Autonomous Interface Agents

Henry Lieberman
Media Laboratory
Massachusetts Institute of Technology
Cambridge, Mass 02139 USA.
+1 617 253 0315
lieber@media.mit.edu

ABSTRACT

Two branches of the trend towards "agents" that are gaining currency are *interface agents*, software that actively assists a user in operating an interactive interface, and *autonomous agents*, software that takes action without user intervention and operates concurrently, either while the user is idle or taking other actions. These two branches are related, but not identical, and are often lumped together under the single term "agent". Much agent work can be classified as either being an interface agent, but not autonomous, or as an autonomous agent, but not operating directly in the interface. We show why it is important to have agents that are both interface agents *and* autonomous agents. We explore some design principles for such agents, and illustrate these principles with a description of Letizia, an autonomous interface agent that makes real-time suggestions for Web pages that a user might be interested in browsing.

Keywords

Agents, interface agents, autonomous agents, Web, browsing, search, learning.

INTRODUCTION

The definition of an *agent* is the subject of much controversy in the field of Human-Computer Interaction. While it is not our purpose here to settle this question, we are interested in the relationship between two particular aspects of agents. First, that the agent operate *in the interface*, as opposed to in the background or "back end" of an application. Second, that the agent *act autonomously*, as opposed to having a sequential conversation with the user. Often, an agent will satisfy one or the other of these characteristics, but it is rare that it will exhibit both at once.

Traditional interface design is oriented toward *conversational* interfaces, where the user and the agent "take turns" acting. Autonomous interface agents lead to a somewhat different design style, brought on by the possibility that the agent may need to interact with the interface while the user is also interacting with the interface. The user may or may not be aware of the agent's activities at any given moment. This design style has its own considerations and tradeoffs, and deserves more attention in the interface design field.

We'll start out by making some general remarks on the controversial question of agentry in interfaces, and the relationship between interface agents and autonomy. In order to illustrate how these principles play out in a real application, we will present Letizia, an autonomous interface agent for browsing the Web. We'll follow the description of Letizia with a discussion of some interface design principles which emerged from our experience with this application.

Agents and assistance

Let's get the "What is an agent?" question out of the way first. My own personal view is that an agent is any program that can be considered by the user to be acting as an assistant or helper, rather than as a tool in the manner of a conventional direct-manipulation interface. An agent should display some [but perhaps not all] of the characteristics that we associate with human intelligence: learning, inference, adaptability, independence, creativity, etc. The user can be said to delegate a task to an agent rather than command the agent to perform the task [15]. What exactly constitutes an "assistant" in any given case may be, of course, a matter for some debate.

I believe that it is the user's view of how the software is acting that counts, so the same interface may properly be viewed as an agent by one person and as a tool by another. Nevertheless, people will tend to agree on whether a program is an agent often enough that such a distinction is useful. One can take an "agential stance" towards a program in the same way as Daniel Dennett [5] proposes a "intentional stance" as a criteria for deciding whether a program is "intelligent". I realize that this view is controversial, and that several influential interface researchers have argued strongly against the desirability of such agents [12].

Having an agent operate directly in the user interface rather than as a “background” or “back-end” process increases the extent to which the user will perceive the software as acting like an assistant. If the user perceives the agent’s actions as actions “that I could have done myself”, the user is more willing to conceptualize the agent in the role of an assistant. Autonomy also increases the feeling of assistance, as we often delegate tasks, not because we cannot do them ourselves, but because we simply do not want to spend the time.

Interface Agents

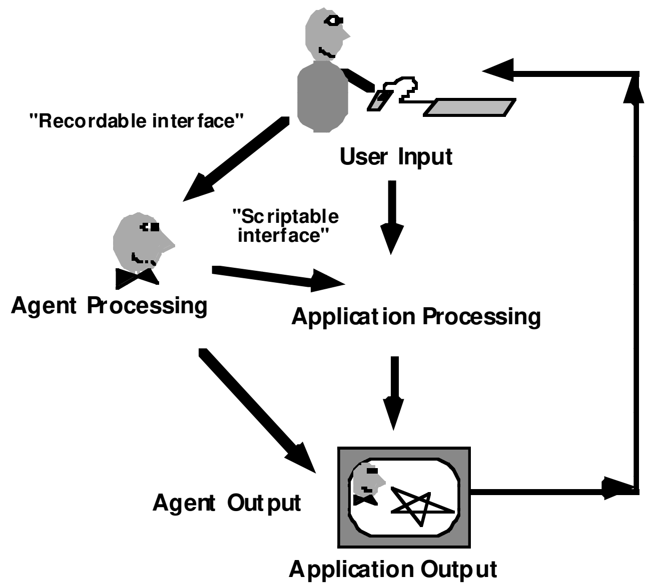
Direct manipulation graphical interfaces display visual representations of physical or conceptual objects, and allow the user to issue commands that change the state of the objects. In a direct-manipulation interface, the changes to the display state are more or less one-to-one with the commands explicitly invoked by the user.

We’ll define an *interface agent* to be a program that can also affect the objects in a direct manipulation interface, but without explicit instruction from the user. The interface agent reads input that the user presents to the interface, and it can make changes to the objects the user sees on the screen, though not necessarily one-to-one with user actions. The agent may observe many user inputs, over a long period of time, before deciding to take a single action, or a single user input may launch a series of actions on the part of the agent, again, possibly over an extended period of time.

An interface agent could be considered to be a "robot" whose sensors and effectors are the input and output capabilities of the interface, and for that reason are sometimes also referred to as "softbots" [7]. Sometimes the interface agent is actually represented anthropomorphically as a face on the screen, such as in the Apple film Knowledge Navigator.

The best-known examples of interface agents are intelligent tutoring systems and context-sensitive help systems. [20] is a good example. In such systems, the user may operate the interface with complete disregard for the agent, but, if called upon, the agent may also display suggestions, or perform direct-manipulation actions on objects in the displayed interface, based on input implicitly collected from the user. Other kinds of interface agents may critique the user's behavior [8], or augment the user's direct-manipulation actions with extra computed information that the user may find helpful [9].

Interface agents are becoming more and more attractive due to the increasing complexity of user interfaces and the tasks to which they are applied. The current growth rate of interactive interfaces [menu operations and other interface choices added per year] is unsustainable [14]. If we insist on maintaining the one-to-one correspondence between interface actions and capabilities of the user interface, the hallmark of direct-manipulation interfaces, we will soon reach the point where no more functionality can be added to our systems. Interface agents provide a way out of this dilemma.



Structure of an interface agent application

Autonomous Agents

An *autonomous agent* is an agent program that operates in parallel with the user. Autonomy says that the agent is, conceptually at least, always running. The agent may discover a condition that might interest the user and independently decide to notify him or her. The agent may remain active based on previous input long after the user has issued other commands or has even turned the computer off.

Why autonomous agents? An assistant may not be of much practical help if he or she needs very explicit instruction all the time and constant supervision while carrying out actions. Assistants can be time-savers when they are allowed to act independently and concurrently. Allowing an interface agent to run off-line and in parallel with the user directing attention to other activities enables the user to truly delegate tasks to the agent.

By contrast, a traditional command-line or menu-driven interface is *conversational*. The user performs input, enters it, the system accepts the input, computes some action, displays the result, and waits for the next input. This has the property that the system is doing nothing while the user is preparing the input, and the user is doing nothing in the interface [except maybe type-ahead] while the system is running. Every action of the system must be explicitly initiated by the user.

Autonomous Interface Agents

It is clear that an agent may exhibit each of these characteristics independently. An interface agent may observe user interface actions and make changes to user interface objects displayed on the screen without having the capability to continue running in parallel with the user. The intelligent tutoring systems and critiquing systems mentioned earlier [8, 9, 14] fall into this category. They may not have any need to independently notify users of external conditions or run while the computer is off.

An agent may have the capability to run autonomously without needing to interact through graphical interface operations [except in the trivial sense in which displaying any text to the user is part of a user interface]. For example, many programs will send you e-mail when there is an update to Web pages you have marked as interesting; these agents are autonomous, but operate outside the user interface. Telescript and Java are only “agent languages” in the sense that they are suited to programming asynchronous, distributed activity. Though these agents may act autonomously off-line, the agent does not operate the interface itself. In 1997, there are even two separate conferences, one for Autonomous Agents and one for Intelligent User Interfaces, within a month of each other!

In order for an agent to be considered both an interface agent and autonomous, it follows that there must be some part of the interface that the agent must operate in an autonomous fashion. The user must be able to directly observe autonomous actions of the agent and the agent must be able to observe actions taken autonomously by the user in the interface. Concretely, the user will see interface elements that appear to "move by themselves" in response to input that the agent appeared to "see for itself" rather than having been explicitly instructed.

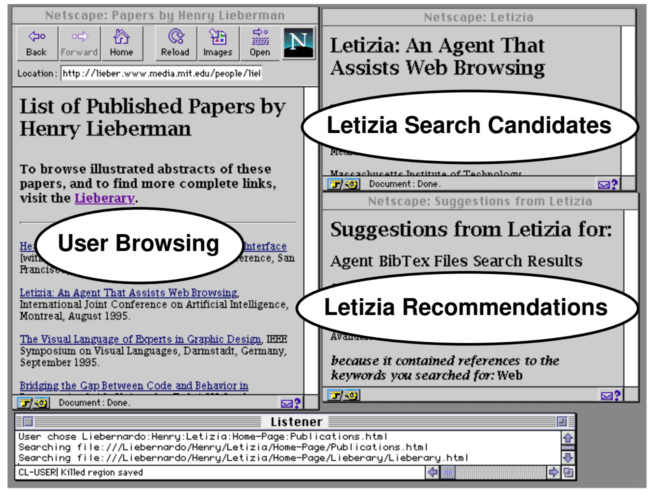
Letizia: An autonomous interface agent for Web browsing

The Web is a domain that is well suited for autonomous interface agents. Web users now feel a crying need for some sort of intelligent assistance, since the direct-manipulation interface of manually following links in a browser is not sufficient to prevent the feeling of being drowned in irrelevant information.

Letizia is an autonomous interface agent that treats search through the Web space as a continuous, cooperative venture between the user and a computer search agent. Letizia records the URLs chosen by the user and reads the pages to compile a profile of the user's interests. A simple keyword-frequency information retrieval measure is used to analyze pages. Letizia is always active, searching the Web space that is "nearby" the user's current position, in parallel with the user's browsing activity. Letizia then uses Netscape's own interface to present its results, using an independent window in which the agent browses pages thought likely to interest the user. The following sections will explain Letizia in more detail, and some of the rationale for its design.

Letizia "channel surfs" the Web

Letizia presents its results to the user continuously, using a kind of "channel surfing" interface style. "Channel surfing" is an American term for a kind of temporal browsing typically performed by television viewers. It refers to the process of sampling information sources for short intervals in the search for something of interest to the viewer. Letizia channel surfs by keeping one or more windows continuously displaying pages that it thinks might be of interest to the user, for short intervals.



Letizia's default screen layout

Letizia's usual interface consists of three Netscape windows. By default, the left-hand window is reserved for user browsing activity. The user may browse in this window in a normal manner, and can completely ignore any agent activity. The two windows on the right side are under control of the agent. The top window displays search candidates, those pages which Letizia is considering to recommend to the user. The bottom right window displays those pages that Letizia actually decides to recommend to the user, passing Letizia's tests for user interest. The user may choose to continue browsing with either his or her own selected pages, or Letizia's suggestions, at any moment in time.

Letizia notices which window is being controlled by the user and only uses windows that are not being actively used by the user. Accepting a suggestion from Letizia simply consists of switching to a window that contains a page recommended by Letizia, and continuing to browse. The user may, of course, save recommended pages on a hotlist or otherwise use them as usual in Netscape.

Letizia is not limited to this 3-window configuration. In practice, a less obtrusive way of running Letizia is just to use two windows, a user window and one Letizia window. Displaying the search candidates is not strictly necessary, but is often useful for demonstration and debugging purposes. Displaying more than two agent windows is also possible, for example, where the two windows could each be working off different user profiles.

Letizia's autonomous browsing vs. conversational search engines

Many of the facilities that are presently referred to as "agents" on the Web are some kind of search engine. A fundamental problem with any kind of search engine is the sequential nature of the interaction. The query interface of a search engine is conversational -- the user asks a question, then waits for the answer. After the answer is received, the user continues the browsing process, either accepting the recommendations of the agent, or doing unrelated browsing. Any other interaction with the agent requires a new search query. The value of a query-based search interface must be balanced against the cost in terms of time it takes to interact with the search agent which could have been spent browsing instead. The consequence of this sequential structure is that either the user is actively working, or the agent is actively working, but never both at the same time.

In autonomous agents like Letizia the view of Web browsing as query-based information retrieval is replaced by a view of Web browsing as a real-time activity. The goal is not to retrieve the "best answer" in any absolute sense, but to make the best use of the most limited and valuable resource -- the attention of the user.

Having a continuously-running agent means that the agent can be bolder in its recommendations to the user. It need not make a perfectly accurate judgment as to which document best meets the user's interest, since it is likely to get another chance to do better at some future time, if the user continues to browse in related areas.

By contrast, a traditional search engine must try very hard to come up with the most accurate result possible in fulfilling the user's request, because once it has returned, it never gets another chance. This places strict requirements on how the search agent determines its choices.

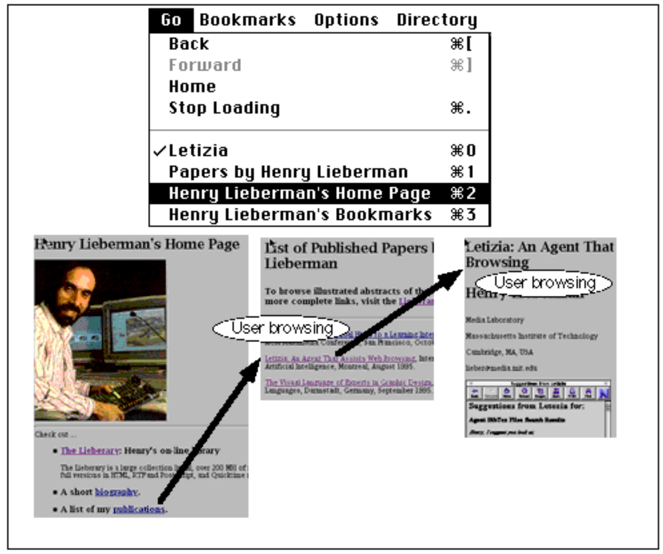
Another problem with the suggestions provided by traditional search engines is that their recommendations are not delivered "in context". The user must perform a mental "context switch" from browsing the space of Web pages to explicitly interacting with the search agent. Letizia bases recommendations on the page the user is examining at the moment, the user often sees both an explicitly selected page and an agent-recommended page simultaneously. The flow of thought of the user's browsing activity is not interrupted by the need to switch to an independent query interface. The agent recommendations come "just in time" as the related pages arise in the user's browsing activity.

Letizia's search and recommendation process

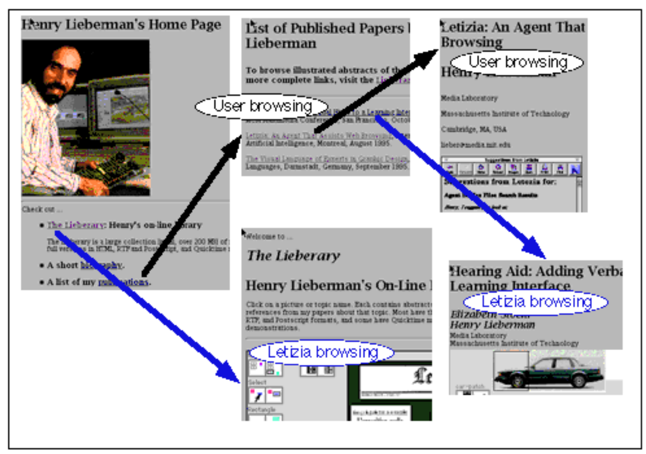
Letizia's search process is based on an observation about the control structure of browsers such as Netscape. The basic structure of a browser is to display a set of choices, the links to other Web pages, along with the page text and pictures. The user then picks one of the links, leading to another Web page, again with links, text and pictures. Picking one of the links again leads to another page, and so on. Each time, the user generally moves "down" in the Web hierarchy.

Netscape's "Go" menu accumulates a stack of previously explored Web pages. You can always "back out" to a previous page at a higher level, but this is not as natural an operation as clicking on a link to go further down one more level. You can see only the titles of the pages on the stack, and it is often unclear how many levels to back up in order to "leave" the current topic.

Thus browsers tend to encourage a depth-first exploration of the Web space. The problem with depth-first search in the user interface of a browser is the same as the problem with depth-first search in traditional AI search spaces -- it is easy to get stuck searching down long and fruitless paths when a better answer lies nearby, but off to the side. Does this happen in real life browsing? Of course it does.



Traditional browsing leads the user into doing a depth first search of the Web



Letizia conducts a concurrent breadth first search rooted from the user's current position

Letizia compensates for the tendency towards depth-first search by doing a breadth-first search, interleaved with the depth-first search performed by the user. Letizia starts by searching every page one link away from the user's current position, then two links away, and so on. It is often the case that pages of considerable interest to the user are not very far from the user's current position, if only the user knew exactly which links to follow to arrive at them.

Uncontrolled breadth-first search is known to be combinatorially explosive, hence impractical. In practice, though, Letizia's breadth-first search is constrained by the interactive behavior of the user. Whenever the user switches from one Web page to another, Letizia's search is immediately refocused to the new page. The state of the search can be stored in a hash table indexed on the current page to continue if the user returns to the page in a short time [a common occurrence]. The user typically looks at a Web page for two or three minutes, then goes on. Web pages tend to be approximately of uniform size. Thus the search tends to remain manageable.

To compute the content of a document, Letizia currently uses a simple keyword frequency measure, TFIDF [term frequency times inverse document frequency]. This well-known technique [19] says that keywords that are relatively common in the document, but relatively rare in general are good indicators of the content. This heuristic is not very reliable, but is quick to compute and serves as a placeholder for more sophisticated techniques. A user profile is accumulated during browsing, and may be written out and read back in across sessions so the user profile can be persistent.

Browsing and query interfaces

There is a fundamental tradeoff between browsing and querying in information-seeking computer interfaces. In browsing, the user must manually follow links from one page to another, but gets to see the information along the path. In conventional query interfaces, the computer automates some of the link-following activity, but at the cost of making the path-following process invisible to the user.

At any given moment, the user could be (1) evaluating retrieved material, (2) interacting with the system in order to instruct it to retrieve more material [e.g. browsing, filling out search query forms, or (3) waiting for the system to retrieve more material. It is up to the agent to balance the tradeoff between these alternatives.

From a human interface standpoint, people tend to find browsing interfaces more fun than querying interfaces. They don't have to commit in advance to evaluation criteria, and often learn something along the way. Even before the Web, there were many query-oriented interfaces which did not achieve the popularity or fire the popular imagination as did the Web. One possible explanation for the sudden growth of the Web is that Mosaic was the first interface that made such a large body of information accessible through a purely browsing interface. The problem with pure browsing interfaces, however, is that they do not take advantage of the computational power of the computer to search large amounts of information rapidly.

Search agents such as Lycos and AltaVista are a way of grafting a query interface onto the browsing interface of the web, so that the benefits of automated searching are obtained. But they replace the desirable browsing interface with a less desirable query interface, reintroducing the problems of such interfaces.

If Web search agents replacing browsing with querying, the idea of Letizia is to replace the query interface of a search engine, with an interface that is more like browsing, namely the standard Netscape interface. Letizia keeps the user within a purely browsing interface while still getting some of the benefit of automated searching.

Bates [3] provides a broad discussion of the spectrum of strategies people use in information retrieval, from the perspective of library science. Computer interfaces can play a variety of roles as collaborators in the information search process, taking various amounts of initiative ranging from low-level keyword searches to automating high-level, multi-step, content-oriented searches. Letizia-like agents can cut across this spectrum.

Streamlining the browsing process

The utility of a Letizia agent lies not in making as accurate a recommendation as possible to the user, but in contributing to increasing the "efficiency" of the browsing process by reducing wasted browsing movements and by increasing the likelihood of taking advantage of browsing opportunities.

A *dead end* is a link that looks like it might be interesting from reading the link text or seeing a link image, but turns out not to be. A *garden path* is a sequence of links that provide just enough incentive to keep following the path, but ultimately results in a dead end. Garden paths may account for a significant percentage of total user browsing time. Time wasted following these paths is one of the great frustrations of the user experience of the Web. Letizia serves the user by helping avoid dead ends and garden paths, which tend not to appear among Letizia's recommendations.

On the other hand, Letizia serves the user by noticing serendipitous connections. Users tend to have "chunky" browsing behavior, spending a coherent interval of time browsing pages on a single topic or a topic and areas related to it, then switching to a [seemingly] completely different topic. Since I once lived in France, I often browse pages related to French news, language, and culture. From this sustained browsing activity, Letizia can pick up my interest in things French. If I then turn to my interest in music and browse pages related to jazz, this interest, too can be noted. If I continue by browsing a calendar of events in the Boston area, bringing me a couple of links away from an announcement about a French jazz festival, this event would be brought to my attention. Thus, Letizia serves an important function in noting the serendipitous connections that occur during browsing.

Design principles for autonomous interface agents

Autonomous interface agents raise a whole set of unique issues in interface design. Below, we discuss some interesting aspects of interface design that emerged from work on Letizia.

Suggest rather than act

Autonomous interface agents work best in situations where their decisions are not critical. Many people are afraid of granting autonomy to interface agents because of the fear that the agent will make a bad decision without their consent. This fear is not without justification.

However, there are many instances where the decisions are not critical and the agent can be cast in the role of making suggestions rather than having responsibility for solving the whole problem. In these situations, the agent does not have to make the absolute best choice in order to be useful, but only offer a suggestion that is "better than nothing", or a "good enough guess".

Web browsing is a good application for an autonomous interface agent because the choice of any one Web page is not a critical decision. When the user is facing a choice between a set of unknown Web links, a recommendation made by the agent can simply increase the chances slightly that the user will make the best choice of what to look at next. Suggesting relies on the user to examine choices by the agent and makes browsing a cooperative activity between the user and the agent.

Some important aspects of this kind of interaction are that the feedback from the agent does not disrupt the work flow of the application interface, and that the agent does not act contrary to the consent of the user. The agent does not insist on acceptance or rejection of its advice.

An important thing to note about much user interaction is that the possibilities for presentation and for user action are almost always very *underconstrained*. There are always a wide range of possibilities for what to show the user and what the user may do next, and these choices are usually made in advance by the application designer. In many cases, these decisions appear arbitrary.

It is in these cases where the system can provide intelligent assistance without becoming too aggressive or disruptive in the user interface. One of the primary roles of agents in the interface should be to provide *agent defaults* for system output and for user action options.

One kind of agent default is *anticipatory feedback*, a display that highlights possibilities in the user interface that the agent would like to suggest that the user consider. Webwatcher displays a modified HTML page where a set of eyeballs identifies a link that has been suggested by the agent.

Agent defaults can either be knowledge-based -- dependent on an agent's predefined knowledge and inference of the probable goals of the user, as in Etzioni's Internet Softbot; or behavior-based-- predicted from empirical observation of the user's behavior, as in Letizia. Combinations of both approaches are possible.

Take advantage of information the user gives the agent "for free"

The actions taken by the user in the user interface constitute information that the system can use to infer the goals and interests of the user "for free" -- that is, without requiring a separate interaction. Even the simplest of interactions that lasts just a couple of seconds can be disruptive enough to the user's workflow that he or she won't bother to do it.

Search engines have an absolutely minimal interaction – bring up a single-line query form, the user types a word, then clicks "enter" -- but it is disruptive enough to the browsing process that it is impractical to use the search engine after every few browsing steps. Letizia's automatic inference of terms that would otherwise have to be typed into a search engine makes bringing up recommendations based on each browsing step possible.

Take advantage of the user's think time

A disadvantage of the traditional conversational interface is that the agent remains idle when the user is thinking about what input to provide next. Running the agent autonomously while the user is thinking takes advantage of compute time that otherwise be wasted. This is especially important in search or exploration tasks when the idle time can be used to provide independent exploration of the search space. Available time can also be used to deepen the results of previous searches or interests, and other lower-priority tasks that might provide useful but would also potentially risk disrupting the primary interaction.

The user's attention may be time-shared

A consequence of running an interface agent autonomously is that the agent cannot assume it has the full attention of the user in the way a traditional, modal interaction does. The user may find themselves providing input to the agent without being explicitly aware of where the input is going. The user may find output happening on the screen at a time that they may not expect it. This can be both good and bad. It can make more information available "at a glance", provide a better sense of context, and reduce the need for context-switching.

The autonomously running Netscape window through which Letizia displays its recommendations to the user need not command the full attention of the user. When the user is actively engaged in browsing, Letizia's actions can be ignored. When the user's attention to the manual browsing session flags, glancing over to the Letizia window can provide timely suggestions for the next browsing action.

Autonomous interface agents may have a different tradeoff between deliberation and action

Well-known in artificial intelligence is the tradeoff between deliberation and action. Since thinking about a problem consumes time in and of itself, there reaches a point where it may not be worth spending more time trying to make a better decision, and it's best to "just do it".

For a traditional search engine, the judgment as to whether a particular document should be considered relevant is critical, because once the search engine returns a list of suggestions for a given query, it never gets another chance. By contrast, when Letizia needs to decide to recommend a document, the decision is less critical because it will usually have future opportunities to make a better decision as to how relevant that document is relative to other documents that it might propose as alternatives. At any moment, Letizia may choose either to spend more time analyzing pages, or to spend time searching more pages.

Running as an autonomous interface agent gives Letizia the opportunity to make a "quick and dirty" guess first, then spend more time, if it is given the opportunity, to make a more refined judgment.

An autonomous interface may not fit the cognitive styles of all users

It is important to realize that users have different cognitive styles and this style of interface may appear distracting or confusing to some users. Some users may be uncomfortable with nonlinear interfaces where more than one thing is going on at once. What sometimes seems distracting about such interfaces is that the user may feel obligated to pay attention to every screen change, even when absorbed in other activity. An experienced user may feel more comfortable with the idea of only keeping “half an eye” on a change occurring in some other part of the screen. "MTV generation" and "Nintendo generation" users may be more comfortable with the idea of visually busy, fast-paced interfaces.

Related work

The closest projects to Letizia are Webwatcher [1] and Lira [2]. Both of these are Web agents whose actions are interleaved with the user's browsing in Netscape, but they run on the server-side, whereas Letizia runs on the client-side, and both require explicit interaction to indicate interest in topics or particular pages. The Remembrance Agent [18] is also an autonomous interface agent, one that reminds the user of relevant files stored on the user's local disk. It "remembers the past" [shows the user relevant material that they have already seen] whereas Letizia "remembers the future" [shows relevant material not yet seen].

Other approaches to Web agents, some more knowledge-intensive than Letizia’s essentially behavior-based approach, are illustrated by [7, 10, 16] and others referenced below.

Few references explicitly discuss the control structure issues for autonomous agents that share interfaces with the user, as we do here, but [17] is an exception, providing good discussion of issues surrounding agent-user collaborative interfaces. In their system, both a user and an agent cooperatively use an airline reservation system. A CHI ‘95 Workshop on “Model Worlds Versus Magic Worlds” [21], [e.g. direct manipulation interfaces vs. autonomous interface agents] also investigated some of the same design issues mentioned here.

[13] presents an early version of Letizia, with a more conventional, conversational interface. The name Letizia comes from [4].

CONCLUSION

The ability of a software agent to operate the same interface operated by the human user, and the ability of a software agent to act independently of, and concurrently with, the human user will become increasingly important characteristics of human-computer interaction. Agents will observe what human users do when they interact with interfaces, and provide assistance by manipulating the interface themselves, while the user is thinking or performing other operations. Increasingly, applications will be designed to be operated both by users and their agents, simultaneously.

We’ve illustrated some of the principles behind autonomous interface agents with a description of Letizia, an agent for assisting a user browsing the Web. Letizia is an interface agent in that it observes input that the user directs at the browser, not at the agent. It reports its results by automatically browsing Web pages in a Netscape window. It is autonomous in that it browses concurrently with the user, searches and analyzes Web pages while the user is browsing, and displays recommendations continually, without explicit user request or other intervention. We think that autonomous interface agents like Letizia will be the next step in the evolution of interfaces beyond direct manipulation.

More information about Letizia, including a Quicktime movie demonstration, can be found at the web page,

`http://www.media.mit.edu/~lieber/Lieberary/Letizia/Letizia-Intro.html`

and at the author's home page,

`http://www.media.mit.edu/~lieber/`

ACKNOWLEDGMENTS

This work was supported in part by grants from the European Community, British Telecom, Apple Computer, the News in the Future Consortium, the Digital Life Consortium, and other sponsors of the MIT Media Lab. Thanks to David Maulsby for a careful reading of an earlier draft.

REFERENCES

1. Robert Armstrong, Dayne Freitag, Thorsten Joachims and Tom Mitchell, WebWatcher: A Learning Apprentice for the World Wide Web, in AAAI Spring Symposium on Information Gathering, Stanford, CA, March 1995.
   `http://www.cs.cmu.edu/afs/cs/project/theo-6/web-agent/www/project-home.html`
2. Marko Balabanovic and Yoav Shoham, Learning Information Retrieval Agents: Experiments with Automated Web Browsing, in AAAI Spring Symposium on Information Gathering, Stanford, CA, March 1995.
   `http://flamingo.stanford.edu/users/marko/bio.html`
3. M. Bates, Where should the person stop and the information search interface start? Information Processing & Management, 26, 575-591, 1990.
   `http://www.gse.ucla.edu/facpage/bates.html`
4. Jorge Luis Borges, The Library of Babel, in *Ficciones*, Grove Press, 1942.
5. Daniel Dennett, *The Intentional Stance*, MIT Press/Bradford Books, Cambridge, MA, 1987.
6. C. Drummond, R. Holte, D. Ionescu, Accelerating Browsing by Automatically Inferring a User's Search

Goal. Proceedings of the Eighth Knowledge-Based Software Engineering Conference pp. 160-167.

http://www.csi.uottawa.ca:80/~holte/Learning/index.html

7. Oren Etzioni and Daniel Weld, A Softbot-Based Interface to the Internet, Communications of the ACM, July 1994.

http://www.cs.washington.edu/homes/weld/pubs.html

8. Gerhard Fischer, Andreas C. Lemke, Thomas Mastaglio and Anders I. Morch, Critics: An Emerging Approach to Knowledge Based Human Computer Interaction, International Journal of Man-Machine Studies, p. 695-721, 35(5), Nov 1991.

9. William C. Hill, James D. Hollan, Dave Wroblewski and Tim McCandless, Edit Wear and Read Wear, CHI 92.

10. Robert C. Holte and Chris Drummond, A Learning Apprentice For Browsing, AAAI Spring Symposium on Software Agents, 1994.

http://www.csi.uottawa.ca:80/~holte/Learning/index.html

11. Craig A. Knoblock, Yigal Arens, and Chun-Nan Hsu Cooperating Agents for Information Retrieval Proceedings of the Second International Conference on Cooperative Information Systems, Toronto, Ontario, Canada, University of Toronto Press, 1994.

http://www.isi.edu/sims/knoblock/info-agents.html

12. Jaron Lanier, Agents of Alienation, Interactions, July 1995.

13. Henry Lieberman, Letizia: An Agent That Assists Web Browsing, International Joint Conference on Artificial Intelligence, Montr⁄al, August 1995.

http://lieber.www.media.mit.edu/people/lieber/Lieberary/Letizia/Letizia.html

14. Henry Lieberman, From Human-Computer Interface to Human-Computer Collaboration, CHI Research Symposium, 1995.

15. Pattie Maes, Agents that Reduce Work and Information Overload, Communications of the ACM, July 94.

16. Mike Perkowitz and Oren Etzioni, Category Translation: Learning to Understand Information on the Internet, International Joint Conference on Artificial Intelligence, Montr⁄al, August 1995.

http://www.cs.washington.edu/homes/map/ila.html

17. Charles Rich and Candace Sidner, Adding a Collaborative Agent to Graphical User Interfaces, ACM Symposium on User Interface Software Technology, 1996.

http://www.merl.com/pub/rich/uist96.ps.Z

18. B. J. Rhodes and Thad Starner, The Remembrance Agent, AAAI Symposium on Acquisition, Learning and Demonstrations, Stanford, CA, 1996.

http://www.media.mit.edu/~rhodes/remembrance.html

19. G. Salton, Automatic Text Processing: The Transformation, *Analysis and Retrieval of Information by Computer*, Addison Wesley, 1989.

20. Ted Selker, COACH: A Teaching Agent That Learns, Communications of the ACM, July 1994.

21. Loren Terveen and Markus Stolze, From Model Worlds to Magic Worlds, SigCHI Bulletin, 27(4), 1995.