Supporting End-Users in the Creation of Dependable Web Clips

Sandeep Lingam
University of Nebraska-Lincoln
Lincoln, USA
slingam@cse.unl.edu

Sebastian Elbaum
University of Nebraska-Lincoln
Lincoln, USA
elbaum@cse.unl.edu

ABSTRACT

Web authoring environments enable end-users to create applications that integrate information from other web sources. Users can create web sites that include built-in components to dynamically incorporate, for example, weather information, stock-quotes, or the latest news from different web sources. Recent surveys conducted among end-users have indicated an increasing interest in creating such applications. Unfortunately, web authoring environments do not provide support beyond a limited set of built-in components. This work addresses this limitation by providing end-user support for "clipping" information from a target web site to incorporate it into the end-user site. The support consists of a mechanism to identify the target clipping with multiple markers to increase robustness, and a dynamic assessment of the retrieved information to quantify its reliability. The clipping approach has been integrated as a feature into a popular web authoring tool on which we present the results of two preliminary studies.

Categories and Subject Descriptors

D.2.6 [**Software Engineering**]: Interactive Environments; D.2.5 [**Software Engineering**]: Testing and Debugging; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Experimentation, Reliability, Human Factors

Keywords

Web authoring tools, Dependability, End-users

1. INTRODUCTION

Web authoring environments have enabled end-users who are non-programmers to design and quickly construct web pages. Today, end-users can create web pages of increasing sophistication with simple drag-drop type operations, while the authoring environment keeps the underlying code infrastructure hidden. Consider the example of Tom, the owner and sole employee of a travel agency, who arranges airline tickets, car rentals, hotel reservations and package tours for his local customers. Tom has designed his own site, and since he cannot afford to pay travel consolidators and specialized programmers, his daily routine consists of monitoring various airlines' and travel web sites for low-price alerts, checking the availability and prices of rental cars and hotel rooms for his customers, and maintaining updated information about package tours from various operators in his web site. In order to obtain that information, Tom has to go through the tedious process of navigating to the appropriate web page, filling in all the relevant details, extracting sections of desired information and integrating them into his own site.

Several aspects of the process just described could be automated, freeing the end-user to perform other activities while reducing the chances of introducing errors in this repetitive task. At a high level, we envision Tom specifying what data to obtain from what web sites, and how those pieces of data should be included in his web site, and then an automated process would create Tom's site and the necessary components so that the target site's content is automatically retrieved and integrated. Tom's situation is a common one, experienced by several end-users who wish to repeatedly retrieve, consolidate, and tailor content such as search results, weather and financial data from other web sources. In a recent survey conducted by Rode and Rosson [21] to determine the types of web applications non-professional programmers would be interested in, 34% of the subjects surveyed expressed interest in applications that required facilities for custom collections of data generated by other web applications.

Still, the process of finding, extracting and integrating data from multiple web sites remains laborious. Several commercial web-authoring environments such as Microsoft Frontpage [7] and Dreamweaver [6] enable end-users to create increasingly sophisticated web-applications capable of integrating information from various web-sources in the form of built-in components. These web-authoring tools, however, do not support integration of information beyond a limited set of components. End-users attempting to retrieve information from sites not covered by these built-in components receive no further support from web authoring tools. Furthermore, deploying such functionality requires a level of technical expertise above an end-user like Tom. Our work addresses this limitation by assisting end-users in the creation of "Web clips".

Web clips are components within the end-user web site which dynamically extract information from other web-sources. For instance, in the example described earlier, Tom's site would include several web clips, each extracting and incorporating information from a target web site as shown in Figure 1.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

Figure 1: Tom's travel site

There are several challenges that must be addressed to provide adequate support for end-users like Tom. First, although we can assume that the end-user is familiar with web authoring tools, we cannot expect end-users to have any programming experience or desire to learn programming. This implies that the support mechanisms for the creation of web clips must be transparently integrated within web-authoring tools, while the underlying support infrastructure and code necessary to deploy the web clips must be automatically generated and hidden from the end-user.

Second, it is reasonable to expect that the target web sources of information will change in ways that go beyond content. For example, a web source may present data in a new format, or in a different sequence, or not include it at all. A deployed web clip must be able to at least detect and alert the end-user about the degree to which the changes in the web source may impact their web site. Following with Tom's scenario, if his web clip is not robust enough to detect changes in the structure of the source web site, it might end up retrieving "cruise deals" instead of "flight deals", thereby resulting in inconvenience, delay or even a transaction loss because of incorrect information. At the same time, we would like for web clips to be resilient to, for example, cosmetic changes, to reduce the number of times a web clip must be created. Hence, it is desirable to develop clips that not only accurately identify what to clip but are also robust in the presence of changes.

To address these challenges, we have designed and prototyped an approach to support an end-user in creating a dependable web clip. The approach is unique in three fundamental aspects: 1) it maintains the web authoring tool interface metaphor so that the end user keeps operating as usual through the standard directives while imposing no additional programming requirements, 2) it increases the robustness of the web clip through a training procedure in which the end user is iteratively queried to infer multiple markers that can help to identify the correct data in the presence of change, and 3) it deploys multiple filters to increase the confidence in the correctness of the retrieved information, and assessment code to provide a measure of correctness associated with the retrieved and integrated data.

2. RELATED WORK

There have been numerous research efforts addressing automated extraction and customization of content from web sites that we now proceed to summarize.

Multiple toolkits and specialized languages enabling content extraction through manual and automated wrapper generation have been discussed by Kuhlins et al.[22] and Laender et al.[23]. On the user end, stand alone languages such as Jedi [18] and W4F (World Wide Web Wrapper Factory) [31] provide sophisticated modules for generating wrappers that enable web content extraction. Furthermore, GUI-based toolkits such as Lapis [26] and Sphinx [25] enable end-users who lack any programming skills to create wrappers by providing enhanced user-interaction. A different type of extraction mechanism is provided by web site companies in the form of web-feeds and web services. Web-based feeds such as RSS [8] provide web content or summaries of web content together with links to the full versions. Web-feeds originated with news and blog sites, but more web sites are now making content available through them. Information is usually delivered as XML content, organized in a pre-defined format, which could be collected by feed aggregators. Web services [9] on the other hand enable creation of Mashups [20], which combine information from multiple web sites into a single one using public interfaces and APIs provided by web site companies.

There are several instances of techniques and tools that operate as web-browser extensions and provide end users with basically web macro recorders of different sophistication. Smart bookmarks are shortcuts to web content that would require multiple steps of navigation to be reached. Systems supporting smart bookmarks such as WebVCR [11] are capable of recording the steps taken during navigation, which could later be replayed automatically to reach the target web page. Furthermore, some such as Webviews [16] also enable some level of content customization to, for example, fit smaller screens. Tools such as Greasemonkey [2], Chickenfoot [14] and Robofox [33] enable end-users to, for example, customize the look and content of a web page, associate actions with triggers in observed web sites, fill and submit forms, and filter content. Essentially, these tools provide programming language capabilities with different levels of abstraction, each for a slightly distinct range of users, to manipulate and utilize web page content as perceived by the user through the browser.

Stand-alone research systems such as Hunter Gatherer [32], Internet Scrapbook [34] and Lixto [13] enable end-users to create "within-web page collections" which can contain text, images, links and other active components like forms. These tools enable users to make selections of the desired content within the visited web page as it is visited. A slightly different approach is taken by web-integration frameworks such as InfoBeans [12], CCC [17], and Robosuite [4] which enable users to create several configurable components, that encapsulate information or specify an information gathering and providing process. Several such components could be integrated into a common interface to create an application where information obtained from one component can be used to drive other components. One particular feature that is worth noting in these systems is example-based clipping guided by a training process where the user demonstrates the selection while the system generates a unique characterization of the selected content. This approach partly inspired

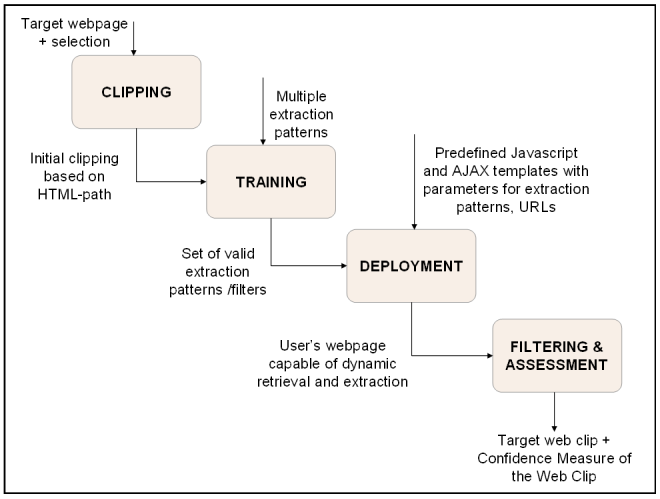

Figure 2: Outline of web clip creation process

our training component. Characteristics such as document structure, XPATH and page fragmentation have been used by systems discussed in [27, 15] to mark the clipped content. Alternate approaches use visual patterns within web pages [19] and tree-edit distances [30] (based on the tree structure of a web page) to identify the selected content. The component of our approach that aims to identify and mark the entities to be clipped combines the merits of the above approaches.

It is worth noticing that although sufficient correctness and robustness have been cited as critical aspects of applications that integrate information from other web sources [29], the approaches just described have not directly addressed these issues. Furthermore, most of the approaches discussed above are either stand alone applications, have been integrated into web browsers, or are constrained by the availability of a specific API or content. Our work addresses that particular niche, the dependability aspects associated with clips developed through web authoring environments that are intended to be published by end users.

3. OUR APPROACH: WEB CLIPPER

The primary goal for *web clipper* is to enable an end-user to create a dependable web clip without imposing additional requirements on or expecting higher levels of expertise from the end user. The following objectives were defined in order to achieve the above goal:

1. To develop and integrate an intuitive and user-friendly selection process into the web-authoring environment that would enable an end-user to create a clipping from any given web site with ease.
2. To enable training of the web clip to better identify the clipped content within the structure of the target web site.
3. To provide an extent to the validity of the extracted information, once the web clip(s) is deployed.
4. To alert the user about structural changes in the "clipped" web site that may affect the reliability of information collected.

Figure 2 gives an overview of the various steps involved in web clip creation - clipping, training, deployment, filtering and assessment. Clipping enables the end-user to make a selection within the target web page and is followed by a training session which generates several valid extraction patterns capable of identifying the clipped content. The deployment stage results in the creation of several scripts which dynamically retrieve and filter content to create the target web clip. It is followed by an assessment of the validity of information within the generated web clip.

3.1 Clipping

The clipping process enables the end-user to select data from a target web page. Consider the example of Tom discussed earlier. One of the web clips that Tom could be interested in, is the "sample airfares" section on *travelocity.com*. There are two basic requirements for the creation of this web clip which are described in the following sections.

3.1.1 Target Clip Selection

Since we have prototyped our approach in Microsoft Frontpage, we will utilize this tool's context to explain our approach. Clipping is available in the tool in the same manner as several other HTML controls, so the incorporation of the target web clip into the user's web page is provided through a simple drag-and-drop operation or by a menu-selection. The *web clipper* control has a custom browser associated with it, which enables the user to navigate to the desired web page and clip content from it as shown in Figure 3.

The selection operation is supported by visual feedback in the form of a border around the corresponding HTML element over which the user is hovering. As the user moves the mouse, every extractable document element is highlighted and the user can make a selection by clicking on it. Options to refine a selection such as zoom in and zoom out, are available through a context menu. These operations are based on the DOM [1] structure of the HTML document where, for example, a zoom out operation selects the first visible HTML element which encloses the selected element. Move up and move down are also available to enable a more detailed DOM traversal.

3.1.2 Extraction Pattern

Once a selection is made, an extraction pattern is generated. An extraction pattern constitutes a unique characteristic of the user's selection that can later be used to dynamically extract it from the target web page. During the clipping process, the user's selection is uniquely identified by its HTML-Path. HTML-Path is a specialized XPATH [10] expression which uniquely identifies any HTML-document portion corresponding to a DOM tree node. For example, the HTML-Path corresponding to the "sample airfares" section on *travelocity.com* shown in Figure 3 enables the identification and extraction of that section within the web page. The URL of the clipped web page and the HTML-Path of the clipped section are extracted during the clipping process and will be later embedded (among other components) in the user's web page for dynamic extraction of content after deployment.

3.2 Training

Although an extraction pattern based on HTML-Path is capable of uniquely identifying a web page element, it is not

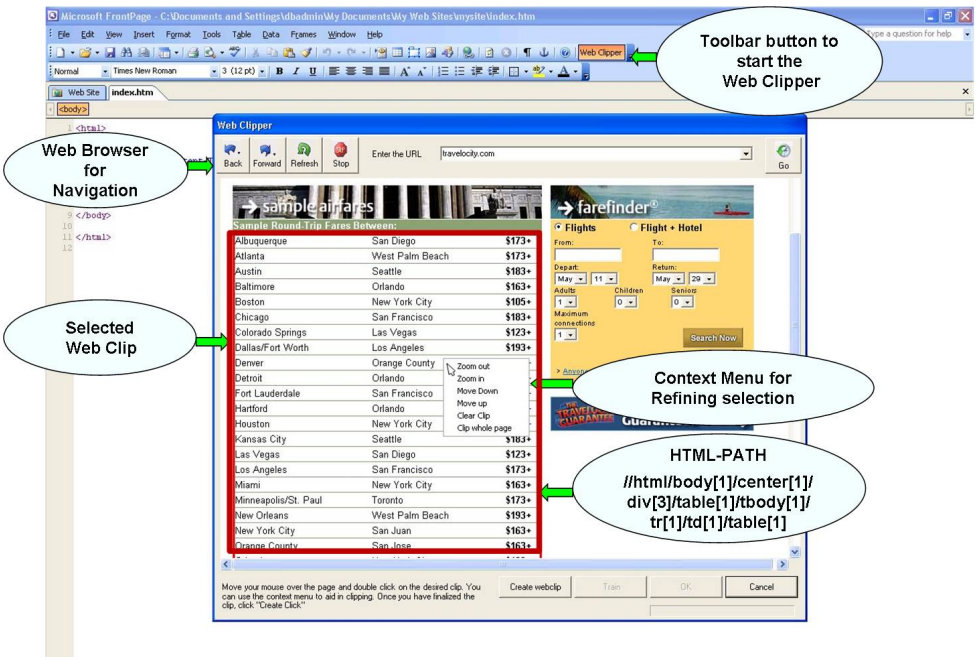

Figure 3: Clipping process

quite robust. Even a small change in the structure of the web page might affect the content retrieved by a web clip based on just the HTML-Path. For instance, in the example described in the previous section, if a <TABLE> element is introduced before the "sample airfares" section, the clipping based on HTML-Path will extract the newly inserted <TABLE> element instead of the selected one, thereby returning incorrect results.

The aim of the training session is to increase the robustness of the web clip by constructing several extraction patterns which uniquely characterize the end-user selection, such that if one of them fails, one of the others might be successful in identifying the selection. Once the clipping process is complete, the user can proceed by clicking "Train" to start a training session (see Figure 4). During the training session, several clips, created using different extraction patterns are shown to the user, one after the other. The user either approves or disapproves the clips based on his or her expectations, thereby generating several unique characterizations for the desired clip. Every time the user marks a clipping as valid, the system generates a filter corresponding to the clipping. Filters are basic snippets of javascript code that will be embedded within the user's web page, and which are capable of identifying the clipped content within the target web page using an extraction pattern.

For instance, continuing with the example discussed earlier (Figure 3), let us assume that the "sample airfares" section is always preceded by the element <DIV> which contains a label for the section, "Sample Round-trip fares between:". Thus, one of the extraction patterns would be a <TABLE> element that immediately follows a <DIV> with the text "Sample Round-trip fares between:". During the training session, the above extraction pattern is applied to the web page and the first table matching the pattern is returned as the web clip. This is presented to the user by

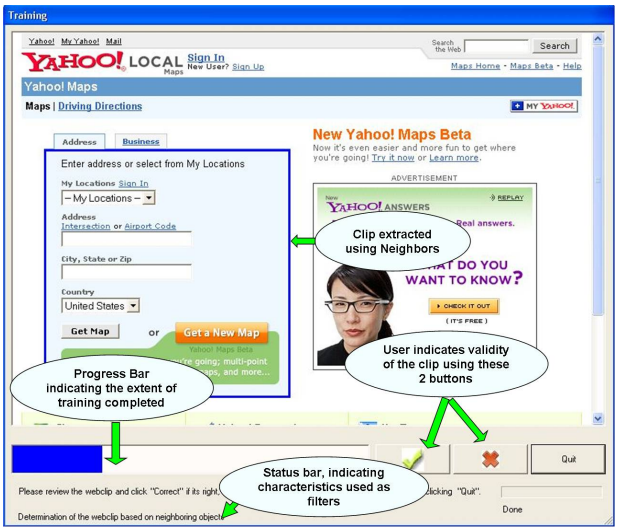

Figure 4: Training session

highlighting the clipped element with a blue border. The user reviews the response and indicates its correctness by clicking on the appropriate button as shown in Figure 4. In case of a correct response, the extraction pattern becomes a unique characterization of the web clip, which could be used in addition to HTML-Path to extract the desired content. The resilience of the web clip to changes within the target web page is dependent upon the existence of multiple valid extraction patterns, hence the user is encouraged to train the system better in order to obtain better results.

Table 1 summarizes the characterization heuristics we used as bases for extraction patterns in our prototype. For each

Pattern	Rationale	Usage and Internal Representation	Example
HTML-Path	HTML-Path provides direct access to the clipped section within the target web page	HTML-Path evaluated based on position of clipped element relative to the document root. Internally represented as an XPATH expression	Sample airfares section on *travelocity.com* was extracted using its path as shown in Figure 3
HTML Element Id	'ID' attribute is likely to be unique and is often used by web pages for updating content or as a stylesheet indicator	'ID' attribute for clipped element captured and stored. Clipped element directly referenced using 'ID' during dynamic extraction	Articles on *slashdot.org* are usually enclosed in a with 'ID' as "articles" and were clipped using this characteristic
Class name and other Attributes	A combination of properties of the clipped element such as class name, style, action, alt, could enable its identification	Absolute indexes of elements in HTML-Path are replaced by name-value pairs of attributes to form *HTML-Path by expressions*. These are internally represented as specialized XPATH expressions	The "top stories" section on *times.com* was identified by the *HTML-Path by expression* /html/body/div[2]/div[2]/div[1] /table/tbody/tr/td[2]/div[1] /div[1]/div[@class='aColumn']
Enclosing Element	Certain sections within web pages are easily identified by their container sections	Relative XPATH expressions enable identification of clipped sections [27]. Surrounding/Enclosing elements which can be referenced directly using 'IDs', 'class names' are used to form relative XPATH expressions	The form to obtain maps on *maps.yahoo.com* was extracted using the enclosing it which has its ID attribute set to "ymap-form"
Neighboring element	Visual patterns such as adjacent or nearby sections, side panels, images, advertisements can enable identification of clipped section [19]	Characteristics of nearby elements such as content type, attributes, distance from clipped section enable its identification. Information pertaining to above (tag name, attributes, distance, content-type) is captured and internally represented as pattern-value pairs	The "stock indexes" section on *nyse.com* is usually preceded by options to e-mail or print the page, and followed by an advertisement with an image.
Labels	Headings and labels of sections describe the nature of content within the clipping and could be used as identifiers for clipped content	Sections containing text within emphasis tags such as $h1 - h6, b, span, i, em$ and shorter than a specific length are recognized as labels. Labels can be contained within the clipped section or within nearby elements and are represented by a combination of regular expressions and pattern-value pairs	A label "Right Now For" usually precedes the weather information for a particular zip code on *weather.com* and enabled its identification
Links	Hyperlinks within/near the clipped content could be a characteristic feature of a clipping	Hyperlinks within/nearby the clipped sections are captured and information pertaining to them such as link text, images attributes, referenced URL and distance is stored to enable dynamic identification of the clipped section	The links - 'Web','News', and 'Images' enabled characterization of *Google's* search form
Content Structure	Structure of content may enable identification of a particular section within an entire web page	A few patterns pertaining to the structure of content in web pages such as number of rows/columns in a table, items in a list, number of siblings, children, have been identified and are being currently used by the prototype for clipping.	The "sample airfares" section on *travelocity.com* was clipped using the characteristic that it was the only table within the web page with exactly 3 columns. (see Figure 3)

Table 1: A list of characteristics used as extraction patterns

characterization type we provide its rationale, usage, internal representation and an example where it can be identified within the sites we analyzed in Section 4. It must be noted here that this set of extraction patterns is easily extendable [24].

3.3 Deployment

The clipping and training processes are followed by the deployment of the user's web page with the clipped content. The URL and extraction patterns captured from the previous stages are used to generate an AJAX [28] script that is linked to the user's web page. This script uses the XMLHttp object to retrieve content dynamically from the target web pages and incorporate it into the end-user's web page. Since retrieved HTML documents may be inconsistent with HTML specification, their content is converted to XHTML to enable construction of a DOM tree and application of DOM and DHTML methods. Any relative URLs are converted to absolute URLs to ensure that links are not broken. Therefore, clicking of a hyperlink or submitting a form would take the user to the target web page. By default, the web clip's style is maintained by importing the stylesheets from the target site, but it can be tailored to match the user's style by pointing to a different stylesheet set. Any javascript associated with the target web clip is either imported or referenced to ensure that features such as validation checks, cookies are not lost due to creation of the web clip.

In addition to the AJAX scripts to support the data retrieval, Javascript filters are generated from pre-defined templates for each of the extraction patterns gathered during training. Validation scripts to enable filter assessment as explained in Section 3.4 are also embedded within the user's web page. Filters and validation scripts enable the extraction and assessment of the clipped content which is placed within an *iframe* element within the user's web page. Just

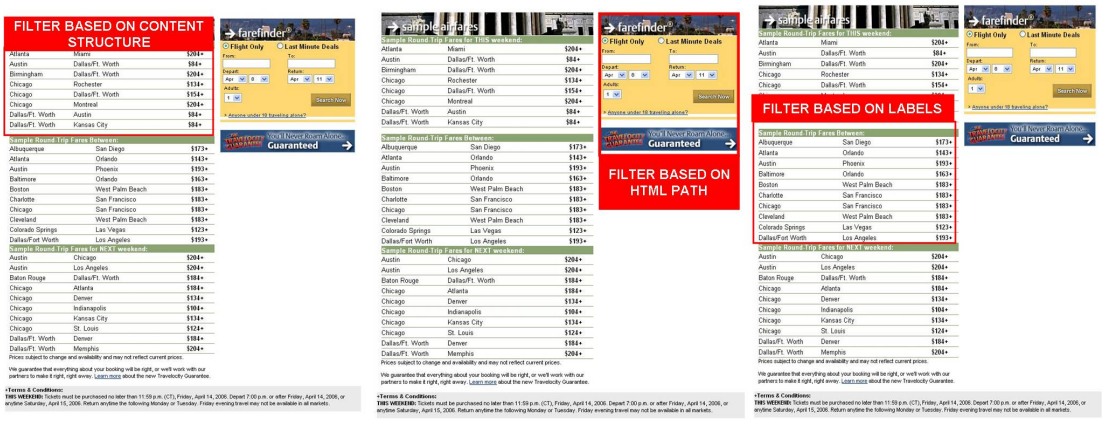

Figure 6: Clips generated by filters using HTML-Path, Content Structure, and Labels

like any other HTML control, the user can move, resize or even annotate the web clip to suit his/her preference. This entire process is automatically triggered upon the completion of the training session and runs in the background without the user having to worry about the mechanics of the process. Web clips thus generated can be currently viewed in Internet Explorer (6.0 and above) and Mozilla Firefox (1.5 and above).

3.4 Filtering and Assessment

After deployment, when the user designed or updated web page is loaded in a browser, a filter score is computed for each filter generated by a valid extraction pattern during training, by analyzing its performance in the presence of the other available filters. If a filter retrieved valid data in the presence of another filter, then its filter score is increased, otherwise it remains the same. At run-time, the filter with the maximum filter score is assumed to be the most trustworthy so it is used to generate the target web clip (any ties in the filter scores are broken by random selection).

For example, Figure 5 gives the web clip that Tom had initially created during the clipping process. Figure 6 gives the web clips generated a few days later by 3 different filters. Table 2 gives a summary of the extraction patterns observed

Figure 5: Tom's desired web clip

Filters / Extraction patterns	Content Structure	HTML-Path	Label	Filter score
Content structure filter	1	0	0	1
HTML-Path filter	0	1	0	1
Label filter	1	0	1	2

Table 2: Filter scores and valid extraction patterns

during training and after deployment, and the scores of the corresponding filters. The filters for clips shown in Figure 6 were generated based on content structure, HTML-Path and labels respectively. In Figure 6a we see that though the content structure is the same as in the pattern marked valid during the training process, the label and the HTML-Path are not the same. In this example we note that the filter using labels (third row) had the highest filter score as most number of patterns marked valid during training were observed even after deployment.

To provide the user with an extent to the validity of the content retrieved, we define *confidence* as the ratio of the maximum filter score to the total number of valid extraction patterns generated during the training session. For example, if six extraction patterns were marked valid during the training process, but only five extraction patterns are valid after deployment for the web clip generated by a particular filter, then the confidence score would be .83. Confidence gives an extent of the similarity between the patterns marked valid during the training process and the patterns observed after deployment. A greater similarity in patterns results in a higher confidence in the content within the dynamically generated web clip.

The current prototype provides this confidence information either by displaying it through the automated annotation of the web site with a *Validity Assessment Bar* above each web clip (as illustrated in Figure 7), or by reporting it to a log file. The prototype is also able to alert the user about changes in the patterns that were observed during the training process, as this might enable the user to make a further assessment of the dependability of the clip. The user can also configure the web clips to provide alerts when the confidence scores fall below a particular threshold.

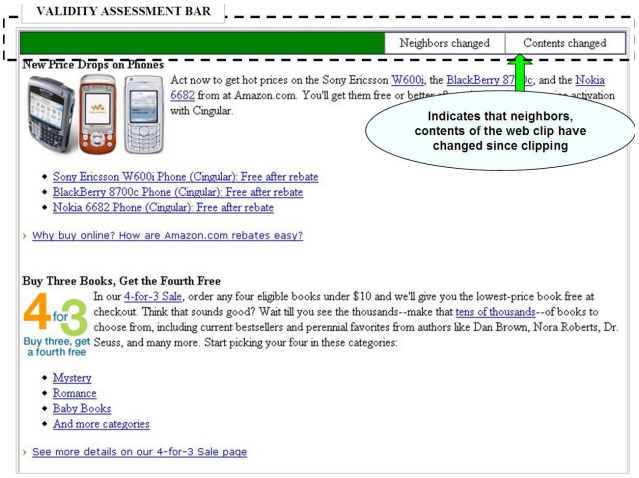

Figure 7: Sample web clip with the Validity Assessment Bar at the top

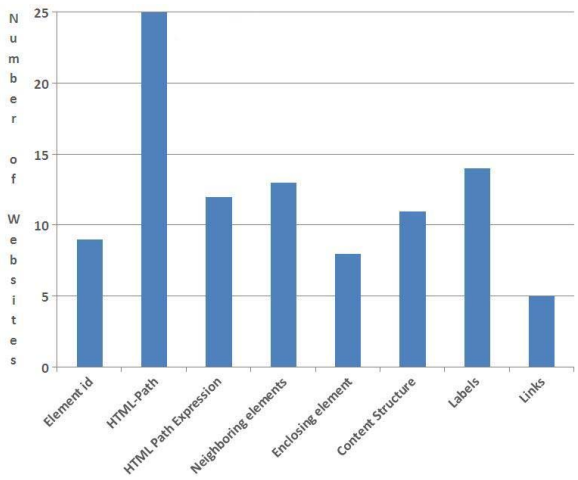

Figure 8: Applicability of extraction patterns for clipping

4. EVALUATION

Our work aim has been to assist an end-user in creating dependable and robust web clips. To evaluate the effectiveness of *web clipper* in achieving this objective, the first author of the paper generated and tested web clips from 25 different web sites to assess:

- Effectiveness of the extraction patterns used in generating web clips
- Dependability of web clips in providing sufficiently correct information over time, and
- Robustness of web clips to changes in the clipped web site

4.1 List of Web sites

Web clips were generated from 25 popular web sites for evaluation purposes. The web sites used for evaluation were chosen for several reasons. First, they had content which gets updated periodically. Second, as the primary focus of our work has been on assisting end-users, we chose web sites which an end-user like Tom might access. Finally, some of these web sites are among the top commercial web sites having high hit counts [5, 3]. Web clips created from these sites include text, links, images, forms and dynamically created content.

4.2 Effectiveness of Extraction Patterns

There are several extraction patterns which could be potentially used to characterize users' clippings such as the ones described in Section 3.2. Clippings were generated from the 25 web sites to assess the effectiveness of these extraction patterns. The chart in Figure 8 gives an indication of the applicability of the various extraction patterns to the 25 web sites. Results indicate that many of the web clips could be 'clipped' using more than one extraction pattern. In all the observed sites, HTML-Path was a valid extraction pattern as every page element has a unique path within a HTML document. Extraction patterns based on *HTML-Path by expressions* were found to be very effective in characterizing clippings as several of the observed sites such as *slashdot.org*, *travelocity.com*, and *espn.com* used "classes", "ids" for logical separation of sections. Labels and content structure were also very effective as extraction patterns, characterizing clippings in more than 50% of the sites. *usaweather.com*, *southwest.com*, *nasdaq.com*, and *yahoo* had a well-defined central structure within the web page for clipped content annotated by explanatory labels. Neighboring and enclosing elements characterized nearly 50% of the clippings in sites such as *careerbuilder.com*, *nyse.com* ands *bettycrocker.com* where navigation panels, side bars and bounding elements served as extraction patterns. Links were not widely applicable as extraction patterns as they were either not present or were not unique to the clipped element. Only a couple of sites such as *astrology.yahoo.com* and *BBC sports* had links which could characterize the clipping.

Overall, *usaweather.com* and *travelocity.com* generated the highest number of valid extraction patterns (six) during the training session. The average number of extraction patterns valid for the observed sites was close to 4. However, the existence of multiple extraction patterns for every observed web site enabled us to generate multiple filters for clipping content, thereby increasing the probability of successful retrieval of desired information in the presence of changes in the target site.

4.3 Dependability of web clips

To assess the dependability of web clips in providing sufficiently correct information over time, the web clips generated were monitored for a period of 16 weeks. Each web clip was periodically refreshed and the confidence score was recorded. Table 3 gives the biweekly confidence scores for the clipped web sites that had undergone changes during the period. Most of the web clips were found to be quite stable, retrieving the desired information, despite changes to the target web site. The average confidence score was nearly 85%.

Certain extraction patterns enabled the identification of the clipping during the training process but were not characteristic features of the clipped section. For example labels containing dates on which the web pages were viewed, titles of news, recipe sections, and number of items within

Web sites/Week	1	3	5	7	9	11	13	15	Type of Change
www.careerbuilder.com	100	100	100	75	100	75	75	75	Html-Path
hotjobs.yahoo.com	100	67	67	67	67	67	67	67	Label
www.cricinfo.com	67	67	67	0	0	0	0	0	Re-designed
www.amazon.com	100	100	67	67	67	67	67	67	Content structure
www.jcpenney.com	100	100	100	80	80	80	80	80	Links
www.epicurious.com	100	75	75	75	75	75	75	75	Html-Path
www.bettycrocker.com	100	100	83	83	83	83	83	83	Neighbors
astrology.yahoo.com	100	75	50	50	50	50	50	50	Html Path + Links
horoscopes.astrology.com	75	75	75	75	50	50	50	50	HTML Path + Neighbors
weather.yahoo.com	100	100	100	100	100	100	100	0	Re-designed
usa.weather.com	100	100	100	100	100	100	83	83	Neighbors
www.techdirt.com	100	75	75	75	75	75	75	75	Neighbors
www.nyse.com	100	100	100	67	67	67	67	67	Html Path
www.health-fitness-tips.com/	100	75	75	75	75	75	75	75	Enclosing Element

Table 3: Biweekly confidence scores

lists were among the extraction patterns observed for sites such as *yahoo jobs*, *epicurious.com* and *amazon.com*. These were eliminated within a short time after deployment after which the confidence scores stabilized for these sites. Certain design changes such as introduction of additional sections, navigation aids and sidebars affected the HTML path, neighbors and enclosing elements of *careerbuilder.com*, *astrology.com* and *bettycrocker*. New advertisements also affected sites such as *espn.com* and *usaweather.com*. Two web sites, *yahoo weather* and *cricinfo.com*, had a major restructure, resulting in the web clips not being able to retrieve the desired information and reporting a zero confidence value. Overall, the web clips were able to retrieve the originally clipped content in most cases or were appropriately annotated with confidence scores enabling an end-user to determine whether the information was sufficiently correct to use.

4.4 Robustness

The previous study observed real web sites over a period of time, but did not take into consideration the degree to which they change. We are now performing a more controlled study where we manipulated the sites through a series of changes to measure the robustness of the generated clips in the presence of such changes.

For evaluation purposes, we define a 'block' as a HTML element which is similar in structure to the clipped element, having the same tag name and same type of content (text, images, links). For example, Figure 9 shows a new block, "Hot deals" section inserted above the initially created web clip of the "Sample round-trip fares between" section. Blocks were created either from existing elements or by modifying elements within the target web page. The following structural modifications were made to the web pages to assess the robustness of web clips:

- Block Insertion. A new block similar in structure to the clipped element, was inserted into the web page immediately above the clipped element as shown in Figure 9 where the "Hot Deals" section is inserted into the web page from which the "Sample Round-trip fares between" section was initially clipped. This would test the ability of web clips to extract desired information when content is inserted into the web page.
- Block Movement. Reorganization of content within a web page might result in moving of sections of information within a web page. Therefore, we tested the robustness of web clips to block movements within the clipped web page. Block movement involved moving one of the existing blocks which was similar in structure to, and above the clipped element to immediately below the clipped element. In cases where the block below the clipped element was similar to it, the block below the clipped element was moved immediately above it. If there was no block similar to the clipped element, an element with the same tag name and same type of content was considered as a block and moved either immediately above or below the clipped element, based on its initial position.
- Block Deletion. Web clips were also tested for robustness against block deletions. To test for block deletions, the nearest block similar in structure, and immediately above the clipped element was deleted. In the absence of such a block, an element above the clipped section with the same tag name and similar content was deleted.
- Enclosing Element Changes. We modified the element enclosing the clipped element to determine the effect it would have on the content returned by the web clip. This helped us evaluate the robustness of web clips to changes in HTML-Path and other characteristics such as enclosing element, element Id. The enclosing element was modified by changing the tag name of the enclosing element, and by removing any of the attributes that it had, which could be used for its identification.

Figure 9: Block-A section similar to clipped section

Web sites/Structural Change	Block Insertion	Block Movement	Block Deletion	Enclosing Element Change	Target Clipping Removed
careerbuilder	50	100	100	50	25
hotjobs.yahoo	33	100	100	67	0
cricinfo	67	67	67	67	33
sports.espn.go	25	50	50	50	25
bbc.co.uk/sport	67	67	67	67	0
walmart	67	67	67	67	0
amazon	67	67	67	67	67
bestbuy	67	100	100	67	0
jcpenney	60	80	100	80	20
epicurious	75	100	75	50	25
bettycrocker	67	67	67	50	17
recipecenter	75	75	75	50	75
astrology.yahoo	75	100	75	75	25
horoscopes.astrology	50	50	25	75	0
travelocity	50	50	67	33	50
southwest	75	75	75	75	25
rentalcars	67	100	100	67	0
weather.yahoo	50	100	100	50	0
usa.weather	100	100	100	67	0
slashdot.org	80	80	80	40	40
techdirt	75	50	25	50	0
nyse	67	67	67	100	0
nasdaq	50	50	50	50	25
health-fitness-tips	75	75	75	50	0
101lifestyle	67	67	67	67	33

Table 4: Confidence scores observed during controlled study

- Target Clipping Removed. Due to design changes within web sites, it is quite probable that the originally clipped content may be completely removed from the source web page. To test the ability of the generated web clip to alert the user during such changes, we removed the original clipping from the source web page and evaluated the confidence scores.

The user's web pages with the web clips were reloaded (refreshed) after making the above modifications to determine whether the changes affected the content retrieved by the web clip. Confidence scores of the web clips are indicated in Table 4. All the web clips were able to retrieve the desired content despite the structural changes made to the target web pages in the first 4 scenarios. The web clips were found to be robust to the modifications, indicating the importance of the training session in generating multiple ways of retrieving the desired content. The presence of multiple extraction patterns enabled the web clips to recover from the failure of one or more extraction patterns and retrieve the desired information using the remaining ones. However, confidence scores varied depending on the effect that the modifications had on the extraction patterns. The removal of target clipping in the fifth scenario resulted in a significant drop in the confidence scores for sites observed. A couple of sites though such as *amazon.com* and *recipecenter.com* had higher confidence scores due to the presence of multiple sections that could be erroneously identified using the same set of patterns. However, the average confidence score was less than 20% with seventeen web clips displaying alert messages within the validity assessment bar asking users to recreate their web clips.

5. CONCLUSION

In this paper, we have presented an approach to support end-users through the entire process of creating a dependable web clip. We have integrated *web clipper* into a popular web-authoring environment, thereby enabling end-users to incorporate web clips as components within their web pages. *Web clipper* addresses the shortcomings of existing tools by introducing the notion of training of web clips and of dynamic confidence evaluation for retrieved information. Training generates a set of unique characterizations which enable the extraction of desired content in multiple ways, thereby increasing the probability of obtaining desired responses despite structural changes in the target web site. Confidence evaluation provides an extent to the validity of the collected information within the web clip, enabling the user to assess sufficient correctness. A preliminary evaluation of the prototype consisting of the creation of web clips from web pages of twenty-five commercial web sites and modified copies of the clipped web pages has been encouraging. Web clips appear stable over time and robust in the presence of changes in the target sites.

This work points to several research directions that we would like to further explore. First, we would like to actually observe real end-users utilizing *web clipper* to quantify to what degree it helps them generate more dependable clips. Furthermore, we would like to broaden the range of end-users to include those that may be operating in an intranet site, perhaps using the clipper to extract and convey data originating in a legacy system. Second, many heuristics to improve the characterization of web elements are emerging and we have just scratched the surface on what can be used to improve the accuracy of our extraction process. Last, we would like to extend the transparency of our clip generation mechanism so that, for example, the end-user can go back to the "original" source page to re-touch it and maintain it without need for re-clipping.

Acknowledgments

This work was supported in part by NSF CAREER Award 0347518 and the EUSES Consortium through NSF-ITR 0324861.

6. REFERENCES

[1] Document Object Model. *http://www.w3.org/DOM*, Oct. 2006.

[2] Grease Monkey. *http://greasemonkey.mozdev.org*, Oct. 2006.

[3] Infoplease - Internet Statistics and Resources. *http://www.infoplease.com/ipa/A0779093.html*, Oct. 2006.

[4] Kapow Robosuite. *http://www.kapowtech.com/pdf/WebIntegrationWP.pdf*, Oct. 2006.

[5] KeyNote. Consumer top 40 sites. *http://www.keynote.com/solutions/performance indices/consumerindex/consumer40.html.*, Oct. 2006.

[6] Macromedia Dreameaver. *http://www.macromedia.com*, Oct. 2006.

[7] Microsoft Frontpage. *http://www.microsoft.com/office/frontpage*, Oct. 2006.

[8] RSS. *http://web.resource.org/rss/1.0/spec*, Oct. 2006.

[9] Web Services. *http://www.w3.org/2002/WS*, Oct. 2006.

[10] XPATH. *http://www.w3.org/TR/XPATH*, Oct. 2006.

[11] V. Anupam, J. Freire, B. Kumar, and D. Lieuwen. Automating Web Navigation with the WebVCR. In *International Journal of Computer and Telecommunications Networking*, 2000.

[12] M. Bauer, D. Dengler, and G. Paul. Instructible information agents for Web mining. In *International Conference on Intelligent User Interfaces*, pages 21–28, 2000.

[13] R. Baumgartner, S. Flesca, and G. Gottlob. Declarative Information Extraction, Web Crawling, and Recursive Wrapping with Lixto. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 21–41, 2001.

[14] M. Bolin, M. Webber, P. Rha, T. Wilson, and R. C. Miller. Automation and customization of rendered web pages. In *ACM Symposium on User interface Software and Technology*, pages 163–172, 2005.

[15] B. Christos, K. Vaggelis, and M. Ioannis. A web-page fragmentation technique for personalized browsing. In *ACM Symposium on Applied Computing*, pages 1146–1147, 2004.

[16] J. Freire, B. Kumar, and D. Lieuwen. WebViews: accessing personalized web content and services. In *International Conference on World Wide Web*, pages 576–586, 2001.

[17] J. Fujima, A. Lunzer, K. Hornbaek, and Y. Tanaka. Clip, connect, clone: combining application elements to build custom interfaces for information access. In *ACM Symposium on User Interface Software and Technology*, pages 175–184, 2004.

[18] G. Huck, P. Fankhauser, K. Aberer, and E. Neuhold. Jedi: Extracting and Synthesizing Information from the Web. In *International Conference of Cooperative Information Systems*, pages 32–42, 1998.

[19] U. Irmak and T. Suel. Interactive wrapper generation with minimal user effort. In *International Conference on World Wide Web*, pages 553–563, 2006.

[20] A. Jhingran. Enterprise information mashups: integrating information, simply. In *International Conference on Very Large Databases*, pages 3–4, 2006.

[21] J.Rode and M. Rosson. Programming at runtime: Requirements and paradigms for nonprogrammer web application development. In *IEEE Symposium on Human Centric Computing Languages and Environment*, pages 23–30, 2003.

[22] S. Kuhlins and R. Tredwell. Toolkits for Generating Wrappers . In *International Conference on Objects, Components, Architectures, Services, and Applications for a Networked World*, pages 184–198, 2002.

[23] A. Laender, B. Ribeiro-Neto, A. Silva, and J. Teixeira. A brief survey of web data extraction tools. In *SIGMOD Rec.*, volume 31, pages 84–93, 2002.

[24] S. Lingam. Supporting end-users in creation of web clips. Master's thesis, University of Nebraska, Lincoln, NE, May 2006.

[25] R. C. Miller and K. Bharat. SPHINX: A Framework for Creating Personal, Site-Specific Web Crawlers. In *International World Wide Web Conference*, pages 119–130, 1998.

[26] R. C. Miller and B. A. Myers. Integrating a Command Shell into a Web Browser. In *USENIX 2000 Annual Technical Conference*, pages 171–182, 2000.

[27] M.Kowalkiewicz, M.Orlowska, T.Kaczmarek, and W.Abramowicz. Towards more personalized Web: Extraction and integration of dynamic content from the Web. In *Asia Pacific Web Conference*, 2006.

[28] L. D. Paulson. Building rich web applications with ajax. *Computer*, 38(10):14–17, 2005.

[29] O. Raz and M. Shaw. An Approach to Preserving Sufficient Correctness in Open Resource Coalitions. In *International Workshop on Software Specification and Design*, page 159, 2000.

[30] D. C. Reis, P. B. Golgher, A. S. Silva, and A. F. Laender. Automatic web news extraction using tree edit distance. In *International Conference on World Wide Web*, pages 502–511, 2004.

[31] A. Sahuguet and F. Azavant. Building light-weight wrappers for legacy web data-sources using W4F. In *International Conference on Very Large Data Bases*, pages 738–741, 1999.

[32] 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 *International World Wide Web Conference*, pages 172–181, 2002.

[33] S.Elbaum, A.Koesnandar. Robofox. *http://esquared.unl.edu/wikka.php?wakka=AboutRobofox*, Oct. 2006.

[34] A. Sugiura and Y. Koseki. Internet scrapbook: automating Web browsing tasks by demonstration. In *ACM Symposium on User interface software and technology*, pages 9–18, 1998.