Learning Probabilistic User Profiles: Applications to Finding Interesting Web Sites, Notifying Users of Relevant Changes to Web Pages, and Locating Grant Opportunities.

M. Ackerman, D. Billsus, S. Gaffney, S. Hettich, G. Khoo, D. Kim, R. Klefstad, C. Lowe, A. Ludeman, J. Muramatsu, K. Omori, M. Pazzani[1], D. Semler, B. Starr, & P. Yap

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
(714) 824-7403

Introduction

The World Wide Web contains a vast amount of information potentially relevant to a user's goals or interests. We describe three agents that help a user locate useful or interesting information. The agents share a common core approach to learning a probabilistic profile from a user's feedback. This profile is then used to find, classify, or rank other sources of information that are likely to be of interest to the user. The agents described in this article are:

- Syskill & Webert: This agent is designed to help a user with long-term information seeking goals, such as finding previously unvisited web sites on a particular technical topic.

- DICA: This agent is designed to monitor user-specified web pages, and notify the user of important changes to these pages. The user provides feedback on which changes are important and DICA learns to notify the user only for significant changes.

- GrantLearner: This agent notifies an individual of new research grant opportunities that meet the learned profile of the individual's research interests.

Although the representation and learning of user profiles is the same in the three systems, they differ along several dimensions, including the type of information processed by the system, how additional information is located by the system and how recommendations are presented to the user. We begin our discussion with Syskill & Webert, since it was the first of the three agents and has undergone the most experimentation and user testing.

Syskill & Webert

Web search engines are designed to assist a user to find information relevant to a short-term goal, such as finding a web page with information on a product one is considering purchasing. In contrast, Syskill & Webert is designed to assist a user in pursuing long-term goals, such as keeping track of interesting developments or information resources in

[1] Corresponding Author.

a particular field. The current version of Syskill & Webert is implemented as a plug-in and collection of Java classes that add extra functionality to a web browser. A user of Syskill & Webert first declares a new topic (or reuses an existing topic). Topics of current users include biomedical computing, independent recording artists, and raising goats. The user may optionally associate with each topic one or more "topic pages" which are typically pages created by a person listing web pages available on that topic. For example, http://golgli.harvard.edu/biopages/all.html contains links to over 2000 web pages on biosciences. Although the maintainers of this page have done a remarkable job of collecting information, the usefulness of the resource would be enhanced if there were an assistant watching the user explore this resource, learning his interests, and recommending which unvisited links should be explored next. Syskill & Webert is designed to provide such assistance.

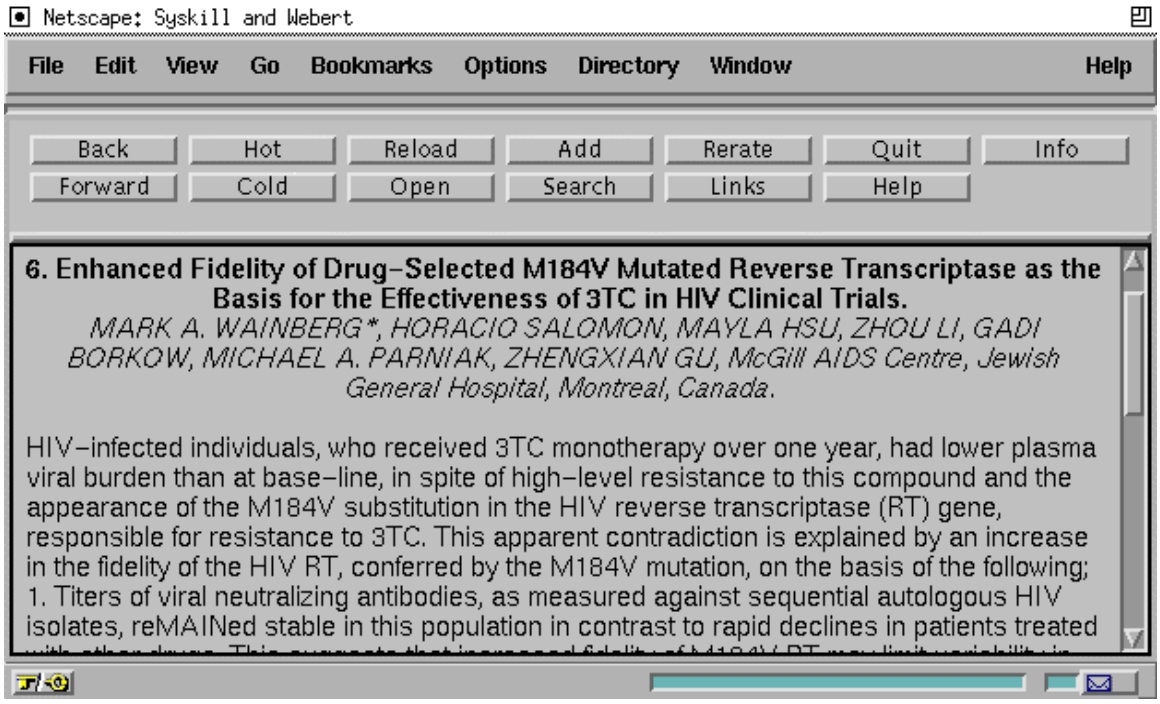

Figure 1. The Syskill & Webert interface for rating web pages adds additional commands to the web browser.

Rather than inferring the user's preferences from behavior (Lieberman, 1995), Syskill & Webert allows the user to provide feedback on the interestingness or usefulness of a web page. The user may click on "Hot" or "Cold" buttons that are added to the browser window (see Figure 1).[2] Syskill & Webert records these ratings to learn a profile of the user's likes and dislikes. A simple Bayesian (Duda & Hart, 1973)classifier creates a user profile from the user's preferences. The profile can be used to calculate the probability that any web page is interesting to the user. There are two ways that Syskill & Webert can locate potentially interesting pages. First, it can locate pages directly accessible from the

[2] The terminology, "Hot" and "Cold" comes from an early version of Syskill & Webert that maintained a "hotlist" for the user's favorite pages and a coldlist for pages the user did not find interesting.

current page. Second, it can construct a query of a search engine such as LYCOS (Maudlin & Leavitt, 1994). Two types of words are used to build a query. First, a query contains words that occur in the most number of pages that have been rated "hot" (ignoring common English words and all HTML commands). Second, a query contains words whose presence in an HTML file helps discriminate pages that are rated hot from cold pages using mutual information. Since LYCOS cannot accept very long queries, we use the seven most informative words that are found in a higher proportion of hot pages than all pages and the seven most commonly occurring words as a query. Once a search engine has returned a page with links to potentially relevant pages, Syskill & Webert retrieves each page and analyzes the HTML to produce its recommendations. Although this process is time consuming,[3] the page ratings of Syskill & Webert can reduce the time spent by a user of manually searching of the Web. In the newest version of Syskill & Webert, the multithreading capabilities of Java are exploited so that page ratings are displayed interactively while other pages are being retrieved.

Syskill & Webert displays its recommendations by annotating links that point to web pages that have been rated. Figure 2 shows an example of a web page returned by LYCOS using a query constructed by Syskill & Webert. The annotations indicate whether the user is likely to be interested in the page and a number indicating the probability that a page is interesting.

Pazzani, Muramatsu & Billsus (1996) provide details on how Syskill & Webert learns and uses a profile. Here we just summarize the most important details. Syskill & Webert represents each page rated by the users as a set of Boolean features. Each feature correspond to the presence or absence of a particular word. Syskill & Webert selects the 96 most informative (Quinlan, 1986; Lewis, 1992) words[4] as features (i.e., words that occur is a greater proportion of "hot" training documents than "cold" or vice versa). Once represented as a set of Boolean features, a variety of machine learning or information filtering algorithm might be used to create a user profile. The publicly available version of Syskill & Webert uses a standard machine learning algorithm, a Bayesian classifier to create a probabilistic profile. The profile contains the estimated prior probability that a page is hot (and cold) and, for each informative feature, the conditional probability that a page contains (and doesn't contain) a word given that the page is hot (and cold). This algorithm was selected because it was experimentally found to be one of the most accurate at this task, it has desirable time and space complexity (linear in the number of examples and features), and its estimates the probability that a page would be interesting to the user for a particular topic. This probability may be used to rank the unseen pages. The probability may also be used to decide whether to recommend the page to user. By selecting a threshold for the probability, it is possible to trade-off precision and recall. We use 0.5 as the default threshold in Syskill & Webert.

[3] If there were a closer integration between the search engine and Syskill & Webert, the considerable overhead of retrieving a page in order to rank it would be effectively eliminated.

[4] In our experiments, having fewer than 50 features, or more than 250 reduces the accuracy of predications, while there was little difference among intermediate values.

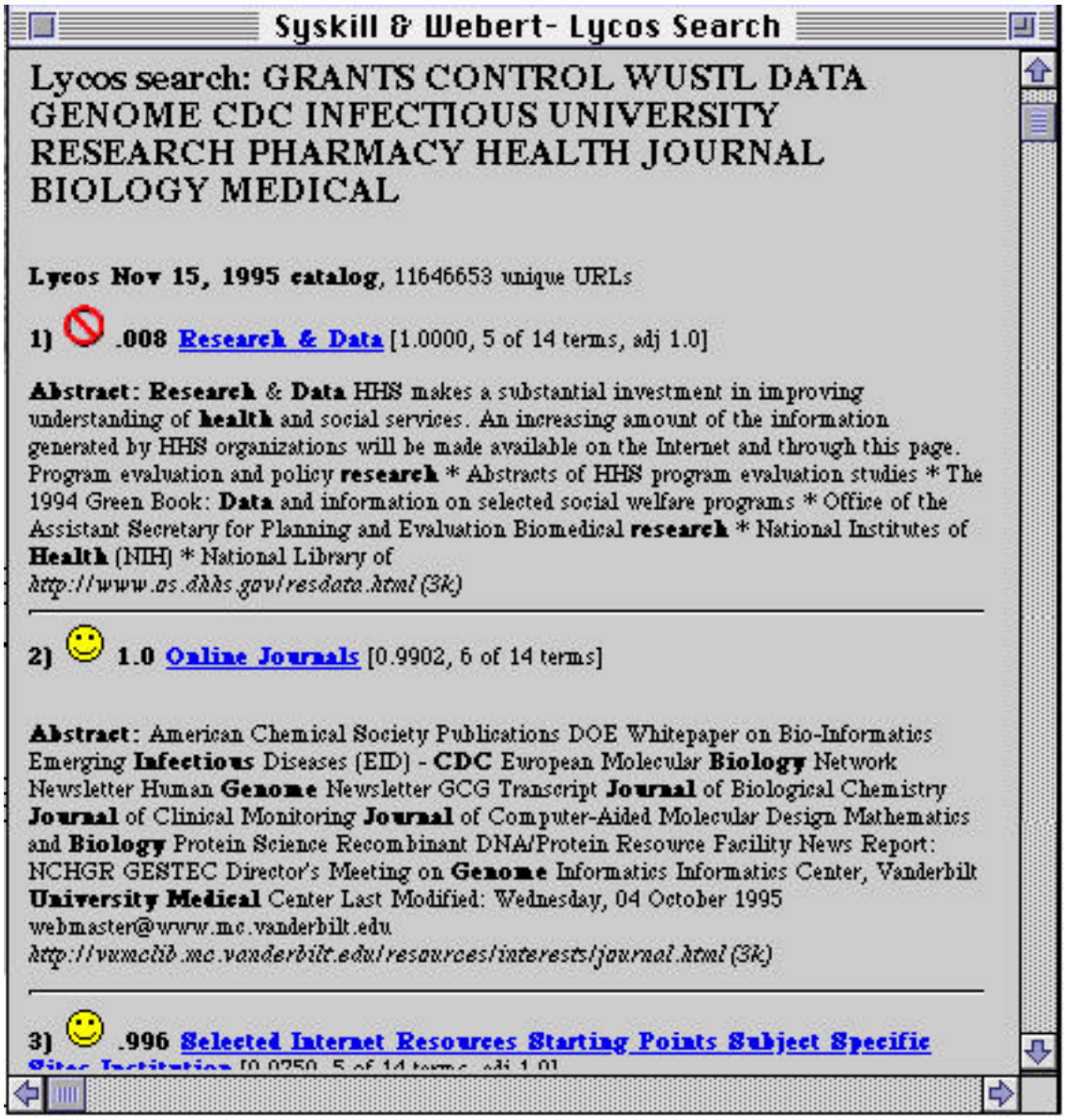

Figure 2. An example of a page annotated by Syskill & Webert. Here, Syskill & Webert constructed a LYCOS query from a user profile for the biomedical topic.

We have run several experiments to evaluate how often Syskill & Webert's prediction agrees with the user's rating of a page. Here we report on one such experiment that was conducted using 48 web pages rated by a user on the topic of films that were available at http://rte66.com/Movies/blurb.html. In this case, the user rated the pages according to whether he liked the film that was described by the page. Each page had information on the people involved with the film and a summary of the film. 25 of the 48 pages were rated hot by the user. To test the ability of Syskill & Webert to learn the user's preferences, it was trained on a subset of the rated pages and evaluated on pages that were not in the training set. Since we are interested in seeing how many pages a user would have to rate to learn a useful profile, we increased the number of training pages from 10 to 35 in increments of 5. Figure 3 shows the average accuracy at predicting unseen pages as

a function of the number of training examples. Each point represents the average of 40 trials. On each trial, *n* pages were randomly selected for training, and a new profile was learned for those pages. The accuracy on the test consisting of all pages not in the training set was computed. The figure shows that with 10 rated pages, Syskill & Webert performs at near chance levels. However, as the number of rated pages increases, the accuracy of its predictions increased. When trained on 35 pages, Syskill & Webert agreed with the user's rating over 80% of the time.

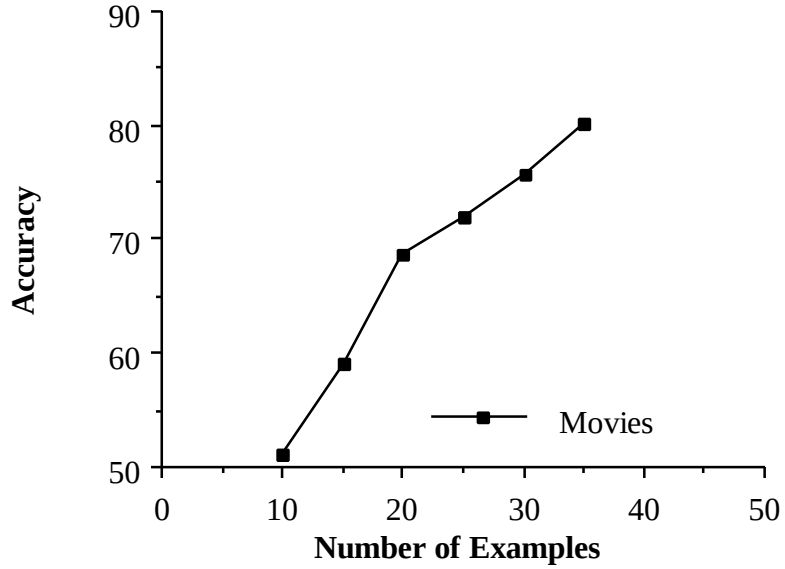

Figure 3. Syskill & Webert's accuracy at predicting the user's preference for movies increases to over 80%.

The user profile provides relatively fine-grained probability estimates that may also be used to order the exploration of new pages. For example, on the biomedical domain, we used a leave-one-out testing methodology to predict the probability that a user would be interested in a page. The ten pages with the highest probability were all correctly classified as interesting and the 10 pages with the lowest probability were all correctly classified as uninteresting. There were 21 pages whose probability of being interesting was above 0.9 and 19 of these were rated as interesting by the user. There were 64 pages whose probability of being interesting was below 0.1 and only 1 was rated as interesting by the user.

The decision to learn a separate profile for each topic in Syskill & Webert was based on our intuition that restricting the scope would simplify the learner's task. In principle, an intelligent agent, like a close colleague, should be able to fairly accurately decide whether any web page would interest a user without regard to whether the topic is films or biomedical research. The experiment we report next investigated whether Syskill & Webert would be able to accurately learn a user's preferences when data from several topics are combined. In particular, we pooled pages from five topics rated by a single user and created a collection of over 450 documents. The user rated 69.5% of these documents as cold. The results of an experiment on this data, ran in the same manner as the previous

experiment are reported in Figure 4. Here, we see that the Bayesian classifier does not reach an accuracy level that that differs substantially from simply guessing that the user would not like any page. The problem arises for two reasons. First, words that are informative in one domain (e.g., recording artists) are irrelevant in another (e.g., biomedical applications). Furthermore, within a particular single domain, it is possible that a simple decision boundary exists that separates the interesting pages from the others. However, it is likely that in the several domains there are a variety of complex "clusters" of interesting articles and the Bayesian classifier is unable to represent this decision boundary One might suspect that a more complex learning algorithm might perform better at this more general problem. However, neither decision tree learners nor multi-layer neural nets were more accurate than the Bayesian classifier due to the relatively small number of examples available for the complex high dimensional concept needed to make accurate predictions across domains. Therefore, we conclude that asking the user to provide a topic and learning a separate profile per topic is an essential, and not particularly burdensome, requirement for an agent such as Syskill & Webert.

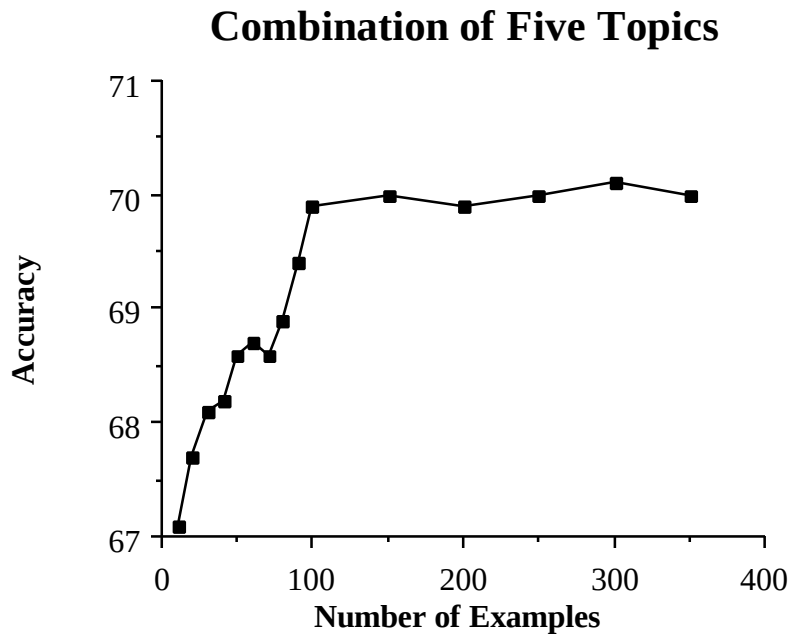

Figure 4. The bias of the Bayesian classifier is not suitable for representing a user profile when there are several different types of pages that are liked by the user.

In summary, Syskill & Webert collects a user's ratings of web pages, learns a profile of the user's preferences for pages of a given topic, retrieves and rates unseen web pages, and annotates links pointed to rated pages with a rating of the page. The current version of Syskill & Webert is available at http://nubian.ics.uci.edu:9001/swb

DICA

The Do-I-Care Agent (DICA) (Starr, Ackerman & Pazzani 1996) project explores a naturally reoccurring problem with the Web. Even after one has discovered web resources, a problem remains: How do you know when to revisit those resources for new material? Consider some common post-discovery situations:

- You know the page where a colleague has his publications. You would like to obtain new papers as they are posted. Furthermore, you would like to know about paper only a particular topic, rather than just all papers.
- You have obtained the page for the low cost clothes merchandiser. You would like to know when there is a sale on jeans or Hawaiian shirts.

These are examples of change events that occur somewhat unpredictably and offer no explicit notification. Today, interested parties must occasionally check by hand. This can be burdensome and tedious, and one may forget to do it on a timely basis. Furthermore, we would like an agent to notify us only of changes that are significant. DICA creates and maintains a web page to report its findings to a user. DICA agents of others users can monitor such a web page to be notified when another user's agent has found an interesting change to the Web.

DICA explores the nature of collaborative resource discovery and information sharing on the Web. Its goal is to augment certain types of computer-supported cooperative work situations. Collaborative efforts to discover Internet resources, such as interesting web pages, require work by users. Unfortunately, users often will not do the extra, altruistic work, leading to adoption failures for social filtering systems. DICA assumes that agents are especially useful in situations where the benefits would be marginal if the task had to be done by hand, and that cooperating agents are especially useful in situations where individuals wish to use the agents for their own purposes anyway.

In order to track interesting changes, an agent needs to know what changes are interesting and where to find them. DICA collaborates with both users and their peers to identify potentially interesting changes. The agent works by soliciting a user's opinions about changes it finds to train the user model; it is in users' best interests to keep their personal agents informed about relevant pages and about the quality of reported changes. Without extra work, the agent can then share these opinions and findings with agents of like-minded users. Individuals engage in resource discovery for themselves, but they can share that effort without any extra work.

Because different changes may be interesting for different reasons, users may maintain multiple Do-I-Care agents specialized on particular topics or kinds of web pages. Each Do-I-Care agent must:

- Periodically visit a user-defined list of target pages.
- Identify any changes since the last time the agent visited.
- Decide whether the changes were interesting
- Notify the user if the change was interesting.
- Accept relevance feedback (Rocchio, 1971) on the interestingness of the change and timeliness of the notification.
- Facilitate information sharing and collaboration between individuals and groups.

DICA maintains a list of web pages for each user's topic that have interesting information. By limiting the search to a small subset of web pages specifically selected because they are likely to contain interesting information, we can greatly improve precision. By accepting

or rejecting Do-I-Care's suggestions, the user refines the types of changes that are interesting. Like Syskill & Webert, a Bayesian classifier learns a profile. The primary difference to the learning algorithm between Syskill & Webert and DICA is that Syskill & Webert extracts features from a web page to determine the interestingness of the page while DICA extracts features from the difference between the current and an earlier version of a web page (computed by using the UNIX "diff" command).

Once an agent spots an interesting change, the user is notified by e-mail, and the change is appended to the agent's associated web page. This web page is also used for relevance feedback and collaborative activity. DICA addresses solve two post-discovery problems: when should a user revisit a known site for new information, and how does a user share new information with others who may be interested. These post-discovery chores are laborious, suffer from inequitable distribution of costs and benefits, and are not both addressed by existing systems. By providing a simple technical mechanism, we have found a collaboration mechanism that provides for group effort through individual incentives.

Grant Learner

The profiles learned by the Bayesian classifier are in no way limited to web pages and can be applied to all kinds of text documents available on the Internet. In a different project we are using the described learning techniques to classify descriptions of funding opportunities. The goal of the "UCI GrantLearner" project is to provide a system that learns to distinguish between interesting and uninteresting funding opportunities, based a user's ratings of these descriptions.

Our current implementation of the GrantLearner is a Java standalone application that runs on any Java compatible platform. It connects to a regularly updated grant database maintained at UCI and provides a user interface to browse this database. Figure 5 shows a screen-shot of the current GrantLearner version. The system displays a list of all grant subjects available in the database. By clicking on a subject line the user can retrieve a corresponding full text description. Grant descriptions usually provide information about the grant's subject area, funds, eligibility and deadlines. Using the "hot" and "cold" buttons the user can indicate whether a grant description is related to her interests. Once the user has rated a minimum number of grants (10 in the current implementation), GrantLearner uses the previously described feature extraction and learning algorithms to learn a profile of the user's interests. The learned profile is then used to process the rest of the database, i. e. all the grant descriptions the user has not looked at. The result of this process is a list of grant descriptions, sorted according to GrantLearner's relevance prediction. A learned user profile can be saved and reused at a later time. For example, users can train GrantLearner with a set of rated descriptions once, and reuse the learned classifier at a later time to get a ranked list of database entries that were added recently. First experiments and feedback from different users suggest that GrantLearner is a helpful tool to locate potentially interesting grant descriptions.

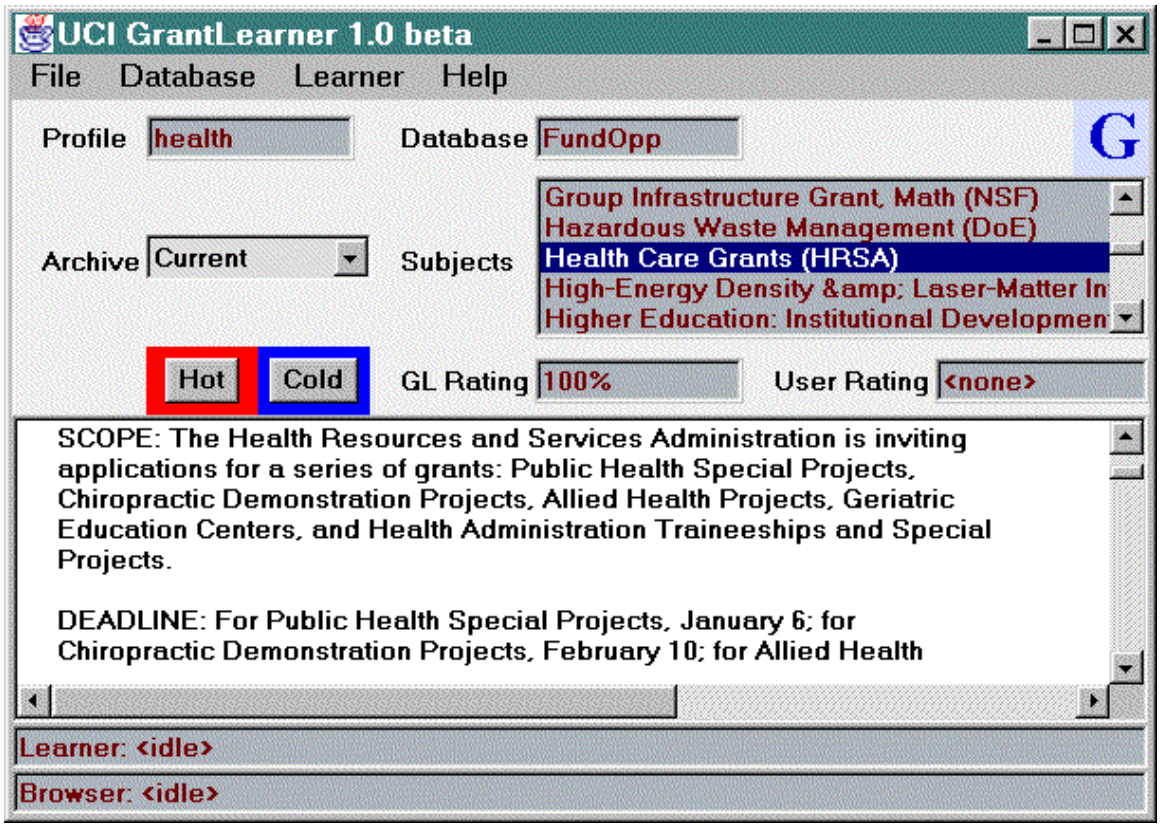

Figure 5. The UCI GrantLearner interface for finding grant opportunities for funded research.

Future versions of GrantLearner will allow to use the same learned profile to filter different grant databases (e.g. IRIS and the NSF grant database). We are also planning to implement an email notification system, such that users are automatically notified when a grant relevant to their interests is added to a database. In addition, we believe that exploiting the structure of descriptions, for example deadline or eligibility sections, is useful to construct a set of special-purpose features that help to prevent the system from suggesting grants one is not eligible for or whose deadlines have passed.

More information about the UCI GrantLearner can be found at http://www.ics.uci.edu/~dbillsus/grantlearner.html.

Current Research Work

Our current research (e.g., Billsus. & Pazzani, 1996) involves exploiting different forms of knowledge or information to improve the accuracy of an agent's recommendations. The goal is to reduce the overall amount of effort required by a user to get useful results from the agent. Our efforts have focused on finding additional information to determine which words should be features in the profile. Although implemented in an experimental version of Syskill & Webert, they are of more general applicability. We have explored two ways of addressing this problem.

- Asking the user for words to use as features for each topic. Syskill & Webert can take advantage of the user's knowledge of the domain to identify key terms.
- Using additional lexical knowledge to filter the informative features. We have experimented with using WORDNET (Miller, 1991), a lexical database containing the relationship between approximately 30,000 commonly occurring English words. When there is no relationship between a word and the topic (e.g., goat) Syskill & Webert can eliminate that word from consideration as a feature.

Figure 6 shows an experiment (run in the same manner as the previous experiments) on the "goat" topic, using profiles learned by a Bayesian classifier in which we compare the accuracy of representing documents by the 96 most informative words, the words that a user indicates are most useful in determining whether a page is interesting, and the subset of the 96 most informative features that are related to the word "goat" according to WORDNET (by following a series of hypernym, antonym, member-holonym, or part-holonym links between the word and the word "goat"). The results show that both approaches to selecting features improve upon the use of just the statistical information available in the training data. On this problem, the user-selected features only provide a benefit when there are few training examples. Using the additional lexical knowledge encoded in the to only retain the subset of the 96 informative features that are related to the topic provides a substantial improvement in accuracy on this problem.

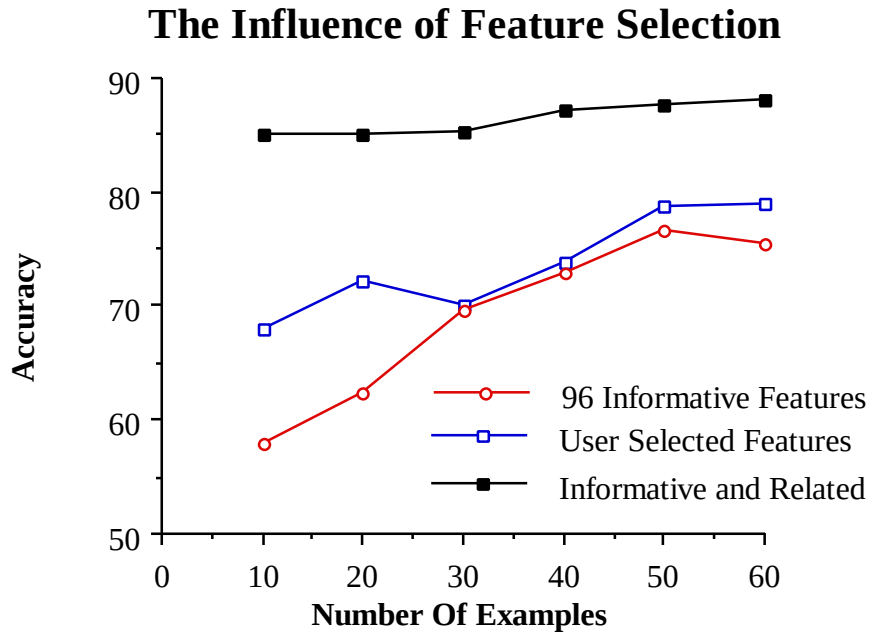

Figure 6. Improving upon the Bayesian classifier by automatically or manually selecting relevant features.

Table 1 lists some informative features found from one training set, and strikes out those for which no relationship exists between the word and the topic. While there is room for improvement, WordNet does remove many words that are unrelated to the topic (and a few such as "farm" that are related). Most of the words that are not eliminated are related to the topic (with a few exceptions e.g., "computer").

Table 1. The stricken words are excluded as features by using WordNet.

pygmy	return	production	cashmere	management
milk	animal	angora	feed	spring
library	breeding	feeding	cheese	~~other~~
program	computer	fair	fiber	~~green~~
~~however~~	~~health~~	~~dairy~~	~~time~~	~~summer~~
~~took~~	~~quality~~	~~early~~	~~normal~~	~~farm~~

Our future plans also include significant enhancement to the Syskill & Webert interface that allows users to share their profiles with other users. In many fields communities of interest already exist, i.e. people know who else is working in their respective fields or who is to be considered an expert in a certain area. We envision a useful information filtering framework that allows people to publish their specialized filtering profiles electronically. These profiles could then be used by other people interested in someone's area of expertise and be utilized as intelligent filtering tools to view large information spaces, such as the web, "through the eyes of an expert". We also believe that a similar approach might be useful in educational settings. Instructors train a profile based on their judgment about the relevance of items to a class they are teaching. These profiles could then be given to students, who obtain an "automated information guide" for their information needs with respect to the class they are taking.

Summary

We have described three examples of intelligent agents under development at UCI. All make use of a simple Bayesian classifier for learning user probabilistic profiles. Experience with the agents has shown that this classifier provides a useful foundation for creating agents that learn from user feedback and recommendations information to the user.

References

Billsus, D. & Pazzani, M. (1996) Revising User Profiles: The Search for Interesting Web Sites. In *Proceedings of the Third International Workshop on Multistrategy Learning (MSL '96)*, AAAI Press.

Duda, R. & Hart, P. (1973). *Pattern classification and scene analysis.* New York: John Wiley & Sons.

Lewis, D. (1992). Representation and Learning in Information Retrieval. Ph.D. thesis, University of Massachusetts.

Lieberman, H. (1995). Letizia: An agent that assists web browsing. Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, August 1995.

Mauldin, M. & Leavitt, J. (1994). Web Agent Related Research at the Center for Machine Translation *Proceedings of the ACM Special Interest Group on Networked Information Discovery and Retrieval.*

Miller, G. (1991). WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4).

Pazzani, M., Muramatsu J., and Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. *Proceedings of the National Conference on Artificial Intelligence*, Portland, OR.

Quinlan, J.R. (1986). Induction of decision trees. Machine Learning, 1, 81–106.

Rocchio, J. (1971). Relevance feedback information retrieval. In Gerald Salton (editor), *The SMART retrieval system- experiments in automated document processing* (pp. 313-323). Prentice-Hall, Englewood Cliffs, NJ.

Starr,, B., Ackerman, M. & Pazzani, M. (1996). "Do-I-Care: A Collaborative Web Agent." Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'96), pp. 273-274.