Maitre-D: a Site-based Agent for Web-Page Recommendation

Richard Lachman
Interactive Cinema Group
MIT Media Lab
richlach@media.mit.edu
January, 1997

1 Introduction

This project explores website-based agents, designed to provide direction and recommendations to visiting clients. Time, processing power and intimate knowledge of the content being served are leveraged on the server-side, to allow suggestions and linkages to be pushed to users, based on their current interests and browsing history.

Agents that can accurately model their user's preferences, and communicate that information to one another as necessary, seem to be a focus of much recent industry attention. However, live users aren't the only information-producing bodies on the net; in fact, user-to-user interactions may make up just a small portion of useful net or Web traffic today. The bulk of information, from a consumer's point of view, sits on static web-sites, waiting to be explicitly pulled to a client. Site designers can only convey as much organizational information as fits on a navigation bar, and can't emphasize or de-emphasize content based on a user's interests. The information on a site is assembled according to how the site-designer thinks a user will want to browse -- and, while intelligent layout and authoring has a lot to be said for it, all hierarchies cannot be all things to all users.

Maitre-D is a preliminary step in addressing that issue. Using a combination of site-indexing, context-sensitive searches, and a layout/display system, it tries to convey "just-in-time" linkages between individual pieces of content. Rather than relying solely on fixed hyperlinks, it suggesting current contextually relevant locations the user may wish to experience.

2 Maitre-D

The package consists of three modules: a search-engine/site-indexer, a Java/Javascript/CGI-script communications layer, and a continuous structure for agent recommendations to the user. Essentially, the actions of a user-modeling agent are represented by a fixed

string of keywords, sent to the webserver as the client browses (this is seen as a two-agent system: client-side and user-side. Maitre-D only implements the server-side functionality). These key terms, along with a conceptual summary of the client's current URL, are used in a local search-engine query. The query-results, influenced by the client's interests and current "state", are translated through a modified version of Mike Murtaugh's "NIF Catalog browser" (Murtaugth96). (Note: this represents just one of many possibly styles of display. For further discussion, see Lachman97a and Lachman97b). The NIF system has been adapted to display a pregenerated, cross-indexed representation of the entire site, as textual tags and summaries. Each page is conceptually related to several others, with a range of relevance visible through text position, colour and saturation. The end result is a sea of text that pushes forward relevant URLs as the user browses, allowing linking directly to associated content throughout the site.

The remainder of this paper will cover the functionality and design of each of Maitre-D's three components, followed by observations and criticisms. I will conclude with a few indications for further explorations.

2.1 The Search Engine

A modified version Architext Excite 1.0 for WebServers was run on an HP900/750 (highlander.media.mit.edu), logging all local pages of a version of the Interactive Cinema site. After a little configuration, a corpus of 819 pages was indexed and searchable. Parts of the server were pulled from the main program and incorporated into Perl scripts, allowing a string of search-terms to be read, queried, and stored as HTTP Cookies. Cookies allow persistent state to be stored in a particular client, and are passed back to the originating server during communications. This system allows distinct user-interests to be modeled, and incorporated a simulation of user-agent profiles into the behaviour of the site-knowledgeable Maitre-D system. This index of pages was fed back into a Perl program, which, using the search-engine's functionality as a library, created a cross-indexed references to related pages. Each page was processed by Architext's concept-based query engine, to find "conceptually similar" pages. The results of this cross-indexing, along with a summary of each page, was stored in a

database format for use by the recommendation-system.

2.2 Communications Layer

The main page for this system spawns an additional Netscape page, as a control-window, on startup. The page contains two frames: one has an embedded Java applet (the recommendation program), the other a Livescript program. The Livescript "spy" constantly checks the location of the user's main browsing window, communicating the information, as it changes, to a Perl script. The Perl script (doSearch.cgi) reads two Cookies from the browser at this time: "profile", a string of keywords representing user-agent interests, and "session", important keywords gleaned from Maitre-D's own conception of overarching user-interests (Note: both of
the cookies are simply set by another CGI script at this time. There is no sophisticated learning algorithm implemented).

The current URL is processed by a version of an Architext library function, and returned as a collection of concept-keywords. These three components are then sent back into the local site-search, to find what the Excite engine thinks is the most relevant URL (excluding the browsers *current* URL, of course).

This information is then passed back through the LiveScript window for communication to the Java applet.

2.3 Modified NIF Catalog

Mike Murtaugh'96 developed the NIF applet to display related concepts and web-locations as text, showing relative importance through position, colour and saturation values. The original system dealt with a fixed, tailored database, and was entirely self-contained.

The existing applet was modified to accept arbitrary input, working in parallel with a Perl script to generate the necessary concept-relations. As a result, however, a lot of the inherent beauty and clarity of the original display was lost (most screen-sizes just aren't made to show 819 unordered and inter-related concepts of varying importance, simultaneously!)

The Java program reads in a database, and lays out the single-line text-titles of all the pages of the site. The result is a light gray wash of characters, unreadable in its neutral state. At the LiveScript layer's queues, a database-record is pushed to the right and to the foreground, unpacking itself to reveal its

3-line document summary. At the same time, documents directly "related" (as determine by Excite's pre-calculated concept query) are emphasized, documents related to *those* pages become slightly emphasized, etc. As the user browses, the display alters to reflect a web of Maitre-D's recommendations.

3. Analysis

The system, as is, is an unholy combination of stable, portable code, and indeterminately-functional, undependable features. The indexing system, cross-referencing tool, database-creation and per-document search programs are all written in Perl, and are easily configurable for installation on other Unix-based webservers. The Architext server itself, however, needs a separate, platform-specific download; ideally, reliance on the actual shareware product should be removed, or a code-merging utility (to combine the off-the-shelf engine with my modifications) provided.

The Java-applet is relatively stable as is; however, as with many Java applications, its graphical signature and performance is markedly different on a number of platforms. Layout, relative image opacities, and font-sizes vary wildly from Unix to Mac to PC, as do applet loading and initialization. Also, the applet's text-parsing needs to be robust enough to deal with a variety of HTML documents as it displays their summaries.

The LiveScript layer is similarly unreliable. Its functionality is Netscape-specific, runs in indeterminate time, and produces platform-specific bugs. However, to date it's the only way to communicate between multiple browser-windows, CGI scripts and Java applets.

4. Conclusions

Automatic indexing is clearly not the best of all possible recommendations-systems. The suggestions Maitre-D produces aren't the result of any more intelligence than an ordinary web-search might produce (with the possible exception of continuity between searches user's past actions). However, the push-style recommendations seem a useful way of directing the user's attention to the products of a search. Also, the presentation of the recommendations is easier to comprehend, rate and react to that, say, the return of an Alta Vista query. Having a limited, known domain, which the host-server can parse, cross-index and process at its leisure, can allow informed

and meaningful recommendations to be made.

Although the features of the NIF-like display are worthwhile to consider, I don't think the human-interface itself is more than serviceable for a prototype. Multiple-window browsing is confusing, and most users don't have the screen real estate to spare. A less intrusive way of displaying a similarly range of possibilities, where *the style of display* tells you something about the information contained therein, is called for.

Finally, I think there is more to be gained by working a server-log parser into the learning system. Watching how long users spent on pages (without their machines idling), tracing clickpaths, etc, might allow the system to recommend paths through content. Rather than simply recommending single pages, the agent could have a sense of continuity, an experience, which better provides what the user is interested in.

In intend to experiment at least with the communication/interface side of this system over s, and can demo a more stable version at that time.

5. References

Architext Excite 1.0:
http://www.atext.com

NIF Catalog applet: © Michael Murtagh, MIT Media Lab, 1996
http://nif.www.media.mit.edu/ecat

Also see

Lachman, Richard. (1997a). "Experiments in Mapping Character Animation to Computational State", to be published in Workshop on Animated Interface Agents, IJCAI '97. Nagoya, Japan. August.

Lachman, Richard. (1997b). "Experiments in Mapping Character Animation to Computation State" Masters Thesis, Massachusetts Institute of Technology