Adaptive Web Sites: Concept and Case Study

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

1 INTRODUCTION AND MOTIVATION

Designing a complex web site so that it readily yields its information is a difficult task. The designer must anticipate the users' needs and structure the site accordingly. Yet users may have vastly differing views of the site's information, their needs may change over time, and their usage patterns may violate the designer's a priori expectations. As a result, web sites are all too often fossils cast in HTML, while user navigation is idiosyncratic and evolving.

Understanding user needs requires understanding how users view the data available and how they actually use the site. For a complex site this can be a difficult task; user tests are expensive and time consuming, and the site's server logs contain massive amounts of data. We propose a *web management assistant*: a system that can process massive amounts of data about site usage and suggest useful adaptations to the webmaster. The use of such assistants is one way to develop **adaptive web sites:** *sites that semi-automatically improve their organization and presentation by learning from visitor access patterns.*

For example, imagine a site devoted to information about automobiles. The webmaster initially decides to organize the site by manufacturer; each auto company will be represented by a dedicated page, containing links to each of their models. However, many visitors to this site intend to comparison shop; a visitor wanting to compare minivans, for example, would have to go to each manufacturer in turn and look up their minivan. Our web management assistant will observe this common behavior pattern and suggest that the webmaster add a "minivans" page containing a link to each minivan directly. Even better, the assistant could suggest a complete *change in view* — creating a way of browsing the information at the site by car model instead of by manufacturer for all models, not just mini vans.

While adaptive web sites are potentially valuable, their feasibility is still unproven; can non-trivial adaptations be automated? will adaptive web sites run amok, yielding chaos

rather than improvement? what is an appropriate division of labor between the automated system and the human webmaster? To investigate these issues empirically, we consider a case study: we analyze the problem of automatic *index page synthesis* based on visitor access patterns. An *index page* is a page consisting of links to a set of pages that cover a particular topic (e.g., minivans).

In this article, we consider the basic design decisions that underlie any kind of adaptive web site, and consider several kinds of adaptive web sites by way of illustration. Next, to demonstrate the potential power of adaptive web sites, we consider the *index page synthesis* case study in more depth. We introduce the IndexFinder page synthesis system and describe its application to a live Web site. More detail about our algorithms and experiments can be found in [6].

1.1 VARIETIES OF ADAPTIVE WEB SITES

Work on adaptive web sites has taken a variety of approaches, from providing automated "tour guides," to making personalized recommendations, to charting the "footprints" left by visitors to a site. All approaches, however, must address the following questions in one way or another.

- **What is the division of labor between the automated assistant and the human webmaster?** The program could be *fully manual* — simply carrying out changes the webmaster specifies — or *fully automated* — allowed to make make changes to the web site on its own. Many programs will be *semi-automated*, requiring webmaster approval to make limited kinds of alterations. For example, a program may be restricted to *non-destructive* adaptations that may add information but not destroy existing structure; examples include adding pages (but not removing them) and adding or highlighting links.

- **Customization or Transformation?** *Customization* is adapting the site's presentation to the needs of each *individual* visitor, based on information about those individuals. Any changes impact only that single user, effectively creating numerous versions of the site, one per user. Customization will be familiar to many readers; many sites allow the creation of personalized home pages. *Transformation* is improving the site's structure based on interactions with *all* visitors. A list of "most popular news stories" is a simple example of a global transformation; the site factors in the interests of past visitors to present links likely to appeal to future visitors, even brand new ones.

- **What aspects of the Web site are open to change?** Manual customization at sites like Yahoo or Excite allows the user to specify values for a small number of parameters (e.g., the stocks in her portfolio, the weather report to receive, etc.).

```
24hrlab-214.sfsu.edu - - [21/Nov/1996:00:01:05 -0800] "GET /home/jones/collectors.html HTTP/1.0" 200 13119
24hrlab-214.sfsu.edu - - [21/Nov/1996:00:01:06 -0800] "GET /home/jones/madewithmac.gif HTTP/1.0" 200 855
24hrlab-214.sfsu.edu - - [21/Nov/1996:00:01:06 -0800] "GET /home/jones/gustop2.gif HTTP/1.0" 200 25460
x67-122.ejack.umn.edu - - [21/Nov/1996:00:01:08 -0800] "GET /home/rich/aircrafts.html HTTP/1.0" 404 617
x67-122.ejack.umn.edu - - [21/Nov/1996:00:01:08 -0800] "GET /general/info.gif HTTP/1.0" 200 331
203.147.0.10 - - [21/Nov/1996:00:01:09 -0800] "GET /home/smith/kitty.html HTTP/1.0" 200 5160
24hrlab-214.sfsu.edu - - [21/Nov/1996:00:01:10 -0800] "GET /home/jones/thumbnails/awing-bo.gif HTTP/1.0" 200 5117
```

Figure 1: Typical user access logs, these from a computer science web site. Each entry corresponds to a single request to the server and includes originating machine, time, and URL requested. Note the series of accesses from each of two users (one from SFSU, one from UMN).

However, other aspects of the pages and the links between pages are etched "in stone". More generally, adaptations could add or remove links, highlight links, change text or formatting, and even create completely new web pages.

- **Are adaptations based on the content of pages or people's navigational choices?** A site that uses content-based adaptation organizes and presents pages based on their content — what the pages say and what they are about. Computer programs are still unable to reliably determine the *meaning* of a web page or other human utterance; content-based approaches (using keywords, for example) are, necessarily, approximate. On the other hand, access-based adaptation uses the way past visitors have interacted with the site to guide how it should be modified. Access information (derived from web server logs, for instance) might contain common patterns of user behavior that suggest valuable adaptations. Naturally, content-based and access-based adaptations are complementary and may be used together.

See the sidebar for examples of different approaches currently found on the Web. In the next section we describe our approach, which may be characterized as a semi-automated, access-based, transformation approach to index-page synthesis.

2 INDEX PAGE SYNTHESIS

Our goal has been to explore the possibilities of adaptive web sites with a test case. We have chosen to investigate the automatic synthesis of new index pages; an index page is a page consisting of links to a set of pages that cover a particular topic (e.g., minivans). Index pages are central to site organization. If we are able to generate index pages that are valued and adopted by the human webmaster, we can have a significant impact on the site's organizational structure over time. Although index page synthesis is an ambitious

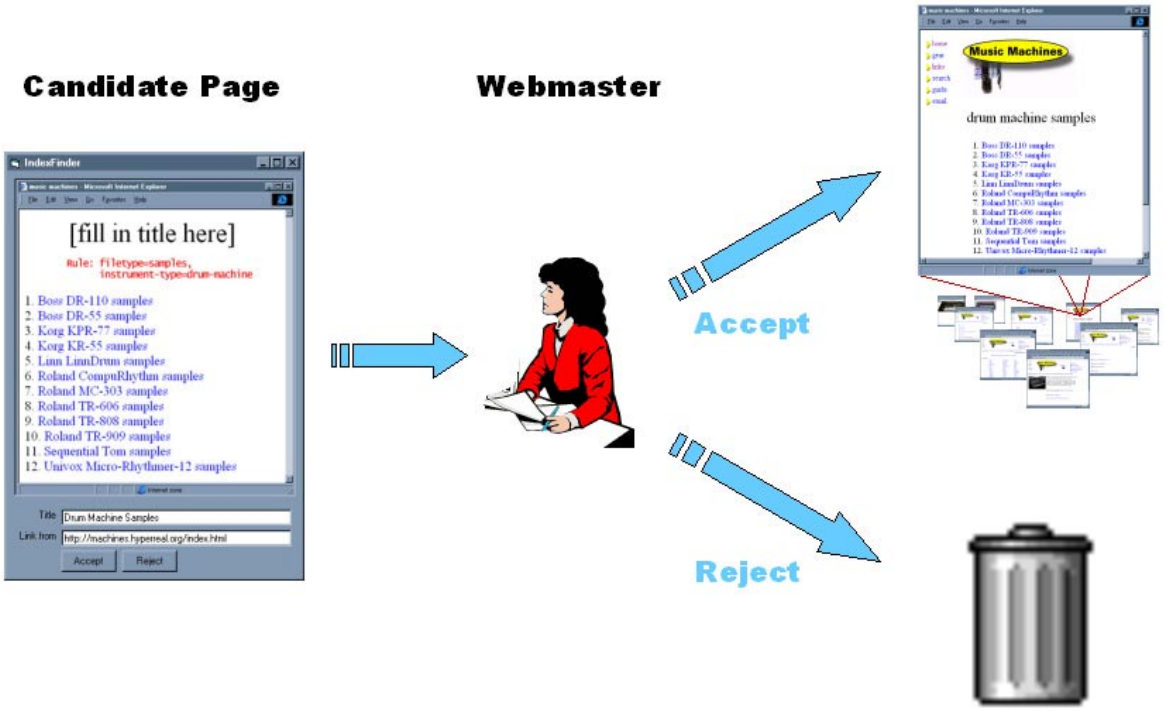

Figure 2: IndexFinder suggests candidate index pages to the webmaster. IndexFinder generates a candidate index page. The human webmaster accepts or rejects the candidate page. If the page is accepted, IndexFinder creates a final version of the page and adds it to the web site; the webmaster specifies the page's title and where in the site it should be linked. Rejected pages are discarded.

and challenging goal, it is a non-destructive transformation; the webmaster need not worry that her site design will be scrambled or her data lost.

We define the **index page synthesis problem**: given a web site and a visitor access log, create new index pages containing collections of links to related but currently unlinked pages. An access log contains one entry for each page requested of the web server (see Figure 1). Each request lists at least the origin (IP address) of the request, the URL requested, and the time of the request. *Related but unlinked* pages are pages that share a common topic but are not currently linked at the site; two pages are considered linked if there exists a link from one to the other or if there exists a page that links to both of them.

The problem of synthesizing a new index page can be decomposed into several sub-problems.

1. What is the content (i.e. set of hyperlinks) of the index page?

2. Does it have a coherent topic? What should its title be?

3. How are the hyperlinks on the page ordered?

4. How are the hyperlinks labeled?

5. Is it appropriate to add the page to the site? If so, where?

IndexFinder does most of this work automatically, leaving to the webmaster the questions of whether and where the page should be added to the site, and how it should be titled. The first two subproblems have been the focus of our research. The third and fourth are adequately handled with simple rules; hyperlinks are labeled with the title of the target page and ordered alphabetically. Naturally, future work may explore other options. When IndexFinder generates a candidate page, the contents are presented to the webmaster. IndexFinder also generates a conceptual description of the page's topic, which is presented in the form of a "rule" defining what appears on the page. Figure 2 shows how IndexFinder and the human webmaster interact. IndexFinder (1) processes the site's access logs and generates a candidate index page (2). The rule shown on the candidate index page specifies that the page contains audio samples of drum machines. The webmaster (3) is prompted to accept or reject the candidate page and to provide a title based on the description. The webmaster also specifies where in the site the page should be linked. If the page is accepted, IndexFinder creates a final version of the page (4), using the supplied title and a graphic template for the site. Finally, the page is linked into the site (5).

The main source of information we rely on is the site's web server log, which records the pages visited by a user at the site. We also rely on the **visit-coherence assumption**: the pages that a user visits during one session at the site are likely to be closely related. One visitor may be interested in electric guitars, and view several pages on that topic; another may be looking for inexpensive keyboards and view all the pertinent pages. We do not assume that *all* pages in a single visit are related. After all, the information we glean from individual visits is noisy; visitors may pursue multiple distinct interests in a single visit, follow the wrong links, or simply get distracted. However, if a large number of visitors continue to visit and re-visit the same set of pages, that provides strong evidence that the pages in the set are related. Thus, we accumulate statistics over many visits by numerous visitors and search for overall trends.

To generate useful index pages, it is not enough to detect frequently co-occurring pages. Valuable index pages typically correspond to a topic or concept that is intuitive to visitors; they cannot be composed of an arbitrary set of links. More formally, the set of links comprising an index page must be both:

- **Pure** — containing no inappropriate links, and
- **Complete** — containing *all* links that accord with the topic.

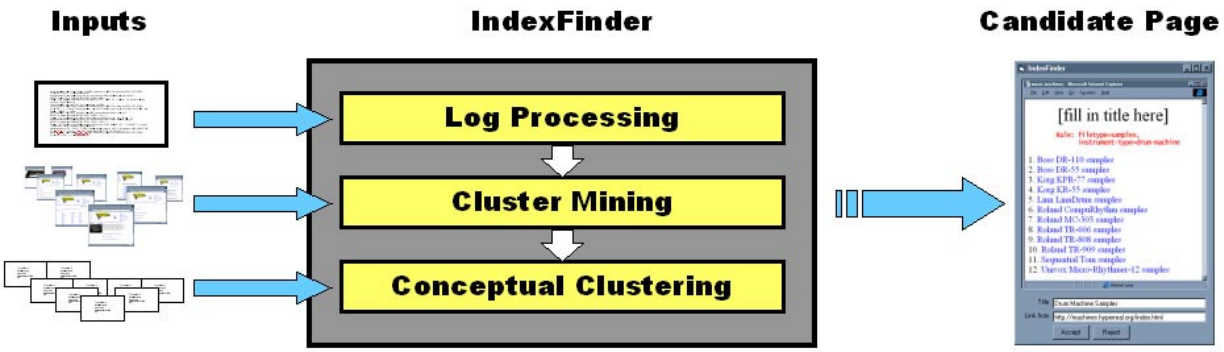

Figure 3: The inner workings of IndexFinder. IndexFinder consists of three basic modules. The log processing module takes the web server logs and computes how often pages co-occur in user visits. The cluster mining module takes this information and a graph of the web site and finds clusters of frequently co-occurring pages. The conceptual clustering module uses conceptual descriptions of the site's pages to convert these clusters into coherent concepts. These are output as candidate index pages and presented to the webmaster.

For example, if the topic of the index page is "minivans, the page should contain no links to information about sports cars or trucks, and there should be no pages at the site about minivans that are not included in the index page.

The *IndexFinder algorithm* takes as input a web server log and a conceptual description of each page at a site. In the case of Music Machines, the webmaster provides a description of each page including the type of instrument (e.g., guitar, keyboard, drum machine), the file type (e.g., text, image, audio sample), the price of the instrument, where it is manufactured, and so on. Descriptions are represented as attribute/value pairs (e.g., filetype=image, price=cheap). Data about how often pages occur together in user visits is extracted from the web server logs; IndexFinder then applies a novel *statistical cluster mining* technique [6] to the data and produces candidate clusters as output. In statistical cluster mining, objects (e.g., web pages) are grouped together based solely on a similarity measure (e.g., how often they co-occur in user visits); the algorithm makes no guarantee that the clusters are pure or complete with respect to some topic. Next, each candidate cluster (represented as a set of positive examples of a concept) is fed to a concept learning algorithm, which then produces the simple concept that most closely describes what the set of pages have in common. The concept is expressed as a conjunction of logical attributes; in Figure 2, for example, these attributes are (1) the instruments are all drum machines; and (2) the files are all audio samples. Pages in the original candidate cluster that do not match the concept description are discarded; similarly, other pages at the site that do fit the description are added. This new complete and pure cluster is presented to

the webmaster, who decides whether or not it should be added to the site as a new index page.

3 DEPLOYING INDEXFINDER

To test IndexFinder in practice we deployed it on a web site and compared user response to the pages IndexFinder generated against the pages generated by other algorithms and against human-authored index pages. We found that IndexFinder outperforms previous algorithms and its pages even compare quite favorably to human-authored index pages.

Our experiments draw on data collected from the **Music Machines** web site (at `http://machines.hyperreal.org`), which is devoted to information about many kinds of electronic musical instruments. Music Machines contains approximately 2500 distinct pages, including HTML pages, plain text, images, and audio samples. The site receives approximately 10,000 hits per day from roughly 1200 distinct visitors. Each page at the site was described by 13 attributes such as the type of instrument, the cost, the country of manufacture, the file type, and other domain-specific attributes. All pages were hand-tagged by the webmaster (one of the authors).

We evaluate IndexFinder by comparing the quality of the index pages it synthesizes with the quality of two other sets of index pages:

- Human-authored: we took four index pages from the Music Machines site and included them in our experiment.
- COBWEB: we applied COBWEB [3], the leading conceptual clustering algorithm, to the task of index page synthesis. Like IndexFinder, COBWEB produces only clusters that are pure and complete with respect to some concept.

Note that, unlike IndexFinder and COBWEB, statistical clustering techniques such as [7, 8] and standard data mining algorithms such as [1] do not yield complete and pure index pages of the sort we want to provide to the Webmaster, so we do not include them in this comparison. In separate experiments, we have shown that PageGather (the statistical cluster mining component of IndexFinder) outperforms standard methods [6].

Both COBWEB and IndexFinder generate a set of clusters, of which the top ten are chosen. Each cluster is converted into a web page containing a link to each page in the cluster. Link text is generated from the title of the linked-to page (or the filename if no title is available). Each cluster is presented to the webmaster, along with its logical description, for naming. For example, presented with the rule ``` ``filetype=image, price=cheap, instrument-type=synth'' ```, the webmaster might name the page "images of cheap synthesizers".

In this way, we generated a large set of index pages; next, we added in the human-authored index pages. To conduct our experiment, five of these index pages are randomly

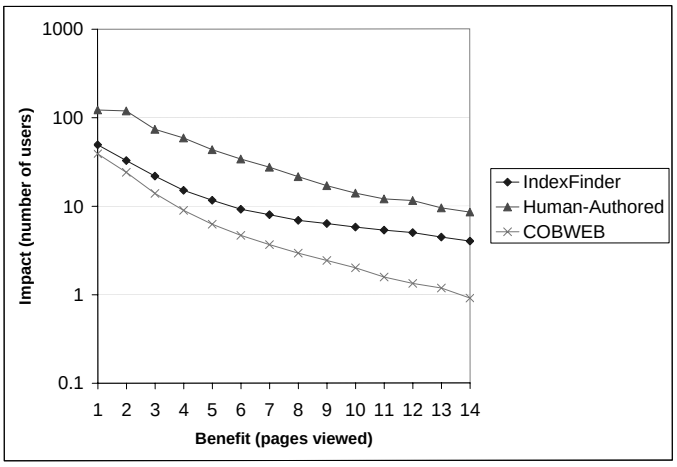

Figure 4: Results from our case study. IndexFinder performs substantially better than COBWEB but not as well as the human-authored pages.

chosen and made accessible on the web site each day. The index pages appear in the site's navigation bar, which appears on all site pages (see `http://machines.hyperreal.org`).

We compared index pages with respect to impact and benefit. *Impact* measures the number of people who use the automatically generated index pages; *benefit* measures the number of links these users follow on each index page. Over the course of each day, we recorded how many users visited each index page, and how many links they clicked on. We computed the number of visitors who clicked on at least one link, the number who clicked on at least two links, and so on. We accumulated these daily counts over the course of a month, and calculated an average daily impact and benefit for each index page. We then averaged these impact and benefit statistics over all index pages generated by a particular algorithm. For each algorithm, we plotted a line of the number of pages viewed (benefit) vs. the number of people who viewed that many pages (impact). If no users viewed more than, say, five pages from any index page for an algorithm, then the line for that algorithm will stop at five. The results of our experiment are shown in Figure 4. Note that impact is measured on a log scale, as it drops off exponentially with increasing benefit. IndexFinder performs substantially better than COBWEB but not as well as the human-authored pages. While IndexFinder and the human clusters seem to be converging, COBWEB drops off more quickly.

4 LESSONS AND FUTURE WORK

We began this work believing that by computing statistics over the visitor access log for Web sites, we would be able to identify valuable adaptations and present them to the site's webmaster. We've learned that to mine this information effectively, we needed to combine the statistical patterns gleaned from the log file with logical descriptions of the contents of each Web page. This observation led us to move from our statistical cluster-mining algorithm, called PageGather, to IndexFinder, which fuses statistical and logical information to synthesize index pages. Logical descriptions of web pages could be derived from XML annotations or from a database of the kind used in Strudel [5].

In future work we plan to move towards a higher degree of automation where IndexFinder suggests possible names for the pages it creates (based on their logical description) as well as appropriate places to insert the pages into the web site (based on visitor access patterns). We also plan to explore intermediate points on the spectrum between customization and transformation. Specifically, we plan to identify classes of visitors to the site (e.g., prospective students visiting a university's Web site) and target specific adaptations at these classes of visitors.

More generally, the IndexFinder case study demonstrates the potential of automatic, and semi-automatic, adaptation to improve web sites for populations of visitors. Beyond the manual customizations so common in today's portal Web sites (e.g., Yahoo), we are beginning to glimpse the potential of data mining of visitor access logs to continually transform and refine Web sites.

References

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th VLDB Conference*, 1994.

[2] 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.

[3] D. Fisher. Iterative Optimization and Simplification of Hierarchical Clusterings. *Journal of Artificial Intelligence Research*, 4, 1996.

[4] N. Good, J. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.

[5] A. Levy, D. Florescu, D. Suciu, J. Kang, and M. Fernandez. Catching the boat with Strudel: experiences with a web-site management system . In *ACM SIGMOD Conference on Management of Data*, 1998.

[6] M. Perkowitz and O. Etzioni. Towards adaptive web sites: Conceptual framework and case study. *Artificial Intelligence*, To appear, 2000.

[7] J. Rocchio. *Document Retrieval Systems — Optimization and Evaluation.* PhD thesis, Harvard University, 1966.

[8] E.M. Voorhees. Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. *Information Processing & Management*, 22:465–476, 1986.

[9] A. Wexelblat and P. Maes. Footprints: History-rich web browsing. In *Proc. Conf. Computer-Assisted Information Retrieval (RIAO)*, pages 75–84, 1997.

Types of web site adaptivity are almost as varied as the web itself; here are a few examples illustrating different popular approaches.

- The typical "portal" site provides a variety of *manual customization* to users, who may select particular links, news topics of interest, or favorite stocks to follow. Automation is minimal: the user arranges her own customized home page. See:
  - The "My Yahoo!" personalizable home page. `http://my.yahoo.com/`
  - Go2net's personalizable home page. `http://www.go2net.com/`
- *Automated customization* is also common on the Web; AVANTI [2], for example, uses both information from user profiles (e.g., whether a user is a tourist) and from user behavior (e.g., whether the user visits many pages about sculpture) in order to decide how to present links (e.g., highlighting a link to information about a museum's famous sculpture garden). See:
  - The AVANTI home page. `http://fit.gmd.de/hci/projects/avanti/`
  - The WebWatcher tour guide. `http://www.cs.cmu.edu/~webwatcher/`
- Another common approach to automated customization is *Collaborative Filtering* (e.g., [4]). In these systems, users rate objects (e.g. web pages or movies) based on how much they like them. Users who give similar ratings to similar objects are presumed to have similar tastes; when a user seeks recommendations of new objects, the site suggests those objects that were highly rated by other users with similar tastes. Companies such as Netperceptions, Inc. provide this sort of technology to a wide range of web sites and particularly to e-tailers such as wine.com or Amazon.com. See:
  - NetPerceptions, Inc. `http://www.netperceptions.com/`
  - Amazon.com. `http://www.amazon.com/`
- A *transformation* approach is taken by Footprints [9], The motivating metaphor is that of travelers creating footpaths in the grass over time; in the web site domain, Visitors leave their "footprints" behind, in the form of counts of how often each link is traversed. Over time, "paths" accumulate in the most heavily traveled areas; new visitors to the site can use these well-worn paths as indicators of the most interesting pages to visit. Footprints are left automatically (and anonymously), and any visitor to the site may see them; visitors need not provide any information about themselves in order to take advantage of the system. See:
  - Papers about Footprints. `http://wex.www.media.mit.edu/people/wex/`

Figure 5: