An Instructable, Adaptive Interface for Discovering and Monitoring Information on the World-Wide Web*

Jude Shavlik, Susan Calcari, Tina Eliassi-Rad, and **Jack Solock**
Computer Sciences Department
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706 USA
+1 608 262 7784
{shavlik, scal, eliassi, jacks}@cs.wisc.edu

ABSTRACT

We are creating a customizable, intelligent interface to the World-Wide Web that assists a user in locating specific, current, and relevant information. The *Wisconsin Adaptive Web Assistant* (WAWA) is capable of accepting instructions regarding what type of information the users are seeking and how to go about looking for it. WAWA compiles these instructions into neural networks, which means that the system's behavior can be modified via training examples. Users can create these training examples by rating pages retrieved by WAWA, but more importantly the system uses techniques from reinforcement learning to internally create its own examples (users can also later provide additional instructions). WAWA uses these neural networks to guide its autonomous navigation of the Web, thereby producing an interface to the Web that users periodically instruct and which in the background searches the Web for relevant information, including periodically revisiting pages that change regularly.

Keywords

intelligent Web interfaces, instructable software agents, machine learning, neural networks, information retrieval

INTRODUCTION

Due to the vastness and dynamic nature of the World Wide Web, there is a tremendous need for flexible information-finding systems that can be easily customized to the personal interests of individuals. The highly distributed nature of the Web, and the fact that the content is constantly being updated, presents a serious challenge to those who want to be aware of news, articles, and data sources being made available daily, since they do not have time to constantly monitor all sources for relevant information.

In addition to the information overload of networked information provided on the Web, there is the issue of the value and quality of the resources. People normally want only very specific information that is related to their own interests and that they deem authoritative. However, they are the sole best judge of what is relevant to them and of high quality. Therefore, the most useful interface to online information will be one that is customizable by individual users for their own purposes.

The user interface also needs to run autonomously in order to free the user from the constant monitoring of the network. The individual user wants the interface to be instructable so that once advice has been given, the tool will work on its own over long periods while they devote their attention to their work. This enables the user to monitor the information that is of interest to them without spending a lot of time on it. The adaptive, intelligent user interface we are developing for this task is called the *Wisconsin Adaptive Web Assistant* (WAWA). It assists users in the discovery, retrieval, and filtering of online information.

WAWA allows individuals to easily create personalized computer assistants for searching and monitoring the Web. We have created a language for describing one's interests and for providing hints regarding where relevant information might be found. Our instructable interface converts this user-provided information into two neural networks. WAWA then uses these scoring functions to guide its search of the Web and to determine which pages to collect for the user. Since the core of the system involves neural networks, we are able to use training examples to refine the user-provided instructions.

WAWA is instructable for particular tasks, such as finding newly released articles, datasets, grant opportunities, or courseware related to one's professional activities, as well for following one's personal hobbies and persistent interests. For example, a cancer researcher might specify some good starting sites on the Web and a description of his or her particular interests. Our system would then periodically visit these sites, including additional sites within some number of hyperlinks of them, collecting the most relevant new pages. Especially suitable topics for our approach are those where

*This paper appears in the *Proceedings of the 1999 International Conference on Intelligent User Interfaces*, Redondo Beach (Los Angeles, CA, USA), January 5-8, 1999.

information changes often and is dispersed over many different sources, as is the case on the Web.

APPROACH TAKEN

At the heart of WAWA are two neural networks, implementing the functions SCORETHISLINK and SCORETHISPAGE. These functions, respectively, guide the system's wandering within the Web and judge the value of the pages encountered. The user mainly programs these two functions by providing *advice*. Following [4], we call our language an advice language, because this name emphasizes that the underlying system does not blindly follow the user-provided instructions, but instead uses machine learning techniques to refine this advice based on the system's experiences.

Table 1 provides a high-level description of WAWA. First, its initial neural networks need to be created. One can view the process of converting advice into a neural network as analogous to compiling a traditional program into machine code, but our system instead compiles instructions into an intermediate language expressed using neural networks. We use the KBANN algorithm [12] to convert the user's instructions into a neural network that initially exactly executes the user's advice. This provides the important advantage that our "machine code" can automatically be refined based on subsequent feedback provided by either the user or the Web, as explained below.

The basic operation of WAWA is heuristic search, with our SCORELINK function providing the sorting heuristic. WAWA collects the 100 pages that SCOREPAGE rates highest. The user can choose to seed the queue of pages to fetch in two ways: either by specifying a set of starting URLs or by providing a simple query that WAWA converts into "query" URLs that are sent to a user-chosen subset of selectable search-engine sites (currently AltaVista, Excite, Info Seek, Lycos, and Yahoo).

Although not mentioned in Table 1, the user may also specify a depth limit that puts an upper bound on the distance the system will wander from the initial URLs.

Before fetching a page (other than those initially in the search queue), WAWA has predicted the value of fetching the page, based on the contents of the "referring" page that linked to it. After fetching and analyzing the text, the system will have a better estimate of the page's value to the user. Any differences between the "before" and "after" estimates constitute an error that can be used by the backpropagation (BP) [7] training algorithm to improve the SCORELINK neural network.

In addition to the above system-internal method of automatically creating training examples via *temporal-difference learning* [11], the user can improve the SCOREPAGE and SCORELINK neural networks in two ways. One, the user can provide additional advice. Observing the system's behavior is likely to invoke thoughts of good additional instructions. WAWA can accept new advice and augment its neural networks at any time. It simply adds to a network additional nodes that represent the compiled advice. Providing such hints can rapidly and drastically improve WAWA's performance.

Table 1: The WAWA Algorithm

Unless they have been saved to disk in a previous session, create the *ScoreLink* and *ScorePage* neural networks by reading the user's initial advice (if any).

Either (a) start by adding the user-provided URLs to the search queue; or
(b) initialize the search queue with URLs that will query the user's chosen set of Web search-engine sites.

Execute the following concurrent processes.

Independent Process #1
While the search queue is not empty nor the maximum number of URLs visited,

Let $URLtoVisit$ = pop(search queue).
Fetch $URLtoVisit$.

Evaluate $URLtoVisit$ using *ScorePage*.
If score is high enough, insert $URLtoVisit$ into the sorted list of best pages found.
Use the score of $URLtoVisit$ to improve the predictions of the *ScoreLink* function.

Evaluate the hyperlinks in $URLtoVisit$ using *ScoreLink* (however, only score those links that have not yet been visited this session).
Insert these new URLs into the (sorted) search queue if they fit within its max-length limitation.

Independent Process #2
Whenever the user provides additional advice, add it to the appropriate neural network.

Independent Process #3
Whenever the person rates a fetched page, use this rating to create a training example for the *ScorePage* neural network.

Although more tedious, the user can also rate pages as a mechanism for providing training examples. This can be useful when the user is unable to articulate why the system is misscoring pages and links, but is able to provide better scores. This standard "learning from labeled examples" methodology has been previously investigated by other researchers, e.g., [6], and we will not further discuss this aspect of WAWA in this article. We do conjecture, though, that most of the improvement to WAWA's neural networks, especially to SCOREPAGE, will result from users providing advice. In our personal experience, it is easy to think of simple advice that would require a large number of labeled examples in order to learn purely inductively. Empirical support for these claims is a topic of experiments in progress.

WAWA's use of neural networks means we need a

mechanism for processing arbitrarily long Web pages with the fixed-sized input vectors that neural networks have. One approach would be to use recurrent neural networks, but instead we borrow an idea from NETTALK [9], though our basic unit is a word rather than an (alphabetic) letter as in NETTALK. WAWA slides a fixed-width (15 words typically) window across a page, and many of the "features" we use to represent a page are defined with respect to the current center of this window.

Basically, we define the score of a page to be the highest score the SCOREPAGE network produces as it is slid across the page. The value of a hyperlink is computed similarly, but WAWA only slides the SCORELINK network over the *hypertext* associated with that hyperlink. (However, in this case the window starts by being centered on the first word in the hypertext, which means the nearby words outside of the hypertext will sometimes fill some of the window positions.)

We next turn to how WAWA represents Web pages and the constructs of its advice language. The input features it extracts (from either HTML or plain text) constitute the primitives in our advice language. We have devoted substantial effort to extracting a large number of possible features from Web pages, since the richness of the set of extracted features determines the expressiveness of our advice language.

A standard representation of free-form text used in information retrieval is the *vector-space model* [8] (or the *bag-of-words* representation). The left side of Fig. 1 illustrates this representation. Basically, word order is lost and all that is used is a vector that records the words present on the page, typically scaled by the number of occurrences of each words and usually normalized with respect to the expected number of occurrences of each word (e.g., TFIDF [8]).

Information retrieval systems also usually discard common ("stop") words and "stem" all words to their root form (e.g., "walked" becomes "walk") [8]. Doing so greatly reduces the dimensionality of the problem, and WAWA also performs these two preprocessing steps.

Instead of solely using the bag-of-words model, however, we use a richer representation that preserves some word-order information. We also take advantage of the structure of HTML documents when a fetched page is so formatted. First, we augment the bag-of-words model, by using several localized bags, some of which are illustrated on the right side of Fig. 1. Besides a bag for all the words on the page, we have word bags for: the words in the title, the page's URL, the sliding window, the left and right-sides of the window, the current hyperlink should the window be inside hypertext, the current section's title, and several others.

In addition to these localized word bags, we also extract features representing several fixed positions on Web pages. Besides the obvious case of the positions in the sliding window (eg, the middle word in the window), we represent the first and last N words (for some fixed N) in the title, the URL, the section titles, etc. Due to its important role in the Web, we also specially

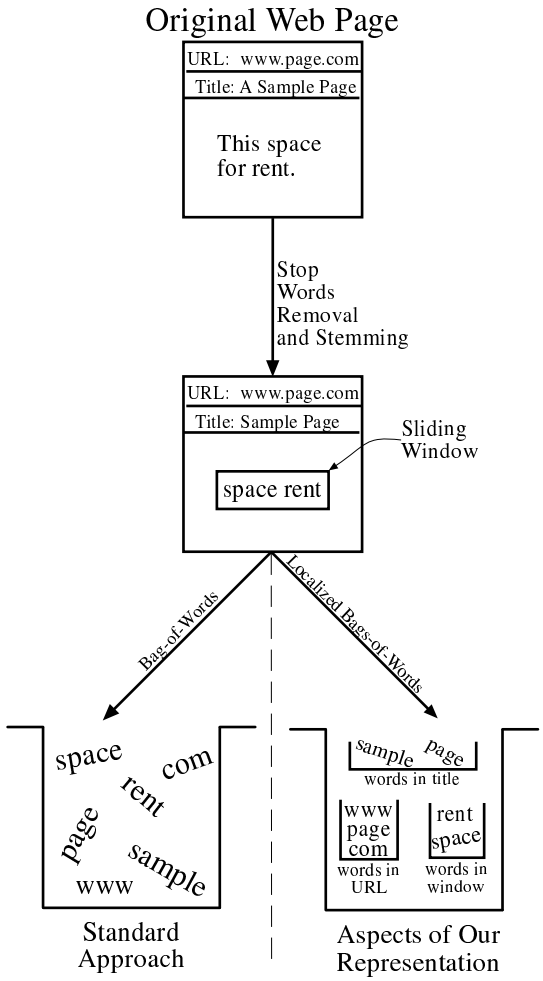

Figure 1: Internally Representing Web Pages

represent the last N fields (i.e., delimited by dots) in the *server* portion of URLs and hyperlinks, e.g. `www acm org` in `http://www.acm.org/sigchi/`.

Besides the input features related to words and their positions on the page, WAWA also constructs various other features, such as the length of the page, the date the page was created (should the page's server provide that info), whether the sliding window currently is inside emphasized HTML text, the sizes of the various word bags, how many words mentioned in advice are present in the various bags, etc.

WAWA's advice language allows users to express more complicated instructions that refer to the basic features of Web pages that WAWA extracts. A sample is:

```
WHEN consecutiveWordsInHypertext
        ( intelligent user interface )
STRONGLY SUGGEST FOLLOWING HYPERLINK
```

This advice would lead to the addition of a node ("hidden unit"), to the SCORELINK neural network, that looks for *user* in the center of the sliding window, with *intelligent* before it and *interface* after it. When such a phrase was found in some hypertext, the score for the associated hyperlink would be increased by a large

amount. One of our major current efforts involves developing an easy-to-use interface for composing advice. We have found that a menu-based design, one that converts users' choices into instructions in WAWA's advice language, produces a good combination of simplicity and expressiveness.

Further details about WAWA and an experimental study of its ability to improve its performance can be found in Shavlik and Eliassi-Rad [10]. The design of an appropriate, cost-effective framework for evaluating synergistic, human-computer learning systems is not clearcut and remains an important open issue.

CURRENT AND RELATED RESEARCH

We are currently involving "real" users in order to better understand what people would like to say to such an instructable Web agent. Based on these interactions, we are improving our advice language. We are also running additional experiments to better study how well WAWA is able to refine user-provided advice.

Another current goal is to embed WAWA into a major, existing Web browser, thereby minimizing new interface features that users must learn in order to interact with our system. Related to this, we are developing methods whereby WAWA can automatically infer plausible training examples by observing users' normal use of their browsers. Finally, we are adding to WAWA additional knowledge about English, such as synonyms [5], and are tagging words with their likely parts of speech [1].

We are collaborating with campus librarians who are specialists in the Internet to assist in building specialized versions of the system for subject-specific tasks. Then individual campus scientists can further customize these systems to their particular interests. Having human librarians provide direct assistance to individual scientists at a detailed level on a one-to-one basis is not economically feasible. Our approach promises to allow the expertise of librarians to be leveraged. In addition, our use of machine learning means that these systems will be able to automatically refine and extend the approximate and incomplete instructions provided by users, as well as adjust as users' interests change.

This collaboration between computer scientists, librarians, and scientists will leverage the expertise and time of each participant. In effect, expertise in all three levels of electronic information retrieval will be represented: the builders of the system, the information specialists who customize retrievals for individual users, and the end users themselves.

Like WAWA, *Syskill and Webert* [6] and *WebWatcher* [2] are Web agents that use machine learning techniques. *Letizia* [3] is a related system that uses lookahead search from the current location in the user's Web browser. Unlike WAWA, these systems are unable to accept (and refine) advice; advice usually is simple to provide and can lead to better learning than rating many Web pages.

CONCLUSION

We are creating an intelligent interface agent that aids in navigating and monitoring the Web for information relevant to the long-term interests of individuals. Users of our WAWA system provide instructions in our *advice language* that informs the system as to which hyperlinks it should follow and which Web pages to collect. These instructions are compiled by WAWA into two neural networks, which it then automatically refines based on additional feedback from the user, as well as training examples it constructs internally based on its experiences. The resulting system is able to autonomously wander the Web, collecting useful pages and periodically revisiting pages whose content regularly changes (e.g., news sites). Based on the training it receives, WAWA improves its performance over time and also is able to quickly adapt when the interests of its user changes.

ACKNOWLEDGEMENTS

This research is partially supported by NSF Grant IRI-9502990, NSF Grant NCR-9712163, and ONR Grant N00014-93-1-0998.

References

[1] E. Brill. Some advances in rule-based part of speech tagging. In *Proc. National Conf. on AI (AAAI-94)*, pages 722–727, 1994.

[2] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the World Wide Web. In *Proc. Intl. Joint Conf. on AI*, pages 770–775, 1997.

[3] H. Lieberman. Letzia: An agent that assists web browsing. In *Proc. Intl. Joint Conf. on AI*, pages 924–929, 1995.

[4] R. Maclin and J. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22:251–281, 1996.

[5] G. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38:39–41, 1995.

[6] M. Pazzani, J. Muramatsu, and D. Billsus. Identifying interesting web sites. In *Proc. National Conf. on AI (AAAI-96)*, pages 54–61, 1996.

[7] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[8] G. Salton. Developments in automatic text retrieval. *Science*, 253:974–979, 1991.

[9] T. Sejnowski and C. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

[10] J. Shavlik and T. Eliassi-Rad. Intelligent agents for Web-based tasks: An advice-taking approach. In *AAAI/ICML Workshop on Learning for Text Categorization*, 1998.

[11] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[12] G. Towell and J. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70:119–165, 1994.