Machine learning used by Personal WebWatcher

Dunja Mladenić
Department of Intelligent Systems, J.Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Phone: (+38)(61) 1773 272, Fax: (+38)(61) 1251-038
Dunja.Mladenic@ijs.si
http://www-ai.ijs.si/DunjaMladenic

March 15, 1999

Abstract

This paper describes design of personal browsing assistant `Personal WebWatcher` that suggests interesting hyperlinks on the requested Web documents. Machine learning is used to generate a model of user's interests. We consider two approaches that differ in the information included in training examples: (1) include information presented to the user, that is a part of the text from the document that contains a hyperlink and (2) include information that was not presented to the user, that is the content of the document pointed to by a hyperlink. We compare two classification algorithms k-Nearest Neighbor and Naive Bayes. Bag of words document representation is used and features are selected using Information gain. Preliminary experiments show that there is no significant difference between the used classifiers and that using only a small number of features gives almost the same results as using all features. In all experiments the achieved classification accuracy is the same or slightly higher than the default accuracy. Since the default accuracy is higher for approach (1) than for approach (2), the results of approach (1) show higher classification accuracy.

1 Introduction

One of the basic paradigms on the Web is that almost anyone having access to the Internet can create and put documents on the Web. Instead of having a well organized, centralized database of documents, we have a large, distributed set of documents that is constantly changing (many new documents are added, some of the existing documents are reorganized, changed, moved or deleted). Most documents accessed by the research community are free of charge, in many cases offered by other researchers without any guarantee that the same document will be there tomorrow. One of the consequences of that 'open-minded structure' of the Web is that finding interesting documents in a huge space of Web documents is often very hard. On the other hand, Web offers possibility for everyone to contribute her/his ideas and creativity. We can see the spontaneous, rapid growth of the Web as a part of overall trends for decentralization and synergy between people that brings also some risk but offers many opportunities.

We can all put our documents on the Web, but the most important part is in getting other Web users know that our documents are available. It is quite common that people exchange the addresses of interesting Web documents (Uniform Resource Locator) that contain research or business information as well as some nice pictures or good Internet shopping spots. There is also growing number of different search engines and other programs usually called intelligent Web agents that aim to help user finding the required information.

Recent work that arises at the intersection of Information Retrieval and Machine Learning offers some novel solutions to this problem, as well as work in Intelligent Agents. For example, Armstrong et al. [1] developed WebWatcher, a system that assists user in locating information on the World Wide Web taking keywords from the user, suggesting hyperlinks and receiving evaluation. Balabanović et al. [2] developed "a system which learns to browse the Internet on behalf of a user". It searches the World Wide Web taking bounded amount of time, selects the best pages and receives an evaluation from the user. The evaluation is used to update the search and selection heuristics. Pazzani et al. [17] collect ratings of the explored Web pages from the user and learn a user profile from them. Pages are separated according to their topic and a separate profile is learned

for each topic. Mitchell et al. [15] proposed a system connected to the user's electronic calendar, that generates sets of rules capturing the user's scheduling preferences and some other information about individual attendees of meetings. It uses these rules to provide advice to the user for new, unscheduled meetings. Lang [13] developed a system for electronic news filtering that uses text-learning to generate models of user interests. Krulwich and Burkey [12] proposed "The ContactFinder agent" that reads and responds to bulletin board messages; assists users by referring them to other people who can help them and; categorizes messages and extracts their topic areas. Maes [14] described "interface agents" that learn from the user as well as from other agents. As examples of such agents they developed agents for: electronic mail handling, meeting scheduling, electronic news filtering and entertainment recommendation. Some of these agents use the context of documents and adopt Information Retrieval approaches (eg. news filtering). Others rely on correlation between different users — performing "social learning" (eg. entertainment recommendation). Holte and Drummond [9], Drummond et al. [5] designed a system that assists browsing of software libraries, taking keywords from the user and using a rule-based system with forward chaining inference, assuming that the library consists of one type of item and the user goal is a single item. Hammond et al. [7, 4] developed a system that uses a "natural language question-based interface to access distributed text information sources" and helps the user to find answers to her/his question in a databases such as FAQ files.

Section 2 gives problem description. The structure of the system is described in Section 3. Section 4 describes the usage of machine learning to generate the model of user's interest. Results of preliminary experiments are given in Section 5.

2 Problem description

Let's now consider one of the typical patterns of Web browsing. Imagine a user sitting behind a Web browser and requesting documents, most of the time by clicking on the hyperlinks on already requested documents that are presented to the user. Predicting and highlighting the clicked hyperlinks is one of the ways to help users finding their way on the Web. In the machine learning setting we used, all the hyperlinks that were shown to the user are considered training examples with a binary-valued class (clicked or un-clicked). Of course there are many other possible settings. For example, asking user to rate each document on a given scale (eg. 1—3) and using the given rating as a class value showing how interesting is document for a user.

For each user a profile is induced independently of other users. This profile can be further used to compare different users and share knowledge between them. For example, instead of having a friend with similar interest sending me the address of 'cool' Web document, the system can automatically suggest me documents that were found interesting by other users that have interests similar to mine. In order to secure the privacy, only knowledge and not the user identity can be exchanged or even this cooperation could apply only for users that explicitly agreed to take part in knowledge sharing. On the other hand, some users might be interested in 'making friends' with similar users and join the list of users whose identity (eg. e-mail) is reviled to similar users from the list.

This sharing of knowledge is related to collaborative approach to intelligent agents design and methods used in multiagent systems. One simple way based on collaborative approach is to give a same set of documents to each user and ask for rating. These ratings are then used to group users accord-

ing to their interests. After that, this grouping can be changing with new documents that are requested by users and new users starting to used the system. In this way instead of system suggesting only documents that fit the user profile, documents visited by other users with similar interests would be also suggested. System Fab by Balabanović and Shoham [3] is based on more advanced combination of collaborative approach and content-based user profiling. Instead of basing similarity on collected ratings from all users for the same set of documents, similarity between users is defined as similarity between their profiles. Each user profile is generated using content (text) of documents and associated rating given by the user.

The other way of cooperation between different users using the same system for user customized Web browsing is on model induction level. Namely, even though each user has separate user profile, all profiles are generated from Web documents. If we could infer from user profiles some knowledge that is independent of a specific set of documents, that knowledge could be shared between users. Our opinion is that such kind of knowledge should be some higher level knowledge. For example, if we have given some background knowledge, find which part of a given background knowledge is frequently used in models for different users (which higher-level attributes are useful). That would be especially valuable for new users, where only small set of documents is available for model induction. Sharing knowledge between different users of the system is out of the scope of this paper. We feel that the first step is to have a good models for individual users.

3 Structure of the system

`Personal WebWatcher` is a system that observes users of the Web and suggests pages they might be interested in. It learns user interests from the pages requested by the user. The learned model is used to suggest hyperlinks on new HTML-pages requested by and presented to the user via Web browser that enables connection to "proxy".

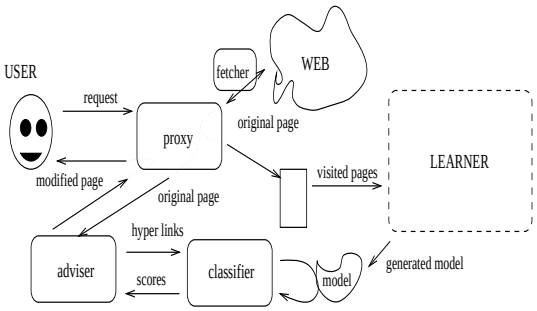

Figure 1: Structure of browsing assistant Personal WebWatcher.

Personal WebWatcher consists of two main parts: a **proxy server** that interacts with the user via Web browser and a **learner** that provides the user-model to the server (see Figure 1). Communication between them is via disk; the **proxy** saves addresses of visited documents (URLs) and the **learner** uses them to generate model of user interests. The first version of the system was implemented in approximately 2500 lines of Perl code and 1500 lines of C^{++} code. To make the system more time and space efficient we implemented a new version in C^{++} that is integrated with the machine learning system Learning Machine [6].

The **proxy server** consists of three main parts: **proxy** (additionally calls external **fetcher** code to fetch the page), **adviser** and **classifier**. **Proxy** waits in an infinite loop for a page request from browser. On request, it fetches the requested document and; if it is an HMTL-document adds advice and; forwards the document to the user. To add advice **proxy** forwards the page to **adviser**, that extracts hyperlinks from document and

calls external code for **classification** that uses generated user-model. A limited number of hyperlinks that are scored above some threshold are recommended to the user, indicating their scores using graphical symbols placed around each advised hyperlink. For example, in Figure 2 three hyperlinks are suggested by PWW: "Machine Learning Information Services" and two project members (Dayne Freitag, Thorsten Joachims). There is a banner at the top of the page showing that PWW is "watching over the user's shoulder".

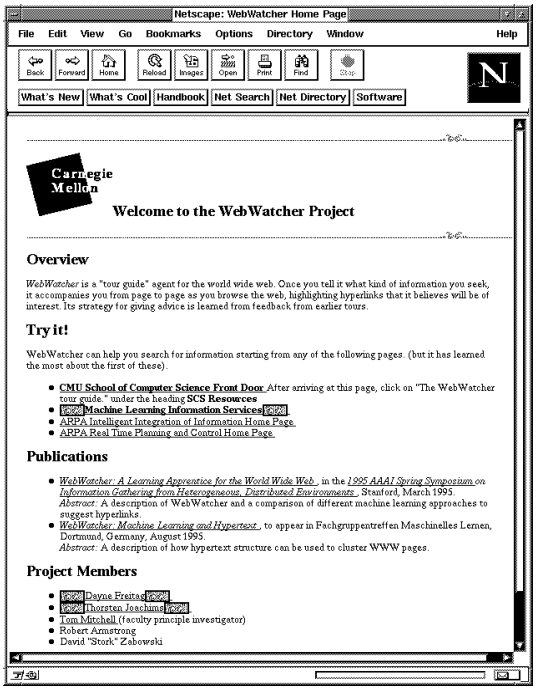

Figure 2: Example of HTML-page presented to the user by Personal WebWatcher.

4 Model of user interests

Helping the user browsing the Web is performed here by highlighting interesting hyperlinks on the requested Web documents. We assume that the interesting hyperlinks are the hyperlinks that are highly probable to be clicked by the user. Our problem is defined as predicting clicked hyperlinks from the set of Web documents visited by the user. All hyperlinks that are present on the visited documents are used for constructing machine learning examples. Since the prediction should be performed while the user is waiting for a document, we are actually predicting interestingness of a document based on the hyperlink pointing to it, and not the document itself. We build descriptions of hyperlinks, and treat these as shortened documents. Hyperlinks whose documents were visited by the user are considered to be positive examples, and all the other to be negative examples of the user interests. The idea is that different hyperlinks were presented to the user and the user chose to visit some of them that meet her/his interests. This simplification is introduced to minimize users involvement in the learning process and enable learning without asking user for page rating.

We consider two approaches that differ in the information included in training examples: (1) include information presented to the user, that is a part of the text from the document that contains a hyperlink and (2) include information that was not presented to the user, that is the content of the document pointed to by a hyperlink.

(1) Generating examples from the information presented to the user on the requested document means using machine learning to model the function $User_{HL} : HyperLink \rightarrow \{pos, neg\}$, where $HyperLink$ (HL) is defined to include the actual hyperlink from HTML-document and some of the its context (titles, neighboring words).

(2) Retrieving documents pointed to by hyperlinks on a requested document means that the model will be induced from documents $User_{DOC} : Document \rightarrow \{pos, neg\}$.

The reason we can afford thinking about retrieving documents is that the classification model is induced "off-line", meaning that the induction is not so time critical as classification. The time critical part is classification, since usually the user is sitting behind the Web browser waiting for the requested document. Before showing the document to the user, we are modifying it by highlighting some of the included hyperlinks according to the results of the classification process in which the document is used.

In both cases, classification is performed on *HyperLinks* that in the second solution can be viewed as a kind of HTML-documents. In (1) the model induced from *HyperLinks* is used for classification while in (2) the model induced from the *Documents* pointed to by hyperlinks is used. In this thesis we use the first approach that requires less time to collect data and results in domain containing less features (smaller vocabulary is used).

5 Experimental results

We measure classification accuracy, defined as a percent of correctly classified examples and calculated over all classes. Our testing is performed with 10-fold cross-validation using the same example splits for testing different approaches.

We observe how the number of selected features (vector size) influences model quality for two learning algorithms: Naive Bayesian classifier on frequency vector as used by Joachims [11] and k-Nearest Neighbor approach on frequency vectors using Euclidean distance between examples and summing class probabilities predicted by k-neighbors as used in [19]. We tested both algorithms on data for each approach $User_{DOC}$ and $User_{HL}$ (see Section 4). Documents are currently represented using the bag-of-words representation and feature selection is performed using information gain [18]. Our experiments are performed on data collected for four users participating in the HomeNet project [8] with the data characteristics given in Table 1.

User id., data source	prob. of interestingness	# of examples	data entropy
usr150101			
Doc	0.094	1 333	0.449
HL	0.104	2 528	0.480
usr150202			
Doc	0.107	3 415	0.492
HL	0.053	4 798	0.301
usr150211			
Doc	0.089	2 038	0.436
HL	0.044	2 221	0.259
usr150502			
Doc	0.100	1 272	0.468
HL	0.100	2 498	0.468

Table 1: Data characteristics for document (Doc) and hyperlink (HL) data for each of the four HomeNet users.

In all experiments k-Nearest Neighbor achieved slightly higher classification accuracy than the Naive Bayesian classifier (see Figures 3, 4, 5, 6), but the difference is significant only in one out of eight experiments (see Figure 5). Using more than approximately 50 features doesn't appear to help for classification accuracy, but it also doesn't hurt. High classification accuracy achieved for all four users by k-Nearest Neighbor algorithm is not much higher than the default accuracy (achieved if negative class is predicted for all testing examples). This indicates that some other model quality estimate should be used, for instance, Precision-Recall combination as used in [16].

In order to draw some conclusion about vector size and quality of algorithms, we need to perform more experiments on different users. These experiments show that increasing vector size probably isn't as beneficial as one could expect. There is no evidence that

algorithms differ substantially in classification accuracy, although k-Nearest Neighbor seems to be more promising. If further experiments confirm the hypothesis that long vectors are not advisable, a closer look at the short vectors should give an idea about the appropriate number of features.

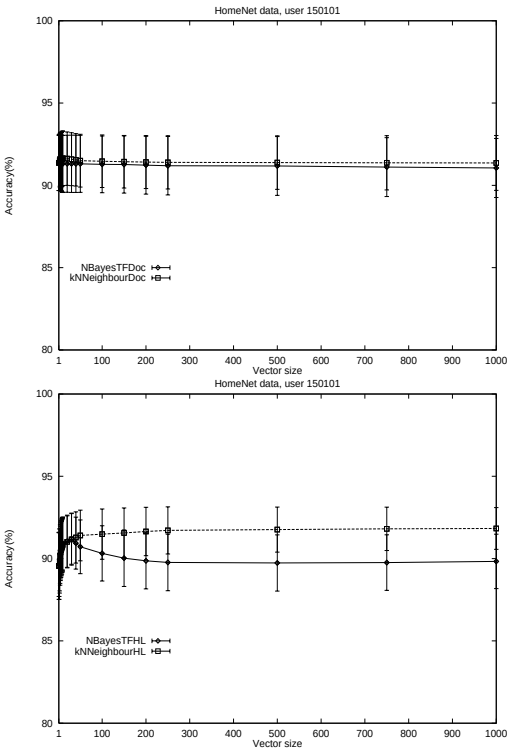

Figure 3: Influence of the number of selected features to classification accuracy of model based on documents (upper) and hyperlinks (lower) for the HomeNet project user 150101. We show mean value and standard deviation of 10-fold cross validation.

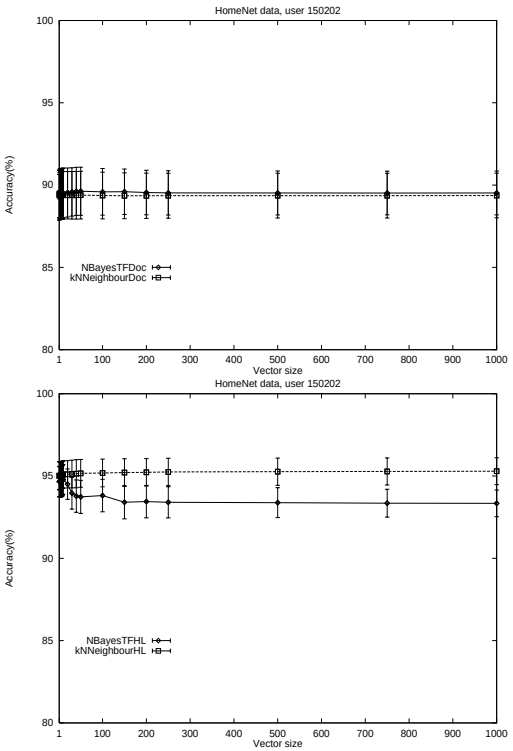

Figure 4: Results for the HomeNet project user 150202.

6 Acknowledgements

This work was financially supported by the Slovenian Ministry for Science and Technology. Part of this work was performed during the authors stay at Carnegie Mellon University. Authors are grateful to Tom Mitchell and his machine learning group at Carnegie Mellon University for generous support, suggestions and fruitful discussions.

References

[1] Armstrong, R., Freitag, D., Joachims, T., Mitchell, T., WebWatcher: A Learning Apprentice for the World Wide Web, *AAAI 1995 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, Stanford, March 1995.

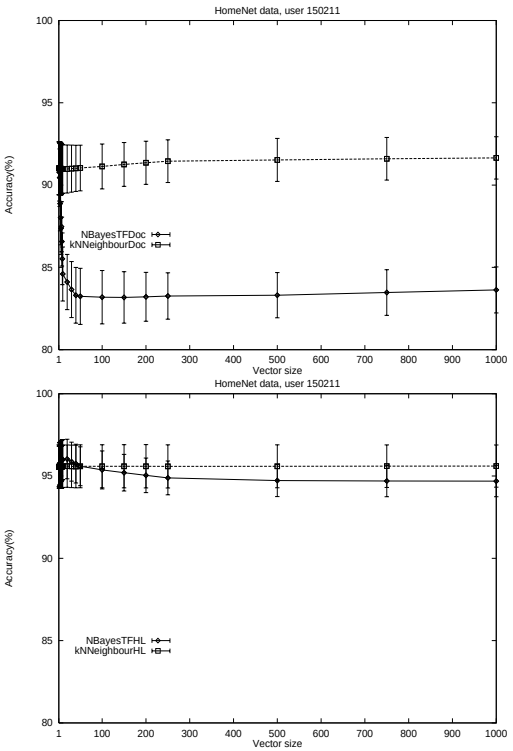

Figure 5: Results for the HomeNet project user 150211.

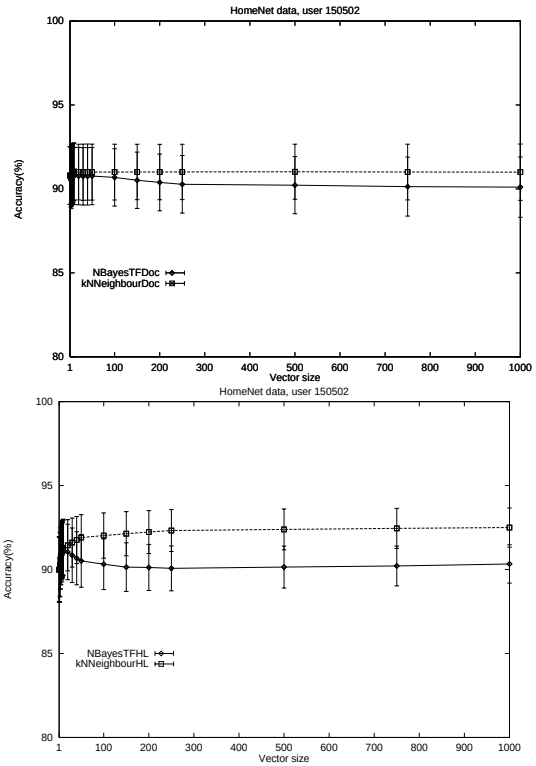

Figure 6: Results for the HomeNet project user 150502.

[2] Balabanović, M., Shoham, Y., Learning Information Retrieval Agents: Experiments with Automated Web Browsing, *AAAI 1995 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, Stanford, March 1995.

[3] Balabanović, M., Shoham, Y., Fab: Content-Based, Collaborative Recommendation, *Communications of the ACM*, pp. 66-70, Vol. 40, no. 3, March 1997.

[4] Burke, R., Hammond, K., Kozlovsky, J., Knowledge-based Information Retrieval for Semi-Structured Text, *Working Notes from AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, pp. 19–24, American Association for Artificial Intelligence, 1995.

[5] Drummond,C., Ionescu, D., Holte, R., A Learning Agent that Assists the Browsing of Software Libraries, *Technical Report TR-95-12*, Computer Science Dept., University of Ottawa, 1995.

[6] Grobelnik, M., Mladenić, D., 1998. Learning Machine: design and implementation, *Technical Report IJS-DP-7824*, Department for Intelligent Systems, J.Stefan Institute.

[7] Hammond, K., Burke, R., Schmitt, K., A Case-Based Approach to Knowledge Navigation, *AAAI Workshop on Indexing and Reuse in Multimedia Systems*, pp. 46–57, American Association for Artificial Intelligence, 1994.

[8] Kraut, R., Scherlis, W., Mukhopadhyay, T., Manning, J. & Kiesler, S. (1996). The HomeNet Field Trial of Residential Internet Services. *Communications of the ACM 12*, (Vol. 39), 55-63. http://homenet.andrew.cmu.edu/Progress/

[9] Holte, R.C., Drummond, C., A Learning Apprentice For Browsing, *AAAI Spring Symposium on Software Agents*, 1994.

[10] Joachims, T., Mitchell, T., Freitag, D., Armstrong, R., WebWatcher: Machine Learning and Hypertext, *Fachgruppentreffen Maschinelles Lernen*, Dortmund, August 1995.

[11] Joachims, T., A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization, *Technical report, CMU-CS-96-118*, School of Computer Science, Carnegie Mellon University, March 1996.

[12] Krulwich, B., Burkey, C., The ContactFinder agent: Answering buletin board questions with referrals, *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI 96*, pp.10—15, 1996.

[13] Lang, K., News Weeder: Learning to Filter Netnews, *Proc. of the 12th International Conference on Machine Learning ICML95*, 1995.

[14] Maes, P., Agents that Reduce Work and Information Overload, *Communications of the ACM* Vol. 37, No. 7, pp.30–40, July 1994.

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

[16] Mladenic, D. (1998). Turning Yahoo into an Automatic Web-Page Classifier. *Proceedings of the 13th European Conference on Aritficial Intelligence ECAI'98* (pp. 473-474).

[17] Pazzani, M., Muramatsu, J., Billsus, D., Syskill & Webert: Identifying interesting web sites, *AAAI Spring Symposium on Machine Learning in Information Access*, Stanford, March 1996 and *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI 96*, pp.54—61, 1996.

[18] Quinlan, J.R., Constructing Decision Tree in *C4.5: Programs for Machine Learning*, pp.17–26, Morgan Kaufman Publishers, 1993.

[19] Yang, Y., An evaluation of statistical approaches to text categorization, *Journal of Information Retrieval*, 1998.