Exploring the Web with RECONNAISSANCE AGENTS

Why surf alone? New agent technology can scout out the online terrain and recommend the best paths for you to follow.

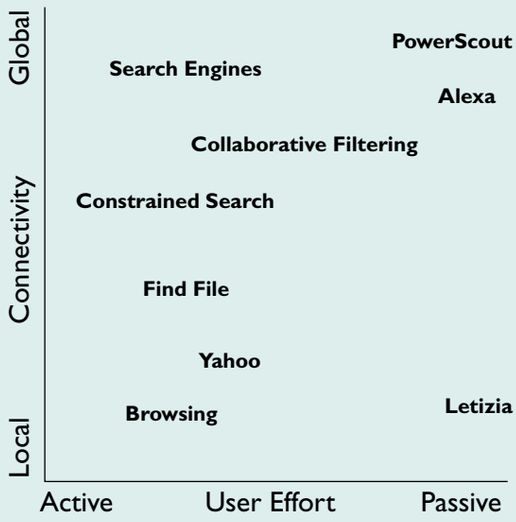
EVERY CLICK ON A LINK IS A LEAP OF FAITH. WHEN you click on the blue underlined text or on a picture on a Web page, there is always a (sometimes much too long) moment of suspense when you are waiting for the page to load. Until you actually see what is behind the link, you don't know whether it will lead to the reward of another interesting page, to the disappointment of a junk page, or worse, to a "404 Not Found" error message.

But what if you had an assistant always looking ahead for you—clicking on the Web links and

Henry Lieberman, Christopher Fry, and Louis Weitzman

CAREN ROSENBLATT

Figure 1. The two dimensions of using agents based on amount of user effort and the connectivity of the data.



checking out the page behind the link before you even get to it? An assistant that, like a good secretary, has a good idea of what you might like. The assistant could warn you if the page was irrelevant or alert you if that link or some other link merited your attention. The assistant could save you time and frustration. The function of such an assistant represents a new category of computer agents that will soon be as common as search engines in browsing assistance on the Web and in large databases and hypermedia networks.

These agents are called *reconnaissance agents*—programs that look ahead in the user's browsing activities and act as an advance scout to save the user needless searching and recommend the best paths to follow. Reconnaissance agents are also among the first representatives of a new class of computer applications—learning agents that infer user preferences and interests by tracking interactions between the user and the machine over the long term.

We'll provide two examples of reconnaissance agents: Letizia and Powerscout. The main difference is that Letizia uses local reconnaissance—searching the neighborhood of the current page, while Powerscout uses global reconnaissance—making use of a traditional search engine to search the Web in general. Both learn user preferences from watching the user's browsing, and both provide continuous, real-time display of recommendations. Reconnaissance agents treat Web browsing as a cooperative search activity between the human user and the computer agent, providing a middle ground between narrowly targeted retrieval that search engines typically provide and completely unconstrained manual browsing.

One description of this landscape of systems organizes them around two axes, one characterizing reconnaissance connectivity (local vs. global) and one characterizing user effort (active vs. passive). Local reconnaissance traces links locally, while global reconnaissance uses global repositories such as search engines. Figure 1 plots a number of tools and agents against these attributes. Typical file browsing is in the lower-left quadrant while standard search engines are located in the upper-left quadrant.

The One-Input Interface. Newcomers to the Web often complain they have trouble using search engines because they feel the interface to a search engine is too complicated. The first time I heard this complaint, I was astonished.

What could possibly be simpler than the interface to a search engine? All you get is a simple box for text entry, you type in anything you want, and then hit "go." (Of course, today's search engines aren't really that simple since they tend to contain advertisements, subject catalogs, and many other features. But the essential functionality lies in the simple query box.) How could you simplify this interface any further?

When you think about the task faced by the new user, however, the complexity becomes apparent. In the user's mind is a complex blend of considerations. They may be looking for something very specific; they may be just interested in generally learning about a given subject. How specific should they be in describing their interests? Should they use a single word, or is it better or worse to type in multiple words? Should they bother learning the advanced query syntax? How should they choose between the myriad search engines, and compare their often-inconsistent syntax and behavior [7]? Would they be better off tracing through the subject catalog rather than the search engines, since most portal sites now offer both? And, regardless of what they choose to type in the search box, they are deluged with a torrent of (aptly named) "hits," and are then faced with the problem of how to determine which, if any, is actually of interest. No wonder they get confused.

The Zero-Input Interface. The only thing simpler than an interface with only one input is an interface that takes no input at all! With a zero-input interface, the computer could already know a great deal about what you're interested in before you enter a word into that search engine. You've already given it the information it needs—the problem is the computer threw that information away. Why do you need to keep telling a system about your interests when it should already know?

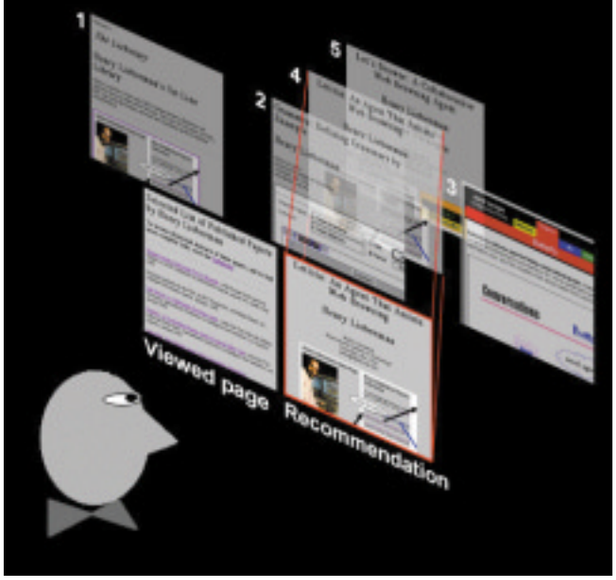
Present-day computers throw away valuable his-

WHAT IF YOU HAD AN ASSISTANT ALWAYS LOOKING AHEAD FOR YOU—clicking on the Web links and checking out the page behind the link before you even get to it? The assistant could warn you if the page was irrelevant or alert you if that link or some other link merited your attention. **THE ASSISTANT COULD SAVE YOU TIME AND FRUSTRATION.**

tory information. Every time you click on a link in a browser, that's an expression of interest in the subject of the link, and hopefully, the subject of the page the link points to. If a person were watching your browsing activity, they would soon have a good idea of what subjects you were interested in. Unfortunately, the only use the browser currently makes of that expression of interest is to fetch the next page. What if the computer kept that information, and over time, used it to learn what you might or might not be interested in? Browsing history, after all, is a rich source of input about your interests.

Systems that track past user behavior and use it to predict future interest are a new form of zero-input interface. Even though they need no input in the sense they do not require explicit interaction themselves, they repurpose input you supply to the computer for other reasons. Such interfaces are rare now because each application has its own self-contained interface and applications cannot reuse input from other applications or keep any significant histories of interaction.

Figure 2. Letizia spirals out from the user's current page, filtering pages through the user's interest profile.



Letizia

If I were with a friend and watching him or her browsing on the Web, I'd soon learn which pages were likely to attract my friend's interest. Given enough time, I might even become pretty good at predicting which link my friend would be likely to choose next. If my friend viewed a site for the first time that I had previously explored thoroughly, I might even be in a position to give a better recommendation to my friend as to what might satisfy his or her interest. And all this could occur without my friend explicitly telling me what those interests were.

Letizia [4, 5] is a zero-input software agent that automates this kind of interaction. It learns a profile of the user's interests by recording and analyzing the user's browsing activity in real time, and provides a continuous stream of recommendations of Web pages.

Browsing with Letizia is a cooperative activity between the human user and the Letizia software agent. The task of exploring the Web is divided between the human, who is good at looking at pages and deciding what to view next, and Letizia, which uses the computer's search power to automate exploration.

Letizia runs simultaneously with Netscape, and determines a list of weighted keywords that represents the subject of the page, similar to the kind of analysis done by search engines. Over time, it learns a profile of the user's interests. Letizia simply uses Netscape as its interface, with one window dedicated to user browsing, and one or more additional windows showing recommendations continuously.

When the user is actively browsing and attention is focused on the current page, the user need not pay any attention to the agent. However, if the user is unsure where to go next, or dissatisfied with the current offerings, he or she can glance over to the recommendation window to look at Letizia's suggestions. Netscape's usual history mechanism allows easy access to past recommendations, just in case the user missed an interesting page when it first appeared.

During the time the user spends looking at a page,

Letizia conducts a search in the local neighborhood surrounding that page. It traces links from the original page, spiraling out to pages one link away, then two links away, and so on, for as long as the user stays on the same page. The moment the user switches pages, Letizia drops the current search and initiates a new search starting from the page the user is currently viewing (see Figure 2). We call this process "reconnaissance" because, like military reconnaissance, Letizia scouts out new territory before the user commits to entering it.

This leads to a higher degree of relevance than search engines. Current search engines are just big bags of pages, taking no account of the connectivity between pages. Because two pages are connected on the Web only when some author thought a user viewing one might want to view the other, that connection is a good indication of relevance. Thus the local neighborhood of a page, obtained by tracing a small number of links from the originating page, is a good approximation to the semantic neighborhood of the page.

If I'm looking for a place to eat lunch and I like Indian food, it's not a very good idea to type "Indian food" to a search engine—I'm likely to get wonderful Indian restaurants, but they might be in New Delhi and Mumbai. What I want is the intersection of my interests and what's in the neighborhood. If the idea of geographic neighborhood is replaced by the idea of semantic neighborhood on the Web, that intersection is what Letizia provides.

Further work on Letizia is concerned with tracking and understanding users' patterns of Web browsing. In one experiment, student Aileen Tang added a real-time display of the agent's tracking of the user's interests. As the user delves deeper into a subject, we observed that the interest function steadily increases, only to dive abruptly as the user changes topics (see Figure 3). Watching such a display lets the user know when he or she is getting off track. Letizia can segment the history into topic-coherent subsessions, and we can detect browsing patterns such as tentative explorations that don't work out, or switching back and forth between two related topics.

Letizia's local reconnaissance is great when the most relevant pages of interest to you are nearby in link space. But what if the best pages are really far away?

Figure 3. Letizia's real-time display of user interests. The graph on the lower right climbs while the user keeps browsing on a single topic (here "Programming by Example"). It dips abruptly when the user switches topics.

Powerscout

Letizia takes advantage of the user's behavior to create a zero-input program that finds pages interesting to the user. But there are a number of zero-input resources that Letizia does not exploit. Letizia only scouts pages close to the current page and can only hope to examine tens of pages, while hundreds of millions of pages exist on the Web, any one of which might be relevant. Users have many interests over time and Letizia only focuses on the current page that the user is viewing. Also, users have a rich browsing history over potentially many months that can be exploited to better understand their true interests. PowerScout is another zero-input reconnaissance agent whose vision addresses some of these more general issues.

PowerScout is a different kind of reconnaissance agent. While it, too, watches you browse and provides a continuous display of recommendations, it uses a different strategy to obtain and display those recommendations. Like people, agents need not surf the Web alone, and PowerScout uses a familiar companion—a conventional search engine—to support its global reconnaissance technique. PowerScout uses its model of the user's interests to compose a complex query to a search engine, and sends that query while the user continues to browse (see Figure 5). If we think of a search engine as being a very simple sort of agent itself (and the more complex among the search engines do have some agent-like features), PowerScout represents an example of how agents with different capabilities can cooperate. By using global

search engines to find documents in the semantic neighborhood of the users current document, the system is performing a different class of browsing—*concept browsing.*

PowerScout introduced the term "concept browsing" to emphasize the idea of browsing links not specified by a document's author, but nonetheless semantically relevant to the document being viewed. This auxiliary set of links may have been overlooked or unknown to the author, or might not have even existed when the page was created.

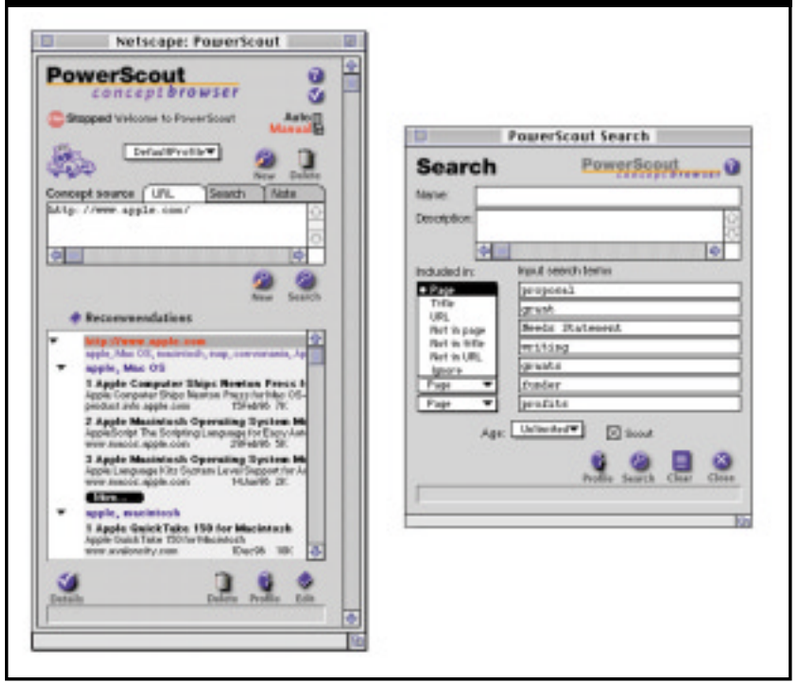
The PowerScout reconnaissance engine formulates the concepts by extracting keywords from the current page. These concepts can be used directly or influenced by user-declared long-term interests that we call "profiles." Figure 4 shows the user viewing a page in the browser on the left while the results of a search is displayed in the PowerScout window on the right. The results are grouped by the concepts used to find them. In this example, the recommendations are organized under the concept of "proposal writing."

Each concept represents a slightly different slant on what is important on the page. PowerScout does the best it can, but it can't read the user's mind. However, the user can quickly look at the concepts and decide which, if any, are important. Beneath each concept is a short list of page hits with summaries. Clicking on the title of a page brings the full page into the browser.

terms used in explicit searches, word frequency, and multiword terms. The details are beyond the scope of this article, but these techniques are the foundation for accurately characterizing a page.

Only one of the profiles is the current profile at any time. By clicking on the "Add to Profile" button, the significant terms of the current page are merged into the term set of the current profile. This process extends the profile with one click. We like to think of it as focusing the profile to match the user's real interest more closely. Both the user's long-term interest

Figure 4. Powerscout's screen. The list of recommendations on the left is grouped by concept and dynamically updated. On the right, Powerscout's search dialogue provides fine control.



Profiles

A profile represents a user's interest in a particular area. The user may create as many profiles as needed to characterize interests. Each profile is made up of a set of ordered terms and a research notebook. The terms are words extracted from a number of different sources including Web pages, explicit searches, and text from documents and email messages. The Research Notebook, as we will discuss, provides the user a way to access and organize the source of the terms in the profile, such as the original URLs, searches, and text.

PowerScout uses a variety of heuristics for extracting and scoring keywords, such as word position, utilization of HTML markup, emphasis of (modeled by the current profile) and short-term interest (modeled by the current page) affect what pages PowerScout recommends.

By keeping areas of interest separated into distinct profiles, one area of interest doesn't pollute another. You may, for example, be interested in computer music and New Orleans architecture, but not care about computer architecture or New Orleans music. When you're browsing for computer music, you don't want to be distracted by articles on New Orleans architecture even though later that day, the converse may be true.

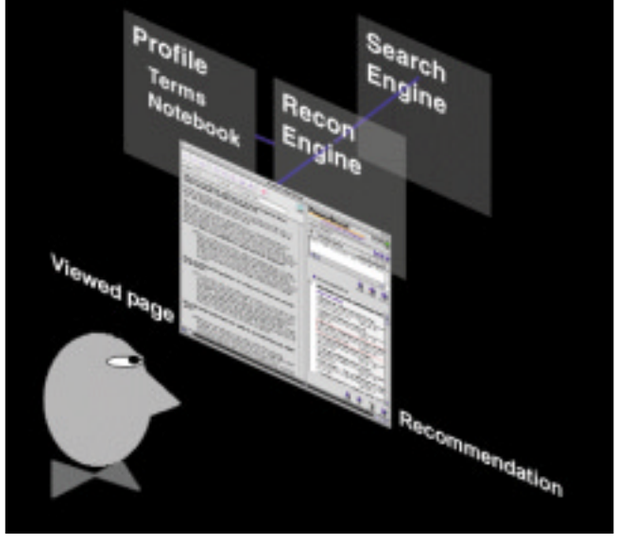
Unlike Letizia, PowerScout does not follow links on the page to get candidates for recommendations. Rather it presents complex queries to a traditional

search engine, such as AltaVista. By automatically constructing queries, PowerScout employs the zero-input principle to perform multiterm, complex queries to search engines without bothering the user with the query language's syntax.

Concept browsing is actually an iterative process of multiple queries based on the current page. Initially, the queries are very specific and exacting. If few results are returned successive iterations relax the constraints on the search. These latter searches are guaranteed to retrieve some results, but will be less relevant than the initial queries. Each query uses several terms, combined in various patterns of "ands" and "ors." There are significant design tradeoffs between the quality of recommendations and the time required to return those recommendations.

PowerScout's Search window complements its automatic reconnaissance. The search window allows the user to manually refine automatically constructed queries or easily construct a new query. This search dialogue box, illustrated in Figure 4, contains a text-edit field for each of seven terms. When the user requests the dialogue box be displayed, it is preloaded with the seven most significant terms from the page being reviewed. The user may then edit the terms as well as indicate where the search engine should look for them (that is, in page's title, URL, body, or temporarily ignored items shown in the expanded pop-up menu on the left). When the user clicks the search button, the complex query is generated and sent to the search engine of choice.

Figure 5. While the user views a page in the browser, PowerScout does global reconnaissance in the background by performing complex queries to a search engine.



Concept Summaries

Unlike Letizia, PowerScout does not automatically display the top recommended page. In fact, there is no single top recommended page. Rather, PowerScout displays a dynamically changing list of recommendations, so the user can get an overview of what's recommended. This is more appropriate to search engines' all-at-once interaction, rather than Letizia's incremental search.

PowerScout's list of concepts and the related page summaries are just like an auxiliary set of related links that the original author of the current page might or might not have included on the page. The recommendations are clustered in related groups by concept. Details of a recommendation can be hidden or expanded while the user reviews the search results. Figure 4 shows the expanded details in the recommendations list.

Often the top recommendation of any given algorithm is not the best for the user's needs. Sorting, categorizing, and summarizing a set of recommended pages is often needed to find the most relevant recommendations.

There is valuable information saved in the Research Notebook of Powerscout's profile that can become a tool in the larger task of collecting and storing information. There are three data types: URLs, searches, and notes. When the user adds a page to the current profile, not only are the significant keywords merged into the profile's terms, but the URL is recorded as well. The user can see these URLs and use them like a folder of bookmarks.

Refining an automatic query results in an explicit search, which can also be saved in a profile. PowerScout's search dialogue box assists the user in making complex queries without having to know the syntax of the search engine. The user can rerun saved searches, such as a periodic query to see what's new on a given topic.

The user may also create notes of text to add to a profile. The full text of a note is kept for later viewing, and the significant keywords in the text are merged into the profile's terms. This is particularly useful when the user wishes to add just a paragraph of a page or perhaps the text of an email message from a colleague on the subject.

Profiles are useful as bookmarks for URLs, searches, and notes, but they can also be valuable

when shared with others. A profile is stored as a file, so it can be easily shared with colleagues. By inspecting someone else's research notebook on a topic, you have access to their URLs, searches, and notes. In addition, you can effectively concept-browse the Web with the interests of the profile's author.

Related Work

The space of search engines and other tools for navigation of the Web is rapidly expanding, and it isn't possible to cover them all here. While search engines could be considered first-generation Web-browsing agents, second-generation agents are becoming more sophisticated in their analysis and classification of documents. Some now use popularity of pages, such as ParaSite [8], and/or analysis of link structure patterns, such as IBM Clever [2]. But most still rely on the conventional paradigm of the user explicitly querying a centralized repository that indexes and ranks documents independent of the individual user. We see the future trend toward navigation assistants more personalized toward individual users' interests and dynamic user behavior. That's where reconnaissance agents come in.

Perhaps the most widespread example of a reconnaissance agent at present is Alexa (www.alexa.com), the service behind Netscape's "What's Related" option. Alexa performs reconnaissance in tracking user-browsing history, and uses collaborative filtering to recommend new pages. Collaborative filtering matches the behavior of each user with other users whose browsing pattern fits most closely, and returns what they looked at subsequently as its predictions. Thus Alexa doesn't have to understand the content of pages, which is both its strength and its weakness.

Northwestern University's Watson [1] also uses tracking user behavior and automatically generated queries to a search engine to recommend pages—as does Powerscout. In addition, Watson has other interesting capabilities, such as searching for pictures or searching for contrasting information instead of that most similar.

Conclusion

Both the Letizia and the PowerScout style of agent have their advantages and disadvantages. Letizia finds pages on a site of interest that the user might overlook because of the complexity of the site's navigation, while PowerScout's global reconnaissance is better at retrieving faraway pages. PowerScout may also work better over a low-bandwidth line, since most of the bandwidth requirements have effectively been absorbed in advance by the search engine.

Letizia's interface design goal was to keep the interaction as minimal as possible, and so it does not show the user directly its profile of the user's interests or its analysis of Web pages. It does not provide any direct means to tell the system what the user is interested in. PowerScout, on the other hand, does provides an extensive interface for its user model and lets the user specify with precision what he or she is or isn't interested in.

As of this writing, the Web consists of an estimated 800 million pages [3]. This wealth of information becomes useful only if we can find what we need. Web users are currently forced to choose between browsing and searching. Both are great tools, yet each limits the process of finding relevant information. A new generation of tools—reconnaissance agents—automatically search while you browse. This permits users to maintain a focus on their quest for information while decreasing the time and frustration of finding material of interest. This blending of browsing and searching empowers users to focus on their task while engaging the system to do what it does best—analyzing, storing, and retrieving relevant information. C

References

1. Budzik, J., Hammond, K.J., Marlow, C.A., and Scheinkman, A. Anticipating information needs: Everyday applications as interfaces to Internet information sources. In *Proceedings of the 1998 World Conference on the WWW, Internet, and Intranet.*
2. Chakrabarti, S., Dom, B., Gibson, D., Kleinberg, J., Raghavan, P., and Rajagopalan, S. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th World-Wide Web Conference.* (1998).
3. Lawrence, S., and Giles, L. Accessibility and distribution of information on the Web. *Nature 400.* (1999), 107–109.
4. Lieberman, H. Letizia: An agent that assists Web browsing. In *Proceedings of the International Joint Conference on Artificial Intelligence IJCAI-95.* (Montreal, Aug. 1995).
5. Lieberman, H. Autonomous interface agents. In *Proceedings of the ACM Conference on Computers and Human Interface.* (Atlanta, May 1997).
6. Salton, G. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer.* Addison Wesley, 1989.
7. Shneiderman, B., Byrd, D., Croft, W.B. Clarifying search: A user-interface framework for text searches. *D-Lib Magazine* (Jan. 1997).
8. Spertus, E. ParaSite: Mining structural information on the Web. In *Proceedings of the Sixth World Wide Web Conference.* (Santa Clara, Calif., 1997).

Henry Lieberman (lieber@media.mit.edu) is a research scientist in the Software Agents Group at the Massachusetts Institute of Technology Media Lab in Cambridge, MA.

Christopher Fry (cfry@shore.net) is a language architect at Glueworks, Lexington, MA.

Louis Weitzman (louisw@us.ibm.com) is a senior software engineer at the IBM Internet Technology Group in Cambridge, MA.

Lieberman's research was supported by the Digital Life Consortium and the News in the Future Consortium, and other sponsors of the MIT Media Laboratory.