Adaptive Sites: Automatically Learning from User Access Patterns

Mike Perkowitz and Oren Etzioni
Department of Computer Science and Engineering, Box 352350
University of Washington, Seattle, WA 98195-2350
`{map, etzioni}@cs.washington.edu`
(206) 616-1845 Fax: (206) 543-2969

Abstract:

Designing a web site is a complex problem. Logs of user accesses to a site provide an opportunity to observe users interacting with that site and make improvements to the site's structure and presentation. We propose adaptive sites: web sites that improve themselves by learning from user access patterns. Adaptive webs can make popular pages more accessible, highlight interesting links, connect related pages, and cluster similar documents together. An adaptive web can perform these self-improvements autonomously or advise a site's webmaster, summarizing access information and making suggestions.

In this paper we define adaptive web sites, explain and formalize several kinds of improvements that an adaptive site can make, and give examples of applying these improvements to existing sites.

Introduction

Designing a web site is a complex and difficult problem (see, for example, [5]). As with any user interface, designers must structure and present their content in a way that is clear and intuitive to users, or those users will become lost and disgruntled. Good design is often facilitated by observing people using the software. However, because traditional software is sold to the customer and used in the privacy of a home or office, software designers have had to resort to testing small groups of users in special labs. On the World Wide Web, however, users interact directly with a server maintained by the inventors of the service or authors of the content. Popular web sites, therefore, facilitate large scale direct observation of real users. Any web site can maintain logs of user accesses, and a designer can use this information to improve the site. Raw data, however, is difficult to use; especially at a large and popular site, access logs may amount to megabytes a day - too much for an overworked webmaster to process regularly. Web server logs, therefore, are ripe targets for automated data mining.

We propose *adaptive sites*: web sites that use information about user access patterns to improve their organization and presentation. Adaptive sites observe user activity and user difficulties and learn about types of users, regular access patterns, and common problems with the site. Adaptive sites are useful for several reasons:

- What a visitor seeks in a site depends on who the user is. An adaptive site can recognize typical user types and customize the site presentation appropriately. Instead of customizing to a single user (as many sites do with cookies), an adaptive site aggregates the experiences of many visitors over time to generalize customize the site for different *types* of users.
- Visitors to a site do not always have the same conceptual model of the material as the site's designer. An adaptive site can recognize when user expectations differ from the site structure.
- Although a web site's structure is usually static, user needs change with time. An adaptive site can learn these patterns and decide what information to present when. (For approaches focused on dynamic customization based on user interests, see [4] and [1].)

For example, the University of Washington's computer science department maintains a web site for its introductory course CSE142. This site contains schedules, announcements, assignments, and other information important to the hundreds of students who take the course every quarter. Enough information is available that important documents can be hard to find or entirely lost in the clutter. Imagine, however, if the site were able to determine what was important and make that information easiest to find. Important pages would be available from the site's front page. Important links would appear at the top of the page or be highlighted. Timely information would be emphasized, and obsolete information would be quietly moved out of the way. These transformations could be performed by an automated "webmaster's assistant" or suggestions could be made to the webmaster, with data to justify those suggestions.

In this paper we present an approach to building adaptive sites. We show how to automatically generate improvements and suggestions from observations of server access logs and discuss the major issues in the design of such sites. All examples will be drawn from two web sites, which will be presented in section 2. In section 3 we discuss the kinds of observations that can be made about a site from server logs and other tools and what can be learned from this informtion. Section 4 presents four major transformations that can be performed on a web site solely on the basis of our described observations. Finally, we conclude with related work.

Two web sites

All examples in this paper will draw heavily from two web sites. The first is the web of the department of computer science at the University of Washington. This site provides information about various aspects of the department, including research projects, educational programs, and faculty, staff, and students. The second site is the course web for CSE142, an introductory course offered in the department. This web provides information for students in the course, including homework assignments, lecture notes, time schedules, and general announcements. These sites can be found at http://www.cs.washington.edu/ and http://www.cs.washington.edu/education/courses/142/96a/ respectively. Note, however, that both sites may change at any time. We have saved copies of both the UWCS and CSE142 front pages as of 12/2/96.

The UWCS front page is broken up into sections corresponding to the main organization of the site: general information, education, research, people and organizations, the region, and spotlight. Each section also contains a number of links that presumably correspond to the most important or popular starting points in each section. These links are ‘‘organized’’ in freeform text. The page also has a search form at the bottom. The pages for each section generally contain more freeform text with links as well as tables of relevant links. Some contain further subsections. Room for improvement is readily apaprent.**[1]**

The CSE142 front page is dominated by a list of links essentially ordered approximately by importance. Of particular interest is the homework page, linked to from the main page. The homework page contains a link for each assignment given out in the class. Each assignment has its own page (which becomes available when the assignment is given out) which further has links to all handouts and information required to do the assignment. After the assignment due date, a solution set is made available on this page as well.

Observation

An adaptive site has two basic components: an *observation module* and a *transformation module*. The observation module monitors user interactions with the site and accumulates important statistics about pages accessed, links traversed, paths followed, and problems encountered. The transformation module draws on this data to make changes to the structure of the site.

A variety of observations can be made from basic web server logs. An entry for a single access typically looks something like this:

```
128.95.170.57 - - [01/Oct/1996:09:48:17 -0700] "GET /education/courses/142/CurrentQtr/ HTTP/1.0" 200 4418
```

This entry contains, among other data, the IP address of the machine from which the access originated, the date and time of the access, and the URL requested. From such data, we can accumulate statistics on page access counts as well as observe time-dependent trends.

This basic information can be provided by any web server. By adding a service such as WebThreads [6], a server can record complete *paths*: the sequence of pages visited and links followed by a single user in a single visit. WebThreads does not require any changes to the original source HTML, but redirects accesses to the site through a program that can recognize individual users and keep track of their navigation through the site. Path data facilitates a number of observations. In addition to recording access counts for pages, we can also record counts for *links*; this enables the webmaster to ask what links on a page are important and should be emphasized. By examining where user paths begin, we can infer the site’s most popular starting pages: the places where people enter the site. Many visitors may not be entering at the site’s front page, perhaps because external links point into the middle of the site. Full paths also enable us to analyze precisely what people are doing and where they are going and to guess at what they are looking for, whether they found it, and whether they got lost in the process. Furthermore, knowing this information about individual visitors allows us to *cluster visitors by type*: we can observe regular access patterns that many users tend to follow and note stereotypical types of visitors. **[2]**

For example, by recording the paths of visitors to the CSE142 site over time, we can make a number of observations about, for instance, the homework pages.

1. The most recent homework assignment is one of the most accessed pages at the site. That is, at any particular time, the access count of the most recent homework made available will be fairly high.
2. Once the due date passes and the solution set is made available, it is the most popular item on the assignment page. That is, on any homework page, the link to the solution set is traversed more than any other.
3. The most recent solution set is among the more popular pages at the site. That is, the most recent solution set made available has a high access count.
4. Before exams, students will visit past solution sets to review them. That is, at certain times, paths will tend to visit

multiple solution sets.

WebThreads is focused on creating web sites that dynamically react to an *individual* user’s navigation, for example by highlighting links the user has not yet followed, customizing web pages for that user, or presenting advertisements she has not seen before. An adaptive site learns from the visits of *many* users to improve the structure of the site for future users. Having made observations, therefore, the adaptive site must next consider possible changes to make.

Transformation

There are several ways for an adaptive assistant to make use of its observations. One way to do this is to summarize the data in human-readable form and present it to the webmaster so she can intelligently improve her design. An advice system of this sort can draw the webmaster’s attention to certain patterns, suggest improvements, and issue regular traffic reports of the system’s heavily travelled routes. Another approach is to allow the site to transform itself in response to the observations it makes. A self-transforming system can explore the space of possible variations on the webmaster’s design by making a series of incremental improvements, each of which improves some aspect of the site.

A complementary approach would be to define a set of HTML extensions to tell the system where it can make changes. ‘‘Adaptive HTML’’ (or *A-HTML*) would add tags to specify lists of items that can be reordered, annotate items to be time-dependent, and so on. Using A-HTML, a webmaster would be able to control where changes could or could not be made and specify dynamic content. We describe some A-HTML extensions below.

In this section, we present several transformations that could be made on any web in response to the sorts of observations described above. These transformations are based on several assumptions.

1. Sites have ‘‘front pages’’ where many visitors enter the site. The front page and pages nearby tend to be index pages, containing links to other pages rather than a great deal of content.
2. The closer a page is to the front page of a site, the easier it is to find and the more likely it is to be visited.
3. The closer a link is to the top of a page, the easier it is to find and the more likely it is to be traversed. According to [5], only 10% of users scroll beyond the first screenful of a web page.
4. Colors, fonts, and graphics can be used to highlight or draw attention to certain links.
5. However, placing too many links on a page or highlighting too many items reduces the page’s appeal and its useability.
6. Multiple pages at a web site may be related by common features, and grouping them together has intuitive appeal to users.
7. Users may perceive a connection between sections of the site that the webmaster never intended; linking these sections may facilitate user navigation. Users may also find irrelevant a connection the webmaster considered important.

These assumptions accord with both intuition and with observations of real users. Based on these assumptions, we present four basic kinds of transformations: promotion and demotion, highlighting, linking and clustering.

Promotion and Demotion

Promotion makes a link or page easier to find by placing a reference to it closer to the front page of the site (on the front page or a nearby index page) or by moving a link closer to the top of a page. Promotion and demotion are based on the *popularity scores* of links and pages: as part of its observations, the system records access counts for pages and traversal counts for links. However, neither data mining technology nor webmasters are quite ready for a system that can rearrange links arbitrarily. Therefore, promotion and demotion will be described in a limited form. The system will be given its own box on any number of the pages at the site (typically the front page and nearby index pages). The system is provided a limited amount of space over which it has total control; the webmaster can be sure that it will do no rearranging outside of its box. This box might be implemented as a frame or simply as a list of limited size at the top of those pages. Promotion, then, means putting a link into the box, and demotion means removing a link. Note that, because the box is of limited size, every promotion implies a corresponding demotion. We define *popularity* as follows

$$Pop(P) = AccessCount(P)$$

$$Pop(L) = TraversalCount(L)$$

That is, the popularity of an object (page or link) is simply how many times it is accessed or traversed. It is not sufficient to place pages and links in the box based on popularity - we must also take into account how accessible the objects already are. Let *Distance(X,Y)* be a measure of how far a page *X* is from page *Y* as a function of both the number of pages traversed and how far down the page each link is. We define the accessibility of an object *X* to be:

$$Acc(X) = \frac{1}{Distance(FrontPage, X)^2}$$

The farther an object is from the front page, the less accessible is it. Preliminary data shows an exponential falloff in accesses to a page as a function of its distance from the front page, and so we use the square of the distance. Let *L(X,Y)* be true when there exists a link from page *X* to page *Y*, *Depth(X,Y)* be the number of links above the link to *Y* on page *X*, and *P* be the set of pages P_1 ... P_n along the minimal path from *X* to *Y*. We define the distance as:

$$Distance(X,Y) = Depth(X,Y), \quad when \quad L(X,Y)$$

$$Distance(X,Y) = |P| + \alpha \sum_{i=1}^{n-1} Distance(P_i, P_{i+1}), \quad otherwise$$

Where *Alpha* is a scaling constant. We should promote an object when its popularity is high but its accessibility is low. Therefore, we define the promotion score of an object *X* as:

$$Pro(X) = \frac{Pop(X)}{Acc(X)}$$

We replace an object *Y* in a box *B* with an object *X* not in *B* if

$$\exists Y((Y \in B) \wedge (Pro(X) > Pro(Y)))$$

and

$$Pro(X) > \pi$$

Where *Pi* is a threshold promotability score required for an object to be promoted at all. If a box has available spaces, the extra spaces are considered to be null objects with *Pro()=0.*

Observation (1) about the homework pages at the CSE142 site reveals an excellent opportunity for promotion: the most recent homework page should be promoted to a prominent place on the front page. Note that as a new homework appears and old ones become outdated, the increase in popularity of the new one and the lack of interest in the old ones should guarantee that the front page has only the most current link.

Highlighting

Highlighting draws attention to an existing link on a page by emphasizing it with fonts, colors, or graphics. Because highlighting is a lightweight alteration, it can be permitted outside of a limited box. Like promotion and demotion, highlighting is based on popularity scores. We define *L* to be the set of links on a page *P*. We rank all L_i in *L* according to $Pop(L_i)$ The top 10%, say, are highlighted. The most effective percentage to highlight should be determined from user testing.

The second observation about the CSE142 pages suggests highlighting the solution set when it appears on its homework page. Each homework page contains only a handful of links. Once the assignment's due date has passed, the solution set is the most popular link on the page and is therefore chosen for highlighting. In this case, we highlight rather than promote as we probably do not want to put a box for promotions on *every* page at the site (though we may choose to promote the solution link to the front page).

Linking

Linking connects two pages that were previously unconnected by adding new hyperlinks between them. Linking is based on inferring *semantic* connections between pages based on correlations in user visits. The fact that many users visit two pages suggests that they are conceptually related in users' minds, even if the webmaster made no explicit connection. Similarly, unlinking is based on observing a lack of correlation; if links between two pages are never followed, we might infer that they are unrelated in users' minds, even though the webmaster connected them. We define the probability of visiting a page *P* as *P(P)*. If they are not linked already, two pages P_1 and P_2 should be linked if their visit probabilities are highly correlated:[3]

$$\rho(P(P_1), P(P_2)) > \delta$$

where *Delta* is a constant.

In the *Research* section of the UWCS page, certain research projects are interrelated; the theory group and the computational biology group, for example, have considerable overlap both in research topics and in personnel. There are, however, no links between the main research pages for these two topics. Even so, many visitors to the web site who visit one page visit the other. This observation suggests the conclusion that these two topics are related and should be linked together. Similarly, the *Spotlight* section of the front page contains a page showing an animation created in an undergraduate graphics class. People who visit this page often then seek out the graphics research page, suggesting that these two pages should also be **linked**. In both these cases, a *semantic* relationship between two pages has been inferred from the fact that they seem to be linked in the minds of visitors, as evidenced by navigation patterns.

Clustering

Clustering associates a collection of related pages and makes them accessible as a group on a newly created page. (see [2] for work that uses clustering to organize documents for browsing). The system recognizes a collection of similar documents that are not grouped together anywhere at the site, creates a new page for them, and adds a reference to the new page. Documents may be considered similar based on their filenames, their locations in the site hierarchy, and their correlation in visitor paths. A set of pages *P* is considered a cluster when

$$\forall X \forall Y (X \in P \wedge Y \in P \rightarrow editdist(X, Y) < k)$$

and

$$\forall X \forall Y (X \in P \land Y \in P \rightarrow \rho(X,Y) > \delta)$$

and

$$\neg \exists P \forall X (X \in P \land L(P,X))$$

Where *L(P,X)* is true when there exists a link from *P* to *X*. The first requirement makes sure that the pages are all similarly named by requiring that the edit distance between their names and paths is smaller than some constant *k*. For example, the pathnames `homework/hw3/hw3solu.c` and `homework/hw10/hw10solu.c` have an edit distance of 4 - two changes (from 3 to 1) and two inserts (inserting 0 twice). The second requirement makes sure that the pages are all correlated in user access paths. This requirement could be dropped, since we may wish to group similar pages together for organizational reasons even if users do not necessarily access them all on a single visit. The third requirement simply makes sure there does not already exist a page at the site which contains links to all the pages. If a page links some, but not all, of the pages, the system may want to add the remaining pages or point this out to the webmaster.

The third and fourth observations above, along with an examination of the filenames at the CSE142 site, suggest that the homework solution sets would make an excellent cluster. The homework directories are called `hw3`, `hw4`, etc., and the solutions are named `hw3solu.c`, `hw4solu.c`, etc. The similar names, the presence of paths that visit multiple solution pages, and the fact that the solutions never appear on a single page together lead the system to suggest that they be given their own page. A link to this page, then, could be promoted to the front page. As with the linking examples given above, we have inferred a conceptual relationship between certain documents based on their locations and access patterns.

Discussion

Limited to a box, promotion and demotion are fairly nonintrusive transformations. The webmaster must deliberately set aside space for the system to use, and it will not stray outside that box. Applied more broadly, promotion and demotion can still be useful, but may have undesirable effects. For example, an unordered list of links is a fine candidate for reordering according to popularity, but if the list is already ordered - alphabetically, say, or chronologically - then allowing the system to play with it will create confusion among users.

Highlighting and promotion are performed under similar circumstances. Highlighting is a weaker transformation; since it changes the appearance of a link but not its position, highlighting may have less overall effect on what visitors notice. At the same time, highlighting is also less intrusive; making a link boldface or changing its color is a much simpler transformation than rearranging links and is less likely to violate the webmaster’s design intentions.

Note that linking differs from promotion in that promotion involves making a page or link available from a place *closer* to the front page of the site so as to make it more accessible to users, whereas linking adds crosslinks between parallel pages in order to make semantic connections explicit. Whereas promotion and highlighting are essentially focused on making existing connections easier to find, linking creates entirely new connections. Linking can be an intrusive transformation - any two pages in the site are potential candidates for linking, and the system may want to add arbitrary links. One way to contain the effect of linking is to allow the system to add a small footer to any page with links to related pages. This footer would essentially carry the message ‘‘If you liked this page, you might also like...’’. Linking also has greater potential for illuminating important aspects of the site that never occurred to the webmaster. Linking is most effective in an advice system; the system can discover new connections, and the webmaster can decide if they are significant. Similarly, clustering can discover connections which never occurred to the webmaster and point them out to the webmaster. In fact, clustering cannot be done without referring to the webmaster to name the new cluster of objects, since the adaptive assistant has no basis for really understanding *why* these pages are connected. In the case of the homework solutions, the assistant would present the proposed cluster to the webmaster, and she would have to recognize that these are all solution sets and decide that they form a cluster worth having. If the adaptive assistant uses a more general clustering approach, it might be capable of discovering even more varied (and surprising!) connections. For example, it might take document content into account. Pages can be transformed into vectors by their word content, and vectors can be clustered based on proximity in word-vector space.:[2]

This approach is more time-consuming but less limited than the approach described above.

Adaptive HTML

A-HTML's extensions are designed to tell an automated assistant where it may and may not make changes to the web site. By thus annotating her pages, a webmaster can facilitate adaptivity without worrying that the assistant will destroy important aspects of the design. A-HTML is a preliminary idea; we here describe several extensions to support the above transformations. The most basic A-HTML tag is a *scope* declaration. By bounding a block of HTML with `<A-HTML>` and `</A-HTML>`, the webmaster specifies an area that the assistant may alter. The `<A-HTML>` tag offers a number of optional arguments. `highlight` specifies whether or not links in the block may be highlighted. `promote` and `demote` specify whether links in the block may be promoted or demoted. `time` tags a block as being time-dependent; the `time` argument may have values such as "monday", "weekday", or "september" to indicate regularly occuring times or an expiration date after which the block is suppressed. The block may also be tagged with keywords that the automated assistant may use for clustering. A-HTML would also extend the `<li>` (list) tag to have several new arguments. `order` indicates how the list should be ordered. If the list is, for example, tagged "alphabetical", the automated assistant may not reorder arbitrarily and must keep the list alphabetical. A list of links tagged "popularity" should be ordered by how popular the links are. If the list is "unordered", the assistant may order it by any criteria it chooses. In addition, lists can have `add` and `delete` tags that specify whether the assistant may add items to the list or remove items from the list. A `cluster` tag tells the assistant that the list is intended to represent a collection of related items. The assistant should enforce this by adding new items that seem related and removing items that do not.

Evaluation

Although the problem of measuring the quality of a web site design is thorny, we have identified several preliminary approaches. Progress on the design of adaptive web sites will include more sophisticated methods of evaluating a site's usability. We propose a basic metric for how usable a site is: how much effort must the average user exert in order to find what she wants? Effort can be defined as a function of the number of links traversed and the difficulty of finding the links on their pages. Full path data (see the Observation section) provides enough information for us to compute how much effort our visitors exert in traversing our site.

In addition to scouring access logs, we can use controlled tests with subjects. Such tests have the advantage of allowing us to observe users as they interact with the site - we get much more information than is encoded in user access logs. As subjects perform tasks such as finding information, downloading software, or locating documents, we may gather data such as:

- Whether the subject succeeded at the task (or realized it was not solvable).
- How long the subject took to solve the goal.
- How much exploration was required (as compared to how long the shortest path to the goal was).
- What frustrations the user experienced in the process.

Careful observation of test subjects would complement the relatively limited access data we get on all of the site's regular visitors. Of course, we can also rely on intermediate measures such as encouraging users to fill out feedback forms and send e-mail messages.

Related work

Our approach is only one of many possible types of adaptive web sites. We discuss other kinds of approaches and related issues, illustrating these issues with examples drawn from current AI research. In general, sites may be adaptive in two basic ways. First, the site may focus on dynamic modification for individual visitors: customizing web pages in real time to suit the needs of a specific user. Second, the site may focus on offline global improvement: altering the underlying structure to make navigation easier for all. Whether we modify our web pages online or offline, the information available in user access logs may not be sufficient for our purposes; we also discuss how to support adaptivity with semantic annotations to HTML pages. Finally, we examine other issues that arise in designing adaptive web sites.

Dynamic modification for individual users can be an effective tool for improving web interfaces. One way for a site to respond to particular visitors is to allow *manual customization*: allowing users to specify display options that are remembered during the entire visit and from one visit to the next. The Microsoft Network (at http://www.msn.com), for example, allows users to create home pages with customized news and information displays. Every time an individual visits her MSN home page, she sees the latest pickings from the site presented according to her customizations.

Path prediction, on the other hand, is attempting to guess where the user will want to go in order to take her there more quickly. The WebWatcher[1] (see http://www.cs.cmu.edu/~webwatcher/) learns to predict what links users will follow on a particular page as a function of their specified interests. WebWatcher observes many users over time and attempts to learn, given a user's current page and stated interests, where she will go next. A link that WebWatcher believes you are likely to follow will be highlighted graphically and duplicated at the top of the page. Visitors to a site are asked, in broad terms, what they are looking for. Before they depart, they are asked if they found what they wanted. WebWatcher uses the paths of people who indicated success as examples of successful navigations. If, for example, many people who were looking for ‘‘personal home pages’’ follow the ‘‘people’’ link, then WebWatcher will tend to highlight that link for future visitors with the same goal.

Instead of predicting a user's next action based on the actions of many, we might try to predict the user's final goal based on what she has done so far, by viewing path prediction as a plan recognition problem. *Plan recognition*[5, 9] is the problem of identifying, from a series of actions, what an agent is trying to accomplish. Lesh [6] poses this problem in a domain-independent framework and investigates it empirically in the Unix domain: by watching over a user's shoulder, can we figure out what she is trying to accomplish (and offer to accomplish it for her)? Lesh models user actions as planning operators. Assuming users behave somewhat rationally, he uses these actions' precondition/postcondition representation to reason from what a user has done to what she must be trying to do. In the web domain, we observe a visitor's navigation through our site and try to determine what page she is seeking. If we can do this quickly and accurately, we can then offer the desired page immediately.
The AVANTI Project[4] (see http://zeus.gmd.de/projects/avanti.html) focuses on dynamic customization based on users' needs and tastes. As with the WebWatcher, AVANTI relies partly on users providing information about themselves when they enter the site. Based on what it knows about the user, AVANTI attempts to predict both the user's eventual goal and her likely next step. AVANTI will prominently present links leading directly to pages it thinks a user will want to see. Additionally, AVANTI will highlight links that accord with the user's interests. AVANTI is illustrated on an example Louvre Museum web site. For example, when a disabled tourist who wishes to visit the museum comes to the site, links regarding handicapped access and tourist information are emphasized. AVANTI relies on users providing some information about themselves in an initial dialogue; the site then uses this information to guide its customization throughout the user's exploration of the site. AVANTI also attempts to guess where the user might go based on what she has looked at so far. For example, if our disabled tourist looks at a number of paintings at the site, AVANTI will emphasize paintings links as it continues to serve pages. As with the WebWatcher, we might ask if we can avoid AVANTI's requirement that users explicitly provide information.

Above, we propose extending HTML to allow the encoding of higher-level directives for the adaptive site. We might want to go beyond the level of A-HTML and use a formal language to describe the content of our pages. HTML and other languages are well suited for specifying how documents should *look*, but they say very little about what they might *mean*. An adaptive web site could use semantic information to better organize the site, to respond to queries, and to better predict what users might want to see. SHOE[7] (at http://www.cs.umd.edu/projects/plus/SHOE/), for example, is a language for adding simple ontologies to web pages. SHOE adds basic ontological declarations to HTML; a page can refer to a particular ontology and declare classifications for itself and relations to other pages. In their example, a man's home page is annotated with information about him, such as the fact that he is a person, his name, his occupation, and his wife's identity (she has her own home page). SHOE is designed to facilitate the exploration of agents and the workings of search tools, but ontological annotation would also be useful for our purposes.

The quest for the self-improving web site raises a number of related questions. An adaptive site will be active twenty-four hours a day, seven days a week. The site will constantly be ingesting and analyzing data, adjusting its concepts and models, and updating its own structure and presentation. Over time, this constant cycle will reflect many hours of experience and refinement. In the past, AI research has focused on single trials and short-lived entities: systems that run their experiments and shut down, to start again the next day with a blank slate. Although such an approach may be applied to the adaptive site challenge, the most intelligent site will surely be one that continually accumulates knowledge about pages, surfers, content, and itself.

User interface design is difficult enough for human beings to perform well. Yet an adaptive web site will have to take into account all the artistry of good design in its self-improvements. We can limit the scope of the system's ability to change itself, thus ensuring that it cannot do too much harm, but this means we also limit its scope for improvement. On the other hand, giving the system free rein for radical transformation might mean giving it free rein for radical screwup.

We might instead put the AI system in the role of advisor to a human master. Instead of making changes under cover of night, our AI system must now intelligently present suggestions to a human being, complete with explanation and justification. Such a solution frees us from the problem of changing details without changing design but presents us with a new interface.

Conclusions

In this paper, we have defined adaptive sites and described how they can augment a webmaster's understanding of how visitors interact with a site. We have presented several transformations and formalized the conditions under which they should be applied. We have also described several different ways of adding adaptivity to a site including using an autonomous assistant, giving advice to the webmaster, and using A-HTML.

We have developed a prototype observation system and are developing an adaptive assistant to provide advice, perform HTML transformations, and respond to A-HTML annotations. Our goal is a fully working system which can be added to an existing web site without fundamental changes to that site. This system will regularly (1) provide feedback to the webmaster about access patterns at a higher level than raw server logs do; (2) advise the webmaster about changes to the site that will improve its appeal and its useability; and (3) autonomously make certain kinds of changes to the site to keep its presentation timely and intuitive to users. Our system will be tested on real web sites, providing data on user access patterns as well as on the effectiveness of our approach.

Acknowledgement

This research was funded in part by Office of Naval Research grant 92-J-1946, by ARPA / Rome Labs grant F30602-95-1-0024, by a gift from Rockwell International Palo Alto Research, and by National Science Foundation grant IRI-9357772.

Endnotes

[1] Automatically improving upon the sickly green color of the page is, unfortunately, beyond the scope of this paper.

[2] Although a service like WebThreads can be extremely useful, path data can also be heuristically inferred from standard server logs. If we assume that accesses originating from the same machine at around the same time

correspond to the visit of a single user making a coherent series of page visits, we can record sequences of pages accessed by individual visitors. We can construct a graph of the site, where nodes represent pages and directed arcs represent links, and match these sequences against the graph to determine the full path followed.

References

1 Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. Webwatcher: A learning apprentice for the world wide web. In *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*, pages 6-12, Stanford University, 1995. AAAI Press. To order a copy, contact sss@aaai.org. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/project-home.html

2 D. Cutting, D. Karger, J. Pedersen, and J. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In *Proceedings of SIGIR*, 1992.

3 Morris H. DeGroot. *Probability and Statistics*. Addison Wesley, second edition, 1986.

4 J. Fink, A; Kobsa, and A. Nill. User-oriented adaptivity and adaptability in the avanti project. In *Designing for the Web: Empirical Studies*, Microsoft Usability Group, Redmond (WA)., 1996. http://zeus.gmd.de/projects/avanti.html

5 H. Kautz. *A Formal Theory Of Plan Recognition*. PhD thesis, University of Rochester, 1987.

6 Neal Lesh and Oren Etzioni. A sound and fast goal recognizer. In *Proc. 15th Int. Joint Conf. on AI*, pages 1704-1710, 1995.

7 S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based web agents. In *Proceedings of the First International Conference on Autonomous Agents*, 1997.

8 Jakob Nielsen. Top Ten Mistakes in Web Design. May, 1996. http://www.sun.com/columns/alertbox/9605.html

9 M. Pollack. Plans as complex mental attitudes. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 77-101. MIT Press, Cambridge, MA, 1990.

10 Webthreads LLC. *WebThreads*. 1996. http://www.webthreads.com/