WebWatcher: A Learning Apprentice for the World Wide Web

Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell
School of Computer Science
Carnegie Mellon University
March 19, 1997

Abstract

We describe an information seeking assistant for the world wide web. This agent, called WebWatcher, interactively helps users locate desired information by employing learned knowledge about which hyperlinks are likely to lead to the target information. Our primary focus to date has been on two issues: (1) organizing WebWatcher to provide interactive advice to Mosaic users while logging their successful and unsuccessful searches as training data, and (2) incorporating machine learning methods to automatically acquire knowledge for selecting an appropriate hyperlink given the current web page viewed by the user and the user's information goal. We describe the initial design of WebWatcher, and the results of our preliminary learning experiments.

1 Overview

Many have noted the need for software to assist people in locating information on the world wide web. This paper presents the initial design and implementation of an agent called WebWatcher that is intended to assist users both by interactively advising them as they traverse web links in search of information, and by searching autonomously on their behalf. In interactive mode, WebWatcher acts as a *learning apprentice* [Mitchell et al., 1985; Mitchell et. al., 1994], providing interactive advice to the Mosaic user regarding which hyperlinks to follow next, then learning by observing the user's reaction to this advice as well as the eventual success or failure of the user's actions. The initial implementation of WebWatcher provides only this interactive mode, and it does not yet possess sufficient knowledge to give widely useful search advice. In this paper we present WebWatcher as a case study in the design of web-based learning agents for information retrieval. We focus in particular on the interface that enables WebWatcher to observe and advise any consenting user browsing any location on the web, and on results of initial experiments with its learning methods.

Figure 1: Original page

2 WebWatcher

This section presents the design of WebWatcher through a scenario of its use. WebWatcher is an information search agent that is "invoked" by following a web hyperlink to its web page, then filling out a Mosaic form to indicate what information is sought (e.g., a publication by some author). WebWatcher then returns the user to (a copy of) the web page from which he or she came, and assists the user as they follow hyperlinks forward through the web in search of the target information. As the user traverses the web, WebWatcher uses its learned knowledge to recommend especially promising hyperlinks to the user by highlighting these links on the user's display. At any point, the user may dismiss WebWatcher, by clicking one of two indicators on the WebWatcher icon, indicating either that the search has succeeded, or that the user wishes to give up on this search.

The sequence of web pages visited by the user in

Figure 2: WebWatcher front door

Figure 3: Paper search form

Figure 4: Copy of original page with WebWatcher advice

Figure 5: Next page (user has followed WebWatcher's advice)

a typical scenario is illustrated in figures 1 through 5. The first screen shows a typical web page, [1] providing information about Machine Learning. Notice in the third paragraph, this page invites the user to try out WebWatcher. If the user clicks on this link, he or she arrives at the front door WebWatcher page (Figure 2), which allows the user to identify the type of information he seeks. In this scenario the user indicates that the goal is to locate a paper, so he is shown a new screen (Figure 3) with a form to elaborate this information request. Once completed, the user is returned to the original page (Figure 4), with WebWatcher now "looking over his shoulder". Notice the WebWatcher icon at the top of the screen, and the highlighted link (bracketed by the WebWatcher eyes icon) halfway down the screen. This highlighted link indicates WebWatcher's advice that the user follow the link to the *University of Illinois / Urbana (UIUC) AI / ML Page*. The user decides to select this recommended link, and arrives at the new web page shown in Figure 5, which contains new advice from WebWatcher. The search continues in this way, with the user directing the search and WebWatcher highlighting recommended links, until the user dismisses WebWatcher by clicking on "I found it" or "I give up".

From the user's perspective WebWatcher is an agent with specialized knowledge about how to search outward from the page on which it was invoked. While WebWatcher suggests which hyperlink the user should take, the user remains firmly in control, and may ignore the system's advice at any step. We feel it is important for the user to remain in control, because WebWatcher's knowledge may provide imperfect advice, and because WebWatcher might not perfectly understand the user's information seeking goal.

From WebWatcher's perspective, the above scenario looks somewhat different. When first invoked it accepts an argument, encoded in the URL that accesses it, which contains the user's "return address." The return address is the URL of the web page from which the user came. Once the user fills out the form specifying his or her information seeking goal, WebWatcher sends the user back to a copy of this original page, after making three changes. First, the WebWatcher banner is added to the top of the page. Second, each hyperlink URL in the original page is replaced by a new URL that points back to the WebWatcher. Third, if the WebWatcher finds that any of the hyperlinks on this page are strongly recommended by its search control knowledge, then it highlights the most promising links in order to suggest them to the user. It sends this modified copy of the return page to the user, and opens a file to begin logging this user's information search as training data. While it waits for the user's next step, it prefetches any web pages it has just recommended to the user, and begins to process these pages to determine their most promising outgoing hyperlink. When the user clicks on the next hyperlink, WebWatcher updates the log for this search, retrieves the page (unless it has already been prefetched), performs similar substitutions, and returns the copy to the user.

This process continues, with WebWatcher tracking the user's search across the Web, providing advice at each step, until the user elects to dismiss the agent. At this point, the WebWatcher closes the log file for this session (indicating either success or failure in the search, depending on which button the user selected when dismissing WebWatcher), and returns the user to the original, unsubstituted copy of the web page he is currently at.

The above scenario describes a typical interaction with the current WebWatcher. We plan to extend the initial system in several ways. For example, WebWatcher could be made to search several pages ahead, by following its own advice while waiting for the user's next input, in order to improve upon the quality of advice it provides. In addition, if it encounters an especially promising page while searching ahead, it might suggest that the user jump directly to this page rather than follow tediously along the path that the agent has already traversed.

3 Learning

The success of WebWatcher depends crucially on the quality of its knowledge for guiding search. Because of the difficulty of hand-crafting this knowledge, and because we wish for many different copies of WebWatcher to become knowledgeable about many different regions of the Web, we are exploring methods for automatically learning this search control knowledge from experience.

3.1 What Should be Learned?

What is the form of the knowledge required by WebWatcher? In general, its task is to suggest an appropriate link given the current user, goal, and web page. Hence, one general form of knowledge that would be useful corresponds to knowledge of the function:

$$LinkUtility : Page \times Goal \times User \times Link \rightarrow [0,1]$$

where $Page$ is the current web page, $Goal$ is the information sought by the user, $User$ is the identity of the user, and $Link$ is one of the hyperlinks found on $Page$. The value of $LinkUtility$ is the probability that following $Link$ from $Page$ leads along a shortest path to a page that satisfies the current $Goal$ for the current $User$.

In the learning experiments reported here, we consider learning a simpler function for which training data is more readily available, and which is still of considerable practical use. This function is:

$$UserChoice? : Page \times Goal \times Link \rightarrow [0,1]$$

[1] This is a copy of the web page http://www.ai.univie.ac.at/oefai/ml/ml-ressources.html, to which we have added the third paragraph inviting the user to invoke WebWatcher.

200 words Underlined	200 words Sentence	100 words Heading	$\approx$ 30 words User goal

Table 1: Encoding of selected information for a given *Page*, *Link*, and *Goal*.

Where the value of *UserChoice?* is the probability that an arbitrary user will select *Link* given the current *Page* and *Goal*. Notice here the *User* is not an explicit input, and the function value predicts only whether users tend to select *Link* – not whether it leads optimally toward to the goal. Notice also that information about the search trajectory by which the user arrived at the current page is not considered.

One reason for focusing on *UserChoice?* in our initial experiments is that the data automatically logged by WebWatcher provides training examples of this function. In particular, each time the user selects a new hyperlink, a training example is logged for each hyperlink on the current page, corresponding to the *Page*, *Goal*, *Link*, and whether the user chose this *Link*.

3.2 How Should Pages, Links and Goals be Represented?

In order to learn and utilize knowledge of the target function *UserChoice?*, it is necessary to first choose an appropriate representation for *Page* $\times$ *Goal* $\times$ *Link*. This representation must be compatible with available learning methods, and must allow the agent to evaluate learned knowledge efficiently (i.e., with a delay negligible compared to typical page access delays on the web). Notice that one issue here is that web pages, information associated with hyperlinks, and user information goals are all predominantly text-based, whereas most machine learning methods assume a more structured data representation such as a feature vector. We have experimented with a variety of representations that re-represent the arbitrary-length text associated with pages, links, and goals as a fixed-length feature vector. This idea is common within information retrieval retrieval systems [Salton and McGill, 1983]. It offers the advantage that the information in an arbitrary amount of text is summarized in a fixed length feature vector compatible with current machine learning methods. It also carries the disadvantage that much information is lost by this re-representation.

The experiments described here all use the same representation. Information about the current *Page*, the user's information search *Goal*, and a particular outgoing *Link* is represented by a vector of approximately 530 boolean features, each feature indicating the occurrence of a particular word within the text that originally defines these three attributes. The vector of 530 features is composed of four concatenated subvectors:

1. *Underlined words in the hyperlink.* 200 boolean features are allocated to encode selected words that occur within the scope of the hypertext link (i.e., the underlined words seen by the user). These 200 features correspond to only the 200 words found to be most informative over all links in the training data (see below.)
2. *Words in the sentence containing the hyperlink.* 200 boolean features are allocated to indicate the occurrence of 200 selected words within the sentence (if any) that contains *Link*.
3. *Words in the headings associated with the hyperlink.* 100 boolean features are allocated to indicate selected words that occur in the headings (if any) under which *Link* is found. This includes words occurring in headings at any level of nesting, as long as *Link* is within the scope of the heading. For example, in Figure 4, any of the words in the headings *Machine Learning Information Services* and *General ML Information Sources* may be used as features to describe the link that was highlighted.
4. *Words used to define the user goal.* These features indicate words entered by the user while defining the information search goal. In our experiments, the only goals considered were searches for technical papers, for which the user could optionally enter the title, author, organization, etc. (see Figure 3). All words entered in this way throughout the training set were included (approximately 30 words, though the exact number varied with the training set used in the particular experiment). The encoding of the boolean feature in this case is assigned a 1 if and only if the word occurs in the user-specified goal *and* occurs in the hyperlink, sentence, or headings associated with this example.

To choose the encodings for the first three fields, it was necessary to select which words would be considered. In each case, the words were selected by first gathering every distinct word that occurred over the training set, then ranking these according to their mutual information with respect to correctly classifying the training data, and finally choosing the top N words in this ranking.[2] Mutual information is a common statistical measure (see, e.g., [Quinlan, 1993]) of the degree to which an individual feature (in this case a word) can correctly classify the observed data.

Figure 1 summarizes the encoding of information about the current *Page*, *Link*, and *Goal*.

3.3 What Learning Method Should be Used?

The task of the learner is to learn the general function *UserChoice?*, given a sample of training data logged from users. In order to explore possible learning approaches and to determine the level of competence achievable by a learning agent, we applied the following four methods to training data

[2] The appendix lists the words selected by this procedure using one of our training sets.

collected by WebWatcher during 30 information search sessions:

- `Winnow` [Littlestone, 1988] learns a boolean concept represented as a single linear threshold function of the instance features. Weights for this threshold function are learned using a multiplicative update rule. In our experiments we enriched the original 530 attributes by a transformation. Each attribute a of an example vector was transformed into two attributes a,$\overline{a}$. One attribute is equivalent with the original, the other is its negation. After the learning phase we removed the threshold and used the output of the learned linear function as an evaluation for instances.
- `Wordstat` attempts to make a prediction whether a link is followed based directly on the statistics of individual words. For each feature in the $Page \times Goal \times Link$ vector, it keeps two counts: a count of the number of times this feature was set over all training examples ($total$), and a count of the number of times this feature was set *and* the instance was classified as positive (pos). The ratio $pos/total$ provides an estimate of the conditional probability that the link will be followed, given that this feature occurs. We experimented with various ways of combining these ratios. Of the approaches we tried, the one that worked best in our experiments, the results of which we report here, involves assuming that these single-word estimates are mutually independent. This assumptions allows us to combine individual estimates in a straightforward way. If $p_1, ..., p_n$ are the individual probabilities, and I is the set of indexes for which a bit is set in a given test vector, then the probability that the corresponding link was followed is determined by $1 - \prod_{i \in I}(1 - p_i)$.
- `TFIDF` with cosine similarity measure [Salton and McGill, 1983; Lang, 1995] is a method developed in information retrieval. In the general case at first a vector V of words is created. In our experiments it is already given by the representation described above. Every instance can now be represented as a vector with the same length as V, replacing every word by a number. These numbers are calculated by the formula $V_i = Freq(Word_i) * [log_2(n) - log_2(DocFreq(Word_i))]$, with n being the total number of examples, $Freq(Word_i)$ the number of occurrences of $Word_i$ in the actual example and $DocFreq(Word_i)$ the number of examples $Word_i$ appears in. The length of the vector is normalized to 1. Prototype vectors for each class of the target concept are created by adding all training vectors of this class. In our case we had a target concept with two classes: positive (link was followed by the user), and negative (link was not followed by the user). The evaluation of an instance is calculated by subtracting the cosine between the instance vector and the negative prototype vector from the cosine between the instance vector and the positive prototype vector.
- `Random` To provide a baseline measure against which to compare the learning methods, we also measured the performance achieved by randomly choosing one link on the page with uniform probability. The mean number of links per page over the data used here is 16, ranging from a minimum of 1 to a maximum of 300.

4 Results

In order to explore the potential of machine learning methods to automatically acquire search control knowledge for WebWatcher, we collected a set of data from 30 sessions using WebWatcher to search for technical papers. In each session the user began at the web page shown in Figure 1, and searched for a particular type of technical paper following links forward from there. Searches were conducted by three different users. The average depth of a search was 6 steps, with 23 of the 30 searches successfully locating a paper. Each search session provided a set of training examples corresponding to all the $Page \times Link$ pairs occurring on each page visited by the user.

4.1 How Accurately Can $UserChoice$? Be Learned?

Given the above representation and learning method, the obvious question is "How well can WebWatcher learn to advise the user?" To estimate the answer to this question, the available data was split into training and testing sets. Each learning method was applied to the set of training sessions and evaluated according to how frequently it recommended the hyperlink taken by the user in the separate testing sessions.

In order to obtain more statistically significant estimates of learning accuracy, the training data was separated into 29 training sessions and one test session, in each of the 30 possible ways. Each learning method was then applied to each training session collection and evaluated on the test session. The results of these 30 experiments were averaged. This procedure was run for each of the four learning methods.

Figure 6 plots the results of this experiment. The vertical axis indicates the fraction of test cases in which the user-selected hyperlink was among those recommended by the learned knowledge. The horizontal axis indicates the number of hyperlinks that the learner was allowed to recommend for each page. Thus, the leftmost point of each line indicates the fraction of cases in which the user chose the learner's highest-rated link. The second point to the left indicates the fraction of cases in which the user chose one of the two highest-rated links, and so on.

Notice that all three learning methods significantly outperform randomly generated advice. For

Figure 6: Accuracy of advice for different methods. The vertical axis indicates the fraction of pages for which the recommended hyperlinks included the link chosen by the user. The horizontal axis indicates the number of hyperlinks recommended per page. The worst case line shows the fraction of pages having N or fewer links total.

Figure 7: Increasing accuracy by reducing coverage. The vertical axis indicates the fraction of test pages for which the learner's top recommendation was taken by the user. The horizontal axis indicates the fraction of test cases covered by advice as the confidence threshold is varied from high confidence (left) to low (right).

example, `Winnow` recommends the user-selected link as its first choice in 30% of the test cases, and among its top three choices in 54% of the cases. Given the mean of 16 links per page in this data, random advice chooses the user-selected link only 6% of the time.

4.2 Can Accuracy be Improved by Sacrificing Coverage?

Some users may prefer that the agent provide more accurate advice, even if this requires that it make recommendations more sparingly. To determine the feasibility of increasing advice accuracy by reducing coverage, we experimented with adding a threshold on the confidence of the advice. For each of the learning methods considered here, the learner's output is a real–valued number that can be used to estimate its confidence in recommending the link. Therefore, it is easy to introduce a confidence threshold in each of these cases.

Figure 7 shows how advice accuracy varies with coverage, as the confidence threshold is varied. For high values of the confidence threshold, the agent provides advice less often, but can usually achieve higher accuracy. In this case, accuracy is measured by the fraction of test cases for which the learner's top ranked hyperlink is the link selected by the user. Thus, the rightmost points in the plots of figure 7 correspond exactly to the leftmost plots in figure 6 (i.e., 100% coverage).

Notice that the accuracy of `Winnow`'s top-ranked recommendation increases from 30% to 53% as its coverage is decreased to a more selective 10% of the cases. Interestingly, while `Wordstat`'s advice is relatively accurate in general, its accuracy degrades drastically at higher thresholds. The presence of features which occur very infrequently in the training set, resulting in poor probability estimates, and the inter-feature independence assumption, which the training set by no means justifies, appear to account for this phenomenon.

5 Conclusions

Software assistance is already needed to deal with the growing flood of information available on the WWW. The design of WebWatcher is based on the assumption that knowledge about how to search the web can be learned by interactively assisting and watching searches performed by humans. If successful, different copies of WebWatcher could easily be attached to any web page for which a specialized search assistant would be useful. Over time, each copy could learn expertise specializing in the types of users, information needs, and information sources commonly encountered through its page.

In the preliminary learning experiments reported here, WebWatcher was able to learn search control knowledge that approximately predicts the hyperlink selected by users, conditional on the current page, link, and goal. These experiments also showed that the accuracy of the agent's advice can be increased by allowing it to give advice only when it has high confidence. While these experimental results are positive, they are based on a small number of training sessions, searching for a particular type of information, from a specific web page. We do not yet know whether the results reported here are representative of what can be expected for other search goals, users, and web localities.

Based on our initial exploration, we are optimistic that a learning apprentice for the world wide web is feasible. Although learned knowledge may provide only imperfect advice, even a modest reduction in the number of hyperlinks considered at each page leads to an exponential improvement in the overall search. Moreover, we believe learning can be made more effective by taking advantage of the abundant data available from many users on the web, and by considering methods beyond those reported here.

For additional information, see the WebWatcher project page, http://www.cs.cmu.edu:8001/afs/cs.cmu.edu/project/theo-6/web-agent/www/project-home.html.

6 Acknowledgments

We thank Ken Lang for providing much of the software for learning over pages of text, and for suggesting the idea of implementing the agent by dynamically editing web pages. Thanks to Michael Mauldin for software and advice on the construction of a web-based text-retrieval system. We are grateful to Rich Caruana and Ken Lang for helpful comments on this paper. This research is supported by a Rotary International fellowship grant, an NSF graduate fellowship, and by Arpa under grant number F33615-93-1-1330.

References

[Mitchell et al., 1985] T. Mitchell, S. Mahadevan, and L. Steinberg, "LEAP: A Learning Apprentice for VLSI Design," *Ninth International Joint Conference on Artificial Intelligence*, August 1985.

[Mitchell et. al., 1994] T.M. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski, "Experience with a Learning Personal Assistant," *Communications of the ACM*, Vol. 37, No. 7, pp. 81-91, July 1994.

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

[Lang, 1995] K. Lang, *NewsWeeder: Learning to Filter Netnews*, to be submitted to the International Conference on Machine Learning, 1995

[Littlestone, 1988] N. Littlestone, "Learning quickly when irrelevant attributes abound," *Machine Learning,* **2**:4, pp. 285–318 .

[Quinlan, 1993] J.R. Quinlan, *C4.5: Programs for Machine Learning,* Morgan Kaufmann, 1993.

Appendix The following lists show the words used to encode$Page \times Link \times Goal$ into a feature vector as summarized in Figure 1. Words are listed in order, beginning with the word with highest mutual information.

Underlined words: papers, uci, other, publications, learning, algorithm, www, illigal, page, illinois, related, ted, belding, mitchell, people, approaches, california, soar, tom, ronny, unit, readme, genetic, symbolic, sources, comparison, explanation, laboratory, cmu's, with, cmu, abstract, machine, pazzani, avrim, more, what, j, systems, michael, project, dortmund, my, subject, personal, institute, conference, their, tf3, our, lhm, esf, do, indexes, characterizing, handouts, information.html, fourier, articifial, tracking, ntrs, weakly, 26, readings, software, information, knowledge, 95, lab, language, esprit, lists, html, dnf, staff, rl, reinforcement, univ, services, pub, links, irvine, on, home, report, cognitive, list, international, that, discovery, data, z, publication, agenda, original, discussed, there, between, tex2html2, tex2html1, method, does, tcb, perception, 468, blum, illigals, actually, it, workshop94, newell, bibilography, bottom, heck, rosenbloom, relationship, mine, ciir, germany, uc, tech, net, middle, engineering, sigart, reports, mailing, network, reasoning, michigan, 10, robotics, cs, groups, online, encore, next, low, tr, 4k, pape, return, preprints, new, programs, kohavi, etc, m, group, computing, department, document, postscript, is, proben1, complete, summary, see, ls8, cultural, director, professor, references, index, problems, gmd, dept, ml, applied, vision, clifford, pan, level, faculty, introduction, call, integrated, sciences, brunk, ofai, general, college, digest, neural, ml94, applications, mit, service, databases, abstracts, austrian, issues, from, top, electronic, w, homepage, emde, starting, image

Sentence words: other, illigal, uci, papers, publications, related, algorithm, page, www, lists, people, learning, soar, ted, belding, illinois, kohavi, unit, selected, ronny, california, bottom, laboratory, symbolic, sites, mitchell, approaches, sources, avrim, comparison, more, cmu's, with, genetic, abstract, home, pazzani, systems, document, email, abstracts, institute, conference, machine, dortmund, next, led, view, their, subject, do, tf3, lhm, esf, background, handouts, recommend, ntrs, carry, indexes, tracking, articifial, 468, 26, thrun, language, explanation, what, lab, same, staff, manner, reinforcement, knowledge, cmu, cognitive, my, tom, html, web, there, services, organized, pub, on, irvine, esprit, 95, links, list, international, middle, tech, readme, personal, sigart, net, ml94, are, our, engineering, bibilography, maybe, discussed, september, actually, quick, illigals, listed, heck, tex2html2, tex2html1, relationship, agenda, event, method, does, newell, rosenbloom, tcb, publication, original, blum, z, germany, ciir, that, rl, uc, information, reasoning, know, report, software, to, department, new, groups, encore, reports, test, previous, further, level, pape, preprints, tr, clusters, readings, 4k, low, return, link, or, group, neural, postscript, discovery, computing, gmd, michigan, robotics, is, issues, maintained, mine, general, proben1, check, fields, investigations, images, complete, director, volumes, ls8, cultural, sciences, colt94, at, mailing, edu, network, stanford, college, want, applications, back, pan, online, 10, faculty, integrated, between, jump, brunk, ofai, problems, electronic, digest

Heading words: ga, personal, pages, computing, organization, lab, this, some, of, other, evolutionary, public, neural, data, knowledge, me, to, information, and, various, program, the, nlp, nets, nn, home, institutions, doing, depts, on, ai, reinforcement, about, return, ntrs, illigal, representation, kr, readings, integrated, science, language, intelligence, departments, processing, links, school, 11, systems, 94, meetings, first, try, mining, soar, html, computer, 3k, applications, possibly, ftp, aug, involved, networks, subjects, page, 10, cognitive, natural, relevant, interpretation, tf3, esf, rules, indirect, lhm, act, interest, robotics, txt, colt94, workshop, david, programmatic, you, goldberg, e, director, professor, apps, cogsci, 15, 681, list, handouts, journal, starting, logic, talks, interests

User goal words: university, carnegie, learning, pazzani, explanation, tom, cmu, genetic, information, machine, irvine, decision, reinforcement, soar, stanford, inductive, mistake, curvilinear, gigus, pattern, steffo, ilp, ronny, higher, goldberg, holland, first, rise, phoebe, avrim, mit, occams, emde, gmd, illinois, dnf, koza, berkeley, quinlan, computational, josef, average, salton