Summarizing Personal Web Browsing Sessions

Mira Dontcheva[1] *Steven M. Drucker*[3] *Geraldine Wade*[4] *David Salesin*[1,2] *Michael F. Cohen*[3]
[1]Computer Science & Engineering
University of Washington
Seattle, WA 98105-4615
{mirad, salesin}@cs.washington.edu

[2]Adobe Systems
801 N. 34th Street
Seattle, WA 98103
salesin@adobe.com

[3]Microsoft Research, [4]Microsoft
One Microsoft Way
Redmond, WA 98052-6399
{sdrucker, gwade,
mcohen}@microsoft.com

ABSTRACT

We describe a system, implemented as a browser extension, that enables users to quickly and easily collect, view, and share personal Web content. Our system employs a novel interaction model, which allows a user to specify webpage extraction patterns by interactively selecting webpage elements and applying these patterns to automatically collect similar content. Further, we present a technique for creating visual summaries of the collected information 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. Informal evaluation shows that our approach works well for popular websites, and that users can quickly learn this interaction model for collecting content from the Web.

ACM Classification H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General Terms Design, Human Factors, Algorithms

Keywords: Information management, webpage extraction patterns, template-based summarization

INTRODUCTION

The proliferation of Web content and the advances in search technology have fundamentally changed our lives. Today, to plan a vacation, you no longer go to a travel agent; when making purchases, you no longer go to your neighborhood specialty store; and when learning about a new topic, you no longer go to the library. Instead, you search the Web. While finding information is becoming easier and easier, collecting and organizing Web research and coming back to it remains difficult, especially when the research is exploratory. Sharing the results of the browsing session with others can be even more difficult. As found by Sellen et al. [20], research browsing 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. Current methods for collecting and organizing Web content include saving bookmarks or tabs, collecting content in documents, or storing pages locally. While useful, 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.

In this paper, we present a new approach to collecting and organizing Web information. Our approach is motivated by Gibson et al.'s analysis [8] of the amount of template content on the Web. Their study shows that 40-50% of the content on the Web is template content and that template material has grown and is continuing to grow at a rate of approximately 6% per year. Studies of users' habits for collecting Web content [19] further show that people are often interested in collecting not only full webpages but also pieces of webpages. These findings motivated us to design an interface that leverages the growing amount of templated information on the Web and allows users to gather only the pieces of content they find relevant to their task.

Our interface includes two parts: user specification of relevant content, and automatic collection of similar content. The user specifies relevant content by interactively selecting webpage elements and labeling each element with a keyword, chosen from a predefined set. The selected content is stored locally and used to create an *extraction pattern* that can be applied to collect more content from similar pages. Extraction patterns specify the locations of selected elements using the structure of the webpage and, optionally, text that must be matched in any new pages. When the user visits a new page, he can automatically collect all of the content that is analogous to the content previously selected as relevant. Further, we provide an interface for collecting content from multiple pages simultaneously by allowing the user to select hyperlinks that point to pages of interest.

To organize and present the gathered content, we employ layout templates that create visual summaries. A layout template specifies how the content should be organized and formatted and defines a set of user interactions available within a summary. The templates filter and present the content according to the labeling specified by the user during content selection.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
UIST'06, October 15–18, 2006, Montreux, Switzerland..
Copyright 2006 ACM 1-59593-313-1/06/0010 ...$5.00.

To demonstrate possible content presentations, we have implemented several layout templates, including a map, calendar, and grid. Our system is not limited to these designs and can incorporate additional layout templates.

Our work has two key contributions: a novel interaction mechanism for collecting Web content semi-automatically, and a presentation technique for the gathered content that uses pre-defined layout templates to construct summaries that are interactive and can be shared with others. We demonstrate our approach on three types of tasks: comparison shopping, travel planning, and researching academic papers. However, our approach can be used for any type of Web research and can be applied to other domains. User feedback shows that people can quickly learn to use our system and find many tasks for which it is useful.

RELATED WORK

Our work has its roots in the area of information workspaces and systems like WebBook [5], Data Mountain [17], and TopicShop [1], which present different mechanisms for presenting and organizing collections of webpages. Our system is most closely related to Hunter Gatherer [19] and Internet Scrapbook [22], which provide the ability to save webpage elements and place them together in a document. Like these two systems, we provide an interface that allows the user to select pieces of webpages; however, we go further by making use of the user selection to create extraction patterns that can then be used to gather content automatically. We also provide a mechanism for organizing the gathered content in richer ways, such as on a map or calendar, and for different purposes, such as for a PDA or printer.

Our use of user-specified labels for relating content from different sources is similar to approaches in Semantic Web applications, such as Piggy Bank [11] and Thresher [10]. Our work differs from these in two ways. First, we create visual summaries of the gathered content, while Piggy Bank and Thresher store the collected data in a database. Second, our interface for labeling the content is more accessible to the average Web user, because the user must simply right-click and select a keyword. Piggy Bank requires that the user write scripts, and Thresher expects that the user has some knowledge of Semantic Web constructs.

Another related system is Chickenfoot [3], which also automates repetitive Web tasks through the use of a high-level interface and pattern matching in webpages. However, our goals, approaches, and results are very different. Chickenfoot is designed for customizing webpage appearance and thus provides a scripting interface for manipulating webpage elements. Our system is designed for collecting and sharing Web content and thus provides an interface for collecting elements and organizing them in different ways.

Automated layout has been explored previously including approaches that use constraints [2] and employ machine learning [14]. Jacobs et al. [13] combine constraints with predefined layout templates to create adaptive documents. Our system does not use constraints, as we rely on Cascading Style Sheets (CSS) and the browser for correct layout. In some cases, we override the browser's layout engine, using scripts, to create more aesthetic results.

Bibliographic reference tools, such as RefWorks [16] and EndNote [6], are specifically designed for collecting and managing academic references. These systems have specialized filters, much like our extraction patterns, for electronic references websites and can automatically import all the data for a particular reference into a database. Our system complements such systems by giving the user an interface for selecting personally relevant bibliographic information. Furthermore, our approach is more general and is applicable to other Web content.

USER INTERFACE

We designed the user interface 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. Our system is implemented as an extension to the Firefox browser and is presented to the user through a toolbar (see Figure 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 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 2) has buttons for opening and saving summaries and a menu for changing the layout template used to compose the summary.

Sample User Scenario

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

The user starts the browsing session by looking through restaurant listings at `www.seattleweekly.com`. When he finds a potential restaurant, he presses the "Select" button, 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 our layout templates are defined with respect to these labels. Figure 1, step 1, shows that the user has selected three elements and has assigned them the "name," "image," and "description" labels.

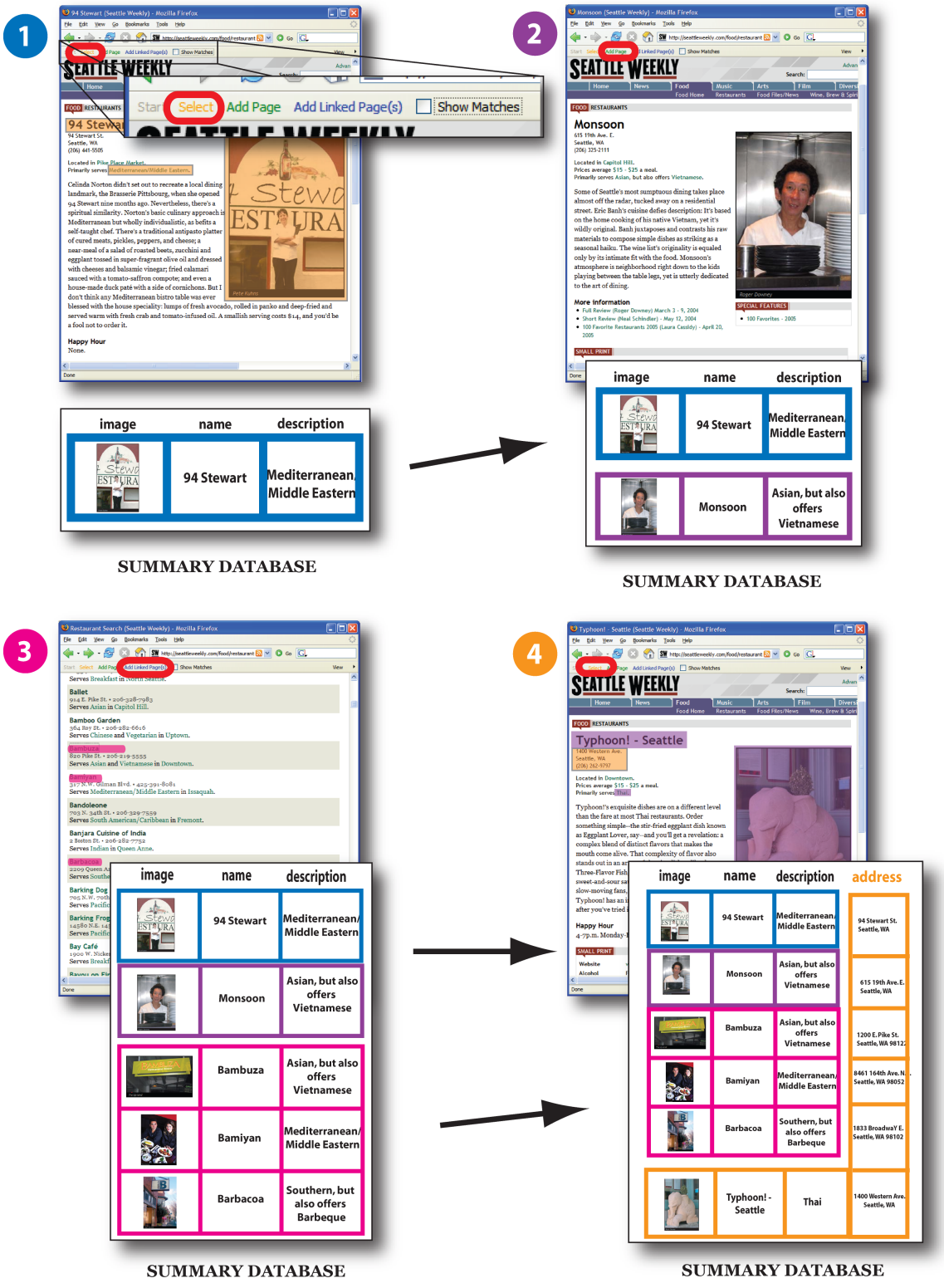

Figure 1: In step 1, the user selects the picture, name, and type of cuisine for a restaurant; labels them with “image,” “name,” and “description”; and adds them to the summary database, shown below the restaurant webpage. When the user finds a new restaurant, as shown in step 2, he can add the same 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.

Figure 2: The grid layout template places all the content in the database on a grid in groups, according to extraction pattern.

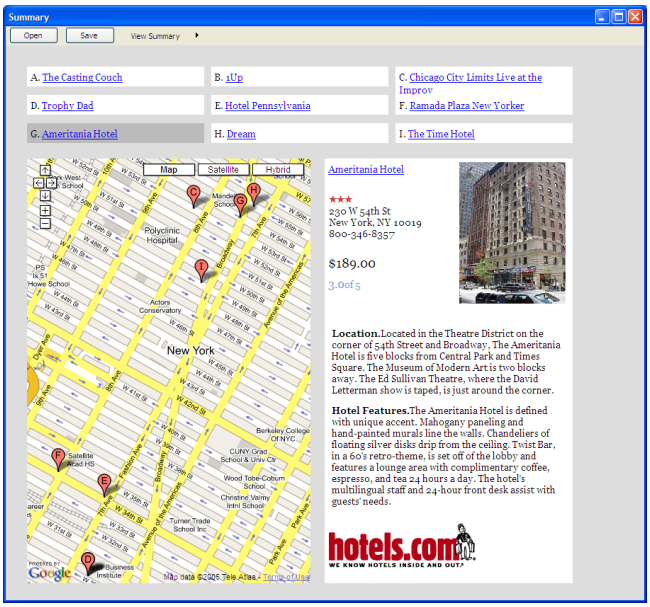

Figure 3: With the map layout template, the user can view the content with respect to a geographical map. The template filters the database and presents only content that includes an address.

To 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 he wants to save, 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 1, step 2). He can also add several restaurants to the summary simultaneously (Figure 1, step 3) by selecting hyperlinks. 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 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 3).

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 our system gathers content using the structure of a given webpage, as we describe in the next 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 experience we have created 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. Next we describe the details of how we make this user experience possible.

SYSTEM DESCRIPTION

We first describe our overall system design and then explain the technical details for each part. Our 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 for a visual description of our system.

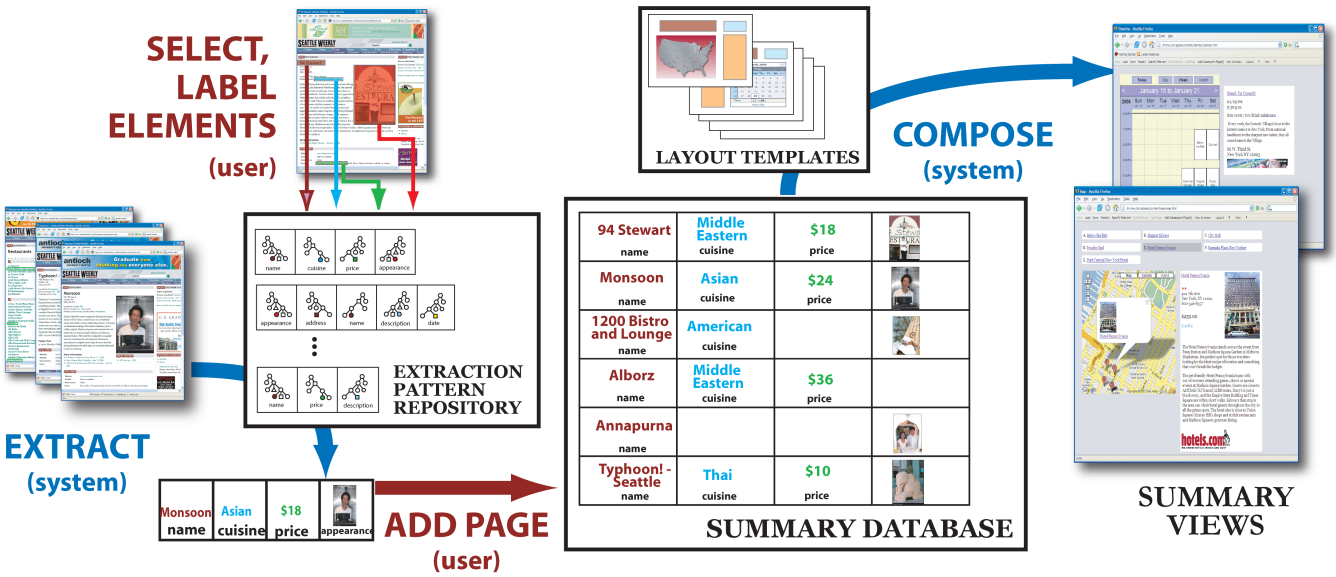

Figure 4: The user-selected page elements define extraction patterns that are stored in the extraction pattern repository. Every new page the user visits is compared with the patterens, and if matching page elements are found, the user can add them to the summary database. The layout templates filter the database and compose summary views. The user can view the database contents at any time with any view.

Gathering content

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.

The page-element selection interface uses the 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. 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, the magnitude of these changes and their effect on the extraction rules is not yet clear.

In addition to the *structural* extraction rules just described, our 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. 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 extraction pattern 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

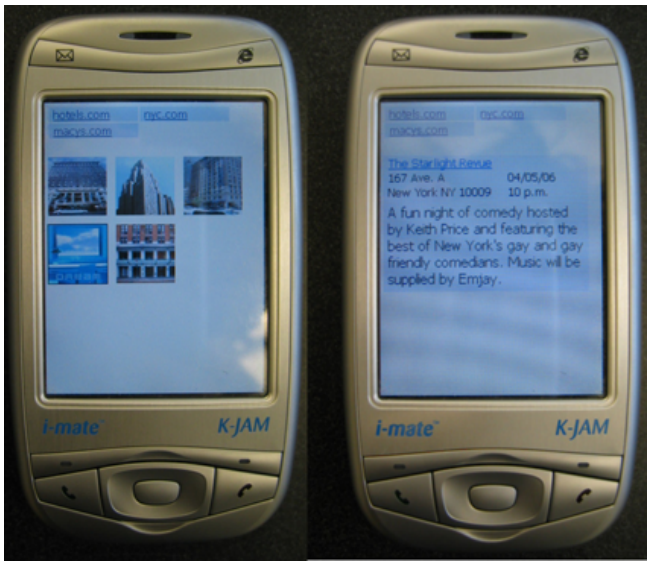

Figure 5: The PDA layout template is designed for a small-screen device so that the user can take the summary anywhere. The layout separates the content into tabs according to website and provides detailed views when the user clicks on an item.

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 for us to automate collecting information from the Web, this automation comes at a cost. Our system 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.

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 we provide the same set of labels for all webpages, we can create layout templates that filter the database using 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 position. 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 we 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.

Figure 6: The print layout template formats the content so that it can be printed.

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, our 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.

We 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 framework 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.

Layout templates

To demonstrate different possibilities for summarizing Web content we implemented a set of layout templates. We 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.

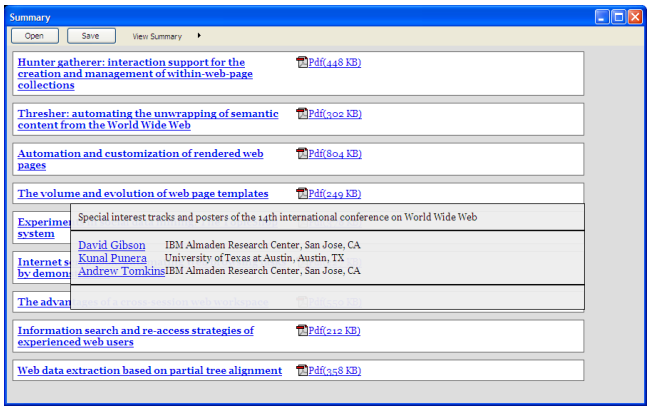

Figure 7: The text layout template is designed for collecting text content, as is common in a literature search. The paper titles are placed in a list, and further details for each paper, such as the authors and abstract, are available through mouse interactions.

The *grid layout template* (Figure 2) places webpage elements found on the same page into one visual element, a box, and arranges these elements according to extraction pattern. If there is only one pattern specified for a website, as in Figure 2, 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 5) 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 5 the user clicked on the "Starlight Review" and is now viewing details about this event.

The *print layout template* (Figure 6) 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 7) is designed for text-intensive tasks, such as literature surveys. The template organizes the names — or, in the case of Figure 7, 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.

We present two anchor-based layouts, a map and a calendar. An anchor-based layout allows us to relate items to one another through a common element. To create this relationship we 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 [21], we use heuristics to find the address, date, and time within the selected nodes. While these heuristics are not capable 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 3) 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 we use the Google Maps API [9].

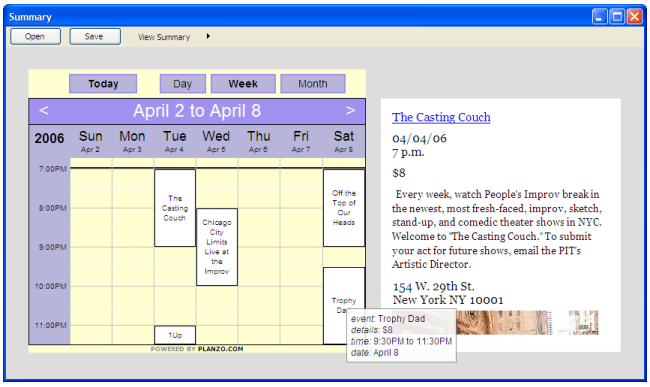

Figure 8: With the calendar layout template, the user can view content with respect to time. When the user clicks on an event, details about the event are displayed in a panel next to the calendar.

The *calendar layout template* (Figure 8) 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 [15].

EVALUATION

We evaluated our system by 1) measuring the performance of our extraction patterns on a set of popular websites, and 2) soliciting user feedback in an informal user study.

Quantitative analysis

To measure the performance of our extraction patterns we used nine popular websites for shopping, travel planning, and academic reference. We selected the websites for the shopping and travel categories from `www.alexa.com`, which tracks Web traffic patterns and reports the most visited websites. For the academic reference category, we chose the three most popular reference websites in computer science. For each website, we visited 25 randomly selected webpages. On the first visited page for each website, we selected 3-6 page elements to build an extraction pattern and then collected content from 24 more pages. We then measured how much of the content was collected from the 24 webpages and computed an *accuracy* for each pattern. We define accuracy as the number of page elements actually collected using the extraction patterns divided by the number of page elements that were actually present in all the visited webpages. Table 1 shows the accuracy of our patterns on the selected websites. We 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.

domain	website	accuracy (%)	initial els	addl. els
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

Table 1: We measure the accuracy of our approach on nine popular websites. Our system is able to collect most of the information with only a few user selections. The fourth column shows the number of initially selected page elements. Websites that use more than one template, such as Ebay and Expedia, require more user assistance. The fifth column shows the number of additional element selections necessary to collect 100% of the content. The additional user selections are still many fewer than would be necessary if the user were to collect the same information manually.

Our results show that popular websites tend to use templates to display content, and our system can robustly extract content from such websites. The two websites where our system has trouble 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, we did measure pages created by the same seller and for those our system was 100% accurate. Similarly, Expedia uses several different layouts for hotels. We suspect the hotel layouts have regional differences and also depend on when the hotel was added to the Expedia database. For this analysis, we selected hotels from five different cities in various parts of the world.

We used structural extraction patterns for all websites, except the CiteSeer website. The CiteSeer website is a wiki and can be edited by anyone; thus the structural layout of pages varies from paper to paper. We collected the name and authors of each paper with structural extraction rules and the link to the PDF of the paper and BibTex reference with content-based extraction rules. The text strings for the content-based rules were "PDF" and "author =".

Informal user evaluation

We conducted an exploratory evaluation of our system with three questions in mind:

1. Can people effectively specify extraction patterns by selecting from a webpage's underlying Document Object Model?
2. Are people happy with the workflow of specifying patterns and subsequently adding extracted items to a summary?
3. What kinds of operations did users feel should be supported in interacting with the summary?

We 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 [12], 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 [4], those that did not save Web content simply hoped they would find it again.

We 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. Our 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.

Observations and feedback

The amount of aid we offered the participants decreased with each task. For the first task, we made sure the participants understood the pattern specification mode and how to add information using the patterns. In the second task we 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, we found that this was in part due to the immediate highlighting response to the cursor. In our design, we aimed to make the selection interface quick and responsive, and perhaps we 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. [18] 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 our 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, we can use our current approach and just group the selected DOM nodes together. For the second case, we can use our approach to find matching nodes, but we 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 we match the address from other pages. For this, we 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. We suspect that for some of our 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 we designed our interface with the goal of minimizing user effort, at the onset of the user evaluation we were concerned about requiring manual specification of extraction patterns. We were 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." We were 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 that we add multi-page extraction patterns. Extending our framework to include content from multiple pages could be done with an approach similar to C3W [7] 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. Our system can be extended to provide such sub-page patterns. Zhai and Liu [23] present an algorithm that automatically extracts all the fields in such a list. We could combine our 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. We 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 that we provide mechanisms for configuring both the label choices and the summary layouts. We 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 labels; however, any new labels would have to be incorporated in the templates. In the future, we hope to create a graphical authoring tool for the templates, which would make it possible to dynamically add new labels as the user is gathering content. We expect that there will be two types of users of our system. There will be those who use the defaults that we 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 our tool. 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`.

CONCLUSIONS AND FUTURE WORK

This paper presents an approach for collecting and viewing content found on the Web in a personalized way. Based on this approach we implemented a system that is applicable for a variety of tasks, including comparison shopping, travel planning, and reference collecting. An evaluation of our system leads us to the firm conclusion that such an approach is much needed and welcomed by Web users. Furthermore, our work suggests new directions for research, such as designing an end-user interface for customizing layouts and using the personalized summaries in collaborative settings.

Currently, the layout templates we use are pre-defined with the help of a designer and are limited in the types of interactions possible. We envision allowing the user to interactively specify the templates either with an authoring tool or directly in the summary by applying filters or interactively selecting which content should be displayed. We plan to provide a set of default templates that the user can customize interactively to create new specialized summary views.

Our system is also appropriate for collaborative settings, when several people are gathering information and interacting with the summary simultaneously. We plan to explore 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.

While we are satisfied that our user study participants didn't find the overhead of selecting and labeling page elements too time consuming, we were curious how they felt about using a public repository of extraction patterns instead of manually selecting page elements. We asked the participants whether they would be willing to share and use a public repository of extraction patterns. The participants were timid about using patterns defined by others as they felt they were no longer in control of the gathered information. Seven of the participants said they would be hesitant to use a public repository, as they were unsure how much work would be required. We anticipate that although a public repository of extraction patterns may not be used by everyone, a public repository could be useful for automatically labeling page elements, especially since four of the nine participants did not remember to label the page elements they selected.

Finally, we plan to evaluate our system more rigorously. We plan to evaluate the robustness of our extraction patterns over time and understand the type and frequency of changes of website templates. Given the positive reaction in the pilot study, we hope to release the system and study users as they carry out their own tasks over several months. We want to explore how people accumulate content and use our framework over time. We expect that users will find new tasks for our system in addition to those we have already explored.

ACKNOWLEDGEMENTS

We thank our study participants for spending time with our system and providing useful feedback on future improvements. We thank Blaine Bell for his assistance with the Google Maps API and Ken Hinckley for helping us with the text. Funding and research facilities were provided by Microsoft Research and the University of Washington GRAIL lab.

REFERENCES

1. B. Amento, L. Terveen, and W. Hill. Experiments in social data mining: The TopicShop system. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 10(1):54–85, 2003.
2. G. J. Badros, A. Borning, K. Marriott, and P. Stuckey. Constraint cascading style sheets for the Web. In *Proceedings of UIST'99*, pages 73–82.
3. M. Bolin, M. Webber, P. Rha, T. Wilson, and R. C. Miller. Automation and customization of rendered web pages. In *Proceedings of UIST'05*, pages 163–172.
4. H. Bruce, W. Jones, and S. Dumais. Keeping and refinding information on the web: What do people do and what do they need to do? In *Proceedings of ASIST 2004*.
5. S. Card, G. Roberston, and W. York. The WebBook and the Web Forager: An information workspace for the World-Wide Web. In *Proceedings of SIGCHI'96*, pages 111–117.
6. EndNote. http://www.endnote.com.
7. J. Fujima, A. Lunzer, K. Hornbæk, and Y. Tanaka. Clip, connect, clone: combining application elements to build custom interfaces for information access. In *Proceedings of UIST '04*, pages 175–184.
8. D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *Special interest tracks and posters of WWW'05*, pages 830–839.
9. Google Inc. http://www.google.com/apis/maps/.
10. A. Hogue and D. Karger. Thresher: Automating the unwrapping of semantic content from the World Wide Web. In *Proceedings of WWW '05*, pages 86–95.
11. D. Huynh, S. Mazzocchi, and D. Karger. Piggy Bank: Experience the semantic web inside your Web browser. In *Proceedings of the International Semantic Web Conference*, 2005.
12. JabRef. http://jabref.sourceforge.net/.
13. C. Jacobs, W. Li, E. Schrier, D. Bargeron, and D. Salesin. Adaptive grid-based document layout. *ACM Transactions on Graphics*, 22(3):838–847, 2003.
14. S. Lok and S. Feiner. A survey of automated layout techniques for information presentations. In *Proceedings of the SmartGraphics Symposium*, Hawthorne, NY, USA, 2001.
15. Planzo. http://www.planzo.com.
16. RefWorks. http://www.refworks.com.
17. G. Robertson, M. Czerwinski, K. Larson, D. Robbins, D. Thiel, and M. van Dantzich. Data Mountain: Using spatial memory for document management. In *Proceedings of UIST'98*, pages 153–162.
18. E. Saund, D. Fleet, D. Larner, and J. Mahoney. Perceptually-supported image editing of text and graphics. In *Proceedings of UIST '03*, pages 183–192.
19. m.c. schraefel, Y. Zhu, D. Modjeska, D. Wigdor, and S. Zhao. Hunter Gatherer: Interaction support for the creation and management of within-web-page collections. In *Proceedings of WWW '02*, pages 172–181.
20. A. J. Sellen, R. Murphy, and K. L. Shaw. How knowledge workers use the web. In *Proceedings of SIGCHI'02*, pages 227–234.
21. J. Stylos, B. A. Myers, and A. Faulring. Citrine: Providing intelligent copy-and-paste. In *Proceedings of UIST '04*, pages 185–188.
22. A. Sugiura and Y. Koseki. Internet Scrapbook: Automating Web browsing tasks by demonstration. In *In Proceedings of UIST '98*, pages 9–18.
23. Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *Proceedings of WWW '05*, pages 76–85.