Learning Information Retrieval Agents: Experiments with Automated Web Browsing

Marko Balabanović* and **Yoav Shoham**
marko@cs.stanford.edu, shoham@robotics.stanford.edu
Department of Computer Science,
Stanford University, Stanford, CA 94305

Abstract

The current exponential growth of the Internet precipitates a need for new tools to help people cope with the volume of information. To complement recent work on creating searchable indexes of the World-Wide Web and systems for filtering incoming e-mail and Usenet news articles, we describe a system which helps users keep abreast of new and interesting information. Every day it presents a selection of interesting web pages. The user evaluates each page, and given this feedback the system adapts and attempts to produce better pages the following day. We present some early results from an AI programming class to whom this was set as a project, and then describe our current implementation. Over the course of 24 days the output of our system was compared to both randomly-selected and human-selected pages. It consistently performed better than the random pages, and was better than the human-selected pages half of the time.

Introduction

In recent years there has been a well-publicized explosion of information available on the Internet, and a corresponding increase in usage. This is particularly true of the World-Wide Web and its associated browsers (e.g. NCSA Mosaic), which allow easier access to the information available and thus make it accessible to a wider audience.

When people access the web, we can classify their activities into two broad categories. They are either *searching* for specific information, or they are *browsing*, looking for something new or interesting (often referred to as *surfing*). There have been several efforts to index the web, and the ability to search such an index would primarily support the searching activity. This paper describes a system which attempts to support the browsing activity.

Inherent in this decision is the principle that users of the system should not have to supply specific search requests or a description of the set of documents in which they are interested, as is appropriate for the searching activity. Instead, we propose a system which presents users with a selection of documents it thinks they will find interesting. Users then evaluate each document, and the system adjusts its parameters in order to try to improve its performance.

There are obvious advantages to the use of machine learning for retrieval and interface agents, as discussed in (Maes & Kozierok 1993). However, note that our application is different from *filtering*, where the system filters a stream of incoming information (such as e-mail or Usenet newsgroups) to reduce the burden on the user. In contrast, the web is a rapidly changing and expanding resource, where the onus is on the user or agent to actively seek out relevant information.

In the remainder of this document we give an overview of our approach, describe some interesting early results from an AI programming class to whom this was set as a project, and finally briefly discuss our current implementation. We assume only a basic familiarity with the World-Wide Web.

Outline of the Agent

In this section we present a general overview of the design of the agent, along with some examples of ways of implementing the various components. Later sections discuss results from a variety of implementations all following this outline.

The system runs in discrete cycles. The following algorithm summarizes its behavior for one cycle:

1. Search the web, using some search heuristic, taking a bounded amount of time.
2. Select the best p pages to present to the user, using some selection heuristic (which may be the same as the search heuristic).
3. Receive an evaluation from the user for each page presented.
4. Update the search and selection heuristics according to this feedback.

*This work was supported in part by the NSF/ARPA/NASA Digital Library project (NSF IRI-9411306), and in part by ARPA grant F49620-94-1-0900.

1. http://www.service.com/PAW/thisweek/sports/1994-Jun-3.new-and-recommended.html
2. http://www.commerce.digital.com/palo-alto/chamber-of-commerce/events/wCup/home.html
3. http://www.service.com/PAW/thisweek/sports/1994-Jun-3.sports-shorts.html
4. http://www.atm.ch.cam.ac.uk/sports/wc94.html
5. http://www.cedar.buffalo.edu/khoub-s/WC94.html

⋮

Figure 1: First five entries from a sample top-ten list produced by a student's program.

Search

There is a clear mapping between the problem of searching the web and the classic AI search paradigm. Each page of the web is a node, and the hypertext links to other pages are the edges of the graph to be searched. For simplicity we will only consider textual HTML pages, and ignore the multimedia information available.

In typical AI domains a good heuristic will rate nodes higher as we progress towards some goal node. In the web domain, the heuristic models how interesting a page is to a user. There is not necessarily any correlation with the heuristic values of nearby pages. However, we assume that it will be beneficial to expand nodes with higher heuristic values.

Clearly the web is too large to search exhaustively. Furthermore, pages will be changing, appearing and disappearing as the search is taking place. The simplest search technique we have looked at is a best-first strategy, with a resource limit to restrict the number of pages the agent could access on each cycle.

Features of Documents

We need to extract some features from each page found so that we can evaluate a search heuristic, and to provide a basis for the learning component. Various approaches which have been tried. The most promising uses the *vector space* information retrieval paradigm, where documents are represented as vectors (Salton & McGill 1983). Assume some dictionary vector $\vec{D}$, where each element d_i is a word. Each document then has a vector $\vec{V}$, where element v_i is the weight of word d_i for that document. If the document does not contain d_i then $v_i = 0$.

Word weights can be determined using a TFIDF[1] scheme, which calculates the "interestingness" value of a word based on how unusual it is. In the simplest case, the weight v_i of a word d_i in a document T is given by:

$$v_i = tf(i) \cdot \log \frac{n}{df(i)}$$

where $tf(i)$ is the number of times word d_i appears in document T (the *term frequency*), $df(i)$ is the number of documents in the collection which contain d_i (the *document frequency*) and n is the number of documents in the collection.

Using this measure we can extract the best words from each document to incorporate into a representation of the users interests, and this in turn can be the set of features for the learning subsystem. One way to do this would be to maintain a vector $\vec{M}$ for each user, where the weight for each word corresponded to the user's preferences. The search heuristic for a document with vector $\vec{V}$ could then be evaluated by simply taking the dot product $\vec{V}.\vec{M}$.

Learning Algorithms

The information supplied to the learning subsystem consists of a number of documents, each with a certain number of keywords picked out (those with the highest weights), and also an associated evaluation supplied by the user.

Its task is to update the weights for words in $\vec{M}$, and potentially apply some decay function. One way to do this is to use an instance-based scheme and attempt to maintain several prototype points in the feature space that define interesting documents (this would entail actually using several $\vec{M}$'s). An even simpler method is to increment or decrement the weights in $\vec{M}$ directly according to the users feedback.

User Interface

We intend to use HTML forms as the primary way of accessing the agent. In this way the system will be usable from any point on the Internet, merely requiring an appropriate browser. In addition, we envisage that in coming years some of the main beneficiaries of systems such as this will be users of mobile computers, linked to wireless networks. In this situation it is imperative that the large information space being accessed is somehow condensed to minimize communications bandwidth and screen space. The limitations of mobile computers primarily affect the ability to browse, not to search (where both the request and the resulting data can be quite small). To this end, a textual e-mail based interface to the system will also be provided, to be usable from a mobile computer linked to a wireless e-mail service.

[1] Term Frequency × Inverse Document Frequency

Early Experiments: Class Project

The outline just described roughly corresponds to a project set for the CS227 (*AI Techniques in Prolog*) class at Stanford University in the spring of 1994.

In order to provide a controlled and safe test-bed for the students' agents, the first task was to cooperatively construct a self-contained "mini-web" of interlinked pages. Although the mini-web contained only 240 nodes and 850 links, it was surprisingly easy to experience the well known "lost in hyperspace" feeling.

Early assignments included building a map of the entire mini-web[2] and inventing heuristics for best-first search, which were tested by searching for pages containing specific keywords. Eventually learning components were also incorporated, creating complete agents similar to the outline originally described.

After testing the agents on the mini-web, the final for the class involved sending them out into the real web to see what they could find. A trial over twenty simulated days was run for each agent (each day actually being ten minutes of real time), with "daily" feedback from a user. The system presented 5 new pages each day, and at the end of the twenty days supplied an ordered "top-ten" list of what it considered the most interesting pages found.

Features

Features ranged from the TFIDF scheme as described, to schemes where words inside HTML tags were rated higher[3], to using structural features such as the length of a document or the number of pictures it contained.

Search

Most of the projects based their search on a standard best-first algorithm, with bounds imposed on the memory and time used for each search.

Learning

Learning methods included a neural network approach, variants on nearest neighbor schemes, and various simpler weight updating strategies.

Results

Given the short time-frame, the systems produced were remarkably successful. However, due to the informal nature of the experiments, only anecdotal results can be reported. In one experiment, a user gave high scores to pages which related to soccer and the World Cup. After the twenty days of feedback, the system presented a top-ten list where the top five pages were World Cup related (figure 1). In another experiment, a preference for Japan-related pages led to a top-ten list containing only such pages. In fact most of the testers reported similar successes.

[2] See `http://robotics.stanford.edu/people/marko/fridge-door/MiniWeb_Map_2x2.ps` for an example.

[3] E.g. in HTML `<B>this</B>` would appear in bold face, and thus would be assumed to be more indicative of the document content.

Current Implementation

Recently we have created a more stable implementation, which we have used to perform an initial experiment. The system is intended to run one cycle every night, presenting users with fresh output each morning.

Features

Before extracting keywords we run the pages through a number of parsing steps common in information retrieval applications: we remove HTML markup tags, remove *stop words* (words so common as to be useless as discriminators, like **the**) and then *stem* the remainder of the words. This reduces words to their 'stems', and thus diminishes redundancy. For instance, **computer**, **computers**, **computing** and **computability** all reduce to **comput**. We use the Porter suffix-stripping algorithm (Porter 1980), as implemented in (Frakes 1992).

Word weights are calculated using a more sophiticated TFIDF scheme which normalizes for document length, following recommendations in (Salton & Buckley 1987):

$$v_i = \frac{\left(0.5 + 0.5\frac{tf(i)}{tf_{max}}\right)\left(\log\frac{n}{df(i)}\right)}{\sqrt{\sum_{d_j \in T}\left(\left(0.5 + 0.5\frac{tf(j)}{tf_{max}}\right)^2\left(\log\frac{n}{df(j)}\right)^2\right)}}$$

where tf_{max} is the maximum term frequency over all words in T. The document frequency component is calculated using a fixed dictionary of approximately 27,000 words gathered from the web and then stemmed. Currently we include only the 10 highest-weighted words in a document vector, for computational reasons. Salton and Buckley (1987) show that restricting the number of words has only a minor effect on the performance of a retrieval system.

Search

A fairly standard best-first search is used. It has been slightly modified it so that it will halt after using up its limit of CPU time, output the best pages found so far, and be ready to resume searching from the same point the next time it is executed.

Learning

Each page $\vec{V_i}$ will be viewed by the user and receive an evaluation e_i (an integer in the range $[-5, +5]$). Given this information we update the weights of $\vec{M}$ by a simple addition:

$$\vec{M} \leftarrow \vec{M} + \sum_{i=1}^{p} e_i \vec{V_i}$$

In the IR literature this is referred to as *relevance feedback* (Rocchio 1971).

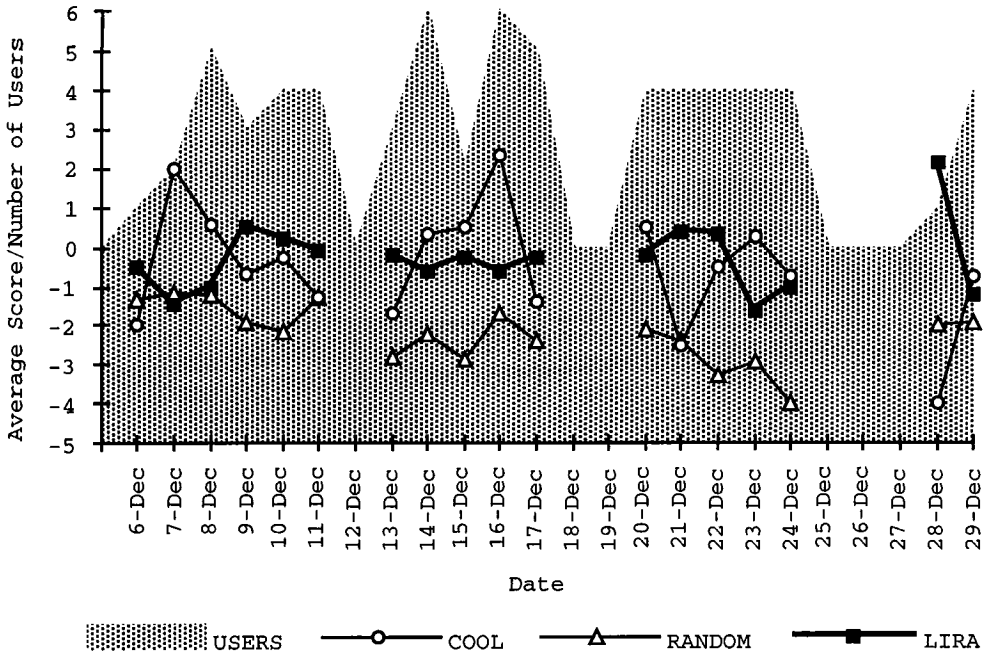

Figure 2: Comparison of our system (**LIRA**) against random (**RANDOM**) and human-selected (**COOL**) pages. The vertical axis shows both the number of users each day and the average score for each source of pages. The gaps indicate holidays or periods when the system was down for debugging.

Results

So far we have performed only a preliminary experiment, recording the evaluation scores given by users each day.

We have attempted to compare the pages produced by our system with both human-selected and randomly selected pages. Each day a separate process takes six pages produced by our system, three random pages and one human-selected page. These are presented to the user in a random order. In this way neither the user nor the experimenter know which page comes from which source. We keep a log of the evaluations received for each of the three sources.

For the human-selected page we make use of the "Cool Site of the Day" link (Davis 1994). Note that this is not tuned to a particular user. The random pages are selected from a static list of several hundred thousand URLs, checked for accessibility. The system does **not** use the evaluations of these control pages to update its weights.

Figure 2 shows the results collected during the first 24 days of operation of the system.

Analysis

Given the short duration of this experiment and the limitations of our first implementation, the results are very encouraging. After the first couple of days, our system's performance becomes consistently better than the randomly-selected pages. It beats the human-selected page half of the time. The greater variability in the score for the human-selected page is probably due to the fact that it is only a single page.

Our prediction was that the scores given to pages from our system would rise slowly, as it learned the preferences of individual users. However, it appears to stay fairly constant. On the face of it, this suggests that apart from a short initial period adaption does not help. However, we believe this actually highlights a difficulty in our experimental methodology. Over time users tend to normalize their scores, so that the same page will receive a different score if it is presented amongst worse pages than if it is presented amongst better pages. Furthermore, users will consciously attempt to influence the algorithm by giving minimum scores to pages relating to topics they are no longer interested in, despite giving pages on those topics high scores in the past.

It has been difficult to advise users on evaluating pages. For instance, users may do some exploration of their own starting from a page provided by the system, and discover something they find interesting which they would not have found otherwise. Should this have a bearing on the evaluation of the page the system provided? Ideally the system would be able to accept feedback on pages which it had not suggested.

Since we anticipate the use of this system over a longer period, this short experiment provides only a promising initial glimpse at its utility. Apart from re-

porting various bugs, the only negative reaction from the test users related to the systems' tendency to focus on one particular set of interests, so that the pages returned would cover a single topic rather than a range.

Related Work

There has been much recent work in a similar vein where the aim is to build an index of the web, and then allow users to search that index: (Mauldin & Leavitt 1994), (Pinkerton 1994). In this other work the system is supporting a directed search, initiated by the user. There is a possibility of an interesting symbiosis between these indexes and our searcher: the searcher could potentially make use of the indexes when searching, and the pages it produced could, if publicly accessible, be indexed by the indexer. The Harvest framework (Bowman *et al.* 1994) is more general and allows indexes and search engines to be created in a modular way.

Another related area of work attempts to automatically filter incoming information, notably (Maes & Kozierok 1993), (Lashkari, Metral, & Maes 1994). In this case, the system is not contributing new information but attempting to organize the existing flow to the user. This kind of application is inherently more risky. For instance, an inadvertently deleted mail message could have disastrous consequences. Thus it is more important not only that the agent have a model of the user but also that the user has a model of the agent, in order to build up trust.

The SIFT system (Yan & Garcia-Molina 1995) shares our goal of continuously informing the user of new information. In contrast, however, SIFT requires that users submit a profile which represents their interests, and then allows them to modify this by relevance feedback.

Although we borrow many techniques from the field of IR, our domain is quite different: there is no search query provided by the user, the collection of documents has not been indexed in advance, and operation is over longer periods of time.

Future Work

One extension of this work we intend to pursue would allow communication and cooperation between instances of the system running on behalf of different users. Since the output of a given system is just another web page, part of the required mechanism already exists. One can imagine systems not tied to a particular user, whose narrow focus of interest could form a specialized information resource for a group of users. Like the designers of the Tapestry system (Goldberg *et al.* 1992), we envision that the shared interests among a group of users would serve to cut down on duplicated searching.

Conclusions

We have described some preliminary designs and encouraging experimental results for agents which discover interesting information on the Internet, adapting to a particular user. The domain appears to be amenable to applying AI techniques, making it interesting not only from a research standpoint, but also as a educational tool.

Acknowledgments

We would like to thank last year's CS227 students, all the members of the Nobotics research group who participated in the experiment described (and gave lots of helpful advice), Glenn Davis for allowing the use of his daily "cool site" selection and Michael Mauldin for allowing the use of a list of URLs and various word frequency lists gathered by the Lycos system.

References

Bowman, C. M.; Danzig, P. B.; Hardy, D. R.; Manber, U.; and Schwartz, M. F. 1994. Harvest: A scalable, customizable discovery and access system. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado—Boulder.

Davis, G. 1994. Cool site of the day. `http://www.infi.net/cool.html`.

Frakes, W. B. 1992. Stemming algorithms. In Frakes, W. B., and Baeza-Yates, R., eds., *Information Retrieval Data Structures and Algorithms.* Englewood Cliffs, NJ: Prentice Hall, Inc. 131–160.

Goldberg, D.; Nichols, D.; Oki, B. M.; and Terry, D. 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 35(12):61–70.

Lashkari, Y.; Metral, M.; and Maes, P. 1994. Collaborative interface agents. In *Proccedings of the 12th National Conference on Artificial Intelligence.*

Maes, P., and Kozierok, R. 1993. Learning interface agents. In *Proccedings of the 11th National Conference on Artificial Intelligence.*

Mauldin, M. L., and Leavitt, J. R. 1994. Web-agent related research at the CMT. In *Proceedings of the ACM Special Interest Group on Networked Information Discovery and Retrieval.*

Pinkerton, B. 1994. Finding what people want: Experiences with the WebCrawler. In *The Second International WWW Conference: Mosaic and the Web.*

Porter, M. 1980. An algorithm for suffix stripping. *Program* 14(3):130–137.

Rocchio, Jr., J. 1971. Relevance feedback in information retrieval. In *The Smart System—Experiments in Automatic Document Processing.* Englewood Cliffs, NJ: Prentice Hall Inc. 313–323.

Salton, G., and Buckley, C. 1987. Term weighting approaches in automatic text retrieval. Technical Report 87-881, Cornell University, Department of Computer Science.

Salton, G., and McGill, M. J. 1983. *An Introduction to Modern Information Retrieval.* McGraw-Hill.

Yan, T. W., and Garcia-Molina, H. 1995. SIFT—a tool for wide-area information dissemination. In *Proceedings of the USENIX Technical Conference.*