Agent-Assisted Internet Browsing

Gabriel L. Somlo and Adele E. Howe
Computer Science Department
Colorado State University
Fort Collins, CO 80523
email: {somlo,howe}@cs.colostate.edu

Introduction

In recent years, the Internet has become a resource relied upon for a constantly growing number of needs: research, news, shopping, entertainment, travel, etc. However, as a growing number of individuals and organizations keep contributing an increasing volume of data, the users, especially novice ones, become more confused and lost in a quagmire of hard to reach, often contradicting pieces of information.

Given a specific task, the problem facing the users is no longer the lack of availability of information related to that task, but extracting it from the huge amount of useless, uninformed, or outright misleading data that has been made available on the Internet. The process of separating useful data from noise, verifying its validity, and integrating multiple sources of information can be extremely time consuming and inefficient if done manually. The waste of time explodes if the need for information is not sporadic, but repetitive.

During the last few years, automating the task of gathering information has been a hot topic in AI and agents research. Many solutions have been proposed, especially in the form of meta-search engines (e.g., (Dreilinger)), WWW browser proxies (e.g., (Chen & Sycara 1998)), and recommendation systems (e.g., (Alexa Internet Co.)). Although these systems have been tremendously helpful, the problem with many of these solutions is that they are centralized – intended to be used by a large number of people. Because of this, their help may be either too general (their model of a "user" is averaged over all people using the service), or too invasive (personalized help from a centralized service requires the user to surrender trust and personal information).

Addressing the Problem

We present a *personal* Internet helper agent, residing on the user's own machine, that learns tastes, preferences and habits from the user and uses this knowledge to help automate a significant portion of the tedium associated with navigating the Internet. For this agent, we focused on four design principles:

- service should be tailored to a specific user,
- user information should not be available outside our system (preserve privacy),
- the agent should minimize annoyance to and interruption of the user, and
- the tedium of waiting for and wading through information request responses should be off-loaded to the agent.

Ultimately (after the learning or tuning period), the performance of the agent will be judged by how well it collects information compared to alternative methods and how much time it saves the user. During learning, we will need to assess the level of burden on the user.

In our design, the agent starts out by monitoring the user's activity, learning from the user's repetitive actions (unsupervised) and from explicit feedback (supervised), building a profile that models the user from the agent's point of view. In time, it will start having "initiative", in the form of pre-fetching information from frequently visited sites, recommending sites to visit that contain information similar to the user's interests (to improve recall), refine (or recommend refinements to) queries to search engines, and read and reorder hits resulted from such queries according to the user's interest (to improve precision).

Because each individual user has a number of interests not necessarily related to each other, it makes sense to split the profile into "categories of interest" (e.g., web agents, computer-generated visual effects, linux, wireless/cellular, news, shopping, travel, etc.). WebMate (Chen & Sycara 1998) offers this feature internally, but when supervised learning occurs, the category to be updated is selected automatically by the agent without giving the user a chance to express an option. Also, the number of distinct preference categories is hardcoded to 10. Our agent will allow the user to create or update such a category explicitly, with an optional recommendation as to what existing category is the best match for a new document.

The proposed Internet agent will use both supervised and unsupervised learning. For supervised learning, the user has the option of pointing out documents of particular interest. These documents are processed, and the results added to the profile under a selected category of

interest.

Unsupervised learning is a continuous activity – the agent maintains a history list with addresses of all the sites visited by the user. If repetition occurs (i.e., the user is interested in monitoring certain sites over a longer period of time), the agent will learn this from the history list and add the address to a separate section of the user profile.

We present a few scenarios in which our Internet agent can be of assistance to users:

1. Many users read news on the Internet (news sites, Usenet, etc.) regularly. Unfortunately, the signal to noise ratio in these environments is low, and a lot of time is wasted deciding what articles are actually worth reading. Luckily, by repeatedly visiting these sites, our agent adds them to the profile, and automatically visits these sites a few hours before we wake up, downloading only the articles that match one of our categories of interest. This way, we have a compilation of news worth reading made available to us just in time when we turn on the computer in the morning. The same example applies to sites that we monitor regularly in connection to our work. The improvement over other Internet agents such as WebMate (Chen & Sycara 1998) is that the list of sites to check will be *learned* by the agent, and thus will change over time as the user's interests shift to new topics.
2. When we need to find information, we usually visit a search engine, and type in a few keywords. Frequently we receive a vast amount of hits totally unrelated to what we are actually looking for (due to the many contexts we didn't think of in which our keywords can be used). Fortunately, while providing supervised feedback to our agent, it also learned dependencies between prominent words in the documents we liked. These dependencies can be used to recommend refinements to our queries, in order to limit the number of different contexts and thus the amount of false positives we receive. Also, the hits from a search engine can be re-ordered by the agent such that documents matching one of our profile categories are placed towards the front of the list.
3. The agent could search for information on its own, in an attempt to predict our needs for the short-term future. Every night, several prominent words from our profile could be selected to form queries to our favorite search engine(s). The hits can be cached and presented to us later when needed. Actually, this is an example of complementary behavior to what is presented in the first scenario, where the agent uses its *already acquired* knowledge. The behavior is *exploratory*, attempting to discover new sources of information that might be of interest to the user.

Implementation and Technical Details

Given our design principles and goals for the agent, we identified several issues that direct its design. First, how should information be collected? Currently, our agent is being implemented as a WWW proxy. This way, it has access to the addresses and contents of all documents downloaded by the user's Internet browser. The interface provided to the user is also done via HTML, in the form of a "Feedback" button appended to each downloaded document. This button is a link to a bogus URL, one that is intercepted by the proxy. A set of options is generated in response to pressing this button, including the selection of a profile category to be updated. Thus, the user chooses when to actively prod the agent to remember an important web page and relate it to a category.

All communication with the agent will occur via HTML. When clicking the "Feedback" button appended by the proxy at the bottom of each downloaded document, the user will be taken to a bogus URL intercepted by the proxy, and served a document created on-the-fly by the agent. This will be the agent's console, from where the user will have access to the profile categories, the search-engine interface, a list of links to pre-fetched documents deemed to be of interest, and a set of links to suggested new documents worth looking at. An example of the console (currently only including the profile categories) is displayed in Figure 1.

The second issue is how to organize and learn from the collected information. Our agent employs a web tailored version of standard information retrieval methods. The profile categories are based on term/document frequencies. TF-IDF (Salton & McGill 1983) word vectors are created from each processed document by assigning each word in the document a weight, according to the following formula:

$$W[word] = TF[word] \cdot \log \frac{D}{DF[word]}$$

where $TF[word]$ is the actual count (term frequency) of *word* in the document being processed, D is the total number of existing documents, and $DF[word]$ is the number of documents in which *word* occurs at least once (document frequency).

Each profile category is also a TF-IDF vector, and updating a category is accomplished by simply adding the new document vector to the vector of the selected category. Similarity between documents and/or categories is measured by computing the dot-product of the respective TF-IDF vectors.

TF-IDF methods have been developed for use with large collection of documents that are known in advance. Hence, D and $DF[word]$ are known and constant across the entire set of documents. Unfortunately, we do not have access to the entire collection of documents available on the Internet, and therefore we need to approximate D and DF incrementally. Previous agents such as WebMate (Chen & Sycara 1998) have been using only the documents selected for (positive) feedback to approximate these values. Because of this, they only become reliable after the user has expressed interest in a large number of documents. Our agent

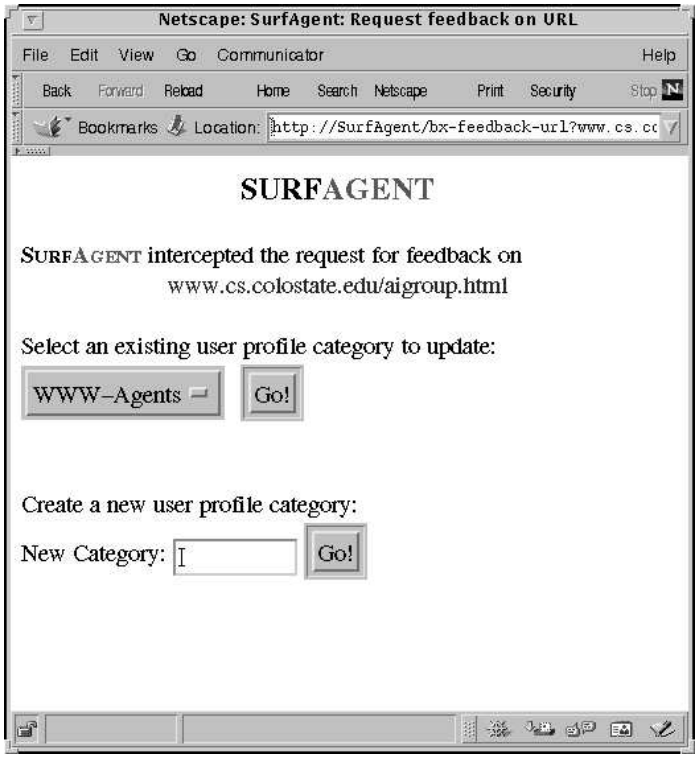

Figure 1: HTML-based interaction between user and agent

brings an improvement to this approximation: since it is a proxy, it has to pass through all the downloaded documents, regardless of their value to the user. While doing so, it is a trivial and inexpensive task to count the words in these documents and update D and DF accordingly, thus having a better approximation for them in a much shorter time. We also will explore the utility of allowing the user to give negative feedback.

In addition to the above change, we are planning to add the unsupervised learning component – the extraction of addresses of frequently re-visited sites from the history list, and extend the supervised learning from TF-IDF vectors and profile categories to word dependencies. Algorithms will be used to compute relationships between the most prominent words occurring in the documents used as positive feedback. We have previously developed one such family of algorithms, DD (Dependency Detection) (Howe & Somlo 1997), that analyzes a trace of events for ones that frequently occur together. Using word dependencies computed this way and a search-engine interface, queries can be automatically refined or extended, or refinement suggestions can be presented to the user. Hits from search engines can be scanned, and re-ordered according to their expected value to the user, as computed by the similarity of their TF-IDF vectors to one or more of the user-profile categories.

So far, the primary sources of information will be what pages are downloaded and whether the user associates a page with a category. Other information is extractable passively by observing the user's interaction with the web browser: the links that are followed or ignored, amount of time spent pausing on a page and the route used to get to a page (through other pages, from bookmarks or directly). For example, the SavvySearch meta-search engine used passively collected information (which links were followed) to learn associations between search engines and query terms (Dreilinger & Howe 1997). Unfortunately, these sources are only indirect and noisy indicators of user interest. For example, the user may have paused on a page because he was distracted with a telephone call while browsing. However, we will be considering how this information can be incorporated in the learning algorithms.

Another issue is what services to supply to the user. As suggested by the scenarios, the basic capability will be to automate routine, repetitive searching and browsing, producing digests off-line based on frequently visited sites and often searched topics. The next step will be to add the "initiative" component to the agent: once enough documents have been processed, the agent will go out on the Internet and pre-fetch information matching the user profile from the sites it has learned to monitor. Also, displaying exploratory behavior, it will attempt to predict the user's needs and find new sources of useful information. Exploration will be accomplished by both querying search engines using prominent words from the user profile, and by following links from the

documents found at the frequently visited locations. The history list will be used to avoid multiple lookups of the same document, and new documents will only be considered interesting if they match one or more profile categories to a certain threshold.

Conclusions and Future Plans

We are currently implementing the design presented in this paper. Once we have tested the basic capability prototype, we are planning on adding capability to monitor Usenet transactions in addition to WWW pages. In addition, we are developing a more sound solution to the TF-IDF approximation problem presented above (or even alternatives to TF-IDF as far as user profiling is concerned).

Our Internet helper agent is being actively developed at this time. Currently, we have completed the learning of profile categories using TF-IDF vectors. The agent is being built on top of the JunkBuster proxy (Junkbusters Corp.), designed to filter out banner ads and cookies from Web documents based on matching regular expressions to their URL's. The efficient implementation in C and the use of HTML to access the agent directly through the browser promises to allow our agent to be among the first that have adequate real-time response and do not annoy users by slowing down their Web access.

Acknowledgments

This research was supported by a Career award from the National Science Foundation, number IRI-9624058. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

Alexa Internet Co. Alexa internet recommendation service. `http://www.alexa.com/`.

Chen, L., and Sycara, K. 1998. WebMate: A personal agent for browsing and searching. In *Proceedings of Autonomous Agents '98*.

Dreilinger, D. The SavvySearch meta-search engine. `http://www.savvysearch.com/`.

Dreilinger, D., and Howe, A. 1997. Experiences with selecting search engines using meta-search. *ACM Transactions on Information Systems* 15(3):195–222.

Howe, A., and Somlo, G. 1997. Modeling discrete event sequences as state transition diagrams. In *Proceedings of the Second Conference on Intelligent Data Analysis*.

Junkbusters Corp. The internet junkbuster proxy. `http://www.junkbusters.com/`.

Salton, G., and McGill, M. 1983. *Introduction to Modern Information Retrieval*. New York: McGraw-Hill.