IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

PERSONALIZED USER MODEL, L.L.P.,	)	
	)	
Plaintiff,	)	
	)	
v.	)	C.A. No. 09-525-LPS
	)	
GOOGLE INC.,	)	**JURY TRIAL DEMANDED**
	)	
Defendant.	)	
GOOGLE, INC.	)	
	)	
Counterclaimant,	)	
	)	
v.	)	
	)	
PERSONALIZED USER MODEL, LLP and YOCHAI KONIG	)	
	)	
Counterdefendants.	)	

DECLARATION OF JOSHUA L. SOHN IN SUPPORT OF ERRATA TO DEFENDANT GOOGLE INC.'S SUMMARY JUDGMENT PAPERS

I, Joshua L. Sohn, declare as follows:

1. I am an attorney in the law firm on Quinn Emanuel Urquhart & Sullivan, LLP, counsel for Defendant Google in this action. I have personal knowledge of the matters set forth in this Declaration and, if called as a witness, could testify competently thereto.

2. Attached hereto as Exhibit 1 is a true and correct copy of an article entitled "Syskill & Webert: Identifying interesting web sites," authored by Michael Pazzani, Jack Muramatsu & Daniel Billsus. This article was marked as Exhibit 1 in the November 17, 2012 deposition of Dr. Pazzani in this Action.

3. Attached hereto as Exhibit 2 is a true and correct copy of an article entitled "Learning and Revising User Profiles: The Identification of Interesting Web Sites," authored by

Michael Pazzani and Daniel Billsus. This article was marked as Exhibit 2 in the November 17, 2012 deposition of Dr. Pazzani in this Action.

I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct.

Executed this 1st day of May 2013, in Washington, D.C.

Joshua L. Sohn

1104462 / 34638

EXHIBIT 1

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

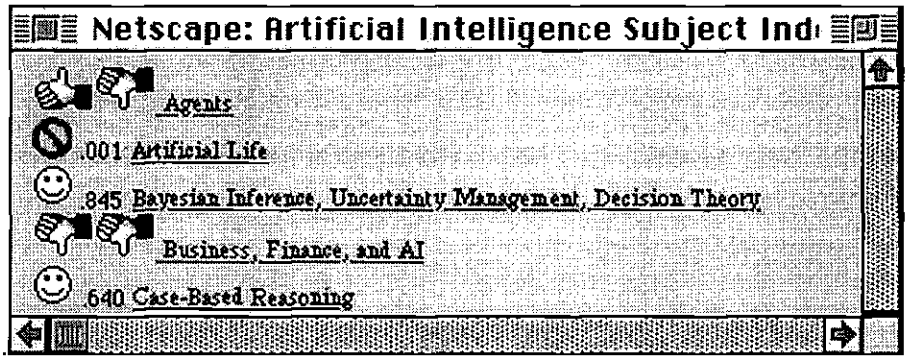

Figure 1. An example of a page annotated by Syskill & Webert.

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

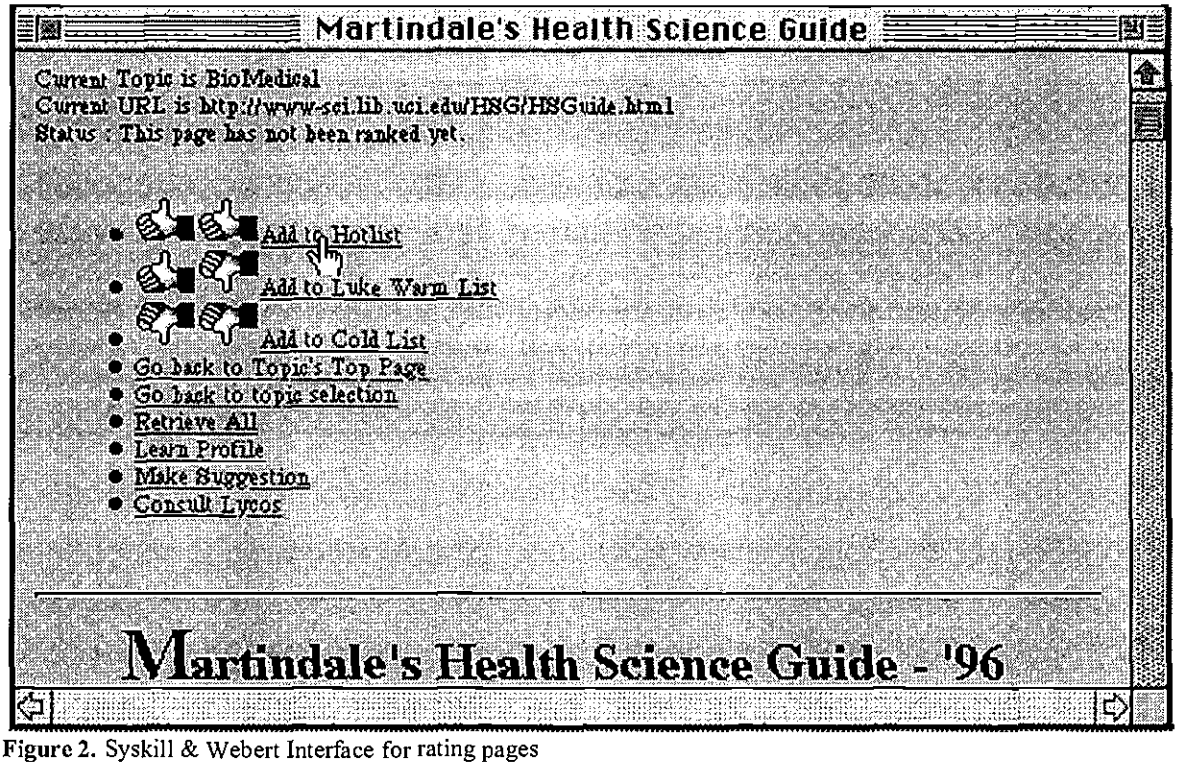

Figure 2. Syskill & Webert Interface for rating pages

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.

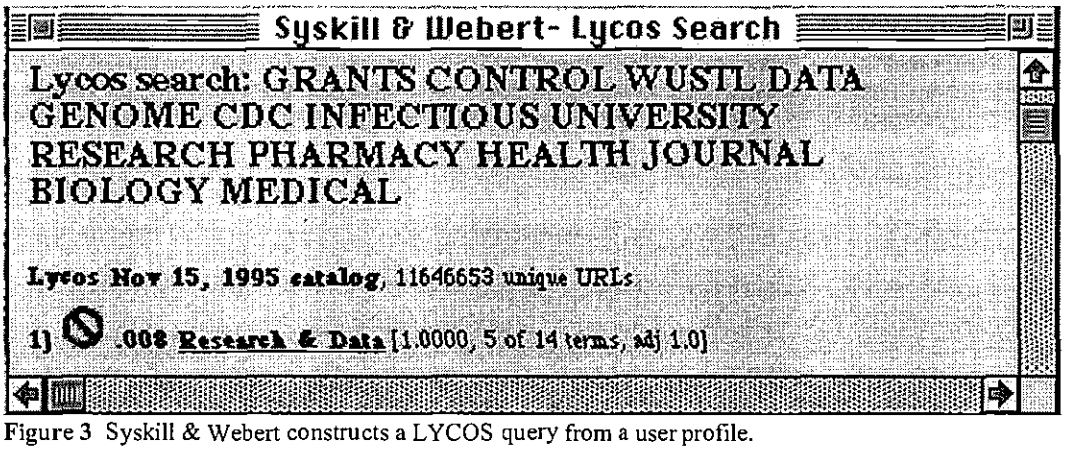

Figure 3 Syskill & Webert constructs a LYCOS query from a user profile.

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

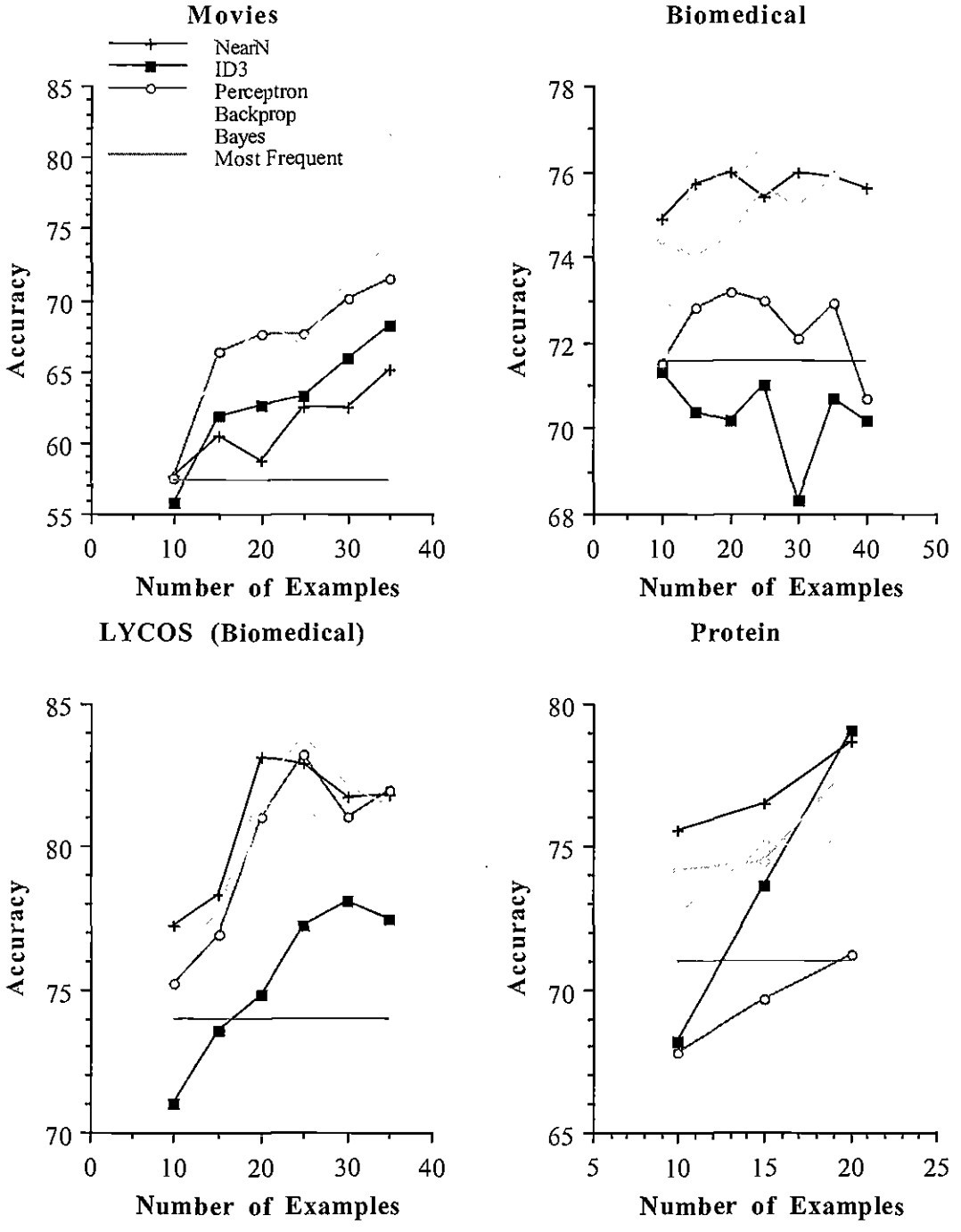

Figure 4. Accuracy of learning algorithms

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.

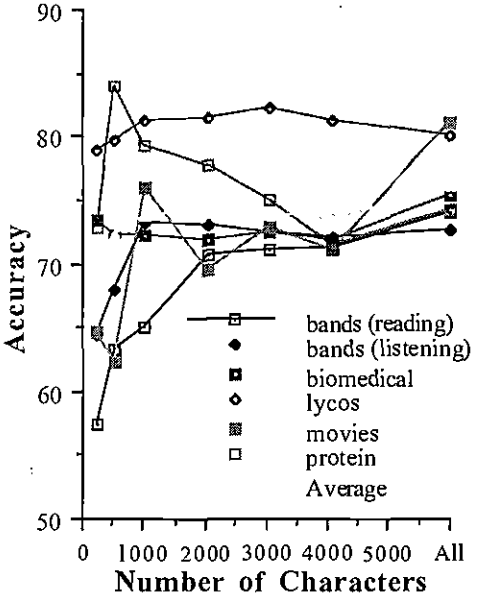

Figure 5 The effect of only analyzing the initial c characters of a file.

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.

EXHIBIT 2

Machine Learning 27, 313–331 (1997)

Learning and Revising User Profiles: The Identification of Interesting Web Sites

MICHAEL PAZZANI pazzani@ics.uci.edu
DANIEL BILLSUS dbillsus@ics.uci.edu
Department of Information and Computer Science, University of California, Irvine, Irvine, CA 92697

Editors: Ryszard S. Michalski and Janusz Wnek

Abstract. We discuss algorithms for learning and revising user profiles that can determine which World Wide Web sites on a given topic would be interesting to a user. We describe the use of a naive Bayesian classifier for this task, and demonstrate that it can incrementally learn profiles from user feedback on the interestingness of Web sites. Furthermore, the Bayesian classifier may easily be extended to revise user provided profiles. In an experimental evaluation we compare the Bayesian classifier to computationally more intensive alternatives, and show that it performs at least as well as these approaches throughout a range of different domains. In addition, we empirically analyze the effects of providing the classifier with background knowledge in form of user defined profiles and examine the use of lexical knowledge for feature selection. We find that both approaches can substantially increase the prediction accuracy.

Keywords: Information filtering, intelligent agents, multistrategy learning, World Wide Web, user profiles

1. Introduction

The World Wide Web (WWW) contains a vast amount of information of varying quality. In this paper, we focus on the problem of assisting a person to find information that satisfies long-term, recurring goals (such as finding information on machine applications in medicine) rather than short-term goals (such as find a paper by a particular author (Armstrong et al., 1995)). We define the "interestingness" of a web page as the relevance of the page with respect to the user's long-term information goals. Feedback on the interestingness of a set of previously visited sites can be used to learn a profile that would predict the interestingness of unseen sites. Since learning accurate classifiers often requires a great deal of data and users may be unwilling to rate many sites before accurate predictions are made, we allow the user to provide an initial profile that is then revised when the user rates visited sites. We will show that revising profiles results in more accurate classifications, particularly with small training sets.

In this paper, we give a brief overview of Syskill & Webert (Pazzani et al., 1996), an intelligent agent we designed to learn profiles. Syskill & Webert identifies informative words from Web pages to use as Boolean features, and learns a naive Bayesian classifier to determine the interestingness of pages. After an initial description of Syskill & Webert, we then discuss a number of issues in the learning and revision of profiles.

- Do more sophisticated and computationally expensive classifiers provide a benefit over the naive Bayesian classifier?

- How can an initial user profile be used to increase the accuracy of initial predictions?
- What additional knowledge can be brought to bear on the problem to increase the accuracy of predictions?

2. Syskill and Webert

Syskill & Webert is designed to help users distinguish interesting Web pages on a particular topic from uninteresting ones. Because the criteria that are used to determine whether a page on one topic is interesting are likely to depend upon the topic, a different profile is learned for each topic. Similarly, different users may not agree on the interestingness of a page. Therefore, each user has a set of profiles, one for each topic.

2.1. User interface

To use Syskill & Webert, users indicate whether they wish to continue exploring an existing topic, or wish to create a new topic. To create a topic, the user first gives the topic a name (e.g., Biomedical) and the URL of an existing *index page*, and optionally an initial profile to distinguish interesting pages on this topic from uninteresting ones. We will discuss the representation and revision of user profiles in Section 4. An index page is simply a web page with many links to other pages on the topic. Fortunately, a number of such pages are manually created and maintained on a variety of topics. For example, `http://www.ohsu.edu/cliniweb/wwwvl/all.html` contains links to over 500 web sites on Biomedical topics.

Once the user has identified a topic, the user can explore the Web, and each page that is displayed is augmented with additional controls that can be selected to collect user ratings on the current page, and to instruct Syskill & Webert to learn a profile, suggest which links on the current page are interesting, or to consult LYCOS (Mauldin & Leavitt, 1994), a Web search engine, to help find interesting pages. Figure 1 shows an example of a Web page annotated with such controls.

The user may rate a page as either hot (two thumbs up) or cold (two thumbs down), and the ratings are saved by Syskill & Webert so that the pages may be used as positive or negative examples when learning a profile[1]. When the user wants advice on which links to explore from the current page, the user selects "Make Suggestion" and the current page is annotated with symbols indicating suggestions or prior ratings. Figure 2 shows an example of a page (`http://ai.iit.nrc.ca/subjects/ai_subjects.html`) after it has been annotated.

In Figure 2, the user has indicated interest in "Machine Learning" (indicated by two thumbs up), and no interest in "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 has not visited the page and Syskill & Webert recommends the page to the user. From this page, the "Neural Networks" link is highly recommended. 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

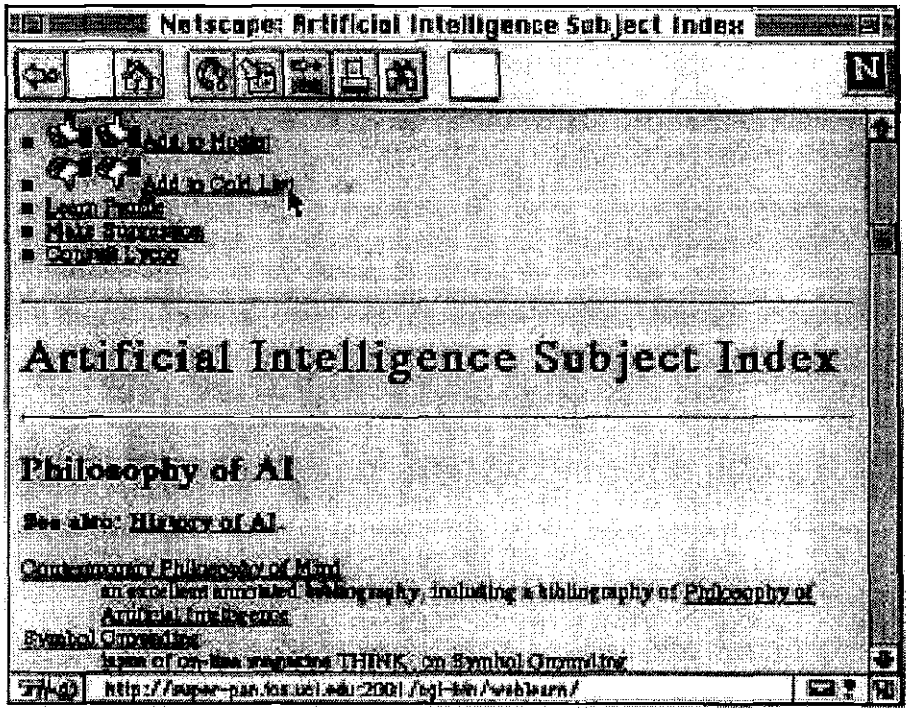

Figure 1. Syskill and Webert interface for rating pages.

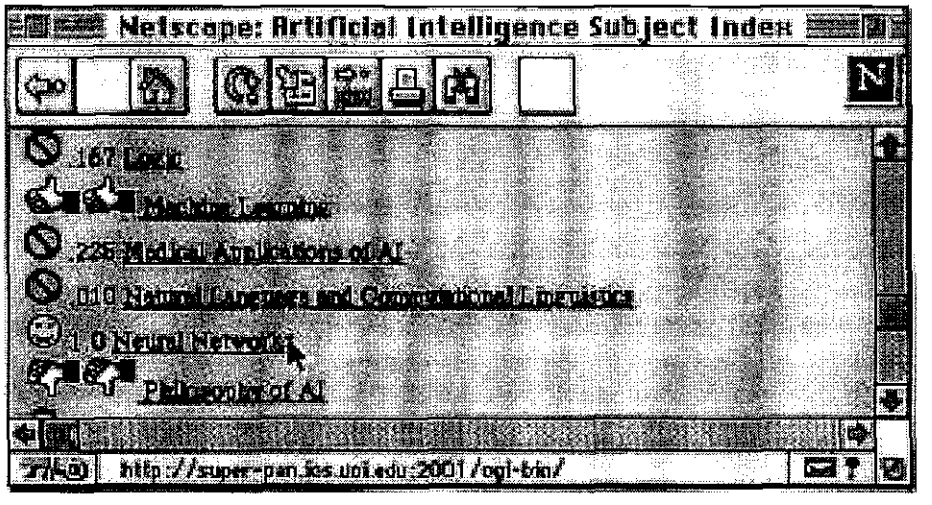

Figure 2. An example of a page annotated by Syskill & Webert.

probability the user would like the page. This probability is determined by a naive Bayesian classifier (Duda & Hart, 1973).

Note that Syskill & Webert is not restricted to just pages that can be accessed from an index page. In particular, it has the ability to construct queries of search engines (consisting of a combination of highly informative words and words that occur frequently on interesting pages), and it can make suggestions about which pages returned from a search engine

are interesting. Pazzani et al. (1996) contains more information on constructing LYCOS queries. In this paper, we focus on learning issues.

2.2. *Learning user profiles*

Supervised 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). Most machine learning programs require that the examples be represented as a set of feature vectors. Therefore, we convert 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. 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 the 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 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 accomplished 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, and $(S_{w=present})$ is the set of pages that contain at least one occurrence of W and S_c are the pages that belong to the class c.

Using this approach, we find the set of k most informative words. In the initial experiments in this paper, we use the 128 most informative words. We will return to the issue of determining how many informative features to use later in this paper.

In a sample of 20 web pages, there are often 5,000 or more unique words, and it is likely that with such a large collection of words and such a small sample of pages, some words would appear to be informative that would not be informative with a larger or more representative sample. This can particularly be a problem with frequently occurring words that happen by chance to occur in a higher (or lower) percentage of hot pages than cold pages. Syskill & Webert, like many information retrieval systems (e.g., Salton (1989)) attempts to mitigate this problem by having a stop list, i.e., a list of approximately 600 frequently occurring English words (and HTML commands) that typically are not very relevant to classification problems. Words on the stop list (e.g., "the," "is," "very," and "if") are always excluded from consideration as informative words.

Table 1. Some of the words used as features.

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

Table 1 shows some of the most informative words obtained from a collection of 80 HTML documents on goats.

Once the HTML source for a given topic has been converted to positive and negative examples represented as feature vectors, it is possible to run many learning algorithms on the data. We use a naive Bayesian classifier in Syskill & Webert, as it has several properties (prediction time, ease of adding prior knowledge) which make it an appropriate learning algorithm for this task. In Section 3, we compare the naive Bayesian classifier to some alternative classification algorithms and explain in greater detail why we chose it as the default algorithm for Syskill & Webert.

The Bayesian classifier (Duda & Hart, 1973) is a probabilistic method for classification. It can be used to determine the probability that an example j belongs to class C_i given values of attributes of the example:

$$P(C_i \mid A_1 = V_{1_j} \& \ldots \& A_n = V_{n_j})$$

If the attribute values are independent, this probability is proportional to:

$$P(C_i) \prod_k P(A_k = V_{k_j} \mid C_i)$$

Both $P(A_k = V_{k_j} \mid C_i)$ (i.e., the probability that a page contains a word given that it is hot) and $P(C_I)$ (e.g., the probability that a page is hot) may be estimated from training data. To determine the most likely class of an example (i.e., either hot or cold), the probability of each class is computed. An example is assigned to the class with the highest probability.

2.3. *Initial experiments*

To determine whether it is possible to learn user preferences for web sites accurately, we collected data from four users on a total of nine different user profiles. Two users rated pages on independent recording artists. One 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 read about the bands (due to the lack of sound output on the computer) and indicated whether he'd be interested in the band. The other topics include Sheep and Goats (whose links were obtained from Inktomi searches), Biomedical (once from a topic page

Table 2. Topics used in our experiments.

User	Topic	URL of topic's index page	Pages
A	Bands (listening)	http://www.iuma.com/IUMA-2.0/olas/location/USA.html	57
A	Bio (Topic)	http://golgi.harvard.edu/biopages/medicine.html	127
A	Bio (Lycos)	*Not applicable*/results of a *LYCOS* search on biomedicine	54
A	Goats	*Not applicable*/results of an *Inktomi* search on goats	70
A	Sheep	*Not applicable*/results of an *Inktomi* search on sheep	70
A	Mail	*Not applicable*/converted to web pages	129
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

and once from a LYCOS search), Protein Science, an archive of electronic mail (that was converted to Web pages so that Syskill & Webert could be used to filter mail), and motion pictures. Table 2 lists the sources of the used web pages, and the amount of data collected.

Syskill & Webert is intended to be used to find unseen pages that are relevant to the user's interests. To evaluate the effectiveness of the learning algorithms, it is necessary to run experiments to see if Syskill & Webert's prediction agrees with the user's 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 n 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. The learning algorithm created a representation for the user preferences. Next, the test data (i.e., all data not used as training 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 40 trials of the Bayesian classifier. Figure 3 shows the average accuracy of Syskill & Webert using a Bayesian classifier as a function of the number of training examples (from 10 to 45 examples). Two domains, protein and movies, did not have enough examples to use the larger training sets. The percentage of cold pages (i.e., the most frequent class in all domains) is displayed next to the domain name in the legend of each graph.

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. On many problems, the accuracy increases substantially as more examples are added. However, on two versions of the problem of identifying interesting independent recording artists Syskill & Webert performs at chance levels and adding additional examples does not provide a substantial benefit. This suggests that either the example representations are inadequate to learn an accurate classifier using the naïve Bayes approach or that the problem is very noisy, i.e., the problem contains a lot of information that is not informative with respect to the "interestingness" of recording artists.

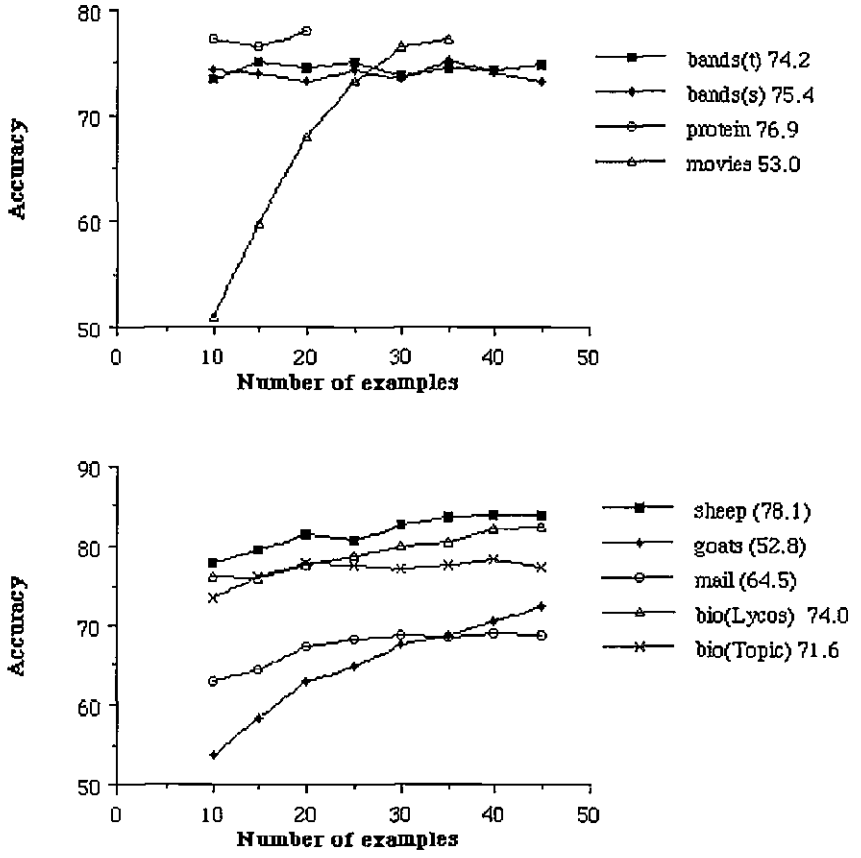

Figure 3. The average accuracy of Syskill & Webert as a function of the number of training examples.

The results illustrate that learning user profiles would be a useful additional capability of Web search engines. For example, one user was interested in only 26% of the biomedical articles returned by LYCOS. In contrast, Syskill & Webert is able to identify with greater than 80% accuracy whether this user would like a biomedical page found by LYCOS.

So far, we have just considered a binary decision of whether a user would or would not be interested in a page. The Bayesian classifier also returns a probability that may be used to rank order pages. 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 on a set of 120 rated pages 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 ten 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 one was rated as interesting by the user.

The decision to use the 128 most informative words as features was made by looking at one initial domain (biomedical) when only 50 examples were collected. Next, we explore the impact of selecting other numbers of features. We show results only for the naive

Table 3. The effect of varying the number of informative features.

Features	Bands Reading	Bands Sound	Biomed Topic	Biomed Lycos	Movies	Protein	Average
16	70.4	73.0	74.5	77.1	74.6	71.6	73.9
32	72.0	71.6	73.6	79.2	76.8	73.8	74.6
64	74.3	71.7	74.1	80.9	73.1	75.0	74.8
96	74.3	75.1	76.9	78.7	72.5	78.8	75.5
128	73.8	73.4	77.3	77.0	74.2	75.1	75.1
200	74.3	73.3	77.1	77.4	71.4	75.0	74.7
256	74.5	73.4	76.9	77.0	71.3	76.5	74.6
400	74.8	73.3	75.5	76.9	69.2	70.6	73.9

Bayesian classifier with 20 training examples (and using all unseen data as test examples). Each domain contains several thousands of distinct words that could be used as potential features. We experimented with selecting 16, 32, 64, 96, 128, 200, 256, and 400 of the most informative words to use as features. The results, averaged over 24 trials, are shown in Table 3. We list six domains separately and also list the average over the six domains.

Table 3 shows that in most domains (and on average) an intermediate number of features performs best. Having too few features can cause problems because important discriminating features are ignored. On the other hand, having too many features can also cause problems if many words that aren't very relevant are used as features. On average for the values we tested, 96 performed best. One might consider using information-gain to select a large group of informative features, and then using existing approaches for feature subset selection (e.g., Kittler, 1986; John et al., 1994) to select some of these features using a criteria other than informativeness. However, such algorithms increase the complexity of the Bayesian classifier, making it impractical to learn a profile interactively. Furthermore, such approaches are likely to overfit the example representation to the training data since in many cases there are more features than examples. We will return to this issue in Section 6 where we propose an approach based on lexical knowledge.

3. Experimental comparisons

Alternatives to the Bayesian classifier were considered for Syskill & Webert. In this section, we compare the Bayesian classifier to several standard machine learning algorithms and present experimental evidence that the Bayesian classifier performs at least as well as these computationally more intensive alternatives.

3.1. Nearest neighbor

The nearest neighbor algorithm (Duda & Hart, 1973) operates by storing all examples in the training set. To classify an unseen instance, it assigns it to the class of the most similar

example. Since all of the features we use are binary features, the most similar example is the one that has the most feature values in common with a test example.

3.2. PEBLS

PEBLS (Cost & Salzberg, 1993) is a nearest neighbor algorithm that makes use of a modification of the value difference metric, MVDM, (Stanfill & Waltz, 1986) for computing the distance between two examples. This distance between two examples is the sum of the value differences of all attributes of the examples. The value difference between two values V_{j_x} and V_{j_y} of attribute A_j is given by:

$$\Delta(V_{j_x}, V_{j_y}) = \sum_i \left| \hat{P}(C_i \mid A_j = V_{j_x}) - \hat{P}(C_i \mid A_j = V_{j_y}) \right|$$

In many ways, PEBLS is similar to a naive Bayesian classifier. However, PEBLS can accurately learn non-linearly separable concepts from Boolean features while the naive Bayesian classifier cannot. If PEBLS were to be consistently more accurate than the Bayesian classifier, then one possible explanation would be that a more complex decision procedure than weighting of evidence across a group of words is necessary.

3.3. Decision trees

Decision tree learners such as ID3 (Quinlan, 1986) build a decision tree by recursively partitioning examples into subgroups until those subgroups contain examples of a single class. A partition is formed by a test on some attribute (e.g., is the feature `database` equal to 0). ID3 selects the test that provides the highest gain in information content.

3.4. Rocchio's algorithm

We have used a version of Rocchio's algorithm (Rocchio, 1971) adapted to text classification by Ittner et al. (1995). Rather than representing a document by a set of Boolean features indicating the presence or absence of a word, Rocchio's method uses the TF-IDF weight for each informative word. TF-IDF is one of the most successful and well-tested weighting schemes in Information Retrieval (IR). The computation of the weights reflects empirical observations regarding text. Terms that appear frequently in one document (TF = term-frequency), but rarely on the outside (IDF = inverse-document-frequency), are more likely to be relevant to the topic of the document. Therefore, the TF-IDF weight of a term in one document is the product of its term-frequency (TF) and the inverse of its document frequency (IDF). In addition, to prevent longer documents from having a better chance of retrieval, the weighted term vectors are normalized to unit length.

Following Ittner et al. (1995), we use the average of the TF-IDF vectors of all examples of the interesting pages, and subtract away a weighted fraction (0.25) of the TF-IDF vectors of the uninteresting pages in order to get a prototype-vector for the interesting class. Subtracting TF-IDF vectors of the uninteresting pages helps to prevent infrequently occurring terms from overly affecting the classification (since it is likely that these terms appear with

similar frequencies in uninteresting pages). The weighted fraction used (0.25) was empirically determined reasonable in TREC-2 (Harman, 1994).. Pages within a certain distance of the prototype (as determined by the cosine similarity measure) are considered interesting. A distance threshold is chosen that maximizes the accuracy on the training set.

3.5. *Neural nets*

We used two approaches to learning with neural nets. In the perceptron approach, there are no hidden units and the single output unit is trained with the delta rule (Widrow & Hoff, 1960). The perceptron is limited to learning linearly separable functions (Minsky & Papert, 1969). We also use multi-layer networks trained with error backpropagation (Rumelhart et al., 1986). We used 12 hidden units in our experiments.

3.6. *Results*

We ran 40 paired trials of seven classification algorithms on the data from the nine domains. Each trial used a training set of size 20 and the remaining data was used as a test set. Twenty examples were chosen as a reasonable intermediate number of examples. Most users would like to get useful feedback as early as possible and most of the classification algorithms have substantial increases in accuracy between 10 and 20 examples on many domains, but not as drastic increases in accuracy between 20 and 50 examples. The results are shown in Table 4. For each domain, we found the algorithm with the highest accuracy and the algorithms that did not have a significant difference from the algorithm with the highest accuracy (as determined by a paired two-tailed t-test at the .05 level). These algorithms are noted with a '+' in Table 4.

Although no one algorithm is clearly superior on these problems, some algorithms do stand out. ID3 is not well suited to this task since it attempts to build trees that test as few features as possible to make a classification and it appears that summing small amounts of evidence over a larger number of features is needed in this task. We have also experimented

Table 4. Average accuracy of the classification algorithms.

Domain	NNeigh	ID3	Percept	BackP	PEBLS	Bayes	Rocchio
Bio(Topic)	74.5	70.2	73.2	76.0+	74.6	77.3+	77.5+
Goats	62.0	64.7	66.3+	67.0+	62.7	62.9	69.4+
Bio(lycos)	80.0+	75.9	80.2+	80.9+	79.9+	78.2	78.1
Mail	63.1	62.4	62.8	64.2	63.3	66.9+	67.9+
Movies	58.6	69.4+	70.5+	67.4	60.0	69.3+	69.0+
Protein	77.0+	73.7	70.4	74.1	77.0+	77.0+	74.6
Sheep	79.3	78.4	78.9	80.5+	79.3	81.5+	78.8
Bands(s)	74.4+	70.7	71.4	73.1+	74.5+	73.4+	73.7+
Bands(t)	75.0+	68.6	69.6	73.9+	75.0+	74.6+	74.5+

with a more advanced decision tree learner C4.5 (Quinlan, 1994) with similar results. Nearest Neighbor performed significantly worse than other approaches in more than half of the experiments. We also experimented with "K Nearest Neighbor" approaches, but modifying the number of neighbors did not result in higher accuracy. Backpropagation, the Bayesian classifier and Rocchio's algorithm consistently perform well on most domains and these algorithms are characterized by summing evidence among a large number of features, but differ according to how the weight of each feature is calculated. There does not appear to be the need for nonlinear classifiers such as neural nets trained by backpropagation, since the Bayesian classifier and Rocchio's method, which are both linear classifiers, perform with similar accuracy to the neural nets. It also does not appear that in this domain there is an advantage in the TF-IDF weights rather than the Boolean features used by the Bayesian classifier.

Although the assumption of attribute independence is clearly an unrealistic one in the context of text classification, we have decided to use the naive Bayesian classifier as the default algorithm in Syskill & Webert for a variety of reasons. It performed well in our experiments, and it also performs well in many domains that contain clear attribute dependencies. Domingos & Pazzani (1996) explore the conditions for the optimality of the naive Bayesian classifier and conclude that it can even be optimal if the estimated probabilities contain large errors. Furthermore, 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. As we shall see in the next section, it is also simple to add some form of prior knowledge to the Bayesian classifier, increasing its accuracy.

4. Using predefined user profiles

There are two drawbacks to Syskill & Webert that can be addressed by providing a learning algorithm with initial knowledge about a user's interests. First, not all the words selected by expected information gain seem related to our intuitions of relevant features for discriminating between interesting and uninteresting pages. This occurs most frequently when there are only a few training examples available. Using many irrelevant features makes the learning task much harder. Second, the classification accuracy leaves some room for improvement particularly with a small number of examples. Since some users might be unwilling to rate many pages before the system can give reliable predictions, initial knowledge about the user's interests can be exploited to give accurate predictions even when there are only a few rated pages.

We elicit an initial profile for a new topic from the user. We ask the user to provide words that are good indicators for an interesting page and allow him to state as many or few words as he can think of. In addition, we ask for words that are good indicators for a page being uninteresting. However, the concept of an "uninteresting web page" is hard to define, because an "uninteresting web page" can be anything diverging from the user's interests. Therefore, it might be hard or somewhat unnatural to think of words that are indicators for uninteresting pages, and the user does not have to state any words if he cannot think of any. However, words like "construction" and "moved" often occur on uninteresting pages.

The initial user profile used in Syskill & Webert consists of a set of probabilities representing the probabilities for the word appearing (or not appearing) in a page, given that the page was rated hot (or cold respectively). Such a probability table $p(word_i \mid class_j)$ contains 4 probabilities, namely $p(word_i\ present \mid hot)$, $p(word_i\ absent \mid hot)$, $p(word_i\ present \mid cold)$, and $p(word_i\ absent \mid cold)$. Note that the user must provide only two of the four probabilities since $p(word_i\ absent \mid class_j)$ is equal to $1 - p(word_i\ present \mid class_j)$. Furthermore, if the user declines to state a probability, default values of .7 for $p(word_i\ present \mid hot)$ and .3 for $p(word_i\ present \mid cold)$ are used. These default values were chosen, as opposed to .5/.5, because we assume that it is more likely that the user will provide words that are indicators for pages being interesting.

After an initial user profile has been assessed, Syskill & Webert needs to determine to what degree it should "believe" in the user's probability estimates and to what extent they should be updated when training data is encountered. This decision should clearly be related to the amount of available training data. When there are initially only a few rated pages available, the system should rely more on the profile given by the user than on estimates formed by looking at only the available rated pages. As more training data becomes available, the system should gradually increase the belief in probability estimates from the data and gradually decrease the weight of the initial user profile.

Since an initial user profile is represented as probability tables, the process of revising the user profile consists of gradually updating the given probabilities in a way to make them reflect the training examples seen. A theoretically sound way of doing this is to express the uncertainty in a probability estimate with a conjugate probability distribution. While observing more training data, this probability distribution can be updated to reflect both the changed expected value of the probability, and the increased confidence in the accuracy of the probability estimate. The probability tables that represent the current profile of the user's interests can be directly used in the simple Bayesian classifier.

Conjugate priors are a technique from Bayesian statistics to update probabilities from data (e.g., Heckerman, 1995). We use the equivalent sample size approach for implementing conjugate priors in Syskill & Webert and by default we weight the prior probability estimate to be equivalent to 50 samples. For example, if the user specifies $p(word_i\ present \mid hot)$ to be 0.8, this is equivalent to having seen 50 hot pages, 40 of which contain the word. After seeing 25 hot pages, 10 of which contain the word, the value of $p(word_i\ present \mid hot)$ used by Syskill & Webert would be 50/75, while if the probability were estimated from the data alone it would be 10/25.

To determine the effect on the classification accuracy of revising probabilities, we ran an experiment comparing three variants of how the Bayesian classifier estimates conditional probabilities and determines which words to use as features:

- **Data**: The 96 most informative words are used as features, and the conditional probabilities are estimated from the data alone, ignoring the initial profile. This is the naive Bayesian classifier used in the previous experiments.
- **Revision**: All words from the user profile are used as features, supplemented with the most informative words for a total of 96 features. The conjugate priors technique is used to estimate probabilities.

- **Fixed**: This approach just uses the words in the user profile as features and always uses the value for conditional probabilities provided by the user. In fact, apart from estimating the class probabilities $p(hot)$ and $p(cold)$, there is no learning involved in this variant. However, it provides a basis to assess the utility of the initial user profile.

The goal of the experiments is to determine if the revision strategy is more accurate than the other two. If this is the case, then there is a benefit in obtaining an initial profile from the user and revising it with feedback. Tables 5, 6 and 7 show the profiles for three representative domains. Figures 4, 5 and 6 show the accuracy of the three variants of the Bayesian classifier on these domains averaged over 25 trials.

The results shown in Figures 4, 5 and 6 clearly demonstrate that the strategy of revising an initial user profile is more accurate than the other two strategies. The benefit of the revision strategy is most dramatic for the goats problem where the other two strategies are approximately 70% accurate and the revision strategy is approximately 85% accurate. Perhaps the most surprising finding is that the "Fixed" strategy does so well. People are usually not very reliable in estimating conditional probabilities. However, it is possible that people are extremely good at identifying keywords that distinguish interesting pages from uninteresting ones. We will explore this hypothesis in the next experiment.

To determine whether the Revision strategy is successful due to the influence of the user specified keywords or the use of initial probability estimates, we have tested a Bayesian classifier that uses only the words in the user profile as features, but estimates all probabilities

Table 5. User profile for the bands domain.

guitar .9 .5	*guitars .9 .5*	*acoustic .9 .5*
independent .9 .1	*nirvana .9 .1*	*pumpkins .9 .2*
alternative .9 .1	*college .9 .3*	*folk .9 .5*
synthesizer .1 .6	*keyboard .1 .6*	*dance .1 .6*

Table 6. User profile for the biomedical domain.

grants .7 .2	*database .8 .1*	*genome .6 .3*
molecular .6 .1	*protein .5 .2*	*prediction .9 .1*
classification. 9 1	*structure .6 .2*	*function .6 .1*
webmaster .05 .1	*com .1 .4*	

Table 7. User profile for the goats domain.

dairy .7 .3	*pygmy .7 .3*	*angora .7 .3*
cashmere .7 .3	*milk .7 .3*	*doe .7 .3*
farm .7 .3	*buck .7 .3*	*wether .7 .3*
sheep .7 .3	*animals .7 .3*	*hay .7 .3*
wine .3 .7	*hill .3 .7*	

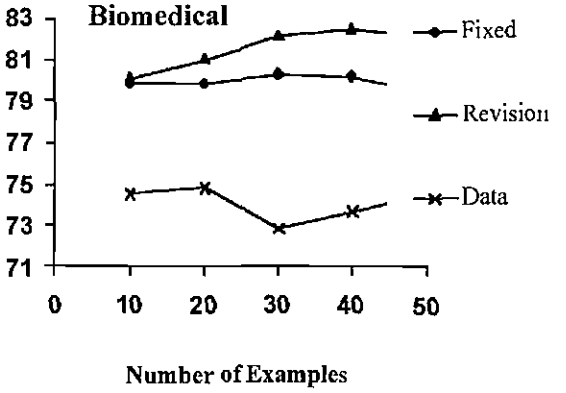

Figure 4. The accuracy of three variants of the Bayesian classifier on the Biomedical domain.

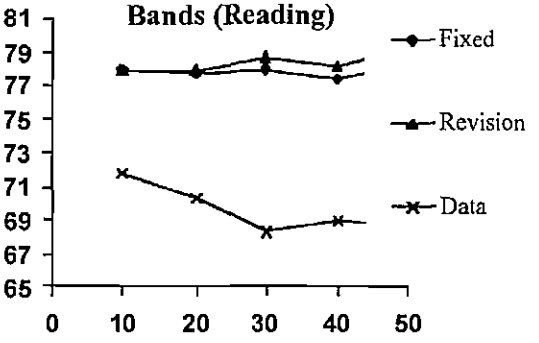

Figure 5. The accuracy of three variants of the Bayesian classifier on the independent recording artists domain.

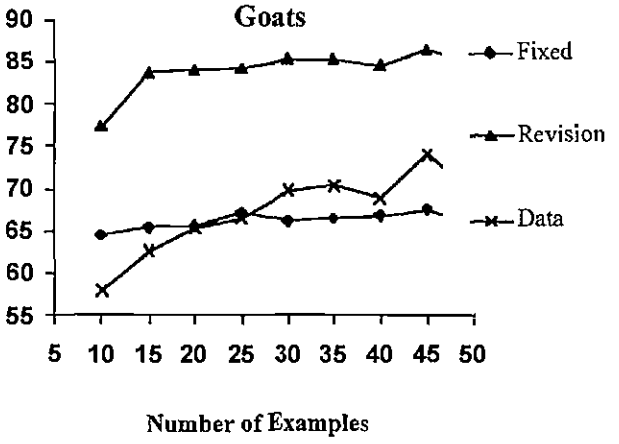

Figure 6. The accuracy of three variants of the Bayesian classifier on the goats domain.

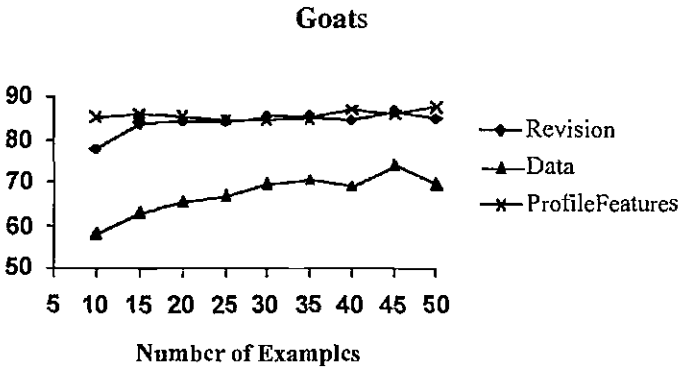

Figure 7. Using only the words from the user profile as features provides a benefit similar to the conjugate prior revision method.

from the data (ignoring the user's estimates). Such a classifier typically uses 10–15 features instead of the larger number used by Syskill & Webert. Although we have found that using a small number of features selected to maximize information gain results in low predictive accuracy, our experiments with a small number of user-selected features show that this technique is very promising. Figure 7 shows that using just the user's features (which is labeled "ProfileFeatures" in the legend) is at least as accurate, and sometimes more accurate than the conjugate priors revision strategy. Similar results were found in other domains not shown here.

5. Using lexical knowledge

The results of the experiment shown in Figure 7 together with an inspection of the words used in user profiles suggest that a considerable benefit could be gained by having knowledge of the relationship between words that could be used instead of a simple stoplist to remove words unrelated to the topic from consideration as features. For example, some of the words selected as informative in the goats domain have included "took," "other," and "however." Such words did not occur in the user's profile, and it is unlikely that in a larger sample, they would be considered informative.

While it might not be possible to easily capture all of a person's intuition about a domain, some knowledge of the words is available. WORDNET (Miller, 1990), is a lexical database containing the relationship between approximately 30,000 commonly occurring English words. When there is no relationship between a word and words in a topic (e.g., goat) Syskill & Webert can eliminate that word from consideration as a feature[2]. This is accomplished by following any of the Hypernym, Antonym, Member-Holonym, Part-Holonym, Similar-to, Pertainnym, or Derived-from links in WordNet. Table 8 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 might be related. Most of the words that are not eliminated are related to the topic (with a few exceptions e.g., "computer"). Figure 8 shows the effect of using WordNet to filter features on the

Table 8. 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~~

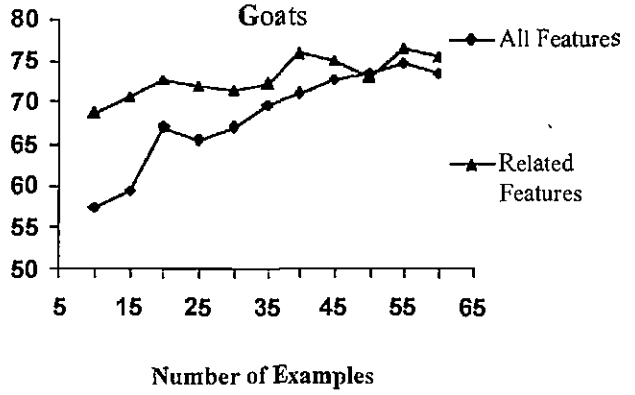

Figure 8. Using only the informative words that are related to the topic results in increased accuracy, particularly with small training sets.

goat domain averaged over 25 trials. The graph shows that there is a substantial increase in accuracy when using lexical information when there are not many training examples. As the number of example increases, the effect is less substantial.

6. Related work

The Bayesian classifier has a long history in text categorization tasks (e.g., Maron, 1961; Lewis 1992). Our work differs from previous work in this area by focusing on incorporating additional knowledge into the classifier, as well as being focused on the World Wide Web.

There are several other agents designed to perform tasks similar to ours. The WebWatcher (Armstrong et al., 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 may be used to suggest new information sources related to ones the user is interested in.

Letizia (Lieberman, 1995) is a software agent that is based on a slightly different learning task. The system monitors the user while he's browsing the WWW, and tries to infer the user's interests based on browsing behavior. The advantage of an approach like this is that the user does not have to explicitly indicate his likes or dislikes. However, the heuristics

used to infer users' interests might give a useful approximation, but they won't result in a profile which is as accurate as a profile based on users' explicit ratings.

Like our work, WebHound (Lashkari, 1995) is designed to suggest new Web pages that may interest a user. WebHound uses a collaborative approach to filtering. In this approach, a user submits a list of pages together with ratings of these pages. The agent finds other users with similar ratings and suggests unread pages that are liked by others with similar interests. One drawback of the collaborative filtering approach is that when new information becomes available, others must first read and rate this information before it may be recommended. In contrast, by learning a user profile, our approach can determine whether a user is likely to be interested in new information without relying on the opinions of other users. Furthermore, the profile learned also contains information that can be used to create queries of Web search engines such as LYCOS. However, one advantage of the collaborative approaches is that they do not require transmission and analysis of the HTML source of Web pages.

7. Future directions

We are planning two types of enhancements to Syskill & Webert. First, we will investigate improvements to the underlying classification technology. However, rather than working on the classification algorithm, the representation of profiles and the use of linguistic and hierarchical knowledge in the forming of features will be pursued. This had a much larger impact than particular classification algorithm used in our experiments.

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 so that the user profile can then be stored on the client rather than on the Syskill and 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 displays its rating as annotations on the current page that is displayed. We are planning on an option that will create a new page with Syskill & Webert's rankings sorted by the probability estimate that the page is hot.
- 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.
- We plan on monitoring the topic page of each topic of a user, and notifying the user when a new link is added to the page that is rated as interesting.

8. 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 nine 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 the presence or absence of a single feature can determine the class assigned by ID3. Additional experimentation showed that revising an initial user profile can increase the accuracy of the classifier and the major benefit of the user profile is in selecting features that are relevant to the classification task. A final experiment demonstrated that benefit may also be achieved by consulting a thesaurus when selecting features.

Notes

1. An earlier version of Syskill & Webert, allowed for an intermediate "lukewarm" rating, but we found that it was used infrequently, and the distinction between lukewarm and cold was not necessary to meet the primary goal of identifying hot pages.
2. We considered using a distance metric between concepts, rather than deciding whether or not there is any relationship. However, early experiences with a distance metric were disappointing in that there did not seem to be a correlation between the number of links traversed to connect two concepts and our "intuitive" measure of the relationship.

References

Armstrong, R., Freitag, D., Joachims, T., & Mitchell, T. (1995). WebWatcher: A learning apprentice for the World Wide Web. *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments* (pp. 6–12). Palo Alto, CA.

Balabanovic, Shoham, & Yun. (1995). An adaptive agent for automated web browsing (Technical Report CS-TN-97-52). Stanford University, Palo Alto, CA.

Cost, S., & Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

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

Domingos, P., & Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the Simple Bayesian Classifier. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 105–112). Morgan Kaufmann, San Fransico, CA.

Duda, R., & Hart, P. (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York.

Harman, D.K. (1994). Overview of the second Text Retrieval Conference (TREC-2). *Proceedings of the Second Text Retrieval Conference (TREC-2)*, NIST Special Publication.

Heckerman, D. (1995). *A Tutorial on Learning with Bayesian Networks* (Technical Report MSR-TR-95-06). Microsoft Corporation.

Ittner, D., Lewis, D., & Ahn, D. (1995). Text categorization of low quality images. *Symposium on Document Analysis and Information Retrieval* (pp. 301–315). UNLV, Las Vegas, NV, ISRI.

John, G., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 121–138). New Brunswick, NJ.

Kittler, J. (1986). Feature selection and extraction. In Young, & Fu, (Eds.), *Handbook of Pattern Recognition and Image Processing*. Academic Press, New York.

Kononenko, I. (1990). Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. In B. Wielinga (Ed.), *Current Trends in Knowledge Acquisition*. IOS Press, Amsterdam.

Lang, K. (1995). NewsWeeder: Learning to filter news. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 331–339). Lake Tahoe, CA.

Lashkari, Y. (1995). The WebHound Personalized Document Filtering System. `http://rg.media.mit.edu/projects/webhound/`

Lewis, D. (1992). Representation and learning in information retrieval. Doctoral dissertation, Department of Computer and Information Science, University of Massachusetts.

Lieberman, H. (1995). Letizia: An agent that assists web browsing. *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 924–929), Montreal, August 1995.

Maron, M. (1961). Automatic indexing: An experimental inquiry. *Journal of the Association for Computing Machinery*, 8:404–417.

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*. The MITRE Corporation, McLean, Virgiana.

Miller, G. (1991). WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4), 235–312.

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

Pazzani, M., Muramatsu J., and Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. *Proceedings of the National Conference on Artificial Intelligence* (pp. 54–61). Portland, OR.

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

Rachlin, Kasif, Salzberg, & Aha, (1994). Towards a better understanding of memory-based reasoning systems. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 242–250). New Brunswick, NJ.

Rocchio, J. (1971). Relevance feedback information retrieval. In Gerald Salton (Ed.), *The SMART Retrieval System—Experiments in Automated Document Processing* (pp. 313–323). Prentice-Hall, Englewood Cliffs, NJ.

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

Salton, G. (1989). *Automatic Text Processing*. Addison-Wesley.

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

Skalak, D. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 293–301). New Brunswick, NJ.

Stanfill, C., & Waltz, D. (1986). Towards memory-based reasoning. *Communications of the ACM*, 29:1213–1228.

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

Received July 3, 1996
Accepted October 23, 1996
Final Manuscript November 7, 1996

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

CERTIFICATE OF SERVICE

I, David E. Moore, hereby certify that on May 1, 2013, the attached document was electronically filed with the Clerk of the Court using CM/ECF which will send notification to the registered attorney(s) of record that the document has been filed and is available for viewing and downloading.

I further certify that on May 1, 2013, the attached document was Electronically Mailed to the following person(s):

Karen Jacobs Louden
Jeremy A. Tigan
Morris, Nichols, Arsht & Tunnell LLP
1201 North Market Street, 18th Fl.
P.O. Box 1347
Wilmington, DE 19899-1347
klouden@mnat.com
jtigan@mnat.com

Marc S. Friedman
Dentons US LLP
1221 Avenue of the Americas
New York, NY 10020-1089
marc.friedman@dentons.com

Jennifer D. Bennett
Matthew P. Larson
Dentons US LLP
1530 Page Mill Road, Ste. 200
Palo Alto, CA 94304-1125
jennifer.bennett@dentons.com
matthew.larson@dentons.com

Mark C. Nelson
Robert Needham
Dentons US LLP
2000 McKinney, Suite 1900
Dallas, TX 75201
mark.nelson@dentons.com
robert.needham@dentons.com

Christian E. Samay
Dentons US LLP
101 JFK Parkway
Short Hills, NJ 07078
christian.samay@dentons.com

/s/ David E. Moore
Richard L. Horwitz
David E. Moore
Bindu A. Palapura
POTTER ANDERSON & CORROON LLP
(302) 984-6000
rhorwitz@potteranderson.com
dmoore@potteranderson.com
bpalapura@potteranderson.com

932168 / 34638