Relations, Cards, and Search Templates: User-Guided Web Data Integration and Layout

Mira Dontcheva[1] *Steven M. Drucker*[3] *David Salesin*[1,2] *Michael F. Cohen*[4]

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

UW CSE Technical Report 2007-04-03

ABSTRACT

We present three new interaction techniques for aiding users in collecting and organizing Web content. First, we demonstrate an interaction technique for creating associations between websites facilitating the automatic collection of related content. Second, we present an authoring interface that allows users to quickly merge content from many different websites into a uniform and personalized representation, which we call a card. Finally, we introduce a novel search paradigm that leverages the relationships in a card to direct search queries to extract relevant content from multiple sources and fill a new series of cards instead of just returning a list of URLs. Preliminary feedback from users is positive and validates our design.

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

General Terms Design, Human Factors, Algorithms

KEYWORDS: Web content extraction, Web search, template-based representation, view and layout editing

INTRODUCTION

For many people today, the World Wide Web is a major source of information. One may go to the Web to make travel plans, shop, read news or favorite blogs, learn about a new place or topic, or even watch a favorite TV show. As more information becomes available, it becomes more difficult to not only find the appropriate information but also to collect, organize and understand that information. Our work focuses on helping people interact and gather Web content. Our goal is to lower the effort necessary for collecting, organizing, managing, and sharing that content.

Previously [7], we developed the summaries framework, which provides interaction techniques that help users collect and organize content semi-automatically. We showed there that users can interactively create webpage extraction patterns that can then be used to automatically collect similar types of Web content. We also demonstrated a variety of representations for the collected content presented as rich visual summaries through the use of layout templates. In this paper we describe our ongoing research to make the process of collecting and organizing Web content even easier. We present three new techniques that build on the existing summaries framework and interaction paradigms.

First, we propose an interaction technique that allows users to specify ***relations between websites*** and use these relations to automatically collect data from multiple websites. When the user relates content collected from one website with another website, he is implicitly defining search queries that are used to collect additional content. For example, a user may be interested in collecting restaurants from one website and reviews for the restaurants from another website. By associating information about the same restaurant, the user specifies a relation between the two websites. Now, when the user collects information about a new restaurant, the reviews for that restaurant are automatically collected from the review website.

Second, we present an interface for ***merging content from multiple websites*** and organizing it visually. With our card designer the user can easily create a new *card* that displays any amount of information from related websites in a uniform and coherent form. A card specifies the information that should be displayed and its visual layout. For example, when collecting information about restaurants, the user can create a card that shows only the name, price, and reviews for a restaurant or one that shows the address and hours as well. Cards can be designed for any size or purpose and can be stored or shared with others.

Finally, we introduce a novel search paradigm for collecting content from the Web with ***search templates***. *A search template* narrows the search space available to the user through general search engines to create a personalized view of available Web content. A search template most often begins from a card that defines a set of source websites and relations between those websites. We demonstrate a simple editor for creating search templates from already created cards. With

this paradigm the user can simply select a search template and type a keyword query. The system searches these sites and presents the search results not just as a list of a hyperlinks but as a uniform and organized collection of cards.

Our three key contributions work in unison: an interface for establishing relations between websites, a methodology to merge content from many websites into cards, and a new search paradigm based on the card structure that returns cards filled with content rather than a list of URLs. We also report on an informal pilot study showing that users are receptive to these new techniques for collecting and organizing Web content and want to use them for their own tasks.

RELATED WORK

Some of the early systems for managing Web content include WebBook [5], Data Mountain [26], and TopicShop [1], which present various mechanisms for presenting and organizing collections of webpages. More recent systems such as Hunter Gatherer [28] and Internet Scrapbook [29] let users collect pieces of webpages and place them together into documents. The Semantic Web community has been addressing this problem with systems like Piggy Bank [14] and Thresher [13], which collect structured content from the Web and store it in databases for later retrieval. With the summaries framework [7] we showed how interactive tools can be used to extract structured content from the Web and how this content can be organized into rich visual summaries. In this work we give the user interactive tools for specifying relations between disparate data sources. We then use these relations to extract structured content from multiple sources simultaneously. To the best of our knowledge, our work is the first interactive system of this kind.

Collecting content using relations

In the last ten years database researchers have explored a number of techniques for data integration [12]. Data integration is the problem of combining 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 [24]. Our work is complementary to database research in that it offers interactive techniques for data integration on a personal scale. We 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. If one were to treat the World Wide Web as a set of databases, then our mappings could be translated into SQL queries. For example, the relations we describe are analogous to a `JOIN` operation over multiple websites.

Another related area of research is end-user programming for the Web. Chickenfoot [4] is a system that allows users to write scripts that customize webpage appearance and automate repetitive Web tasks. RecipeSheet [23] is a general-purpose framework for applying data flow operations typically found in spreadsheets to other types of content, such as text, webpages, or XML. C3W [10] is one application of the RecipeSheet framework that allows users to interactively build customized Web interfaces by clipping and connecting pieces of webpages. Marmite [33] and Yahoo Pipes [34] are recent graphical tools that also use a data-flow architecture but focus specifically on creating Web-based mashups. Our work is related to all of these systems in that it provides a simple graphical interface for mixing content from different sources together; however, our intended audience includes essentially all Web users and not just programmers. While previous systems let users graphically create scripts, our goal is to let users perform the task of collecting and organizing Web content. As a result, we do not expose the underlying programming model and allow the user to manipulate the content directly.

Interactive layout editing

Interactive layout editors have been around for a long time and are used in many commercial packages. Some of the earliest work [30] allowed users to interactively draw and specify constraints. Subsequent work has looked at inferring layout constraints from layout snapshots [21] or user interaction [19]. Our interactive authoring tool follows some of the same ideas and allows users to position layout containers interactively. Our card designer is similar to commercial HTML editors in that it generates HTML templates; however, it is designed for a novice user, and thus it attempts to automate layout as much as possible. Furthermore, our interactive editor is designed around a metaphor for merging content from different websites together. Thus, in addition to specifying layout parameters, it also includes information about the type of content it can display. When merging content from different websites the original design present in the websites no longer applies. With our card designer the user can reintroduce aesthetics and design and view all of the collected content in the same context.

To the best of our knowledge, we are the first to add an interactive layout editor to a system for collecting and organizing Web content. Typically, information management systems, such as PiggyBank [14] and TopicShop [1], focus on providing filtering and sorting functionality. With our system users can create personalized, uniform, and aesthetic presentations of Web content. We view this as an important aspect of collecting and organizing Web content.

Formatting search results

Most search engines today do not format search results. They merely provide the user with a list of hyperlinks and snippets. Stuff I've Seen [8] personalizes this list of hyperlinks by reordering the list using recent user actions. Clusty [32] goes one step further and clusters the search results according to topic, and Grokker [11] provides a visual interface for displaying the clusters as interactive graphical elements. Our work goes beyond clustering and reorganizing URLs. First, it extracts content from the search results. Then, it reorganizes the extracted content so that the entire collection appears uniform despite the fact that it may originate from multiple web-

sites.We propose template-based search as a new approach to Web search personalization.

THE SUMMARIES FRAMEWORK

The new interaction techniques we describe in this paper were built on top of the summaries framework we presented previously [7], and thus we first provide a brief description of this framework to give the reader context for the remainder of the paper.

The summaries framework provides an interface for interactively creating extraction patterns for webpages. The user can create patterns for any of the content included in a webpage and assign it meaning through a tag. Once a pattern has been created, it can be used to collect content from similar webpages automatically. Because the extraction patterns create structured content from unstructured webpages, the user can view his collection of content with pre-defined layout templates. The framework is implemented as an extension to the Firefox browser and is presented to the user through a toolbar. The toolbar opens a window where the user can collect Web content and also includes buttons for creating extraction patterns and saving content. In this work we continue to use the extension platform and provide functionality to the user through a toolbar and collection window. The entire system is implemented as an extension and is written in Javascript and XUL.

The extraction patterns that are part of the summaries framework use the structure of the webpage to collect content. However, there are other approaches for extracting structured content from webpages such as conditional random fields [20], context free grammars [31], or specialized extraction languages [16]. In this work, we use the extraction techniques that are already part of the summaries framework and focus our efforts on applications that use the extracted content.

EXAMPLE SCENARIO

We present our three new interaction techniques in the context of an example, which shows the steps taken by a user as he looks for a restaurant for a night out in Seattle. Figures 1 and 2 outline the process graphically (as does the associated video).

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 2a). 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 extracted and added to the collection. He can also collect hyperlinks pointing to potentially interesting restaurants, and the reviews for those restaurants will be automatically collected from `yelp.com` (Figure 2b).

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 schedule (Figure 2a). 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 collected automatically.

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 on which to design his card. He draws the outline first and then draws containers inside of the card. He places content from his collection right into the containers by drawing a line from the content to the container. 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 2c). 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 that the user can at any time return to the original content.

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 2d). 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 one of the cards designed by the user. To create a search template, he 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 3 shows a query for "chocolate cake" with the "recipes" search template, which retrieves and reformats recipes from `allrecipes.com` and `cooking.com`. Figure 4 shows several queries for different types of cars. Each card includes car specifications and reviews from `autos.msn.com` and `edmunds.com`.

Figure 1: Previously in [7] we developed the summaries framework, which allows users to semi-automatically extract content from webpages. The user can interactively select and tag webpage elements, (a), to create extraction patterns. Extraction patterns can be defined for any website allowing the user to collect related information into one collection, (b).

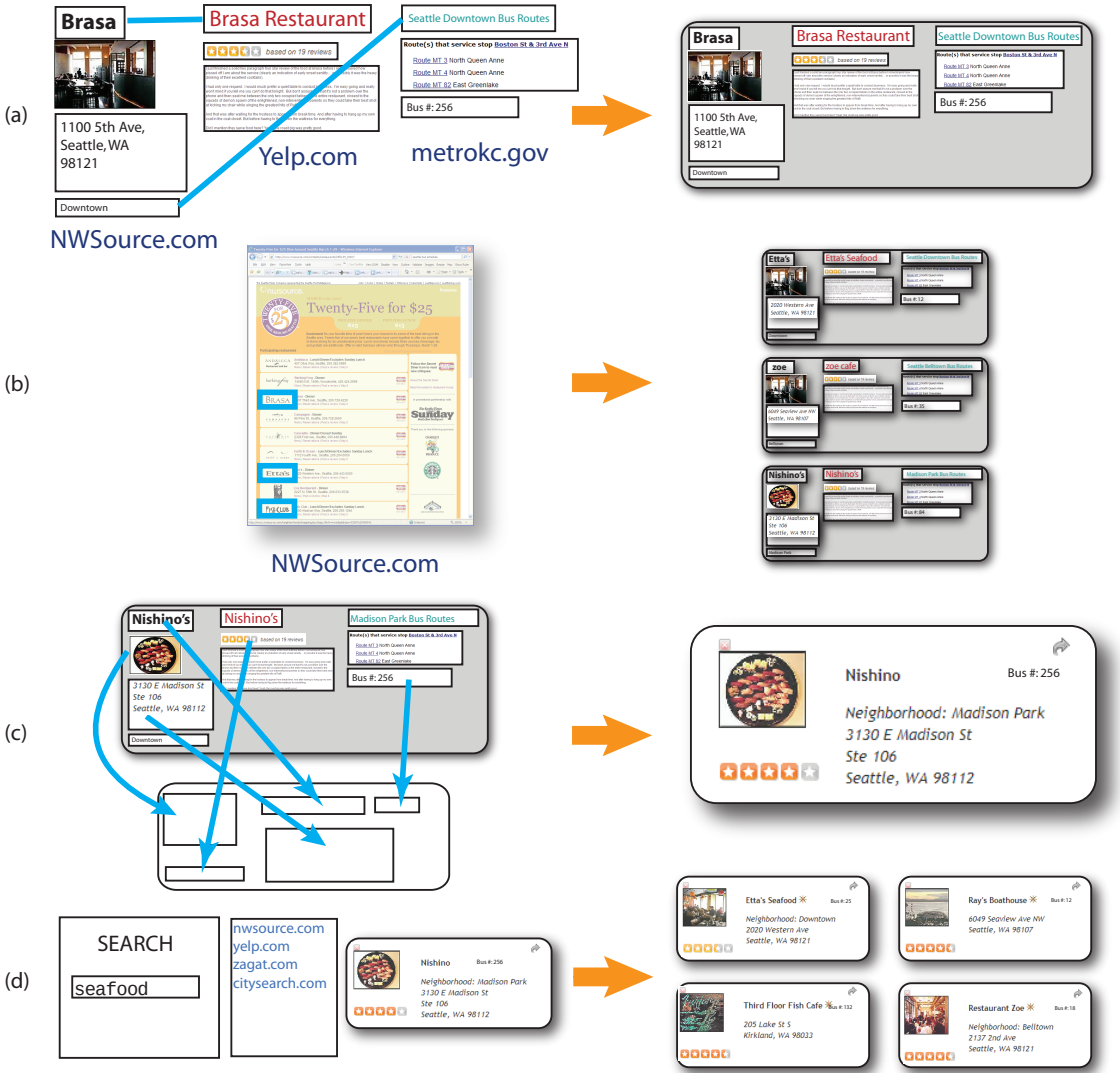

Figure 2: **(a)** The user draws lines between webpage elements, thereby creating directional relations from `nwsource.com` to `yelp.com` and `metrokc.gov`. **(b)** The user can collect more restaurants by selecting hyperlinks that point to interesting restaurants. Each new restaurant triggers the system 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.

Next we give a system overview and then describe each part of the system in detail.

SYSTEM OVERVIEW

The system includes a data repository, a set of user-defined cards, and a set of search templates. The *data repository* holds all of the content collected by the user according to the source webpage and semantic tags of the webpage elements. We 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 on 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.

RETRIEVAL USING RELATIONSHIPS

All of the content collected by the user is associated with semantic metadata or tags. The summaries framework provides these tags, but any other extraction algorithm would also provide this semantic information. When the user connects content collected from different webpages, he creates a relation. We 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 $r = website_A.tag_i \rightarrow website_B.tag_j$, we can express the process of automatically collecting content for any

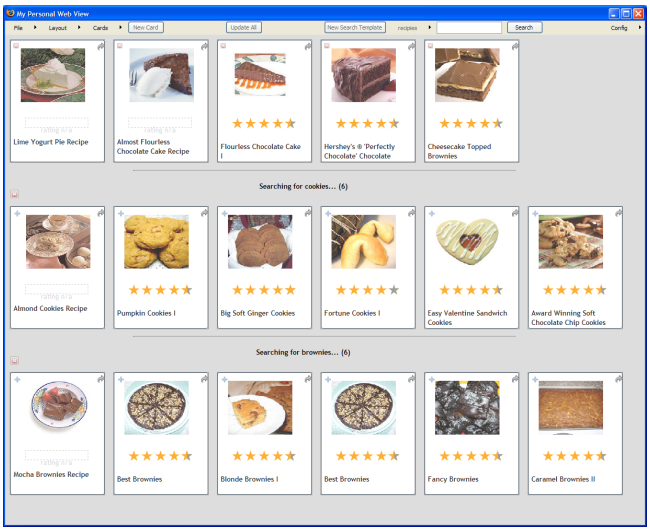

Figure 3: With the recipe search template, the user collects recipes from `cooking.com` and `allrecipes.com`. Here the user has a collection of five cards and has made two queries, one for "cookies" and and another for "brownies". The system automatically collects, extracts, and displays relevant recipes.

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, we use a number of different techniques to make this query feasible. We use the Google Search AJAX API to find webpages within $website_B$ that are relevant. To extract content from each of the search results, we employ the extraction patterns we developed previously [7]. Finally, we 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.

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. [22] pose the query formulation problem as a graph partitioning problem. Dong et al. [6] propose propagating information across relations to better inform similarity computation. Next, we describe how we address these two types of ambiguity.

Query formulation

To formulate the keyword query, we typically use only the extracted text content. We find that this type of query is usually sufficient and returns the appropriate result within the top eight search results. Sometimes, however, the query may include too many keywords, and the search results are irrelevant or cannot be extracted. In such cases, we reformulate the query using heuristics. If characters such as '/','-','+', or

':' appear in the text, we split the string whenever they appear and issue several queries using the partial strings. We 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 is time consuming because of the delay caused by the search and extraction process. We focus on finding good heuristics that could quickly retrieve results that were 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.

Search result comparison

For each query we 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 we compare 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 San Francisco). Because the relation between `nwsource.com` and `yelp.com` links the names of the restaurants, we compare the name "Ambrosia" to all the names of the extracted restaurants. We compare the strings by calculating the longest common substring. We 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, we compare any additional extracted elements. We again compute the longest common substring for corresponding webpage elements. In this example, we 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. We display the highest ranked extracted content but provide all 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.

Limitations

In the current implementation, we extract content from only eight search results because the Google AJAX Search API limits the search results to a maximum of eight. For our purposes using eight results has been sufficient and limits 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

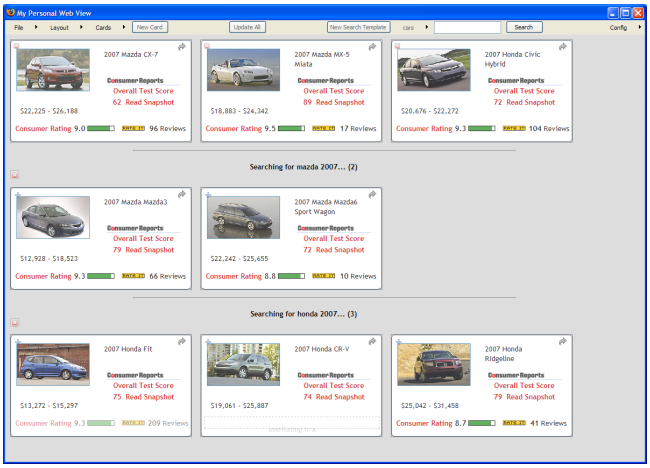

Figure 4: The user is shopping for cars and using a car search template to find new cars. He has three cards in his collection and has made two queries: "mazda 2007" and "honda 2007." Each card includes car reviews from `autos.msn.com` and `edmunds.com`.

of the restaurant chain. For those types of queries narrowing the search through one of the approaches mentioned above would be necessary.

Our 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 dynamic webpages, in subsequent work we hope to leverage research into macro recording systems such as WebVCR [2], Turquoise [25], Web Macros [27], TrIAs [3], PLOW [18], and Creo [9]. These systems allow users to record a series of interactions, store them as scripts, and replay them at any time to retrieve dynamic pages. Madhavan et al. [24] are developing information retrieval approaches to this problem that do not require user intervention.

Finally, in the current implementation we allow 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 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, we plan to explore other types of relations and also introduce transformations into the relations.

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

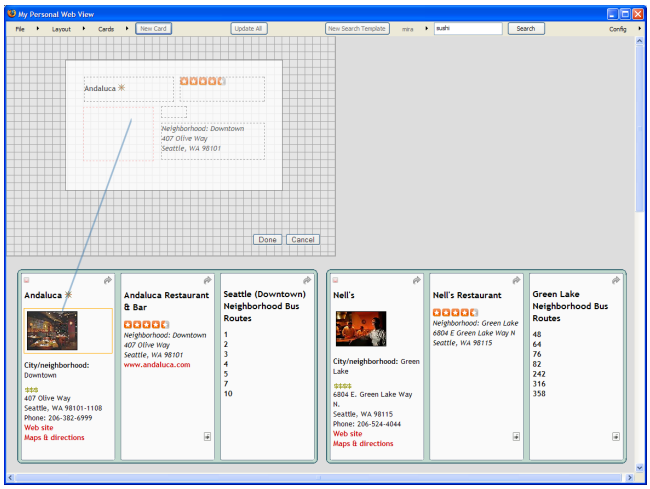

Figure 5: To design a new card, the user opens a canvas and begins drawing. He first draws the outline of the card and then draws containers. To place content in the containers, the user clicks and draws a line from the webpage element to the container. Here, the user is adding an image to the card.

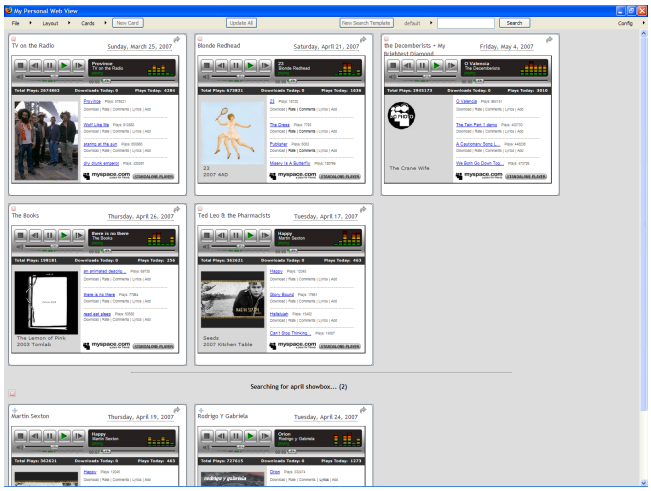

Figure 6: Here the user has related `upcoming.org` to `myspace.com` to automatically collect music samples for upcoming shows.

— 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. Our 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 our 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 his collection of Web content (see Figure 5). 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 ranking process, it is rendered semi-transparent to alert the user that they may want to confirm the information by clicking on it to go to 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 to add and remove more content. Currently, the cards are organized in a grid, but they could be arranged in any number of ways, such as on a map, calendar, or free-form canvas.

Cards can be interactive in nature, encoding interactions specific to the type of data that is collected. Our authoring tool does not currently provide capabilities for scripting. To include such capabilities, we could combine our authoring framework with the Exibit API [15] and offer automatic filtering and sorting. Arbitrary interactions would require exposing a scripting interface to the user.

TEMPLATE-BASED SEARCH

Our third contribution involves the use of *search templates* that combine search directly with the structure of the user designed cards and underlying relations. The user can thus bypass visiting webpages directly and collect content through a search template. Figure 7 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 7a), the system sends the query to a general search engine (Figure 7b), 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 7c) and triggers any relations that are in the template to collect additional content (Figure 7d-e). Due to limitations on the number of search results provided 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 7f). 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.

Figure 6 shows a collection of upcoming shows. In this example the user has related upcoming shows at `upcoming.org` with band webpages at `myspace.com`. Whenever the user adds a show 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 object, a Flash object, we can extract it just as we extract any other

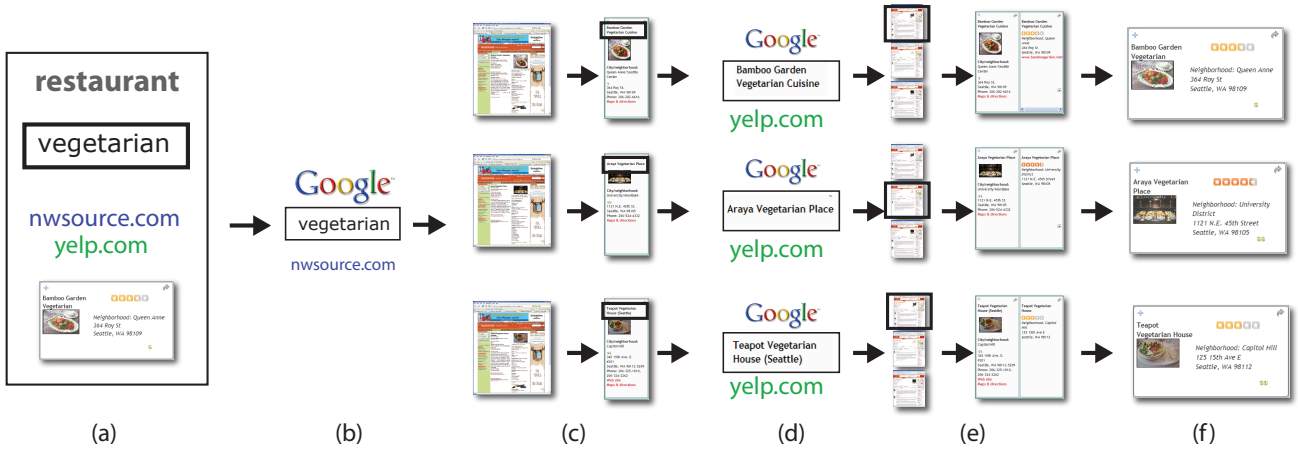

Figure 7: **(a)**The user types the query "vegetarian" into the restaurant search template to collect vegetarian restaurants. This search template includes two websites and a relation. The `nwsource.com` website is considered primary because information from `yelp.com` is collected through a relation. **(b)**The system issues the query to a general search engine but 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.

webpage elements. The music player retains full functionality, and the user can hit the play button on the control to listen to the music.

The success of template-based search lies in the ability to extract semantic information from webpages. Although semantic extraction is only in its infancy, we 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.

EXPLORATORY USER STUDY

We conducted an exploratory study to solicit feedback on our system. We 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 participants were active Web researchers, both personally and professionally. Three of the participants reported performing Web research activities several times a week, and the other three said they used the Web for research several times a month. Two of the participants had extensive bookmark collections, and the remaining four participants reported saving information infrequently and organizing only when absolutely necessary. The primary organization techniques were using email and adding information to documents.

We 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. 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 (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 the information to their collection. The users did not have to specify which content to collect or how to collect the content. They had to only click a button adding some of the information on the webpage to their collection. 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 asked to design a card for the restaurants in their collection. For the last task, the participants were asked to create a restaurant search template and make several queries for more restaurants.

Observations and feedback

Overall, the participants were very positive about using 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 has suggestions to improve different aspects of 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. Our system can easily be extended to include drag-and-drop interactions. It is only through iterations with users that we are able uncover such user preferences.

Relations. All participants were able to create relations. One of the participants said, "It's pretty obvious what the

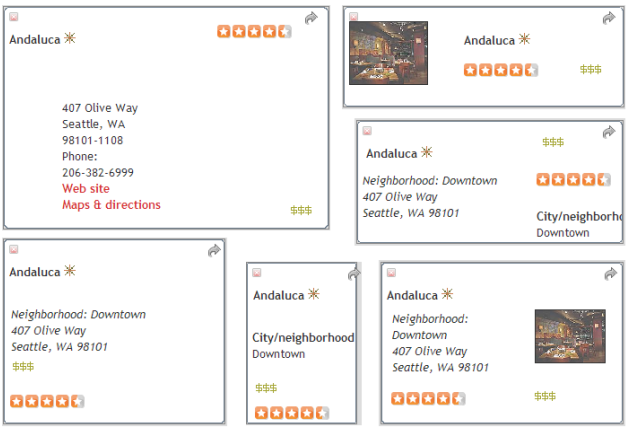

Figure 8: These cards were created by our study participants.

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 fairly simple, we had the participants collect content from only two websites, thus requiring only one relation instantiation. We 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.

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 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 8 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. [17].

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 we 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 collect more content. 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.

CONCLUSIONS AND FUTURE WORK

In this paper we present three new interaction techniques for helping users with collecting and organizing Web content. We apply these techniques to a variety of tasks including comparison shopping, event planning, and data collecting. Our results and user feedback lead us to the firm conclusion that these techniques are useful and welcomed by users. Our work combines content extraction and Web search to provide services and tools that are much needed and can help users with challenging information tasks.

Our 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. While we wait for the promise of the Semantic Web, we have built an interface that gives people enough value that they might effectively build components of the Semantic Web without any help from content providers. With our interface, users specify relations between websites to accomplish their own tasks. These relations are persistent and can be shared with others. We plan to create a public repository of such relations, and this repository can be the beginning of the web of relationships that the Semantic Web envisions. Such a web of relationships can enable a new shift in Web applications and bring about a World Wide Web that is both more personal and collaborative.

We asked our study participants whether they would use a public repository of relations, cards, and search templates, and they were very positive about using such a repository.

Most of them suspected that they would be consumers rather than producers of such community artifacts. As with any task, there are users who will try to accomplish their work as quickly as possible and others who will take the time to create exactly what they need. To this end, we plan to continue evolving the card designer to provide light-weight card authoring for the novice, while including more advanced features such as automated reflowing and layout.

Our search templates present a new interface to Web search. We plan to explore approaches for providing more feedback so that the user can understand search results and quickly and easily iterate through queries.

Finally, we hope to release the system and study users as they carry out their own tasks. We want to explore which websites people relate together, how often they create new cards, and how well they can use search templates.

REFERENCES

1. B. Amento, L. Terveen, and W. Hill. Experiments in social data mining: The topicshop system. *ACM Trans. on Computer-Human Interaction (TOCHI)*, pages 54–85, 2003.
2. V. Anupam, J. Freire, B. Kumar, and D. Lieuwen. Automating web navigation with the webvcr. In *Proc. of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications netowrking*, pages 503–517, Amsterdam, The Netherlands, The Netherlands, 2000. North-Holland Publishing Co.
3. M. Bauer, D. Dengler, and G. 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 Press.
4. M. Bolin, M. Webber, P. Rha, T. Wilson, and R. 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 Press.
5. S. Card, G. Roberston, and W. York. The webbook and the web forager: An information workspace for the world-wide web. In *Proc. of the SIGCHI conference on Human factors in computing systems*, 1996.
6. X. Dong, A. Halevy, and J. 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 Press.
7. M. Dontcheva, S. M. Drucker, G. Wade, D. Salesin, and M. F. Cohen. Summarizing personal web browsing sessions. In *UIST '06: Proc. of the 19th annual ACM symposium on User interface software and technology*, pages 115–124, New York, NY, USA, 2006. ACM Press.
8. S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff ive seen: A system for personal information retrieval and re-use. In *Proc. of SIGIR 2003*, 2003.
9. A. Faaborg and H. Lieberman. A goal-oriented web browser. In *CHI '06: Proc. of the SIGCHI conference on Human Factors in computing systems*, pages 751–760, New York, NY, USA, 2006. ACM Press.
10. J. Fujima, A. Lunzer, K. Hornbæk, and Y. Tanaka. Clip, connect, clone: combining application elements to build custom interfaces for information access. In *UIST '04: Proc. of the 17th annual ACM symposium on User interface software and technology*, pages 175–184, New York, NY, USA, 2004. ACM Press.
11. Groxis. Grokker, 2005. http://www.grokker.com/.
12. A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data integration: The teenage years. In *VLDB*, pages 9–16, 2006.
13. A. Hogue and D. Karger. Thresher: automating the unwrapping of semantic content from the world wide web. In *WWW '05: Proc. of the 14th international conference on World Wide Web*, pages 86–95, New York, NY, USA, 2005. ACM Press.
14. D. Huynh, S. Mazzocchi, and D. Karger. Piggy bank: Experience the semantic web inside your web browser. In *ISWC '05: Proc. of 4th international Semantic Web Conference*, 2005.
15. D. Huynh, R. Miller, and D. Karger. Exhibit: Lightweight structured data publishing. In *WWW '07: Proc. of the 16th international conference on World Wide Web (to appear)*, New York, NY, USA, 2007. ACM Press.
16. U. Irmak and T. Suel. Interactive wrapper generation with minimal user effort. In *WWW '06: Proc. of the 15th international conference on World Wide Web*, pages 553–563, New York, NY, USA, 2006. ACM Press.
17. C. Jacobs, W. Li, E. Schrier, D. Bargeron, and D. Salesin. Adaptive document layout. *Communications of the ACM*, 47:60 – 66, 2004.
18. H. Jung, J. Allen, N. Chambers, L. Galescu, M. Swift, and W. Taysom. One-shot procedure learning from instruction and observation. In *Proc. of the International FLAIRS Conference: Special Track on Natural Language and Knowledge Representation*, 2006.
19. S. Karsenty, C. Weikart, and J. A. Landay. Inferring graphical constraints with rockit. In *CHI '93: Proc. of the SIGCHI conference on Human factors in computing systems*, page 531, New York, NY, USA, 1993. ACM Press.
20. T. Kristjansson, A. Culotta, P. Viola, and A. McCallum. Interactive information extraction with constrained conditional random fields. In *AAAI '04: Proc. of the 19th international conference on artificial intelligence*, pages 412–418, 2004.
21. D. Kurlander and S. Feiner. Inferring constraints from multiple snapshots. *ACM Trans. Graph.*, 12(4):277–304, 1993.
22. J. Liu, X. Dong, and A. Y. Halevy. Answering structured queries on unstructured data. In *WebDB: Proc. of Ninth International Workshop on Web and Databases*, 2006.
23. A. Lunzer and K. Hornbæk. Recipesheet: creating, combining and controlling information processors. In *UIST '06: Proc. of the 19th annual ACM symposium on User interface software and technology*, pages 145–154, New York, NY, USA, 2006. ACM Press.
24. J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR*, pages 342–350, 2007.
25. R. Miller and B. Myers. Creating dynamic world wide web pages by demonstration, 1997.
26. G. Robertson, M. Czerwinski, K. Larson, D. Robbins, D. Thiel, and M. van Dantzich. Data mountain: using spatial memory for document management. In *Proc. of the 11th annual ACM symposium on User interface software and technology (UIST 1998)*, pages 153–162, 1998.
27. A. Safonov. Web macros by example: users managing the www of applications. In *CHI '99: CHI '99 extended abstracts on Human factors in computing systems*, pages 71–72, New York, NY, USA, 1999. ACM Press.
28. m. 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 *WWW '02: Proc. of the 11th international conference on World Wide Web*, pages 172–181, New York, NY, USA, 2002. ACM Press.
29. A. Sugiura and Y. Koseki. Internet scrapbook: automating web browsing tasks by demonstration. In *UIST '98: Proc. of the 11th annual ACM symposium on User interface software and technology*, pages 9–18, New York, NY, USA, 1998. ACM Press.
30. I. E. Sutherland. Sketch pad a man-machine graphical communication system. In *Proc. of the AFIPS Conference*, volume 23, pages 323–328, New York, NY, USA, 1963. ACM Press.
31. P. Viola and M. Narasimhan. Learning to extract information from semi-structured text using a discriminative context free grammar. In *SIGIR '05: Proc. of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 330–337, New York, NY, USA, 2005. ACM Press.
32. Vivisimo. Clusty, 2004. http://www.clusty.com/.
33. J. Wong and J. Hong. Making mashups with marmite: Towards end-user programming for the web. In *"Proc. of the SIGCHI conference on Human factors in computing system (to appear)"*, New York, NY, USA, 2007. ACM Press.
34. Yahoo! Inc. http://pipes.yahoo.com/.