Syskill & Webert: Identifying interesting web sites

Michael Pazzani, Jack Muramatsu & Daniel Billsus
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717
pazzani@ics.uci.edu

Abstract

We describe Syskill & Webert, a software agent that learns to rate pages on the World Wide Web (WWW), deciding what pages might interest a user. The user rates explored pages on a three point scale, and Syskill & Webert learns a user profile by analyzing the information on each page. The user profile can be used in two ways. First, it can be used to suggest which links a user would be interested in exploring. Second, it can be used to construct a LYCOS query to find pages that would interest a user. We compare six different algorithms from machine learning and information retrieval on this task. We find that the naive Bayesian classifier offers several advantages over other learning algorithms on this task. Furthermore, we find that an initial portion of a web page is sufficient for making predictions on its interestingness substantially reducing the amount of network transmission required to make predictions.

1 Introduction

There is a vast amount of information on the World Wide Web (WWW) and more is becoming available daily. How can a user locate information that might be useful to that user? In this paper, we discuss Syskill & Webert, a software agent that learns a profile of a user's interest, and uses this profile to identify interesting web pages in two ways. First, by having the user rate some of the links from a manually collected "index page" Syskill & Webert can suggest which other links might interest the user. Syskill & Webert can annotate any HTML page with information on whether the user would be interested in visiting each page linked from that page. Second, Syskill & Webert can construct a LYCOS (Maudlin & Leavitt, 1994) query and retrieve pages that might match a user's interest, and then annotate this result of the LYCOS search. Figure 1 shows a Web page (http://ai.iit.nrc.ca/subjects/ai_subjects.html) that has been annotated by Syskill and Webert. This web page is a subject listing of AI topics that serves as an example of one index page. In this case, the user has indicated strong interest in "Machine Learning" and "Reinforcement Learning" (indicated by two thumbs up), a mild interest in "Agents" (indicated by one thumb up and one thumb down) and no interest in "Business, Finance and AI" and "Philosophy of AI' (indicated by two thumbs down). The other annotations are the predictions made by Syskill & Webert about whether the user would be interested in each unexplored page. A smiley face indicates that the user hasn't visited the page and Syskill & Webert recommends the page to the user. For this topic, these pages are "Neural Networks," "Evolutionary Algorithms," "Bayesian Inference," "General AI," and "Case-Based Reasoning." The international symbol for "no" is used to indicate a page hasn't been visited and the learned user profile indicates the page should be avoided. Following any prediction is a number between 0 and 1 indicating the probability the user would like the page.

In this paper, we first describe how the Syskill & Webert interface is used and the functionality that it provides. Next, we describe the underlying technology for learning a user profile and how we addressed the issues involved in applying machine learning algorithms to classify HTML texts rather than classified attribute-value vectors. We describe experiments that compare the accuracy of several algorithms at learning user profiles. Finally, we relate Syskill & Webert to other agents for learning on the Web.

2 Syskill & Webert

Syskill & Webert learns a separate profile for each topic of each user. We decided to learn a profile for user topics rather than users for two reasons. First, we believe that many users have multiple interests and it will be possible to learn a more accurate profile for each topic separately since the factors that make one topic interesting are unlikely to make another interesting. Second, associated with each topic is a URL that we call an *index* page. The index page is a manually constructed page that typically contains a hundred or more links to other information providers. For example, the Web page at `http://golgi.harvard.edu/biopages/all.html` contains links to over 400 sites on the topic of Biosciences. Syskill & Webert allows a user to explore the Web using the index page as a starting point. In one mode of using Syskill & Webert, it learns a profile from the user's ratings of pages and uses this profile to suggest other pages accessible from the index page. To collect ratings, the HTML source of users' pages is intercepted, and an

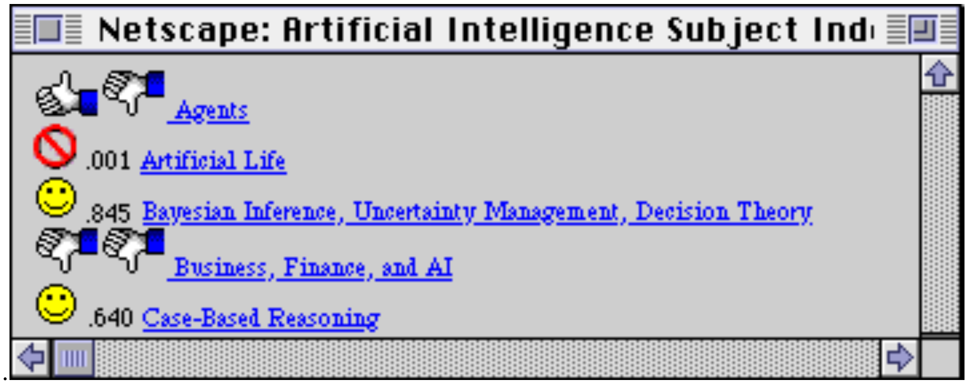

Figure 1. An example of a page annotated by Syskill & Webert.

additional functionality is added to each page (see Figure 2). This functionality allows the user to rate a page as either hot (two thumbs up), lukewarm (one thumb up and one thumb down), or cold (two thumbs down). The user can return to the index page or switch topics. Furthermore, the user can instruct Syskill & Webert to learn a user-profile for the current topic, make suggestions or consult LYCOS to search the Web.

When a user rates a page, the HTML source of the page is copied to a local file and a summary of the rating is made. The summary contains the classification (hot, cold, or lukewarm), the URL and local file, the date the file was copied (to allow for the bookkeeping that would occur when a file changes), and the page's title (to allow for the production of a summary of the ratings).

Syskill & Webert adds functionality to the page (see Figure 2) for learning a user profile, using this user profile to suggest which links to explore from the index page, and forming LYCOS queries. The user profile is learned by analyzing all of the previous classifications of pages by the user on this topic. Syskill & Webert also contains a function to retrieve and locally store the HTML source of all links accessible from the current page. Syskill & Webert analyzes the HTML source of a page to determine whether the page matches the user's profile. To avoid network transmission overhead during our experiments, we

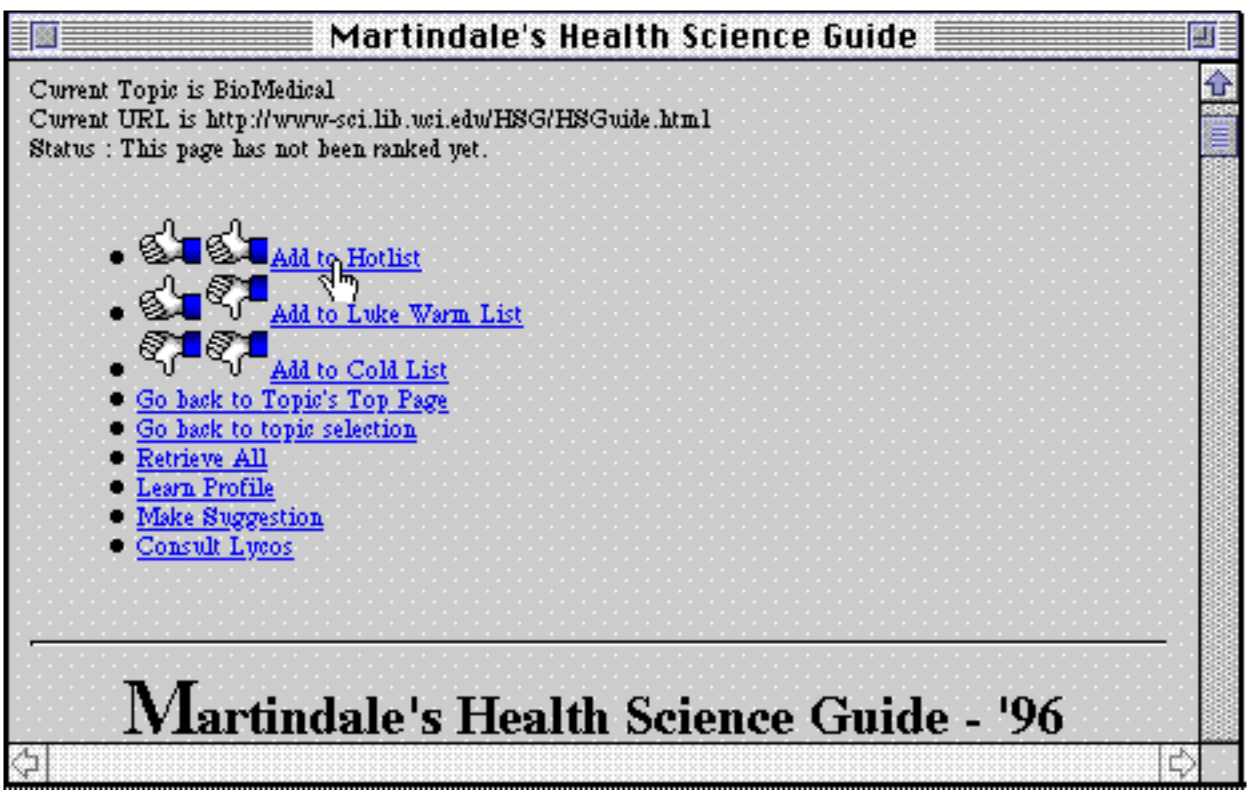

Figure 2. Syskill & Webert Interface for rating pages

prefetch all pages. This also ensures that experiments are repeatable if a page changes or is no longer accessible

Once the user profile has been learned, it can be used to determine whether the user would be interested in another page. However, this decision is made by analyzing the HTML source of a page, and it requires the page to be retrieved first. To get around network delays, we allow the user to prefetch all pages accessible from the index page and store them locally. Once this has been done, Syskill & Webert can learn a new profile and make suggestions about pages to visit quickly. Section 5 discusses one means of avoiding a significant amount of network transmission overhead. Once the HTML has been analyzed, Syskill & Webert annotates each link on the page with an icon indicating the user's rating or its prediction of the user's rating together with the estimated probability that a user would like the page. Following any prediction is a number between 0 and 1 indicating the probability the user would like the page. The default version of Syskill & Webert uses a simple Bayesian classifier (Duda & Hart, 1973) to determine this probability. Note that these ratings and predictions are specific to one user and do not reflect on how other users might rate the pages.

As described above, Syskill & Webert is limited to making suggestions about which link to follow from a single page. This is useful if someone has collected a nearly comprehensive set of links about a topic. Syskill & Webert contains another feature that is useful in finding pages that might interest a user anywhere on the Web (provided the pages have been indexed by LYCOS). The user profile contains information on two types of words that occur in pages that have been rated. First, it contains words that occur in the most number of pages that have been rated "hot." For these words, we do not consider whether they have also occurred in pages that have other ratings. However, we ignore common English words and all HTML commands. The second set of words we use are those whose presence in an HTML file helps discriminate pages that are rated hot from other pages. As described in Section 3, we use mutual information to identify discriminating words. Since LYCOS cannot accept very long queries, we use the 7 most discriminating words that are found in a higher proportion of hot pages than all pages and the 7 most commonly occurring words as a query. The discriminating words are useful in distinguishing pages of a given topic but do not describe the topic. For example (see Figure 3) the discriminating words for one user about the Biosciences are “grants,” “control,” “WUSTL,” “data,” “genome,” “CDC,” and “infectious.” The common words are useful for defining a topic. In the example in Figure 3 these are “university,” “research,” “pharmacy,” “health,” “journal,” “biology,” and “medical.”

LYCOS indexes a large percentage of the Web and can quickly identify URLs whose pages contain certain keywords. However, it requires a user to filter the results. Syskill & Webert can be used to filter the results of LYCOS (provided the pages are fetched). For example, Figure 3 shows part of a LYCOS result that has been augmented by Syskill & Webert to contain a recommendation against visiting one page.

3 Learning a user profile.

Learning algorithms require a set of positive examples of some concepts (such as web pages one is interested in) and negative examples (such as web pages one is not interested in). In this paper, we learn a concept that distinguishes pages rated as hot by the user from other pages (combining the two classes lukewarm and cold, since few pages are rated lukewarm, and we are primarily interested in finding pages a user would consider hot). Most learning programs require that the examples be represented as a set of feature vectors. Therefore, we have constructed a method of converting the HTML source of a web page into a Boolean feature vector. Each feature has a Boolean value that indicates whether a particular “word” is present (at least once) or absent in a particular web page.

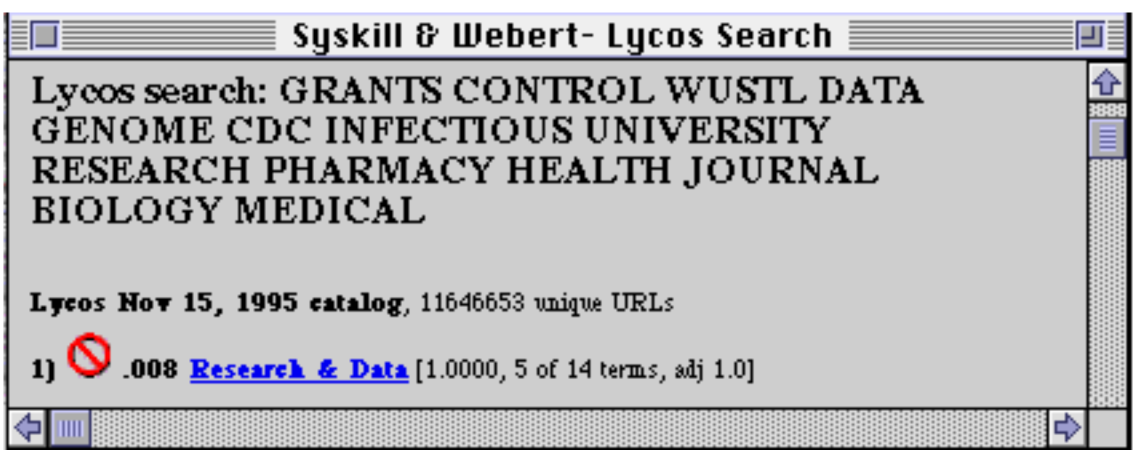

Figure 3 Syskill & Webert constructs a LYCOS query from a user profile.

For the purposes of this paper, a word is a sequence of letters, delimited by nonletters. For example, the URL `<A HREF= http://golgi.harvard.edu/biopages/all.html>` contains nine “words” a, href, http, golgi, harvard, edu, biopages, all, and html. All words are converted to upper case.

Not all words that appear in an HTML document are used as features. We use an information-based approach, similar to that used by an early version of the NewsWeeder program (Lang, 1995) to determine which words to use as features. Intuitively, one would like words that occur frequently in pages on the hotlist, but infrequently on pages on the coldlist (or vice versa). This is achieved by finding the expected information gain ($E(W,S)$) (e.g., Quinlan, 1986) that the presence or absence of a word (W) gives toward the classification of elements of a set of pages (S):

$$E(W,S) = I(S) - [P(W{=}present)I(S_{w=present}) + P(W{=}absent)I(S_{w=absent})]$$

where

$$I(S) = \sum_{c \in \{hot,\, cold\}} -p(S_c)\log_2(p(S_c))$$

$P(W{=}present)$ is the probability that W is present on a page, $S_{w=present}$ is the set of pages that contain at least one occurrence of W, and S_c are the pages that belong to the class. Using this approach, we find the set of k most informative words. In the experiment discussed in Section 4, we use the 128 most informative words. In experimentation not reported here, we’ve found that values of k between 75 and 150 produce acceptable accuracies. Table 1 shows some of the most informative words obtained from a collection of 140 HTML documents on independent rock bands.

Table 1. Some of the words used as features.

nirvana	suite	lo
fi	snailmail	him
pop	records	rockin
little	singles	recruited
july	jams	songwriting
college	rr	his
following	today	write
handling	drums	vocals
island	tribute	previous
smashing	haunting	bass
favorite	airplay	noise

Once the HTML source for a given topic has been converted to positive and negative examples represented as feature vectors, it’s possible to run many learning algorithms on the data. We have investigated a variety of machine learning algorithms including the Bayesian classifier (Duda & Hart, 1973), the nearest neighbor algorithm (Duda & Hart, 1973), ID3 (Quinlan, 1986), perceptrons (Widrow & Hoff, 1960) and multi-layer networks (with 12 hidden units) trained with error backpropagation (Rummelhart, Hinton & Williams, 1986).

4 Experimental Evaluation

To determine whether it is possible to learn user preferences accurately, we have had four users use the Syskill & Webert interface to rate pages. A total of six different user profiles were collected (since one user rated pages on three different topics). The topics are summarized in Table 2 together with the total number of pages that have been rated by the user. Two users rated the pages on independent recording artists. One (A) listened to an excerpt of songs, and indicated whether the song was liked. Of course, the machine learning algorithms only analyze the HTML source describing the bands and do not analyze associated sounds or pictures. Another user (B) read about the bands (due to the lack of sound output on the computer) and indicated whether he’d be interested in the band.

Syskill & Webert is intended to be used to find unseen pages the user would like. In order to evaluate the effectiveness of the learning algorithms, it is necessary to run experiments to see if Syskill & Webert’s prediction agrees with the users preferences. Therefore we use a subset of the rated pages for training the algorithm and evaluate the effectiveness on the remaining rated pages. For an individual trial of an experiment, we randomly selected k pages to use as a training set, and reserved the remainder of the data as a test set. From the training set, we found the 128 most informative features, and then recoded the training set as feature vectors to be used by the learning algorithm. We tried five learning algorithms on each training set. The learning algorithm created a representation for the user preferences. Next, the test data was converted to feature vectors using the features found informative on the training set. Finally, the learned user preferences were used to determine whether pages in the test set would interest the user. For each trial, we recorded the accuracy of the learned preferences (i.e., the percent of test examples for which the learned preferences agreed with the user’s interest). We ran 30 paired trials of each algorithm. Figure 4 shows the average accuracy of each algorithm as a function of the number of training examples for four problems. The results on the two problems not shown are similar.

Table 2. Topics used in our experiments.

User	Topic	URL of topic's index page	Pages
A	Biomedical	`http://golgi.harvard.edu/biopages/medicine.html`	127
A	Lycos	`not applicable`	54
A	Bands(listening)	`http://www.iuma.com/IUMA-2.0/olas/location/USA.html`	57
B	Bands (reading)	`http://www.iuma.com/IUMA-2.0/olas/location/USA.html`	154
C	Movies	`http://rte66.com/Movies/blurb.html`	48
D	Protein	`http://golgi.harvard.edu/sequences.html`	26

The results are promising in that on most of the problems the predictions are substantially better than simply guessing that the user would not be interested in a page (which is the most frequent prediction on all topics). However, no one algorithm is clearly superior on this set. To get a more detailed idea of which algorithms perform well, we ran another experiment with 24 trials and 20 training examples in each domain and the algorithm(s) that were most and least accurate. In each case, we used a paired, two tailed t-test to find other algorithms that were not significantly different from the best and the worst. On the biomedical domain, the naive Bayesian classifier was most accurate and ID3 was least. On the LYCOS search domain, nearest neighbor was most accurate and ID3 was least. On the bands (listening) problem, nearest neighbor, the naive Bayesian classifier and backpropagation were most accurate and ID3 was least. On the bands (reading) problem nearest neighbor and backpropagation were most accurate and ID3 was least. On the movies domain the naive Bayesian classifier was most accurate and nearest neighbor and ID3 were least. On the protein problem, nearest neighbor, ID3 and the naive Bayesian classifier were most accurate and backprop and the perceptron were least accurate. In summary, it appears that ID3 is not particularly suited to this problem, as one might imagine since it learns simple necessary and sufficient descriptions about category membership. A typical decision made by ID3 looks at the presence or absence of 3-5 words and this may be too brittle for a domain such as this. Although one must be careful not to read too much into averaging accuracies across domains, the naive Bayesian classifier has the highest average accuracy with 20 training examples: 77.1 (standard deviation 4.4). In contrast, backprop is 75.0 (3.9), nearest neighbor is 75.0 (5.5), and ID3 is 70.6 (3.6). We have also experimented with a more advanced decision tree learner using pruning with similar results.

We have decided to use the naive Bayesian classifier as the default algorithm in Syskill & Webert for a variety of reasons. It is very fast for both learning and predicting. Its learning time is linear in the number of examples and its prediction time is independent of the number of examples. It is trivial to create an incremental version of the naive Bayesian classifier. It provides relatively fine-grained probability estimates that may be used to order the exploration of new pages in addition to rating them. 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.

In the final experiment, we'll concentrate our experimentation on the naive Bayesian classifier with 20 training examples. We choose a small number of examples since most users will want to get results after ranking such a small number of pages. We investigate whether it is possible to get similar accuracy with less work by looking at an initial portion of the page during learning and prediction.

As described so far, it is necessary to store the complete HTML source of every page rated by the user and to evaluate an unseen page it is necessary to retrieve the entire HTML to convert the page to a feature vector. Here, we investigate an alternate approach. Instead of analyzing the entire HTML to find informative words, only the words contained in the initial c characters are used during learning when creating a profile and only those words in the initial c characters are used to predict whether a user would like the page. Of course, we still select the 128 most informative words in the initial portions of the pages. Note that we look at an initial sequence of characters, rather than words, since the protocol for transmission of segments of the file is based on characters.

We ran an experiment with the naive Bayesian classifier on the six domains with 20 training examples where we varied the value of *c*. The values used were 256, 512, 1024, 2048, 3072, 4096 and infinity (i.e., using the entire document). The results of this experiment, averaged over 24 trials, are shown in Figure 5. We plot each domain separately and also plot an average over the six domains.

While in some domains (protein, bands and LYCOS), there is an advantage in not analyzing the entire document, on average there is a small decrease in accuracy. For

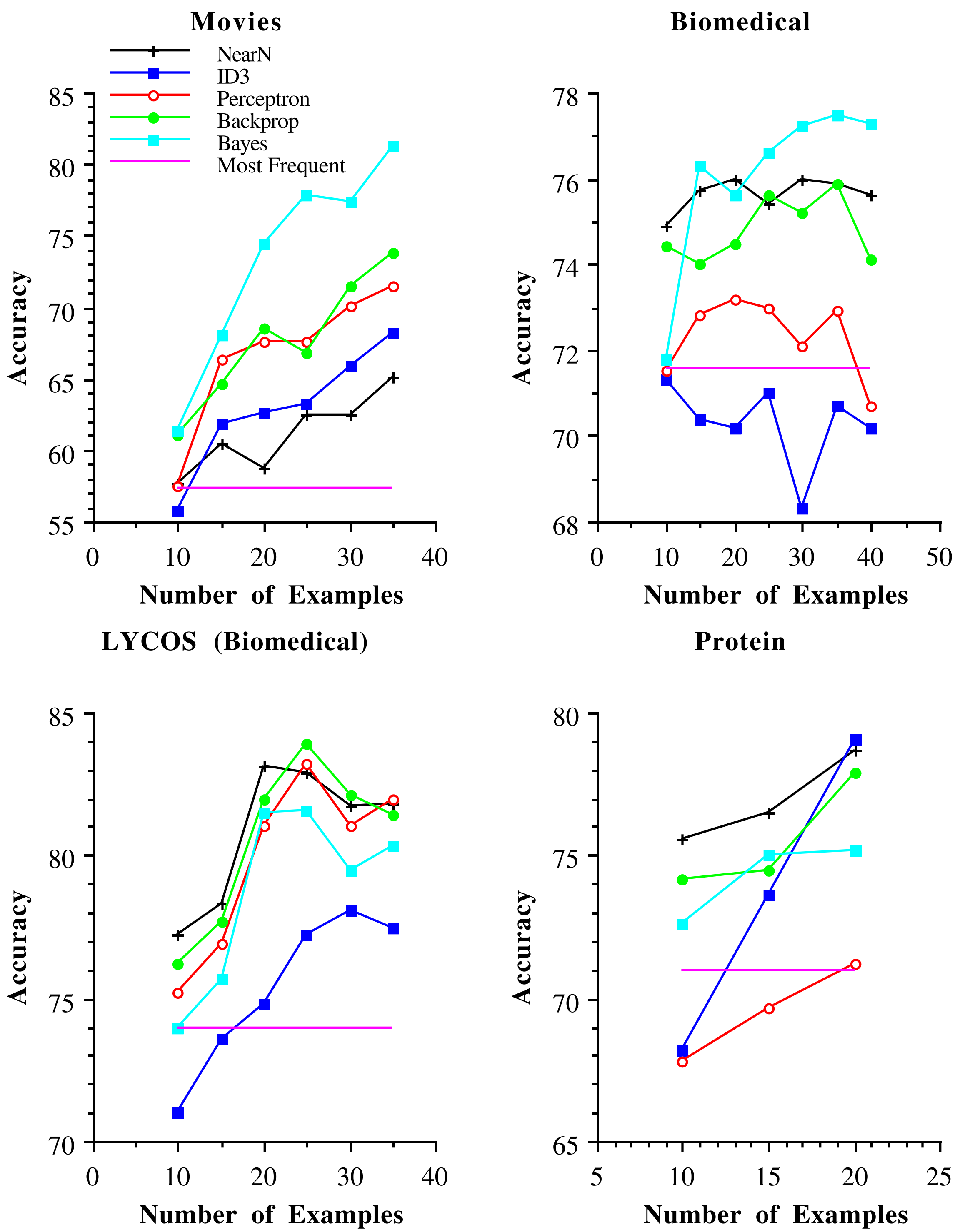

Figure 4. Accuracy of learning algorithms

example, only looking at the first 2048 characters yields an average accuracy of 74.2 while looking at all characters yields an accuracy of 76.3· We have run experiments with all of the other learning algorithms and the general pattern is the same. On average, it is best to analyze the entire file, but the first 1024 -2048 characters is sufficient to achieve nearly the same accuracy. Usually, less than 512 characters results in a significant decrease in accuracy.

Although there is a small decrease in accuracy when looking at the initial segment of the file, in the next version of Syskill & Webert we will store and process only an initial segment to reduce the transmission overhead associated with fetching pages to rank. We also note that many users of Syskill & Webert rate a page as interesting without looking at the entire page and that many information retrieval systems index abstracts rather than the full text of a document.

Anecdotally, we have observed that some errors in Syskill & Webert are caused by analyzing too much of a page. For example, Syskill & Webert sometimes rates a page as interesting to a user when it is not. Sometimes this occurs because the page itself is uninteresting, while at the bottom of the page, there are pointers and short descriptions of related interesting sites. This may explain why in some domains the accuracy of Syskill and Webert is improved when only analyzing an initial segment.

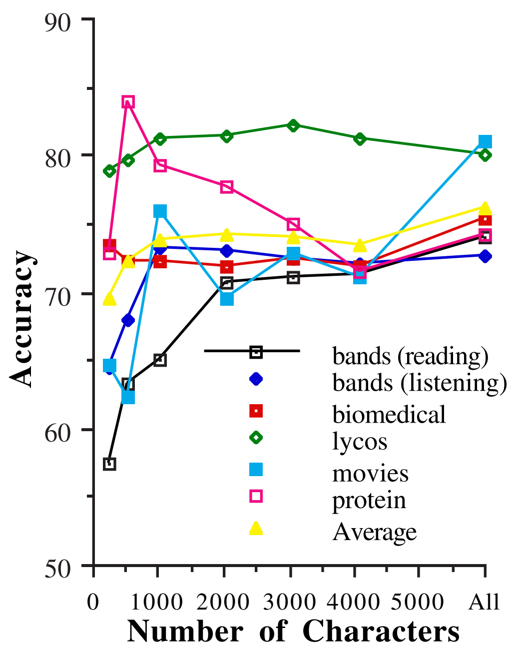

Figure 5 The effect of only analyzing the initial *c* characters of a file.

5 Related work

The methods developed for our learning agent are related to work in information retrieval and relevance feedback (e.g., Salton & Buckley, 1990; Croft & Harper, 1979). However, rather than learning to adapt user queries, we are developing a user profile that may be used for classification tasks such as filtering new information as it becomes available. We experimented with several IR algorithms to perform classification tasks (e.g., by weighting the nearest neighbor similarity metric by the TF-IDF weight, or applying a variant of Rocchio's method (Rocchio, 1971). We do not report on the results in this paper due to space limitations, but in our initial experiments the Bayesian classifier was nearly always more accurate.

There are several other agents designed to perform tasks similar to ours. The WebWatcher (Armstrong, Freitag, Joachims, and Mitchell, 1995) system is designed to help a user retrieve information from Web sites. When given a description of a goal (such as retrieving a paper by a particular author), it suggests which links to follow to get from a starting location to a goal location. It learns by watching a user traverse the WWW and it helps the user when similar goals occur in the future. The WebWatcher and the work described here serve different goals. In particular, the user preference profile learned by Syskill & Webert may be used to suggest new information sources related to ones the user is interested in.

6 Future Work

We are planning two types of enhancements to Syskill & Webert. First, we will investigate improvements to the underlying classification technology:

- Explore the use of ordinal rather than Boolean features. Boolean features indicate whether a word is present or absent in a document. The ordinal features could indicate the number of times a word is present. Note that words include items such as "jpg" and "html" so these may be used to make decisions based on the number of pictures or links if they are informative. Like extensions to TF-IDF, these may be normalized to the length of the document.
- Using linguistic and hierarchical knowledge in the forming of features. Linguistic routines such as stemming (i.e., finding the root forms of words) may improve the accuracy of Syskill & Webert by having a smaller number of more informative features. Currently, words in the pages are used as features without any knowledge of the relationship between words such as "protein" and "proteins." Semantic knowledge such as the relationship between "pigs" and "swine" may also prove useful. Similarly, knowing that "gif," "jpg" and "jpeg" are all extensions of graphic files would facilitate Syskill & Webert learning that a user has a preference for (or against) pages with in-line graphics.

Another set of enhancements to Syskill & Webert involve the redesign of the user interface to make it more interactive. We are currently reimplementing many of its capabilities as a Netscape plugin. One important advantage of this reimplementation is that the user profile can then be stored on the client rather than on the Syskill & Webert server. We are also exploring several other enhancements to the interface that will make it easier to use (and as a consequence allow us to collect more data for our experiments).

- Implementing routines that interactively annotate the index page with Syskill & Webert's predictions as the initial 2k of each link is processed. Currently, no annotations are added until all links have been retrieved and rated. We allow the user to prefetch links so that the rating can occur rapidly, but this does require patience the first time Syskill & Webert is used and disk space to store local copies of files.
- Currently, Syskill & Webert retrieves the original source of a page to determine its interestingness. Several of the Web search engines such as LYCOS store a summary of the page. We are implementing routines to use this summary for rating the interestingness of a page. Combined with the previous option, this will reorder the suggestion made by LYCOS based on the user's profile. This may be particularly useful with "CyberSearch" which is a copy of much of the LYCOS database on CD-ROM eliminating the network connection overhead as well as the network transmission overhead

7 Conclusions

We have introduced an agent that collects user evaluations of the interestingness of pages on the World Wide Web. We have shown that a user profile may be learned from this information and that this user profile can be used to determine what other pages might interest the user. Such pages can be found immediately accessible from a user-defined index page for a given topic or by using a Web search engine. Experiments on six topics with four users showed that the Bayesian classifier performs well at this classification task, both in terms of accuracy and efficiency. Other learning algorithms that make classifications based on combining evidence from a large number of features also performed well. ID3 was not very accurate perhaps since it tries to minimize the number of features it tests to make a classification and accurate classifications cannot be based on the presence or absence of a few words. Further experimentation showed that nearly the same classification accuracy could be achieved by looking only at the initial portion of a page, suggesting an enhancement to the interface that reduces storage space and network transmission overhead.

Acknowledgments

The research reported here was supported in part by NSF grant IRI-9310413 and ARPA grant F49620-92-J-0430 monitored by AFOSR.

References

Armstrong, R. Freitag, D., Joachims, T., and Mitchell, T. (1995). WebWatcher: A learning apprentice for the World Wide Web.

Croft, W.B. & Harper, D. (1979). Using probabilistic models of document retrieval without relevance. *Journal of Documentation, 35,* 285-295.

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

Lang, K. (1995). NewsWeeder: Learning to filter news. *Proceedings of the Twelfth International Conference on Machine Learning.* Lake Tahoe, CA.

Maudlin, 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*

Minsky, M., & Papert, S. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

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

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In D. Rumelhart and J. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations,* (pp 318-362). Cambridge, MA: MIT Press.

J. Rocchio (1971) Relevance Feedback in Information Retrieval. In Gerald Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing, pages* 313 - 323. Prentice Hall, Englewood Cliffs, NJ.

Salton, G. & Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science, 41,* 288-297.

Widrow, G., & Hoff, M. (1960). Adaptive switching circuits. Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4.