Building Worldwide Mentoring Tools: Content Analysis of Visited Web Page and Matching

W. Curtiss Priest, Ph.D., Director, Center for Information, Technology & Society
Jared Luxenberg, Center Fellow
August 21, 2002

Presented October 18th, 2002
Association for the Advancement of Computing in Education
E-Learn 2002 World Conference on E-Learning in
Corporate, Government, Healthcare, & Higher Education
Montreal, Canada

Abstract

Peer-to-Peer (P2P) Communications has become increasingly more recognized as a powerful tool for learning. In a project as part of www.eLearningSpace.org we have designed, built and are testing a new tool for P2P. We wished to have online mentors matched to online learners, but, the problems of successful matching, taking into account the current interests of the learner is challenging. Our design is to: 1. Give K-12 learners web-based learning resources 2. The student has a hover button that tracks their journey and says "Find me a Mentor" 3. They press that buttons: a. The contents of the currently visited web page is read b. We interpret the key concepts or topics of the visited page c. We employ a sophisticated matching tool (like the "find similar" for web pages at Google) to get a ranked list of relevant mentors d. Using an intermediate server, we find which of these mentors are online e. We make the match within seconds and place the learner in touch with the mentor via IM

Background

Mentor-Matcher is a knowledge sharing tool. To grasp how it performs knowledge sharing, it is useful to put this tool in perspective of other knowledge sharing tools. Once it is defined in those contexts, the technique for constructing the tool is addressed, followed by how it is currently being piloted.

The history of computer-based knowledge sharing tools goes back to the creation of online databases, such as by Lockheed Dialog, which terminals could connect to via packet networks in the early 70's.

Also created in the early '70's was a knowledge sharing tool called EIES by Murray Turoff at the Stevens Institute of Technology. Called computer teleconferencing, this tool and others like it permitted an asynchronous dialog among a group of participants where, again, using a dial-up terminal, users could dial to a central computer and post notes to a set of discussion topics.

Quite separately, on the UNIX side of computing, a hodgepodge of "news feeds" were created called Usenet. Either using telephone connectivity, directly, or by obtaining a "feed" from a computer networked to other computers, the user would receive a stream of messages sent to the "group" on the local machine. The user could either reply to a message (called threading) or open a new subject of discussion by announcing it in the subject line of the message.

Yet another mode of knowledge sharing was the Bulletin Board System (BBS). Here permanent messages could be posted in "Forums" and some of the first online chat sessions and chat rooms were created using software such as PCBoard. Other knowledge sharing occurred through the exchange of freeware and shareware files, also placed under topic headings (or directories).

In the '80s, with the prevalence of the PC, it was now possible to distribute knowledge sharing systems -- a form of communications which in the '90s was called Peer-to-Peer communications (P2P).

This author developed (and patented) the first P2P knowledge sharing system that combined e-mail communications, a knowledge database, and a topic structure. A topic structure (or controlled vocabulary) need be thought of as no more than a set of directories containing messages sent in the "context" of the topic.

A later implementation of such a P2P knowledge sharing system was Microsoft's Outlook which permitted the user to exchange information within "categories." Like the Priest implementation, both the category and the information were bundled into a header, a header different from a typical e-mail header.

With the advent of the Internet (net), a wholly different form of knowledge sharing grew, which actually resembled the original online database format more than later P2P implementations.

The net permitted thousands of different sites to flourish, and each contained knowledge of one kind or another. Early pioneers such as Yahoo and Lycos saw the need to comb these sites for information and provide a portal to an index of this information via web-based search engines. The technique behind the scene was the "webcrawler" which gathered words from site after site, stuffing them into a single searchable index. So these tools resembled the early text database tools of the '70s except that the "databases" were much more widely distributed.

On the net, the BBS chat rooms gave way, first to IRC (Internet Relay Chat) and then to P2P chat such as AOL Instant Messenger.

Mentor-Matcher Design

In a prior project we had designed a desktop agent (Tracker) that followed K-12 students using the web for learning resources. In that 81% of a student's time is spent outside of school (P. Kenneth Komoski, "The 81 Percent Solution," ***Education Week***, January 26, 1994, p. 52), the goal was to provide a way to encourage learners to use the net to learn more. And, hopefully, the more passive activity of watching around six hours of television a day might be reduced and replaced with something more constructive.

A learning resource portal site was constructed (http://www.eLearningSpace.org aka eLs) with support from the U.S. Department of Education. Students receive a "learning credit" for each hour spent learning at some 4000 identified resource sites and can exchange those credits for donated goods (such as brand name shoes, etc.) via an online auction site (much like eBay).

The time was right to combine knowledge sharing tools in a way that merged the abilities of web-based search engines and instant messaging.

The design problem was "how to elicit mentoring strengths of online mentors and match those abilities with the current learning needs of the learners "

The national standards efforts had produced a set of topics and sub-topics in the area of math, science, and reading. So, using Javascript we built a topic/sub-topic list where sub-topics and sub-sub-topics can be display by clicking on a plus sign next to the higher topic -- a technique used in Acrobat and other indexed lists.

This approach works well because mentors have a well defined set of strengths that do not change much over time. However, a student's learning interest changes by the hour.

Somehow we needed to make a match between a topically defined person -- the mentor, and a learner who is selecting and viewing web resources.

This problem is similar to the web search engine problem of finding web pages similar to the currently viewed web page. Ah, but the mentor topics were stored in a database; how to make these behave like web pages?

To solve this problem we create a set of local web pages with a key for each page back to each mentor. But how to populate the pages with lots of words?

Since several encyclopedias are online, we could build a tool that "expanded" a knowledge topic -- say "planets" -- into a web page of up to 30,000 words discussing planets. To do this we built an intermediate tool called "WordScooper."

After the mentor signs on to the project and provides a set of topics and/or sub-topics, the server immediately has WordScooper query Microsoft's Encarta for words on that subject. We created rules where WordScooper behaves a bit like a webcrawler, taking words from various levels of encyclopedia pages by following the links on the top page. We ignore pages that are not subordinate to the top web page.

So, now we have reduced the design problem to trying to match the learner's currently viewed web page (an indicator of their knowledge needs) with our locally stored set of web pages (which are keyed back to the mentors).

Knowing that there are several dozen web search engines and that many of these are available in several ways to search one's own web pages, we went to http://kresch.com/search/search.htm which lists all known search engines and their features. While the feature of "find similar" was **not** in their summary of features, they **did** have active links to sample sites employing the particular engine. Using this we reduced the list to four possibilities. Of these, the engine by Thunderstone (Webinator) appeared to be widely used by colleges and universities and, best of all, one could obtain the search engine software and run it under the local server's OS (in our case NT5). This is free, provided the server can access the Thunderstone site to receive authorization to run. [Alternatively, the entire text-based, SQL featured search engine (called Texis) can be purchased for $11,000 (U.S.).]

The matching and messaging process operates as this: 1.) Webinator is instructed to bring down the web page currently viewed by the learner and convert the page to plain text; 2.) Webinator is instructed to use its "metamorph" facility to create an internal set of thirty frequent and unique words which define the content of the web page; 3.) a search engine query is submitted with that list, searching all of our local web pages of mentor-related word-expanded topics; 4.) a rank order list of relevant mentors is returned; 5.) an intermediate server of ours determines which mentors are currently online, and displays a list to the learner of online mentors and the word that caused the match -- each item in the list is a "link" to each online mentor; 6.) the learner clicks on one of the links; our online AOL-IM-like server sends a message to the mentor including the student's name and a display of the URL the student is currently viewing; the mentor may accept or refuse the "assignment;" if he "yes"," our intermediate server sends a message of acceptance to the learner and a two-way chat occurs; 7.) we log the entire chat session and its duration so that the learner can receive learning credits (as above) by verifying that the sites are on the list of educational web pages approved by eLs and logging the minutes in the student's eLs portfolio. The data gained by tracking the student's web browsing had to be detailed (keystrokes, etc.) and be in a form that was easily accessible from ASP scripts.

In first researching tools, we wished to use client side Javascript to track web browsing. This approach presented some control issues. Netscape would not allow tracking outside of the domain the Javascript page was hosted on, and some of the features of the script did not work properly in Internet Explorer (IE). Since Javascript implementations vary from browser to browser, and some users have Javascript turned off, this solution was not feasible. We looked for other ways to track web surfing. We looked at server-side solutions, such as a backwards proxy, but the overhead associated with these out weighed their benefits. We eventually found a way to automate IE on the client side, eliminating the need to run Internet traffic through a central server.

Microsoft provides an ActiveX interface to launch an instance of IE. Events are fired when the user visits URLs, closes the window, reloads the page, etc. This allows total control over the browser, and it provided the degree of control we needed.*
The field test stage has begun, and the tool has met and exceeded all expectations for performance.

In summary, the project illustrates a useful way to combine two different ways of accessing knowledge: 1.) use of keywords or a controlled vocabulary, and 2.) full-text searches. The choice of one and/or the other relates to the ability or inability to use a controlled vocabulary as pieces of this design problem have demonstrated.

---

* The authors express their concern that all roads seem to lead to IE and consider the recent decision by the U.S. Justice Department suit to be an inadequate and myopic decision against Microsoft in this regard.