Stanford University Digital Library Project
Working Paper SIDL-WP-1995-0023

Please Do Not Distribute

An Adaptive Agent for Automated Web Browsing

Marko Balabanović* Yoav Shoham Yeogirl Yun

{marko,shoham,yygirl}@cs.stanford.edu
Department of Computer Science,
Stanford University,
Stanford, CA 94305, USA

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 learns to browse the Internet on behalf of a user. 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. After demonstrating that our system is able to learn a model of a user with a single well-defined interest, we present an initial experiment where 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.

1 Introduction

It is becoming a cliché to talk about the explosion of information available on the Internet, and the corresponding increase in usage. This is particularly true of the World-Wide Web [Berners-Lee *et al.*, 1992] and its associated browsers, which provide easy access to a wider audience. The inevitable result of this growth is that current technologies for accessing information on the Web are inadequate. New users of the Web will simultaneously experience the excitement of boundless information and the frustration of trying to actually find anything.

There has been much work on software systems (often called *agents*) to help reduce this information overload. We have seen systems which reduce the amount of information coming to a user by *filtering*, and we have seen *indexers* which provide convenient ways to search the vast information spaces to find a specific item. We propose a complementary system, which will help users keep abreast of new and interesting things. Rather than supporting the *searching* task, we are supporting the *browsing* task, often referred to as *surfing*. A defining aspect of the surfing activity is that users do not have a specific goal in mind. Instead they are hoping to increase their awareness of the "fringes" of the Web.

*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.

Undoubtedly most readers are already familiar with the Web, and we apologize to them for this brief description. The Web is essentially a collection of electronic documents accessible via a protocol (HTTP) which allows uniform access to varied resources available on the Internet. It has become very popular due to readily available easy-to-use graphical browsers (e.g., NCSA Mosaic, Netscape Navigator). All data is presented uniformly as a series of pages. Highlighted phrases (*links*) can be clicked on in pages created with the HTML mark-up language, allowing a hypertext jump to a new page. A uniform resource locator (URL) is the address of a page.

Our system 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. The documents are found using a heuristic search on the Web, with the heuristic changing as feedback from the user is received.

Clearly this is a task for machine learning: the users do not at any stage specify a search query, but merely give feedback. And unlike an e-mail filtering application, for instance, where the computer could delete an important message, there is no serious penalty for mistakes. We present only new information the user is unlikely to have seen otherwise. Provided the amount of information is not excessive, this is a no-lose proposition for the user.

The domain described is an appropriate testing ground for basic AI techniques, especially search and machine learning algorithms. The problem we tackle was originally set as a term project to an AI programming class at Stanford University, as described in [Balabanović and Shoham, 1995].

In the remainder of this paper we describe our current prototype implementation, and give some early results from initial experiments.

2 Implementation

2.1 Overview

The system runs in discrete cycles. The behavior of one cycle can be summarized as follows:

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.

One cycle is run per user per day, so that a user has a fresh page of output waiting every morning.

The system has been written in a mixture of Python and C++. The following sections describe the most important aspects of our implementation.

2.2 Feature Extraction

Some features need to be extracted from each page found in order to provide a basis for computing the search heuristic. The approach we have taken is to attempt to extract a fixed number of keywords from each document: we assume we can represent the users' interests purely with keywords and associated weights. Although this has obvious limitations, in practice many information retrieval systems use single-word schemes successfully, and in any case this is a good place to start.

In the *vector space* information retrieval paradigm documents are represented as vectors [Salton and 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 $v_i = 0$.

In the typical information retrieval setting there is a collection of documents from which an inverted index is created. Particular documents can be retrieved based on a similarity measure between the document vector and a query vector, often just the cosine of the angle between them. In our framework the query vector is analogous to the system's model of the user, or *user profile*, which we call $\vec{M}$. The score for a page can be calculated by measuring how well it matches this profile, which is just a comparison between the page's vector and $\vec{M}$.

In order to create the vector representation of a Web page we parse it, extract individual words (ignoring HTML mark-up 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 decreases 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].

We calculate word weights using a TFIDF[1] scheme, normalizing for document length, following recommendations in [Salton and Buckley, 1988]. The weight v_i of a word d_i in a document T is given by:

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

where $tf(i)$ 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*), n is the number of documents in the collection and tf_{max} is the maximum term frequency over all words in T.

The document frequency component is calculated using a fixed dictionary words gathered from the Web and then stemmed—a total of approximately 27,000 stems.

For computational reasons the prototype described here used vectors truncated to contain only the 10 highest weighted terms. Recent experiments have shown that both retaining many terms for massive query expansion [Buckley *et al.*, 1995] and carefully selecting a smaller number of terms [Yochum, 1995] can yield comparably good results. We intend to perform our own set of experiments to see what works best for this domain.

2.3 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. 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 heuristic values of nearby pages. However, we assume it is beneficial to expand nodes with high heuristic values. The first experiment (section 3.1) shows that this assumption does hold for some heuristic functions.

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

The search heuristic evaluates a score for each page by simply taking the dot product $\vec{V}.\vec{M}$, where $\vec{V}$ represents the page and $\vec{M}$ the current user profile. Note that this means we can never learn an exclusive-or function (e.g. the user is interested in both children and baseball, but not little league).

In the current prototype, the first time a search is performed for a user the profile $\vec{M}$ is initialized to be empty (so that search is initially random). The profile could also be initialized by scanning a user's Web browser history file, which contains information about previously visited pages, or a user's "hot-list" of Web links, which contains links the user wants to be able to find easily in the future.

[1] Term Frequency × Inverse Document Frequency

In our implementation we consider only plain text or HTML pages. A proxy HTTP server provides us with uniform access to pages which have been provided using different protocols (e.g. Gopher, WAIS, Usenet news, FTP).

2.4 Selection and Weight Update

When the search has used up its allocated time it outputs the p highest-scoring pages found according to the selection heuristic. This primarily uses the scores from the search heuristic, but in addition attempts to ensure a better mixture of pages by selecting at most one from each site. If a page scores very highly, it is likely that related pages at the same site will also score highly. However, showing users pages which are children or siblings of each other in the graph both wastes their time and narrows the feedback available to the system. A second check performed prevents a user from seeing a page with the same contents twice over the course of the experiment.

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]. The update rule we use is equivalent to the "Ide regular" rule [Salton and Buckley, 1990] with additional weighting from the users' evaluations. In fact Salton and Buckley found that the "Ide dec-hi" rule generally gave better performance, where only one negative-scoring document is included. However the absence of an initial user-provided search query in our system increases the importance of negative feedback, and it is not clear that results from experiments in a traditional IR setting will carry over to our domain.

In machine learning terms this is a very simple variant of an exemplar-based scheme [Kibler and Aha, 1987], where we incrementally move a single prototype point and classify instances according to their distance from this point.

2.5 User Interface

The primary way of accessing the system is through HTML forms. In this way the system is usable from any point on the Internet, merely requiring an appropriate browser. Figure 1 shows an example screen.

3 Experiments

3.1 Learning an objective "interestingness" criterion

We have performed two very different experiments to test this system. In the first one we attempted to verify that the algorithm could converge on a user's interests if given a well-defined interestingness criterion. We chose the topic "music". The scoring strategy was as follows:

+5 for pages relating to music.

+2 for pages which although not directly related to music, looked as if they might lead to a music-related page.

-5 for pages unrelated to music.

The experiment was run for 16 iterations ("days") of the algorithm, with 20 minutes of CPU time on a Sun Sparc 10 allowed for each iteration. This corresponds to several hundred URLs accessed per iteration,

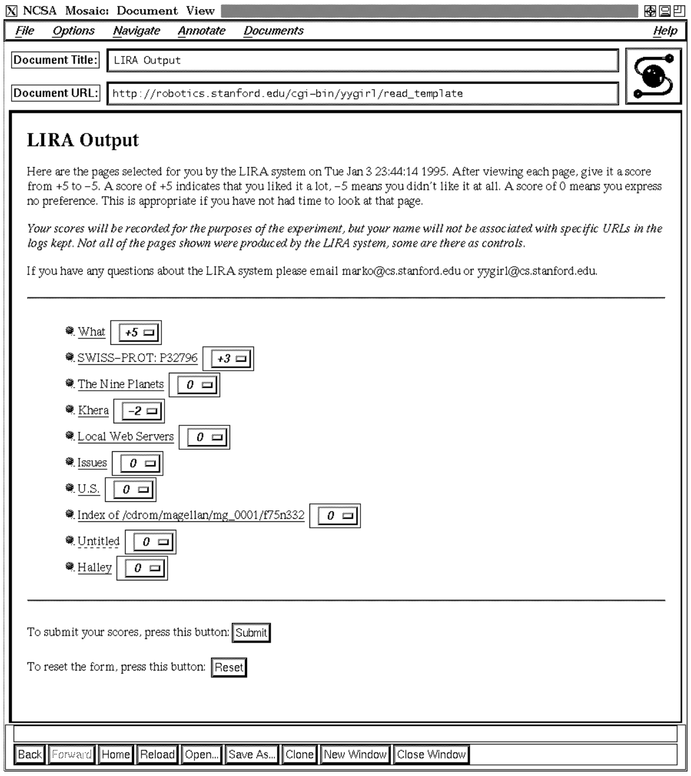

Figure 1: The interface to the system. The user clicks on a link to see the relevant page. A pop-up menu by each link allows the evaluation to be entered.

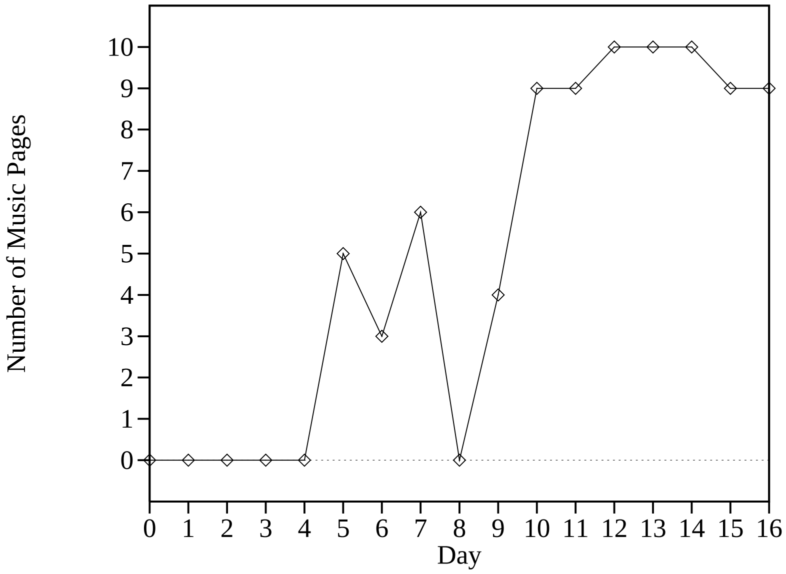

Figure 2: Results of an experiment where only music-related pages where rated highly.

depending on the network load. The results (Figure 2)[2] show that the system was able to successfully learn the concept of music-related pages, with 9 or 10 out of the 10 pages shown every day relating to music after the 10^{th} day.

music	19.7208
album	8.32765
song	6.08677
record	5.89186
band	5.56142
page	5.28602
young	4.90579
regga	4.13141
artist	3.66437
ska	3.47215
link	2.84541
list	2.31812
search	2.30645
art	2.21375
tour	2.0546
mail	1.90863
show	1.87562
sampl	1.75524
indi	1.70879
homepag	1.68255
databas	1.56257
number	1.45883
indielist	1.43027
sound	1.39552
hors	1.3842
collect	1.3754
email	1.28597
perform	1.26534
vibe	1.25311
radio	1.24483

Figure 3: The highest-weighted words and their weights from the user profile $\vec{M}$ after the end of the experiment to find music-related pages. These words have been stemmed, e.g. `regga` was originally `reggae`.

Figure 3 shows the highest-weighted words from the resulting profile $\vec{M}$. The first nine are clearly music related (`Page` and `Young` refer to the musicians Jimmy Page and Neil Young). The total vector contains 1,110 words.

Although this experiment demonstrates that the system is able to converge on such simple concepts, it is not indicative of an appropriate task. If the user is able to articulate a well-defined interest, it would be more efficient to enter it as a search query for one of the available indexes. However, typically users interests are not so easy to define, and their evaluations of pages will not be so sharply partitioned.

3.2 Learning subjective "interestingness" criteria

In the second experiment, for which we only have preliminary results, we attempted to measure how well the system did with a more natural interestingness criterion. The usual relevance-based IR measures of recall

[2] More details of this experiment are available on-line at `http://robotics.stanford.edu/people/marko/lira/demo1.html`

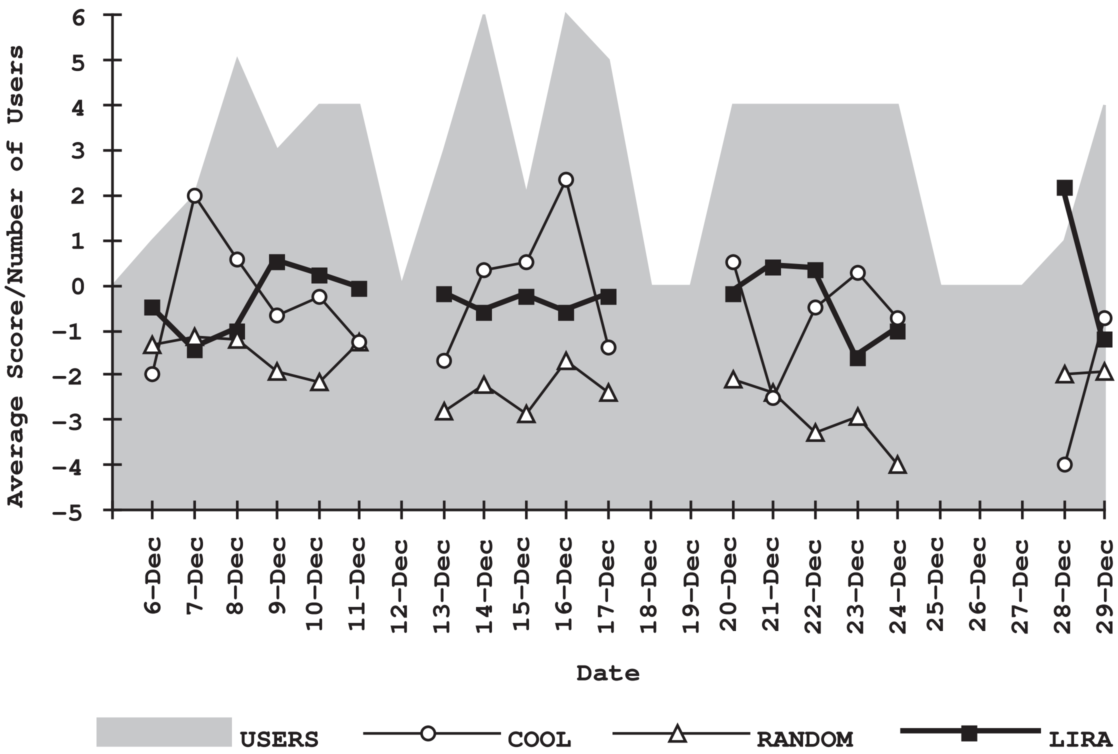

Figure 4: Comparison of our system (`LIRA`) against random (`RANDOM`) and human-selected (`COOL`) pages. The number of users each day and the average score for each source of pages are both shown on the vertical axis. The gaps indicate holidays or periods when the system was down for debugging.

and precision are difficult to apply in this setting, for two reasons: firstly, the browsing task implies the user has no set goal in mind, so the notion of relevance is not really appropriate; secondly, the collection being searched is not fixed but rather changing rapidly. It **would** be possible to use precision if we defined a relevance threshold somewhere in the range of the user's evaluation scores (e.g., all documents rated above 0 are relevant), but we have decided to exploit the finer-grained nature of the raw evaluation scores.

Our motivation was to convince ourselves of the feasibility of the overall approach. We decided a positive indicator for this would be if the system could equal or better the performance of a popular page on the Web where a single hand-picked "cool site" is featured every day. In this way our system could prove its value to users in comparison with an existing widely used resource. To provide a simple control we decided to also compare against randomly picked Web pages.

Each day a separate process took six pages produced by our system, three random pages and one human-selected page. These were presented to the user in a random order. In this way neither the user nor the experimenter knew which page comes from which source. A log was kept of the evaluations received for each of the three sources.

The human-selected page described is 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. The system was allocated 20 minutes of CPU time on a Sun Sparc 10 per user per day.

Figure 4 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 only a single such page was available.

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.

Once the overall feasibility of this approach has been established, we intend to conduct more formal experiments with a larger user population over a longer period of time. This will also allow comparisons between different feature extraction, searching and learning schemes.

4 Limitations

Before discussing our plans for future work, we summarize the limitations of the present prototype:

1. The features used are just keywords. We have ignored things such as the length and structure of pages, the number of links and the sites from which they come.

2. Only textual pages are examined, thus eliminating many of the resources available on the Web. As graphic designers and artists start to take over the production of Web pages from computer scientists, an increasing amount of text is embedded in images, rendering it inaccessible to our system. The problems involved in assessing the content of the available multimedia resources are even greater.

3. We do not solve the problems associated with the transience of Web pages: some are fairly static, others have a limited life and yet others are generated afresh for each request. Currently a URL is never re-visited, which is a temporary solution at best.

4. The evaluation system makes it hard for users to be consistent, and so makes the interpretation of results more problematic (as explained in section 3.2).

5. The pages returned by the system are often very similar to each other, as pointed out by many of the users.

6. By requiring one search process per user, the system does not scale well, neither in terms of computation time nor network load.

5 Future Work

The prototype described has been very useful in showing feasibility, but in order to perform bigger experiments a system which scales up better is required. We envisage a system with multiple *search agents* which collect Web pages in a variety of ways. These would include the three types of agents described already, namely best-first search, human selection of pages and random selection of pages. Further comparisons could be performed with agents which use existing Web indexes rather than doing their own search, although usually the queries would need to be far smaller than our current user profiles.

The search agents would deposit pages in a central repository, where individual users' *personal agents* would retrieve those which best matched their own users profile. Feedback would go both to the personal agent (who would keep track of a users profile) and to the respective search agents (who would now be serving groups of users rather than individuals, and hopefully taking advantage of their shared interests). It would be easy to measure the success of search agents by the number of their pages which were selected to be viewed by users, and the resulting evaluations.

We intend to use this new system to investigate issues of scale, in particular to discover how performance degrades as the number of users is increased for a fixed pool of search agents.

An intriguing thought for the future involves the different uses to which we can put the user profile. Over a period of time this will become an increasingly accurate predictor of the user's interests. One can imagine using this valuable resource for many tasks: filtering incoming e-mail, picking out interesting Usenet news articles, creating personalized newspapers, automatic selection of goods to browse in an on-line shop, and so on. It will be important as adaptive agents proliferate to ensure that users retain control of their own profiles, and that all of a user's agents can share this information.

6 Related Work

Recently there has been a proliferation of systems to assist a user who is attempting for find information on the World-Wide Web. Most of these systems can be classified into one of four categories:

Indexes

Web indexes are essentially retrospective information retrieval systems[3]. The index is built over a collection of documents found by a Web search process, which typically searches exhaustively rather than to fulfill a particular query. Examples of publicly available indexes are Lycos [Mauldin and Leavitt, 1994] and WebCrawler (originally described in [Pinkerton, 1994]). It is not common for Web indexes to provide a relevance feedback facility.

Information filtering systems

Although information filtering shares many techniques with retrospective information retrieval, three primary differences were elucidated by Belkin and Croft [1992]. Firstly, information filtering user profiles represent long-term interests, while IR queries typically represent a short-term goal, which can be satisfied by retrieving a particular set of documents. Secondly, IR applications assume that the corpus of documents does not change often, whereas information filtering assumes a constant stream of time-sensitive documents. Lastly, filtering is the act of removing irrelevant items from this stream, whereas IR is the act of finding relevant items in the database.

Information filtering as been applied to e-mail [Maes and Kozierok, 1993] and Usenet news groups [Sheth and Maes, 1993; Yan and Garcia-Molina, 1995], using relevance feedback to build user profiles. Several commercial services now offer simple filtering systems for proprietary collections of documents (e.g. Ziff-Davis' *Personal View*, the San Jose Mercury News' *NewsHound*).

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.

[3] As characterized by the "ad-hoc" queries in the TREC conferences [Harman, 1995].

Social or collaborative filtering systems

Social filtering systems share our goal described in section 5 of capitalizing on the shared interests of users. The key process here is to match up similar users, and make use of evaluations or annotations supplied by others. Examples of such systems are Tapestry [Goldberg *et al.*, 1992] for e-mail, Firefly[4] for music and Webhunter [Lashkari, 1995] for Web pages.

Due to the recency of this paradigm there has not yet been sufficient experimentation to allow meaningful comparisons with the other approaches described.

Assisted browsing systems

Rather than providing the user with selected pages, some systems instead assist the user in their browsing. The WebWatcher system [Armstrong *et al.*, 1995] requires the user to state a particular goal, and then interactively offers advice on which link to follow next given the page the user is looking at. The system learns by keeping track of whether its advice is followed, and also from asking the user to signal success or failure when their search has been completed.

Our application borrows many techniques from the field of IR, and is similar to information filtering in that user profiles represent long-term interests. However it differs from both in that the user profile is used to actively search the dynamic document collection, rather than look up in a precalculated index or sift through an existing stream of documents.

The routing task as defined by the TREC conferences is also related to our application. However routing is a batch task: the system starts with a query and a fixed set of ranked training documents, and then attempts to rank a fixed set of test documents. Our system attempts to learn incrementally, and the set of documents seen on each iteration depends on feedback received in previous iterations.

Our system is exploring the effectiveness of "pure" relevance feedback, as, unusually for an IR application, there is no query ever supplied by the user. There have been some encouraging results for relevance feedback recently. For instance, in an information filtering experiment which attempted to select appropriate Bellcore technical memos for a group of employees, Foltz and Dumais [1992] found that using profiles automatically created from documents ranked by the user as relevant actually outperformed profiles hand-crafted by users. More recently experiments in the TREC-3 conference showed similar results, with automatically generated profiles for the routing task outperforming even skilled human searchers using an interactive IR system [Harman, 1995].

The absence of queries in our "automated browsing" paradigm can be beneficial to users in a number of ways: they no longer have to grapple with different interfaces or query languages, they do not need to have any understanding of the vocabulary or other properties of the space of documents being searched, and indeed they do not even need to be able to articulate what they are interested in. Any system using relevance feedback will exhibit these advantages to some degree [Salton and Buckley, 1990], so it is surprising that so few of the available Web indexes include this feature.

7 Conclusions

Our research was motivated by two beliefs:

1. The rapid expansion of the World-Wide Web necessitates new tools for information discovery.
2. The Web provides an excellent domain for experimentation with AI techniques.

We have provided a system for exploration of the Web that adapts to individual users, and have shown the following:

[4] `http://www.agents-inc.com/`

- Even a simple relevance feedback rule can play a useful role as part of such a tool, adapting to a particular user starting from scratch.
- A heuristic search of the World-Wide Web can recover pages that users rate as interesting.
- Term-weighting formulae from the IR literature can successfully be used to extract salient features from Web pages for use by an AI system.

We have verified the feasibility of our system in a comparison against random and human-selected pages. It consistently scored higher than the random pages, and beat the human-selected page half of the time.

Additionally we feel we have identified a valuable environment for the teaching of AI techniques (such as heuristic search and machine learning), which, although it is a real-world setting, carries none of the difficulties of sensing or control so often associated with such domains.

Acknowledgments

We would like to thank 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

[Armstrong *et al.*, 1995] Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. WebWatcher: A learning apprentice for the World-Wide Web. In *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogenous, Distributed Resources*, Stanford, CA, March 1995.

[Balabanović and Shoham, 1995] Marko Balabanović and Yoav Shoham. Learning information retrieval agents: Experiments with automaed web browsing. In *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogenous, Distributed Resources*, Stanford, CA, March 1995.

[Belkin and Croft, 1992] Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, December 1992.

[Berners-Lee *et al.*, 1992] T. Berners-Lee, R. Cailliau, J-F. Groff, and B. Pollermann. World-Wide Web: The information universe. *Electronic Networking: Research, Applications and Policy*, 1(2):52–58, Spring 1992.

[Buckley *et al.*, 1995] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using SMART: TREC-3. In *Proceedings of the 3rd Text REtrieval Conference*, Gaithersburg, MD, November 1995.

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

[Foltz and Dumais, 1992] Peter W. Foltz and Susan T. Dumais. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35(12):51–60, December 1992.

[Frakes, 1992] William B. Frakes. Stemming algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval Data Structures and Algorithms*, pages 131–160. Prentice Hall, Inc., Englewood Cliffs, NJ, 1992.

[Goldberg *et al.*, 1992] David Goldberg, David Nichols, Brain M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[Harman, 1995] Donna Harman. Overview of the third Text REtrieval Conference (TREC-3). In *Proceedings of the 3rd Text REtrieval Conference*, Gaithersburg, MD, November 1995.

[Kibler and Aha, 1987] Dennis Kibler and David W. Aha. Learning representative exemplars of concepts: An initial case study. In *Proceedings of the 4th International Workshop on Machine Learning*, Irvine, CA, 1987.

[Lashkari, 1995] Yezdi Lashkari. Feature guided automated collaborative filtering. Master's thesis, MIT Department of Media Arts and Sciences, July 1995.

[Maes and Kozierok, 1993] Pattie Maes and Robyn Kozierok. Learning interface agents. In *Proccedings of the 11th National Conference on Artificial Intelligence*, pages 459–465, Washington, DC, July 1993.

[Mauldin and Leavitt, 1994] Michael L. Mauldin and John R.R. Leavitt. Web-agent related research at the CMT. In *Proceedings of the ACM Special Interest Group on Networked Information Discovery and Retrieval (SIGNIDR-94)*, McLean, VA, August 1994.

[Pinkerton, 1994] Brian Pinkerton. Finding what people want: Experiences with the WebCrawler. In *The Second International WWW Conference: Mosaic and the Web*, Chicago, IL, October 1994.

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

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

[Salton and Buckley, 1988] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[Salton and Buckley, 1990] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, June 1990.

[Salton and McGill, 1983] Gerard Salton and Michael J. McGill. *An Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[Sheth and Maes, 1993] Beerud Sheth and Pattie Maes. Evolving agents for personalized information filtering. In *Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications*, Orlando, FL, March 1993.

[Yan and Garcia-Molina, 1995] Tak W. Yan and Hector Garcia-Molina. SIFT—a tool for wide-area information dissemination. In *Proceedings of the USENIX Technical Conference*, pages 177–186, New Orleans, LA, January 1995.

[Yochum, 1995] Julian A. Yochum. Research in automatic profile creation and relevance ranking with LMDS. In *Proceedings of the 3rd Text REtrieval Conference*, Gaithersburg, MD, November 1995.