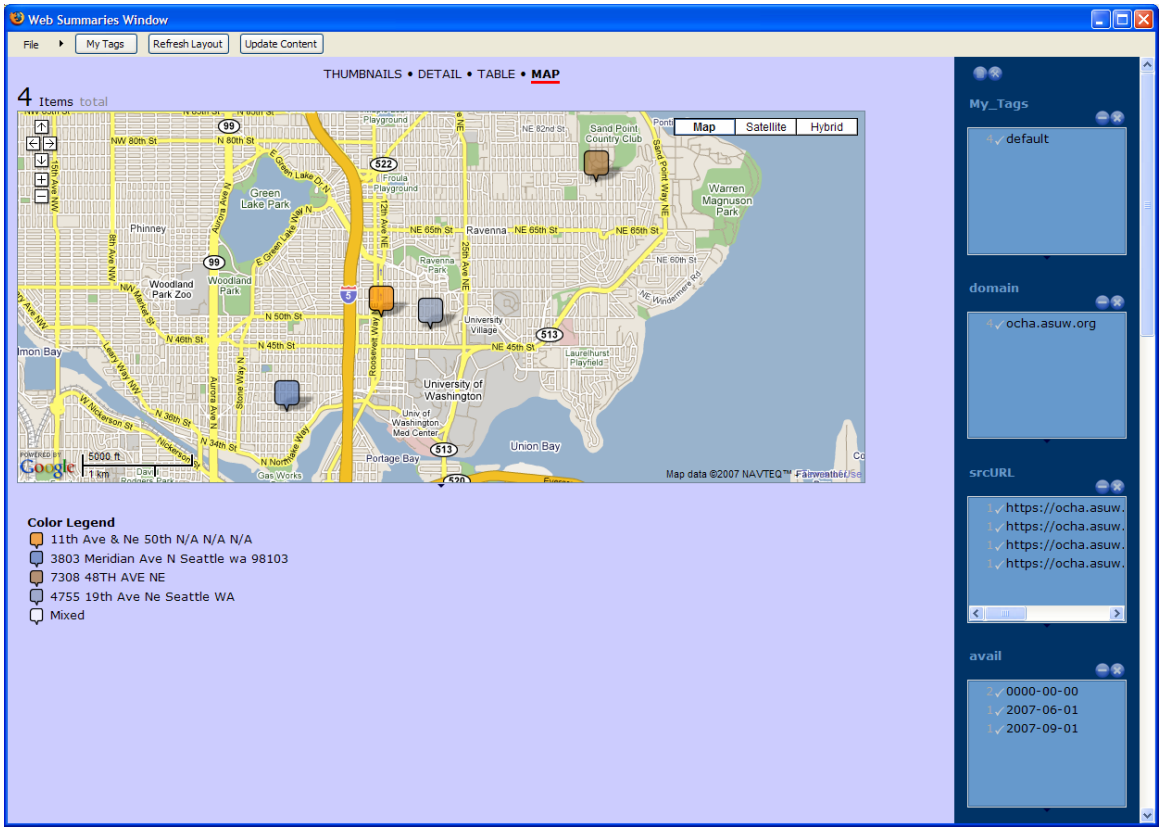


Figure 6.10: One participant collected advertisements for available apartments.

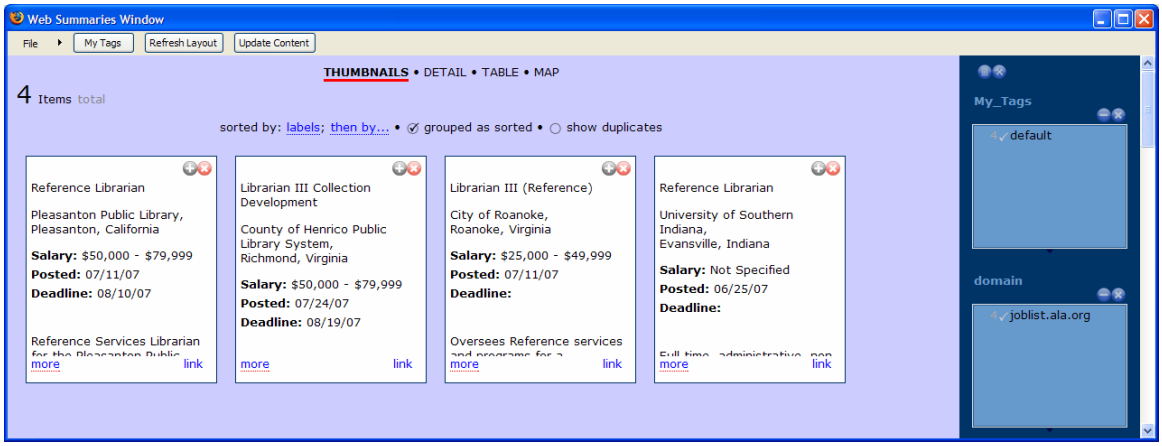


Figure 6.11: Another collected listings for job openings.

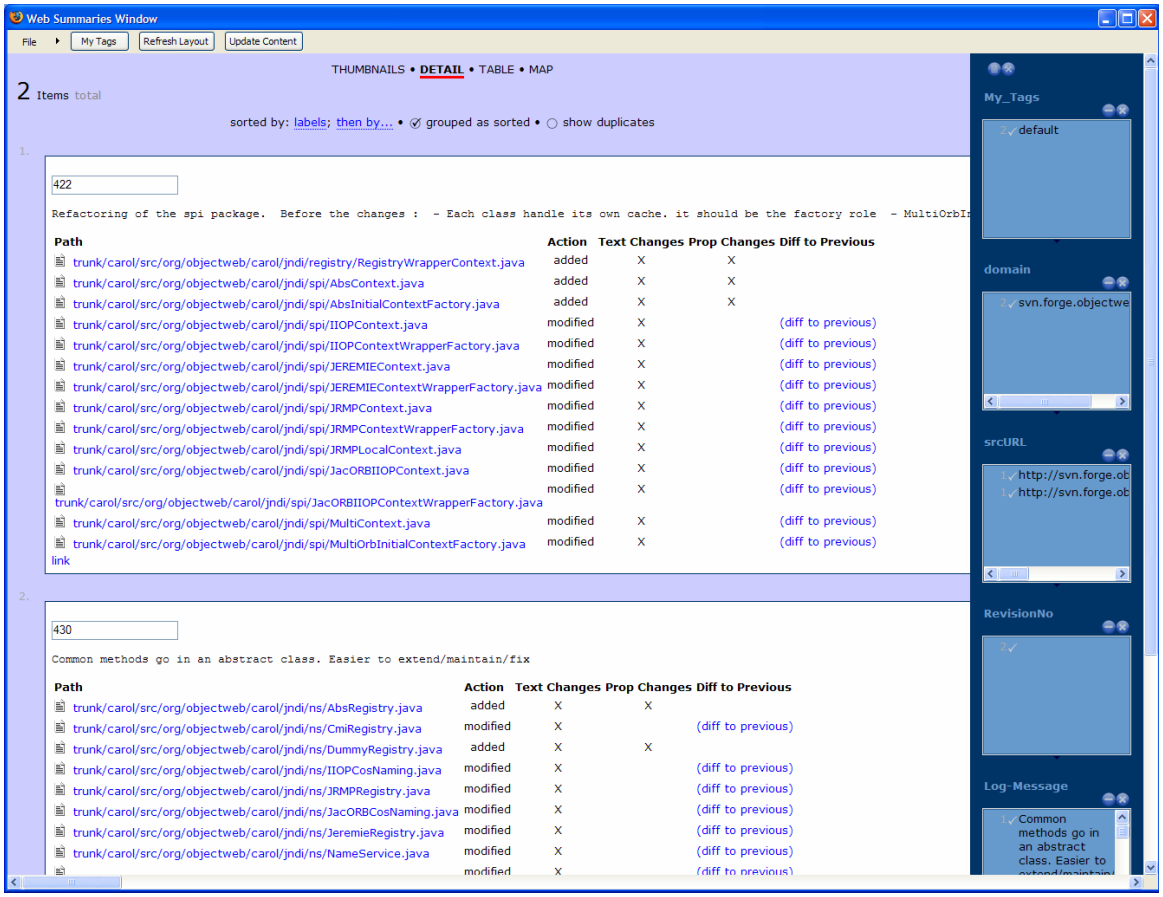


Figure 6.12: One participant used the Web Summaries tool to collect information about open source CVS repositories.

the participants knew that this was a study, they may have been hesitant to store too much important data in a format that may not be accessible later.

Many users were positive about the ability to store rich information from webpages.

> *"It let me collect richer info about the sites I visited than the text files I usually use to take notes. I liked having images and active links, for example."*

> *"I liked the fact that in addition to text, I could tag pictures, and this made the thumbnail view really useful: I could see what recipes were for rather than just read the title and that made browsing easier."*

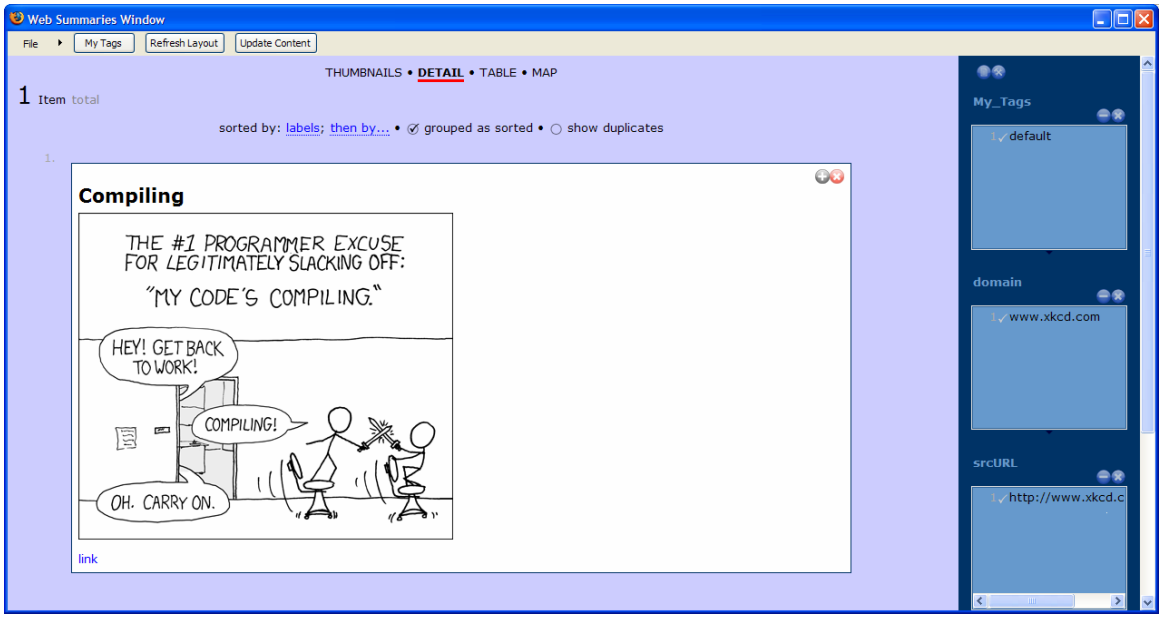


Figure 6.13: One participant used Web Summaries to collect comics and automatically retrieve the most recent comic strip.

While the log data shows that the participants were not accessing existing summaries often, many of the participants asked to be informed of future releases. Two of the participants requested that they continue using the tool so that they could continue with the summaries they had created.

The log data shows that the participants often changed views. Web Summaries recorded 318 view change events. Of those 318 events, 104 were changes to the detail view; 98 were changes to the table view; 70 were changes to the thumbnail view; and 46 were changes to the map view. Since the thumbnail view was the default view, it is largely under-represented in these statistics. User feedback on favorite views varied according to the person and task.

> *"At first I preferred the table view, because it presented all the data side by side for comparison. Then, as the summaries became more complex, and there was no way (that I could find) to resize columns, change font size, etc., I started using the thumbnail view instead."*

> *"Table works well when there are lots of items. Thumbnail works well for a few items when the first few fields contain crucial information."*

Table 6.1 shows that users deleted items in the summary frequently. This number is likely inflated because of the duplication of content during pattern modification. When users modify an extraction pattern, the system adds another item to their collection with the newly created pattern. The participants filtered their summaries often, but they did very little organization through tagging. It is possible that since the participants had already tagged the content during the clipping phase, they didn't feel the need to tag the content collections as well. This minimal amount of organization could also be due to the small sizes of the summary collections. The summaries included on average ten items. Finally, the subjects used the dynamic update functionality and clicked on links stored in the summary frequently.

6.3.5 Sharing patterns with the community pattern repository

One of the goals of this study was to better understand the challenges surrounding a public repository of extraction patterns. My analysis of the logged data reveals that users are interested in using a collaborative pattern repository but that the interface for conveying which patterns are available and what type of information they can extract needs to be finely tuned to make it easy to select an appropriate pattern. In total there were 54 downloads of 30 unique patterns from the community repository with only 9 downloads of patterns by their original authors. Ten patterns were downloaded more than once and all of the patterns that were reused were for the assigned task websites. There was very little overlap in browsing habits between the 15 active participants; thus, the assigned tasks served the study well in simulating a larger participant population.

Many participants downloaded a pattern and then modified it to fit their preferences. Users said:

> *"I loved when I could use other people's extraction pattern. These were a bit mysterious regarding when they would appear. Often I used other peoples' as a base and revised them to my liking."*

> *"I liked the way I could hover over a pattern in the list and see which elements on the page were selected."*

The types of patterns the participants would create and use was somewhat personal. Some preferred collecting lots of information, others preferred collecting less information.

"I tended to look for patterns with the most matches. This may not have left me with the "best" pattern for the page, though."

"I checked first to see if there are some good pattern out there. If not I'll create one myself. I want a simple one, not with a lot of information."

Some users preferred to create their own patterns because they found the community pattern repository interface confusing or simply found it faster to create their own patterns.

"It seems like work to figure out what patterns are there and whether I want to use them. And making patterns still feels fun and easy so I'd rather make my own."

"When I tried using existing patterns I ran into some difficulties, and it turned out to be faster to construct my own."

Despite a small and fairly private community, some participants didn't trust the community patterns and preferred to create their own.

"I do my own thing because I don't trust other people. Sharing is nice if you are in a hurry, but I usually created my own for personal reasons."

"I generally create my own. Creating extraction patterns is not that hard - it's actually much harder to figure out what other people's extraction patterns are. If I trusted other people's extraction patterns, I would probably use them more."

Many participants found it easier to create their own patterns rather than learn how to use the community pattern repository interface. Participants who were in a hurry didn't have much patience for the interface and preferred to interact with the already familiar clipping interface. When participants did find appropriate community patterns quickly, they used them.

6.4 Discussion

This field study shows that users enjoy creating extraction patterns and find automatic Web content extraction capabilities for structured Web content very useful. On many occasions users also collect less structured information, such as articles, and for such tasks automatic extraction is not necessary. Future development of tools and applications for automatically aggregating and extracting Web content needs to address the inherent user need for both structured and unstructured information. Good user interfaces for managing semantic Web content must provide more than just tables, grids, and maps. In future work, I plan to explore interaction techniques and visualization paradigms for a heterogeneous set of Web content that includes both highly structured information, such as the price and address of a hotel, and highly unstructured information, such as long personal reviews or descriptions of amenities.

For many of the subjects the timing of the study was highly critical. Users with content intensive tasks were much more willing to spend time with the tool and learn its interface. The participant who integrated Web Summaries into her own research was compelled to do so because of an impending conference deadline. Most daily tasks, however, are transient and short-lived, and the subjects did not find Web Summaries useful and easy to integrate into their everyday tasks. I believe this was due to the conscious upfront effort required by Web Summaries. The user must open the summary window and actively save content. I designed Web Summaries for content intensive tasks, thus it is not that surprising that it is not as well suited to transient tasks. However, users who do not use Web Summaries often, may turn it off and forget that the tool is available. Future studies on exploratory Web research tasks should be aware of the timing sensitivity that is inherent in this type of research, and Web content tools should be targeted to address the different types of tasks users experience.

Based on my observations I divide Web tasks into four categories - short transient, long transient, short permanent, and long permanent tasks. *Short transient* tasks are typically finished quickly, such as finding a birthday present for next week, or finding a restaurant for a night out. *Long transient* tasks include making more than one arrangement and possibly coordinating with others such as planning a vacation or work trip. They may also include learning about a new topic, such a gardening or a new health concern. *Short permanent* tasks are often also called monitoring Web tasks [50] and include reading news or blogs every day or checking favorite sport websites. Finally,

long permanent tasks are tasks that are longstanding interests and involve gathering, collecting and organizing information over a long period of time. This categorization should viewed as a continuum. Some tasks start out as short transient tasks but may become longer transient tasks. Similarly, short transient tasks may become short permanent tasks. I designed Web Summaries for long transient and permanent tasks. I hope to adapt Web Summaries to shorter more transient tasks by allowing the user to retroactively build summaries thereby remove the active upfront need for managing content. In order to become well integrated into a user's daily Web usage, Semantic Web tools need to be sensitive not only to long and permanent Web tasks but also to transient and short-lived tasks, otherwise users will not integrate them into their Web browsing habits. Since Web browsing is such an integral part of user's lives, tools that support information management must be fluidly integrated into the browser and be available at any time.

Finally, I explored the role of an online repository of extraction patterns. Such a repository could grow to become a collaborative user-defined Semantic Web. I found that when participants were in a hurry and wanted to quickly accomplish their task, they were more willing to use others' patterns. When their were not in a hurry or they had an important task, they created their own patterns. An online repository of semantic information is still very new to users and visualizations for exposing the available information must be carefully designed such that the semantic information can aid rather than stop users from accomplishing their tasks. In the future I plan to explore visualization techniques and interfaces for exposing community information about a webpage to the user. An alternative to asking the user to select among many possible extraction patterns is to automatically select a good pattern using user preferences or statistics.

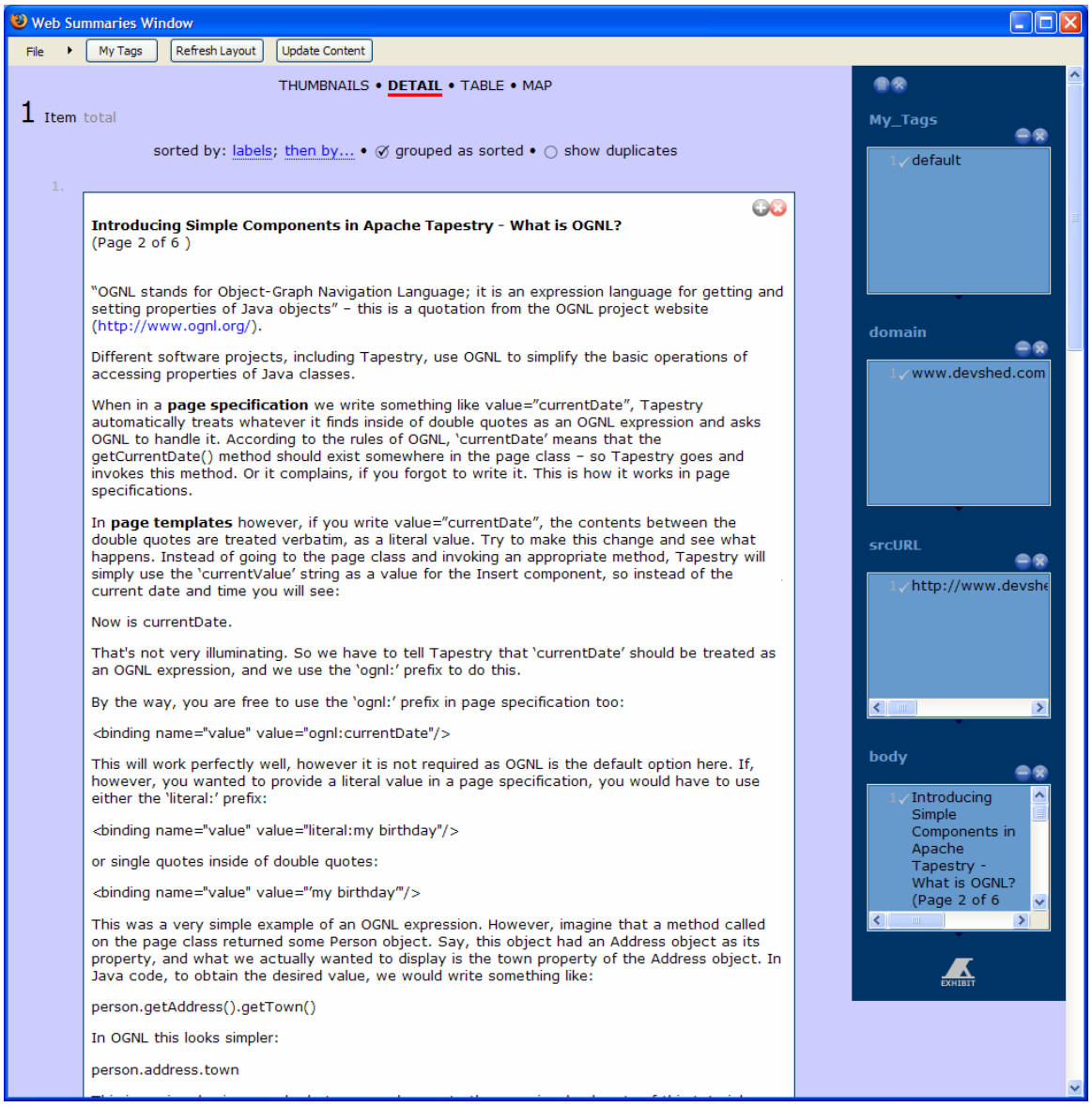


Figure 6.14: Some participants collected entire articles from the Web.

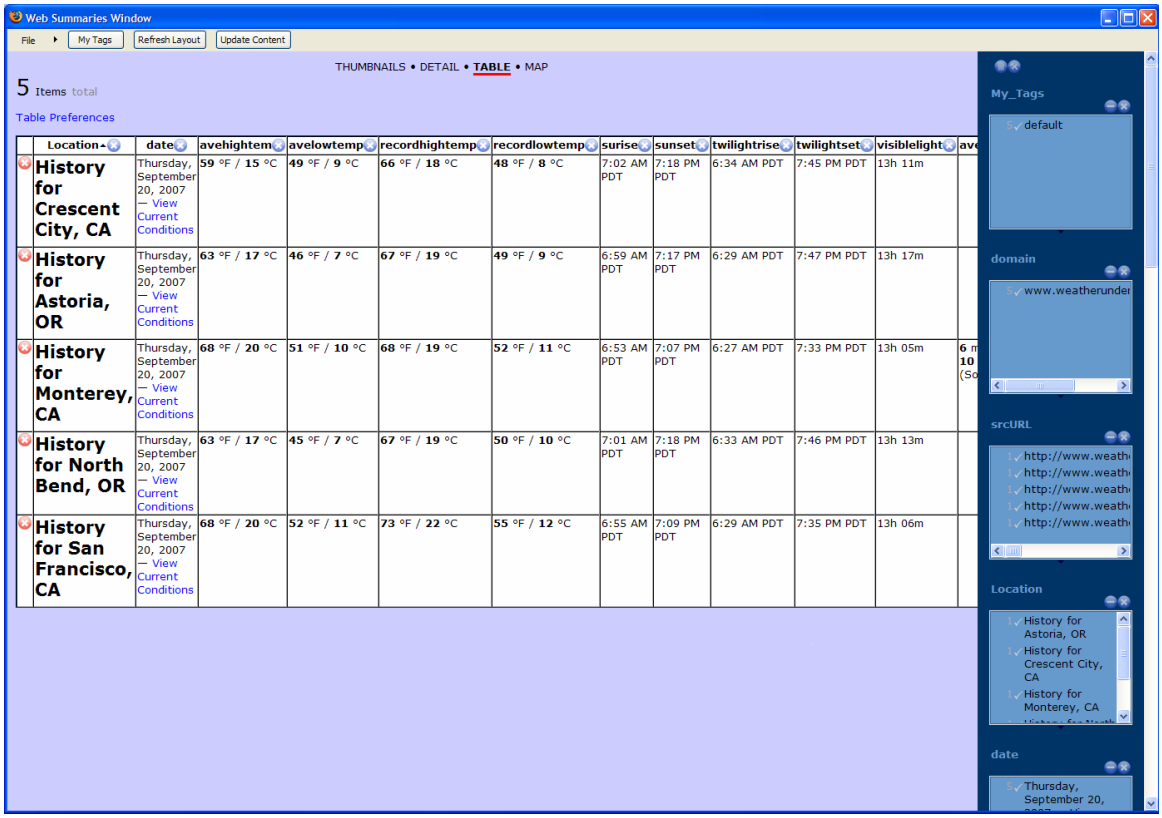


Figure 6.15: One participant collected detailed historical weather data.

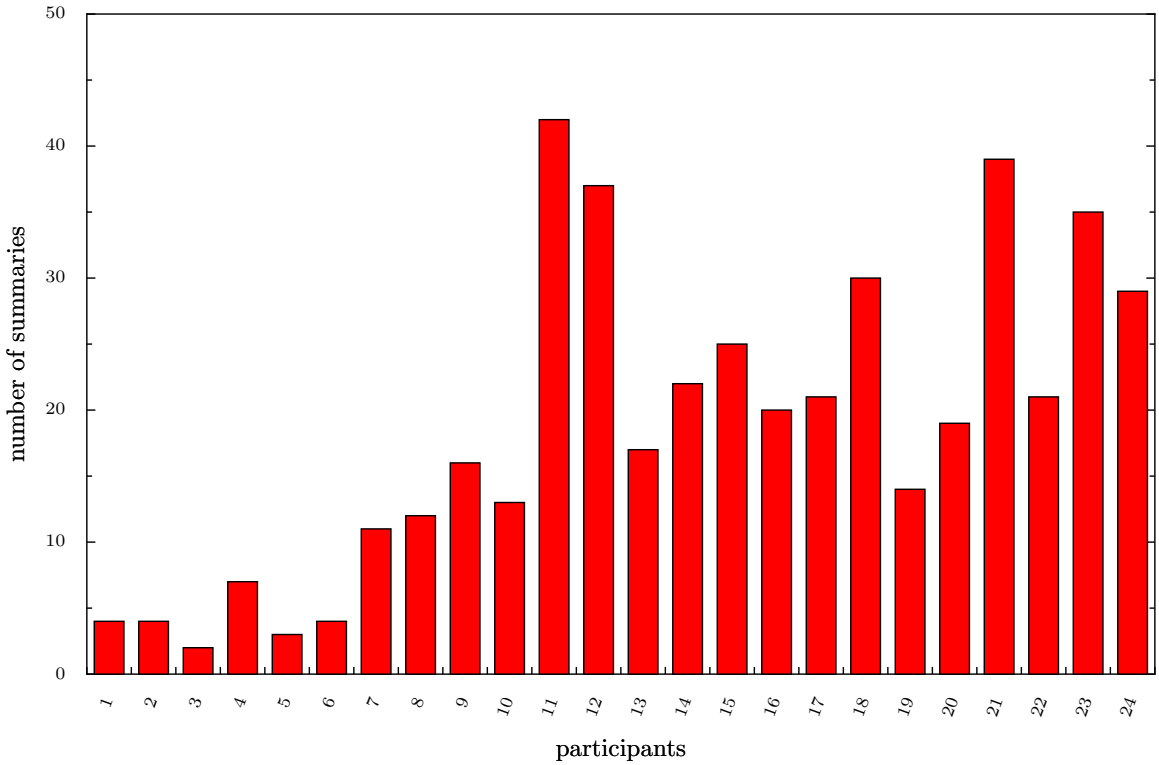


Figure 6.16: This figure shows the number of summaries created by each participant.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Contributions

In this thesis, I have demonstrated that automation can successfully be combined with interaction techniques to help users more effectively collect and organize Web content. My contributions include a set of interaction techniques for gathering content semi-automatically with extraction patterns and relations, novel presentation principles for composing Web content collections into rich visual summaries through layout templates and personalized cards, and a novel search paradigm that transforms a keyword query into a personalized visualization of the returned content. Additionally, the study of webpage structure over time is the first of its kind, and the evaluation of user behavior with tools that include automatic retrieval can help inform future research in this area. Finally, this thesis presents an approach to building a machine-readable Web as a side-effect of using tools to accomplish personal tasks.

7.2 Future work

The Web has transformed the world in which we live and work. I believe the work presented in this thesis presents a way to transform the Web. Today the Web is the same for everyone everywhere, but each person has their own preferences and contexts for viewing and using the Web. The techniques described in this thesis give users the ability to view and present the parts of the Web they care about in their own personalized way. In this section I describe new areas of research and propose some potential projects.

7.2.1 Personal multi-scale views of the Web

Most webpages on the Web are designed for desktop and laptop displays and scale poorly to small mobile devices. Thus, it becomes hard to read the news on a PDA while waiting for the bus, or to make an online purchase while away at a conference. I propose letting users define personal views

for webpages for their personal devices. With Web Summaries the user can already specify which parts of a webpage he cares about, and he can also create a card that can display the content in the most suitable configuration. These views might be defined on a laptop or directly on the mobile devices. Since content will vary over time, the views will have to adapt to different configurations. Research on adaptive document layout [41, 77] can be leveraged to make personal views adaptive to a variety of devices and types of content. In addition to giving users personalized windows into the Web, personal views can lower the bandwidth requirements for browsing the Web on mobile devices. Some important issues to consider will be navigation strategies for such personalized versions of the Web, discovery of new content, and display of parts of the Web that have not yet been personalized.

7.2.2 Retrospective summaries

The Web Summaries system was designed to help people manage Web content, and thus it is closely related to systems that address problem solving and sensemaking. Through the longitudinal study, I found that despite the automation and summarization of the large amounts of content already provided in Web Summaries, users still found the tool difficult as they had to actively make decisions about which content to save. I propose letting users create summaries retroactively using their browsing history. This type of on-the-fly sensemaking and problem solving can alleviate the organization hardships usually imposed by content intensive tasks. As long as users have seen the content, it is considered saved and can be accessed and analyzed at any later time. Search technology is in part successful because it helps users with retroactive retrieval and lets users avoid the difficult and cumbersome process of organization. Some of the open research challenges are how to make this sensemaking on the fly as lightweight as possible and offer users a general set of tools for problem solving for a large number of problems.

7.2.3 Rich search tools

Search templates present a new interface to Web search, but they are only a first step in starting to develop new search interfaces. People have complex information needs with many constraints. The search tools of the future will be able to let users specify their preferences. My research is one of the first to address this growing problem. One of the current limitations of search templates is that they do not let users easily iterate over search results. Future research directions include providing

more feedback with the rich search results so that the user can iterate through queries, using search results directly in the query process, and building hierarchical search templates that allows users to combine information needs into one high-level template.

7.2.4 Collaboration and public repositories

In the longitudinal study I took first steps towards combining the Web's ability to connect people with the automation tools I have proposed in this thesis. The study showed that there are many challenges in harnessing collaboration to help users accomplish their tasks.

First, Web Summaries must be extended to handle groups collaborating together on one summary. Aspects that will be critical in this development include visualizations for presenting content collected by different people, changes to the visual summary over time, and the integration of other types of content, such as personal files, into the visual summary.

Second, this work has strong ties to the vision of the Semantic Web, in which not only can computers understand all of the information embedded in a webpage but they can also understand relationships and concepts. I have built interactive tools that give people enough value that they might effectively build components of the Semantic Web without any help from content providers. My approach leads to many new research directions including visualization paradigms for exposing semantic information and relations among websites, effective aggregation and transformation of content from many disparate sources, and using collaborative semantic information for the assessment of the quality of information on the Web.

7.3 Summary

In this thesis, I have presented a set of semi-automatic interaction techniques for retrieving content from the Web, presentation principles for user-driven organization of content from any number of sources, and a new search paradigm for the Web that transform keyword search into a goal-oriented rich visual experience. To demonstrate the efficacy of these ideas I combined them into a working system and evaluated them through a longitudinal user study. To further reaffirm the suitability of my approach, I also analyzed changes in webpage structure over time and found that structural extraction is well suited to current trends on the Web.

REFERENCES

[1] David Abrams, Ron Baecker, and Mark Chignell. Information archiving with bookmarks: personal web space construction and organization. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–48, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[2] Amazon.com, Inc. http://www.amazon.com/.

[3] Brian Amento, Loren Terveen, and Will Hill. Experiments in social data mining: The topicshop system. *ACM Trans. on Computer-Human Interaction*, pages 54–85, 2003.

[4] Vinod Anupam, Juliana Freire, Bharat Kumar, and Daniel Lieuwen. Automating web navigation with the webvcr. In *Proc. of the WWW conference on Computer networks*, pages 503–517, 2000.

[5] Greg J. Badros, Alan Borning, Kim Marriott, and Peter Stuckey. Constraint cascading style sheets for the web. In *UIST '99: Proc. of the 12th annual ACM symposium on User interface software and technology*, pages 73–82, New York, NY, USA, 1999. ACM Press.

[6] Patrick Baudisch, Xing Xie, Chong Wang, and Wei-Ying Ma. Collapse-to-zoom: viewing web pages on small screen devices by interactively removing irrelevant content. In *UIST '04: Proc. of the 17th annual ACM symposium on User interface software and technology*, pages 91–94, New York, NY, USA, 2004. ACM Press.

[7] Mathias Bauer, Dietmar Dengler, and Gabriele Paul. Instructible information agents for web mining. In *IUI '00: Proc. of the 5th international conference on Intelligent user interfaces*, pages 21–28, New York, NY, USA, 2000. ACM.

[8] Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. Automation and customization of rendered web pages. In *UIST '05: Proc. of the 18th annual ACM symposium on User interface software and technology*, pages 163–172, New York, NY, USA, 2005. ACM.

[9] Olha Bondarenko and Ruud Janssen. Documents at hand: Learning from paper to improve digital technologies. In *Proc. of the SIGCHI conference on Human factors in computing systems*, 2005.

[10] Harry Bruce, William Jones, and Susan Dumais. Keeping and re-finding information on the web: What do people do and what do they need to do? In *Proc. of ASIST*, 2004.

[11] Orkut Buyukkokten, Hector Garcia-Molina, and Andreas Paepcke. Seeing the whole in parts: text summarization for web browsing on handheld devices. In *WWW '01: Proc. of the 10th international conference on World Wide Web*, pages 652–662, New York, NY, USA, 2001. ACM Press.

[12] Deng Cai, Shipent Yu, Ji-Rong Wen, and Wei-Ying Ma. Extracting content structure for web pages based on visual representation. In *Proc. of 5th Asia Pacific Web Conference*, 2003.

[13] Stuart K. Card, George G. Robertson, and Jock D. Mackinlay. The information visualizer, an information workspace. In *CHI '91: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 181–186, New York, NY, USA, 1991. ACM Press.

[14] Stuart K. Card, George G. Robertson, and William York. The webbook and the web forager: an information workspace for the world-wide web. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 111–ff., New York, NY, USA, 1996. ACM.

[15] Bay-Wei Chang, Jock D. Mackinlay, Polle T. Zellweger, and Takeo Igarashi. A negotiation architecture for fluid documents. In *UIST '98: Proc. of the 11th annual ACM symposium on User interface software and technology*, pages 123–132, New York, NY, USA, 1998. ACM Press.

[16] Yu Chen, Wei-Ying Ma, and Hong-Jiang Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *WWW '03: Proc. of the 12th international conference on World Wide Web*, pages 225–233, New York, NY, USA, 2003. ACM Press.

[17] Mary Czerwinski, Maarten van Dantzich, George Robertson, and Hunter Hoffman. The contribution of thumbnail image, mouse-over text and spatial location memory to web page retrieval in 3d. In *Proc. of Interact '99, the 7th IFIP Conference on Human Computer Interaction*, pages 163–170, 1999.

[18] Michael C. Daconta, Leo J. Obrst, and Kevit T. Smith. *The Semantic Web: A Guide ot the Future of XML, Web Services, and Knowledge Management*. Wiley Publishing, Inc., 2003.

[19] del.icio.us, Inc. http://del.icio.us/.

[20] Xin Dong, Alon Halevy, and Jayant Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 85–96, New York, NY, USA, 2005. ACM.

[21] Mira Dontcheva, Steven M. Drucker, David Salesin, and Michael F. Cohen. Changes in webpage structure over time. Technical Report TR2007-04-02, University of Washingtion, 2007.

[22] Mira Dontcheva, Steven M. Drucker, David Salesin, and Michael F. Cohen. Relations, cards, and search templates: user-guided web data integration and layout. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 61–70, New York, NY, USA, 2007. ACM.

[23] Mira Dontcheva, Steven M. Drucker, Geraldine Wade, David Salesin, and Michael F. Cohen. Summarizing personal web browsing sessions. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 115–124, New York, NY, USA, 2006. ACM.

[24] Mira Dontcheva, Sharon Lin, Steven M. Drucker, David Salesin, and Michael F. Cohen. Experiences with content extraction on the web. In *Proceedings of CHI'08 Workshop on HCI and the Semantic Web*, 2008 (to appear).

[25] Jennifer English, Marti Hearst, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Hierarchical faceted metadata in site search interfaces. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, pages 628–639, New York, NY, USA, 2002. ACM Press.

[26] Alexander Faaborg and Henry Lieberman. A goal-oriented web browser. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 751–760, New York, NY, USA, 2006. ACM.

[27] Dennis Fetterly, Mark Manasse, Marc Najork, and Janet Wiener. A large-scale study of the evolution of web pages. In *WWW '03: Proc. of the 12th international conference on World Wide Web*, pages 669–678, New York, NY, USA, 2003. ACM Press.

[28] Jun Fujima, Aran Lunzer, Kasper Hornbæk, and Yuzuru Tanaka. Clip, connect, clone: combining application elements to build custom interfaces for information access. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 175–184, New York, NY, USA, 2004. ACM.

[29] George W. Furnas and Samuel J. Rauch. Considerations for information environments and the navique workspace. In *DL '98: Proc. of the third ACM conference on Digital libraries*, pages 79–88, New York, NY, USA, 1998. ACM Press.

[30] David Gibson, Kunal Punera, and Andrew Tomkins. The volume and evolution of web page templates. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 830–839, New York, NY, USA, 2005. ACM Press.

[31] Google Inc. http://www.google.com/apis/maps/.

[32] Suhit Gupta, Gail Kaiser, David Neistadt, and Peter Grimm. Dom-based content extraction of html documents. In *WWW '03: Proc. of the 12th international conference on World Wide Web*, pages 207–214, New York, NY, USA, 2003. ACM Press.

[33] Alon Halevy, Anand Rajaraman, and Joann Ordille. Data integration: the teenage years. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 9–16. VLDB Endowment, 2006.

[34] Kirstie Hawkey and Kori Inkpen. Web browsing today: the impact of changing contexts on user activity. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1443–1446, New York, NY, USA, 2005. ACM Press.

[35] Andrew Hogue and David Karger. Thresher: automating the unwrapping of semantic content from the world wide web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 86–95, New York, NY, USA, 2005. ACM.

[36] Darris Hupp and Robert C. Miller. Smart bookmarks: automatic retroactive macro recording on the web. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 81–90, New York, NY, USA, 2007. ACM.

[37] David Huynh, Stefano Mazzocchi, and David Karger. Piggy bank: Experience the semantic web inside your web browser. In *Proc. of International Semantic Web Conference*, 2005.

[38] David F. Huynh, David R. Karger, and Robert C. Miller. Exhibit: lightweight structured data publishing. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 737–746, New York, NY, USA, 2007. ACM.

[39] Utku Irmak and Torsten Suel. Interactive wrapper generation with minimal user effort. In *In Proc. of WWW '06*, pages 553–563, 2006.

[40] Charles Jacobs, Wilmot Li, Evan Schrier, David Bargeron, and David Salesin. Adaptive grid-based document layout. *ACM Trans. on Graphics*, 22(3):838–847, 2003.

[41] Charles Jacobs, Wilmot Li, Evan Schrier, David Bargeron, and David Salesin. Adaptive document layout. *Communications of the ACM*, 47:60 – 66, 2004.

[42] Natalie Jhaveri and Kari-Jouko Räihä. The advantages of a cross-session web workspace. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1949–1952, New York, NY, USA, 2005. ACM Press.

[43] William Jones, Harry Bruce, and Susan Dumais. Once found, what then?: A study of "keeping" behaviors in the personal use of web information. In *Proc. of ASIST*, 2002.

[44] Hyuckchul Jung, James Allen, Nathanael Chambers, Lucian Galescu, Mary Swift, and William Taysom. One-shot procedure learning from instruction and observation. In *Proc. of FLAIRS: Special Track on Natural Language and Knowledge Representation*, 2006.

[45] Shaun Kaasten and Saul Greenberg. Integrating back, history and bookmarks in web browsers. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 379–380, New York, NY, USA, 2001. ACM Press.

[46] Shaun Kaasten, Saul Greenberg, and Christopher Edwards. How people recognize previously seen www pages from titles, urls and thumbnails. In *Proc. of Human Computer Interaction*, 2002.

[47] Brewster Kahle. The internet archive. `http://www.archive.org`.

[48] Hyunmo Kang and Ben Shneiderman. Mediafinder: an interface for dynamic personal media management with semantic regions. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 764–765, New York, NY, USA, 2003. ACM Press.

[49] Hyunmo Kang and Ben Shneiderman. Exploring personal media: A spatial interface supporting user-defined semantic regions. *Univerisity of Maryland Techreport, HCIL-2004-05 , CS-TR-4668 , ISR-TR-2005-51*, 2004.

[50] Melanie Kellar, Carolyn Watters, and Kori M. Inkpen. An exploration of web-based monitoring: implications for design. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 377–386, New York, NY, USA, 2007. ACM.

[51] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, and Juliana S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Rec.*, 31(2):84–93, 2002.

[52] Heidi Lam and Patrick Baudisch. Summary thumbnails: readable overviews for small screen web browsers. In *Proc. of the SIGCHI conference on Human factors in computing systems*, pages 681 – 690, 2005.

[53] V Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklady*, 10:707–710, 1966.

[54] Lipyeow Lim, Min Wang, Sriram Padmanabhan, Jeffrey Scott Vitter, and Ramesh C. Agarwal. Characterizing web document change. In *WAIM '01: Proc. of the Second International Conference on Advances in Web-Age Information Management*, pages 133–144, London, UK, 2001. Springer-Verlag.

[55] Jing Liu, Xin Dong, and Alon Y. Halevy. Answering structured queries on unstructured data. In *Proc. of WebDB*, 2006.

[56] Jayant Madhavan, Shirley Cohen, Xin Luna Dong, Alon Y. Halevy, Shawn R. Jeffery, David Ko, and Cong Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR*, pages 342–350, 2007.

[57] Richard Mander, Gitta Salomon, and Yin Yin Wong. A 'pile' metaphor for supporting casual organization of information. In *CHI '92: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 627–634, New York, NY, USA, 1992. ACM Press.

[58] Catherine C. Marshall and III Frank M. Shipman. Spatial hypertext and the practice of information triage. In *HYPERTEXT '97: Proc. of the eighth ACM conference on Hypertext*, pages 124–133, New York, NY, USA, 1997. ACM Press.

[59] Rupesh R. Mehta, Pabitra Mitra, and Harish Karnick. Extracting semantic structure of web documents using content and visual information. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 928–929, New York, NY, USA, 2005. ACM Press.

[60] R. Miller and B. Myers. Creating dynamic world wide web pages by demonstration. Technical Report CMU-CS-97-131 and CMU-HCII-97-101, Carnegie Mellon University School of Computer Science, 1997.

[61] Saikat Mukherjee and I. V. Ramakrishnan. Browsing fatigue in handhelds: semantic bookmarking spells relief. In *WWW '05: Proc. of the 14th international conference on World Wide Web*, pages 593–602, New York, NY, USA, 2005. ACM Press.

[62] Saikat Mukherjee, Guizhen Yang, Wenfang Tan, and I. V. Ramakrishnan. Automatic discovery of semantic structures in html documents. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 245, Washington, DC, USA, 2003. IEEE Computer Society.

[63] Hrvoje Nikšić. Gnu wget. `http://www.gnu.org/software/wget/`.

[64] Alexandros Ntoulas, Junghoo Cho, and Christopher Olston. What's new on the web?: the evolution of the web from a search engine perspective. In *WWW '04: Proc. of the 13th international conference on World Wide Web*, pages 1–12, New York, NY, USA, 2004. ACM Press.

[65] Vicki L. O'Day and Robin Jeffries. Orienteering in an information landscape: how information seekers get from here to there. In *CHI '93: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 438–445, New York, NY, USA, 1993. ACM Press.

[66] Kenton O'Hara and Abigail Sellen. A comparison of reading paper and on-line documents. In *CHI '97: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 335–342, New York, NY, USA, 1997. ACM Press.

[67] Peter Pirolli and Stuart Card. Information foraging in information access environments. In *CHI '95: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 51–58, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[68] Planzo. http://www.planzo.com.

[69] Dennis Quan, David Huynh, and David Karger. Haystack: A platform for authoring end user semantic web applications. In *ISWC '03: Proc. of 2nd international Semantic Web Conference*, 2003.

[70] George Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel, and Maarten van Dantzich. Data mountain: using spatial memory for document management. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 153–162, New York, NY, USA, 1998. ACM.

[71] Steven F. Roth, Peter Lucas, Jeffrey A. Senn, Christina C. Gomberg, Michael B. Burks, Philip J. Stroffolino, John A. Kolojejchick, and Carolyn Dunmire. Visage: A user interface environment for exploring information. In *Proc. of the IEEE Symposium on Information Visualization*, pages 3–12, 1996.

[72] Daniel M. Russell, Mark J. Stefik, Peter Pirolli, and Stuart K. Card. The cost structure of sensemaking. In *CHI '93: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 269–276, New York, NY, USA, 1993. ACM Press.

[73] Alex Safonov. Web macros by example: users managing the www of applications. In *SIGCHI Extended Abstracts*, pages 71–72, 1999.

[74] Eric Saund, David Fleet, Daniel Larner, and James Mahoney. Perceptually-supported image editing of text and graphics. In *UIST '03: Proc. of the 16th annual ACM symposium on User interface software and technology*, pages 183–192, New York, NY, USA, 2003. ACM Press.

[75] Bill N. Schilit, Gene Golovchinsky, and Morgan N. Price. Beyond paper: supporting active reading with free form digital ink annotations. In *CHI '98: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 249–256, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[76] m.c. schraefel, Yuxiang Zhu, David Modjeska, Daniel Wigdor, and Shengdong Zhao. Hunter gatherer: interaction support for the creation and management of within-web-page collections. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 172–181, New York, NY, USA, 2002. ACM.

[77] Evan Schrier, Mira Dontcheva, Charles Jacobs, Geraldine Wade, and David Salesin. Adaptive layout of dynamically aggregated documens. In *IUI '08: Proc. of the 13th international conference on Intelligent user interfaces*, page (to appear), 2008.

[78] Abigail J. Sellen and Richard H. R. Harper. *The Myth of the Paperless Office*. The MIT Press, 2001.

[79] Abigail J. Sellen, Rachel Murphy, and Kate L. Shaw. How knowledge workers use the web. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 227–234, New York, NY, USA, 2002. ACM.

[80] Frank M. Shipman, Haowei Hsieh, J. Michael Moore, and Anna Zacchi. Supporting personal collections across digital libraries in spatial hypertext. In *JCDL '04: Proc. of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 358–367, New York, NY, USA, 2004. ACM Press.

[81] Frank M. Shipman and Catherine C. Marshall. Spatial hypertext: an alternative to navigational and semantic links. *ACM Computing Surveys*, 31(4es):14, 1999.

[82] Frank Shipmann, Catherine Marshall, and Mark LeMere. Beyond location: hypertext workspaces and non-linear views. In *Proc. of the tenth ACM Conference on Hypertext and hypermedia*, pages 121–130, 1999.

[83] Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. In *IEEE Trans. on Visualization and Computer Graphics*, 2002.

[84] Jeffrey Stylos, Brad A. Myers, and Andrew Faulring. Citrine: providing intelligent copy-and-paste. In *UIST '04: Proc. of the 17th annual ACM symposium on User interface software and technology*, pages 185–188, New York, NY, USA, 2004. ACM Press.

[85] Atsushi Sugiura and Yoshiyuki Koseki. Internet scrapbook: automating web browsing tasks by demonstration. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 9–18, New York, NY, USA, 1998. ACM.

[86] Kuo-Chung Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433, 1979.

[87] Wikimedia Foundation, Inc. http://www.wikipedia.org/.

[88] Jeffrey Wong and Jason I. Hong. Making mashups with marmite: towards end-user programming for the web. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1435–1444, New York, NY, USA, 2007. ACM.

[89] Allison Woodruff, Ruth Rosenholtz, Julie B. Morrison, Andrew Faulring, and Peter Pirolli. A comparison of the use of text summaries, plain thumbnails, and enhanced thumbnails for web search tasks. *Journal of the American Society for Information Science and Technology*, pages 172 – 185, 2002.

[90] World Wide Web Consortium. Cascading style sheets. http://www.w3.org/Style/CSS/.

[91] World Wide Web Consortium. The extensible stylesheet language. http://www.w3.org/Style/XSL/.

[92] Yahoo! Inc. http://www.flickr.com/.

[93] Yahoo! Inc. http://pipes.yahoo.com/.

[94] Christopher C. Yang and Fu Lee Wang. Fractal summarization for mobile devices to access large documents on the web. In *WWW '03: Proc. of the 12th international conference on World Wide Web*, pages 215–224, New York, NY, USA, 2003. ACM Press.

[95] Yanhong Zhai and Bing Liu. Web data extraction based on partial tree alignment. In *WWW '05: Proc. of the 14th international conference on World Wide Web*, pages 76–85, New York, NY, USA, 2005. ACM Press.

VITA

Mira Dontcheva received her B.S.E in Computer Engineering from the University of Michigan in 2000. In 2003 she received her M.S. in Computer Science from the University of Washington. She stayed at UW to continue her studies, and in January 2008 she received her Ph.D in Computer Science.