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
phone: (714) 824-5888
fax (714) 824-4056
http://www.ics.uci.edu/~pazzani/

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 a 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 four different learning algorithms and TF-IDF, an approach to weighting words used in information retrieval

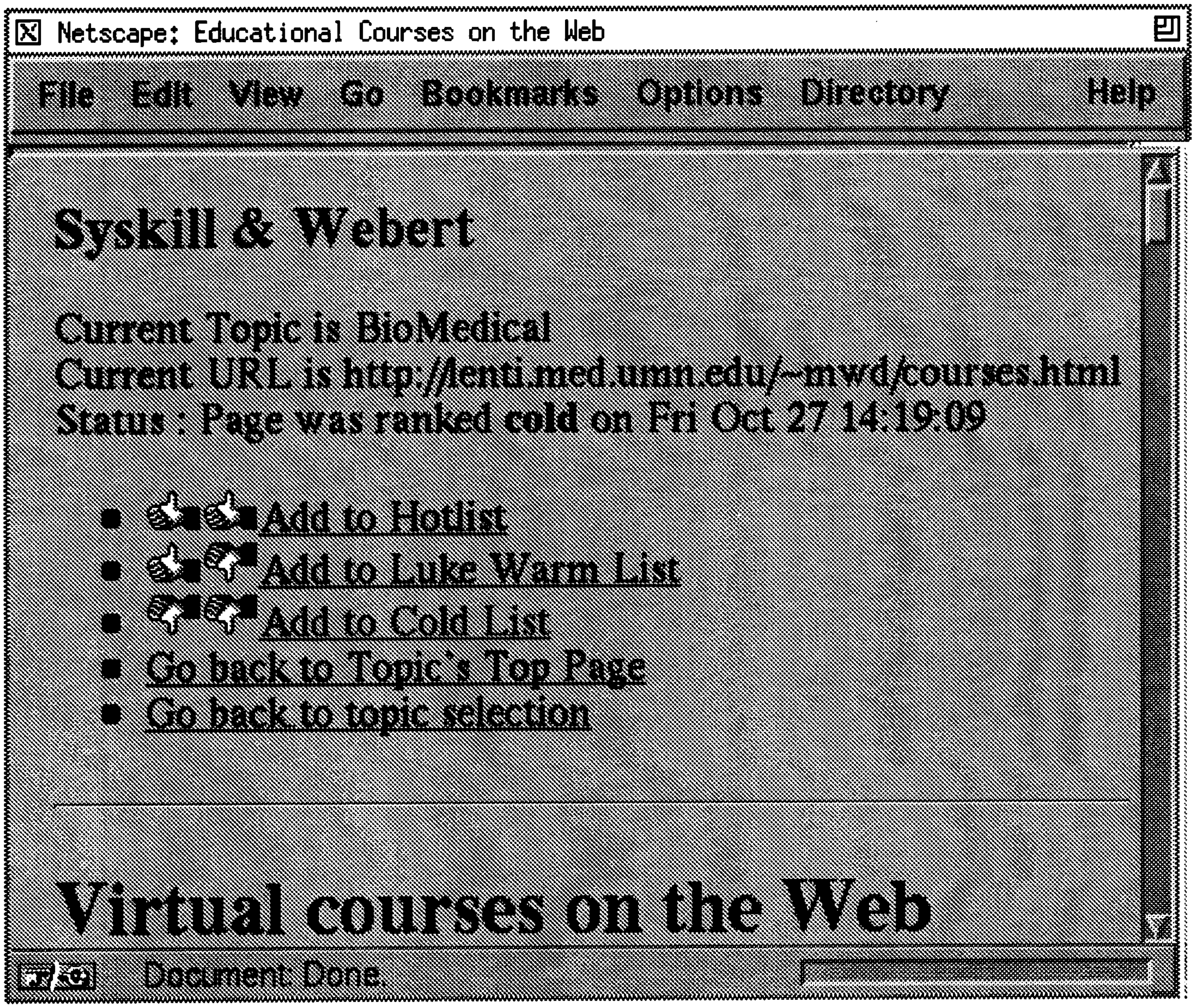
1 Introduction

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. Second, Syskill & Webert can construct a LYCOS (Maudlin & Leavitt, 1994) query and retrieve pages that might match a user's interest, and then rate these pages.

In this paper, we first describe the Syskill & Webert interface 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 classified HTML texts rather than classified attribute-value vectors and 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 few hundred 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 additional functionality is added to each page (see Figure 1). 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). In addition, the user can return to the index page, or switch topics.


Netscape: Educational Courses on the Web

Syskill & Webert

Current Topic is BioMedical
Current URL is http://lenti.med.umn.edu/~mwd/courses.html
Status : Page was ranked cold on Fri Oct 27 14:19:09

- Add to Hotlist
- Add to Luke Warm List
- Add to Cold List
- Go back to Topic's Top Page
- Go back to topic selection

Virtual courses on the Web

Document: Done.

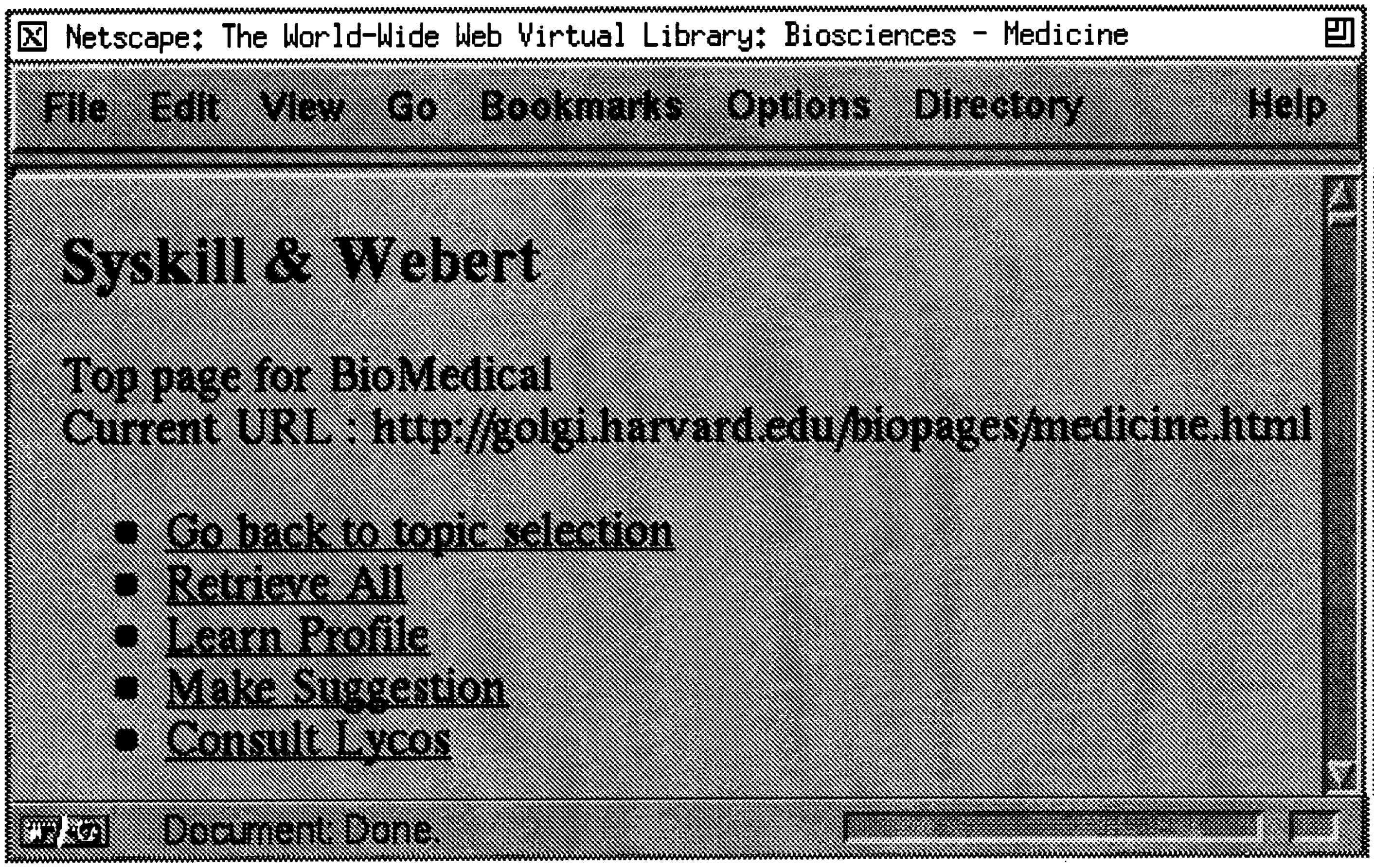
Figure 1. Syskill & Webert interface for rating pages.

When a user rates a page, the HTML source of the page is copied to a local file[1], and a summary of the rating is made as shown below. 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[2] to the index 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.

[1] The entire HTML source is currently saved, to allow for experimentation in feature extraction methods.

[2] In addition, it contains a function to retrieval and locally store the HTML source of all links [2] In addition, it contains a function to retrieval and locally store the HTML source of all links accessible from the current page. Syskil & 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 prefetch all pages.


Netscape: The World-Wide Web Virtual Library: Biosciences - Medicine
File Edit View Go Bookmarks Options Directory Help

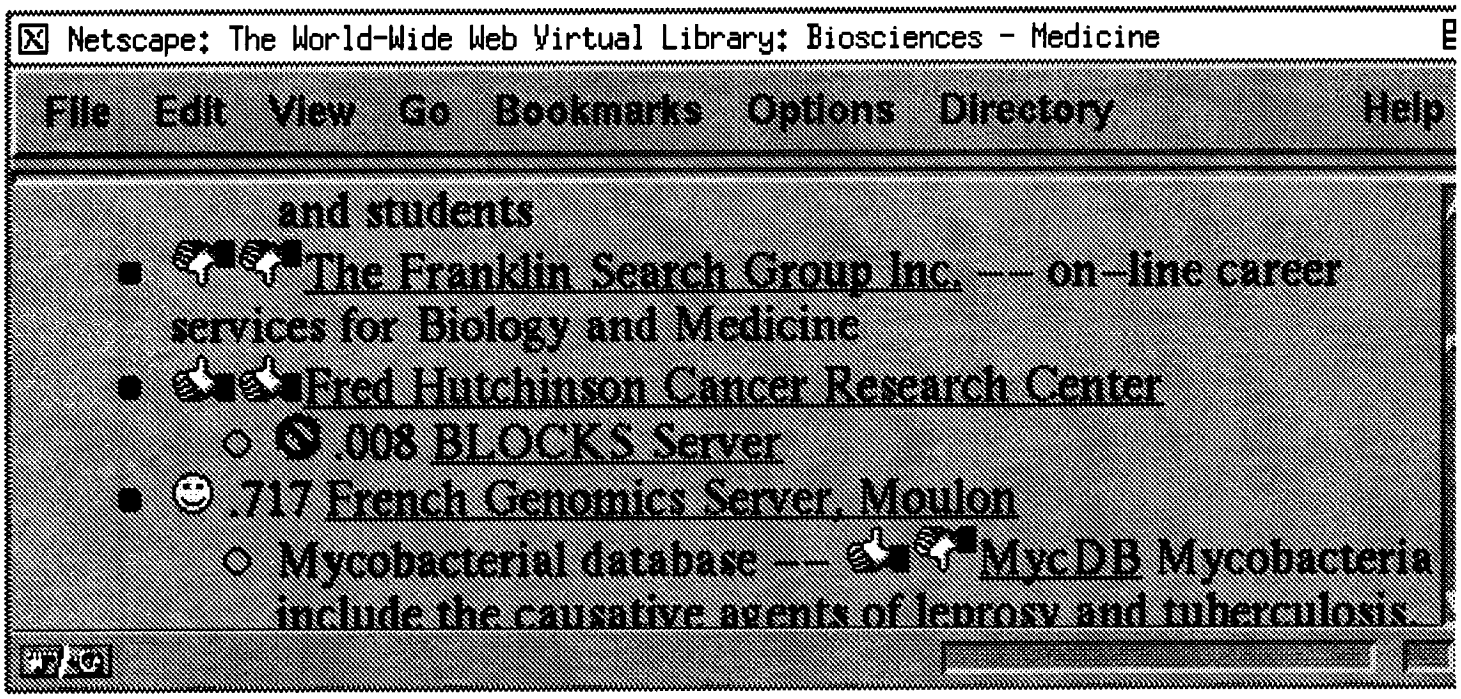
Syskill & Webert

Top page for BioMedical
Current URL : http://golgi.harvard.edu/biopages/medicine.html

- Go back to topic selection
- Retrieve All
- Learn Profile
- Make Suggestion
- Consult Lycos

Figure 2. The Syskill & Webert interface for learning and using a profile.

Once the user profile has been learned, the profile 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, the Syskill & Webert can learn a new profile and make suggestions about pages to visit relatively quickly. Syskill & Webert annotates each link on the index page with an icon indicating the user's rating or its prediction of the user's rating . Two thumbs up indicates the user has visited the page and rated it hot, one thumb up and one down indicates a previous lukewarm rating, and two thumbs down indicates a previous cold rating. A smiley face indicates that the user hasn't visited the page and Syskill & Webert recommends the page to the user. The international symbol for "no" is used to indicate the 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. 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.

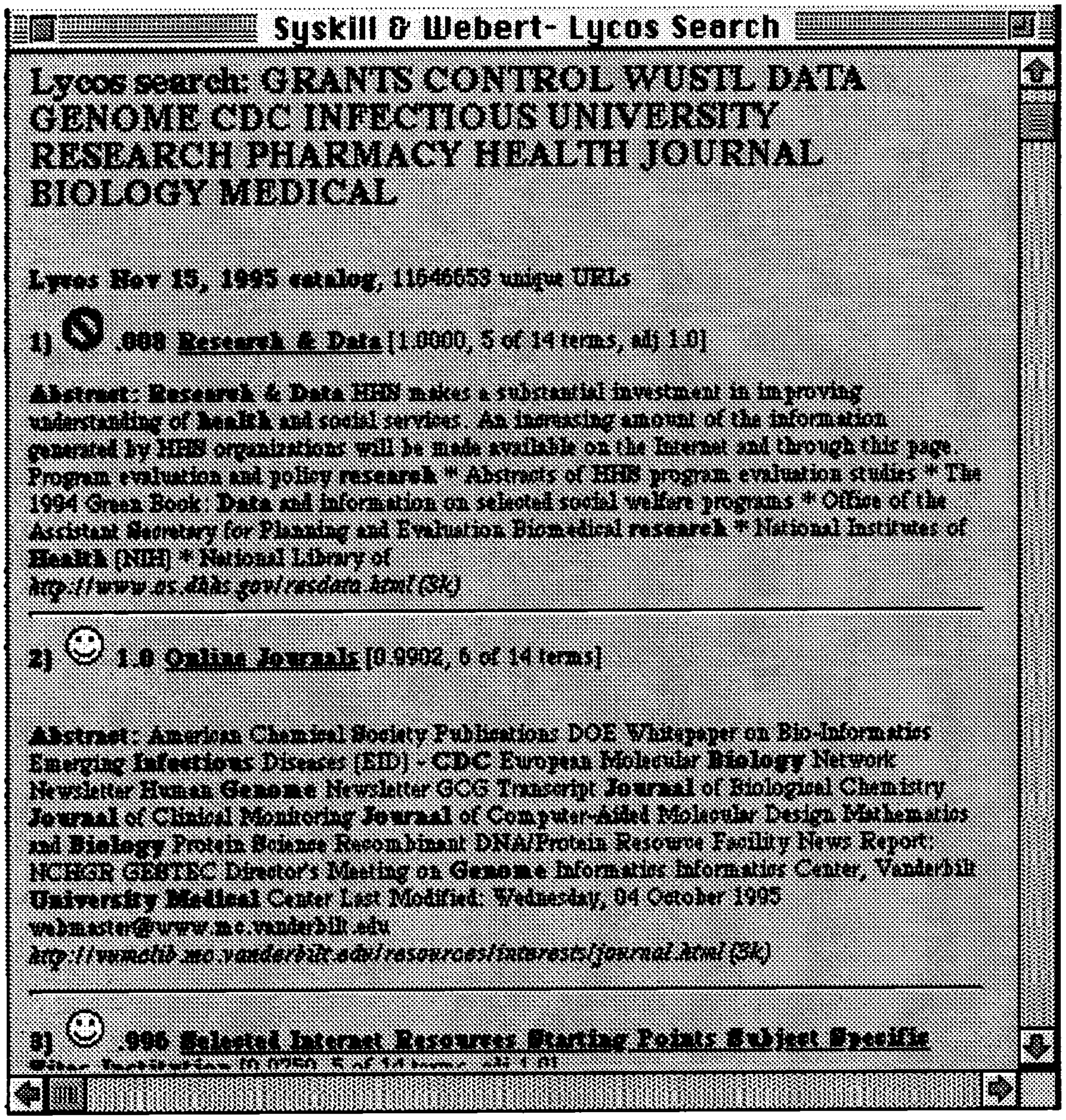

Netscape: The World-Wide Web Virtual Library: Biosciences - Medicine

and students

- The Franklin Search Group Inc. — on-line career services for Biology and Medicine
- Fred Hutchinson Cancer Research Center
  - .008 BLOCKS Server
- .717 French Genomics Server, Moulon
  - Mycobacterial database — MycDB Mycobacteria include the causative agents of leprosy and tuberculosis.

Figure 3. Once a profile is learned, Syskill & Webert indicates previous and predicted rankings with icons.

As described above, Syskill & Webert is limited to making suggestions about which link to follow from a single index 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 to find 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 features. Since LYCOS cannot accept very long queries, we use the 7 most discriminating words and the 6 most commonly occurring words as a query. Experimentally, we have found that longer queries occasionally exhaust the resources of LYCOS. 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," "genome," "data," "institute," "WUSTL" and "pharmacy." The common words are useful for defining a topic. In the example in Figure 3 these are "journal," "biology," "university," "medicine," "health," and "research."


Syskill & Webert- Lycos Search

Lycos search: GRANTS CONTROL WUSTL DATA GENOME CDC INFECTIOUS UNIVERSITY RESEARCH PHARMACY HEALTH JOURNAL BIOLOGY MEDICAL

Lycos Nov 15, 1995 catalog, 11646653 unique URLs

1) .008 Research & Data [1.0000, 5 of 14 terms, adj 1.0]

Abstract: Research & Data HHS makes a substantial investment in improving understanding of health and social services. An increasing amount of the information generated by HHS organizations will be made available on the Internet and through this page. Program evaluation and policy research * Abstracts of HHS program evaluation studies * The 1994 Green Book: Data and information on selected social welfare programs * Office of the Assistant Secretary for Planning and Evaluation Biomedical research * National Institutes of Health (NIH) * National Library of
http://www.os.dhhs.gov/resdata.html (5k)

2) 1.0 Online Journals [0.9902, 6 of 14 terms]

Abstract: American Chemical Society Publications DOE Whitepaper on Bio-Informatics Emerging Infectious Diseases (EID) - CDC European Molecular Biology Network Newsletter Human Genome Newsletter GCG Transcript Journal of Biological Chemistry Journal of Clinical Monitoring Journal of Computer-Aided Molecular Design Mathematics and Biology Protein Science Recombinant DNA/Protein Resource Facility News Report: NCHGR GESTEC Director's Meeting on Genome Informatics Informatics Center, Vanderbilt University Medical Center Last Modified: Wednesday, 04 October 1995
webmaster@www.mc.vanderbilt.edu
http://www.mclib.mc.vanderbilt.edu/resources/interests/journal.html (5k)

3) .996 Selected Internet Resources Starting Points Subject Specific

Figure 4. Syskill & Webert constructs a LYCOS query from a user profile.

A strength of LYCOS is that it 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 have been prefetched). For example, Figure 4 shows part of LYCOS output that has been augmented by Syskill & Webert to contain a recommendation against visiting one page and for visiting others.

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. 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 accomplished by finding the mutual information (e.g., Quinlan, 1984) between the presence or absence of a word and the classification of a page.

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. 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	cause	fabulous	becomes

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 are particularly interested in those algorithms that may be run quickly, so that it would be possible to develop a user profile while the user is browsing. For this reason, we did not investigate neural network algorithms (e.g., Rumelhart, Hinton & Williams, 1986). We concentrated on Bayesian classifiers, a nearest neighbor algorithm and a decision tree learner. In addition, we compare our results to TF-IDF, an approach from information retrieval adapted to perform the task of classification.

3.1 Bayesian classifier

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:

$$P(C_i|A_1=V_{1j} \& \ldots \& A_n=V_{nj})$$

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

$$P(C_i)\prod_k P(A_k=V_{kj}|C_i)$$

Both $P(A_k=V_{kj}|C_i)$ and $P(C_i)$ may be estimated from training data. To determine the most likely class of an example, the probability of each class is computed. An example is assigned to the class with the highest probability.

3.2 Nearest Neighbor

The nearest neighbor algorithm 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.

3.3 Decision Trees

Decision tree learners such as ID3 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.

3.3 TF-IDF

TF-IDF is one of the most successful and well-tested techniques in Information Retrieval (IR). A document is represented as a vector of weighted terms. 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.

In Syskill & Webert we use the average of the TF-IDF vectors of all examples of one class in order to get a prototype-vector for the class (similar to the NewsWeeder program, Lang, 1995). To determine the most likely class of an example we convert it to a TF-IDF vector and then apply the cosine similarity measure to the example vector and each class prototype. An example is assigned to the class that has the smallest angle between the TF-IDF vector of the example and the class prototype.

4 Experimental Evaluation

To determine whether it is possible to learn user preferences accurately, we asked one user interested in music to rate web pages starting at a page that describes independent recording artists. Another user rated pages on the BioSciences starting with a page located at http://golgi.harvard.edu/biopages/all.html and pages found by LYCOS on this same topic. In each case, we used these pages as training and test data for an experimental evaluation. 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 three learning algorithms on each training set: a simple Bayesian classifier, Nearest Neighbor (NN) and ID3 were used. 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. We also tested TF-IDF using a similar scheme, except that TF-IDF operated directly on the HTML pages and did not require converting the pages to feature vectors. 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 24 trials of each algorithm. Figure 5 shows the average accuracy of each algorithm as a function of the number of training examples.

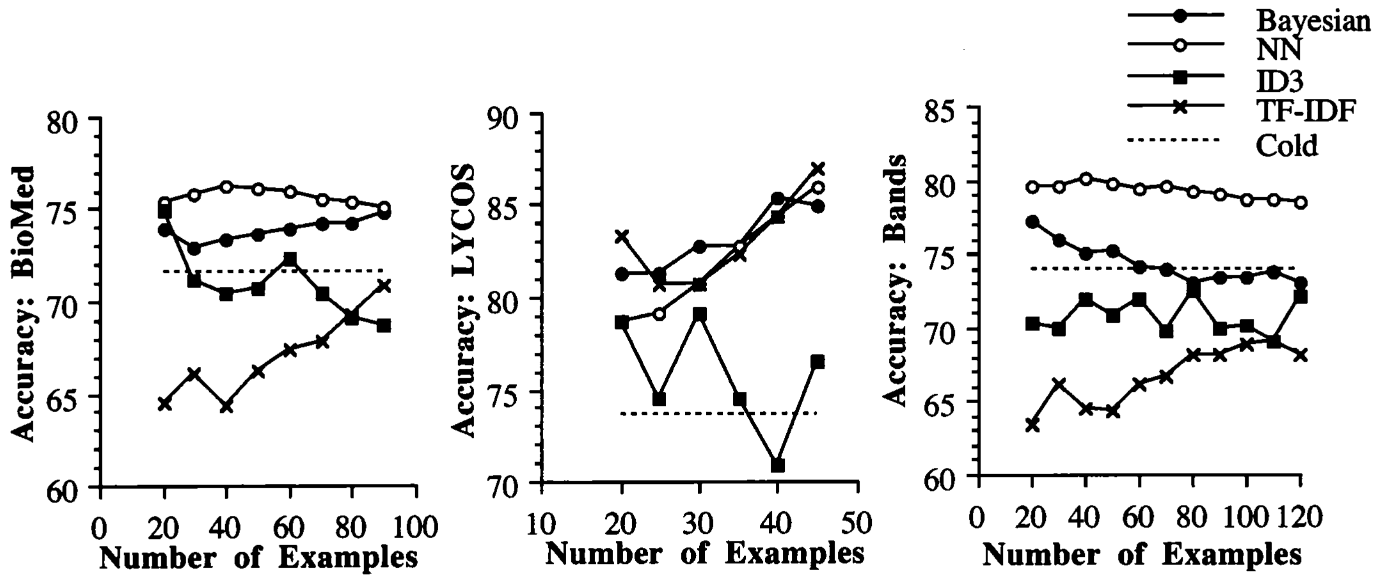


Figure 5. The average accuracy of each learning algorithm at predicting a user's preferences of three different topics.

More experimentation is needed to make conclusions about trends, but 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. In this domain, those approaches that combine pieces of evidence appear to work well, and the nearest neighbor algorithm seems to work very well particularly with large numbers of examples. The TF-IDF algorithm does not appear to have an advantage over the machine learning algorithms. However, we have observed that its predictive accuracy on some of these problems can be improved by restricting the number of words considered as terms. In Figure 5, we use all terms (except terms that appear once and the 40 most frequent terms). In Figure 6 we compare this approach (all) to use the 128 most informative (as we have done with the machine learning algorithms) and the 3000 most frequent (with the exception of the 40 most frequent). Restricting the number of terms to 3000 does tend to increase the accuracy. We have performed similar experiments with the machine learning algorithms and have not surprisingly also found that increasing or decreasing the number of features can have a major effect on the accuracy of individual algorithms.

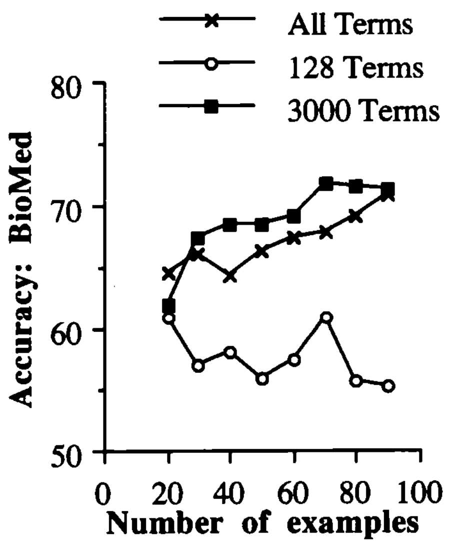


Figure 6. Restricting the number of terms used by TF-IDF.

5 Future Work

In order to evaluate unseen pages, it is necessary to retrieve the entire HTML to convert the page to a feature vector. We are considering an extension that just searches the first *k* (e.g., 2000) characters rather than the entire document. This may reduce the transmission overhead when using the agent interactively. Another alternative we are considering is to just use the summary provided by LYCOS to determine the ranking of any page. This may be particularly useful with "CyberSearch" which is a copy of much of the LYCOS database on CD-ROM.

We currently store local copies of pages that have been rated and pages that are too be rated. This permits easy and repeatable experimentation, but consumes more storage than is necessary. Once we settle on a particular learning algorithm for Syskill & Webert, these storage requirements can be reduced by keeping only a the necessary information from each HTML page.

6. Related work

The methods developed for our learning agent are related to work in information retrieval and relevance feedback (e.g., Salton & Buckey, 1990; Croft & Harper, 1979). However, rather than learning to adapt user queries, we are developing a user profile that may be used for filtering new information as it becomes available.

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 may be used to suggest new information sources related to ones the user is interested in.

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.

Balabanovic, Shoham, and Yun (1995) have developed an agent that searches links for pages that might interest a user, using the TF-IDF algorithm to make a user profile.

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.

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.

Balabanovic, Shoham, and Yun (1995). An adaptive agent for automated web browsing, Journal of Visual Communication and Image Representation 6(4).

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.

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

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

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

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*

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.

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