Machine learning for better Web browsing

Dunja Mladenić
Dpt. of Intelligent Systems, J.Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Dunja.Mladenic@ijs.si

Abstract

This paper describes a way to help the user browsing the Web using Machine Learning techniques. Two machine learning algorithms, Naive Bayes and k-Nearest Neighbor, are compared on two approaches that differ in the information included in the training examples. Experiments on real-world data confirm that the proposed approaches are promising.

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, resulting in a large, distributed set of documents that is frequently changing. 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. The spontaneous, rapid growth of the Web can be seen as a part of overall trends for decentralization and synergy between people that brings some risk but offers many opportunities.

2 Machine learning setting

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 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 order to help users browsing the Web, a profile is induced for each user independently of other users. This profile can be further used to compare different users and to 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 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 [4] and methods used in multiagent systems.

A way of cooperation between different users using the same system for user customized Web browsing is on the model induction level. Namely, even though each user has a separate user profile, they have a similar form. If we could infer from the user profiles some higher level knowledge that is independent of a specific set of documents, that knowledge could be shared between users. For instance, if we have given some background knowledge, find which part of a given background knowledge is frequently used in different models (which higher-level attributes are useful). That would be especially valuable for new users, where only a small set of documents is available for the 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. The next Section is dedicated to definition of framework for the usage of machine learning techniques on our problem domain. Some problem characteristics and possible approaches are described here, while a detailed description of the used machine learning techniques can be found in [5].

3 On-line learning set of models

If we concentrate on a single user browsing the Web, there is a flow of examples (documents presented to the user) whose nature and frequency depends on the user's selection and intensity of the Web browsing. Figure 1 illustrates the flow of examples in our problem (text documents are arriving over time). During the collection of examples, at some point we decide to use the collected set of examples to generate a model that is used for classification. For instance, at the beginning of observation, the set of examples E_1 is used to generate the model M_1 (see Figure 1). After some time, we collect some new examples (the set of examples E_2) and want to generate a new model M_2. There are several possibilities for the data selection for the generation of the new model M_2: from the latest examples only (E_2), from all examples ($E_1 \cup E_2$), by modification of the old model M_1 using the latest examples E_2. In all the three cases we assume that only the last generated model (in this case M_2) is used for classification. There is also a fourth possibility, namely, to generate each model from a separate set of examples and use some combination of the models for classification (M_1 generated from E_1, M_2 generated from E_2, ...). This last approach was proposed by Breiman [1] in machine learning on large datasets and on-line learning (arcing). In arcing, the subsets of incoming examples are taken and used to generate a new model. Subsets are formed to include examples that are incorrectly classified by old (aggregated model) and probabilistically include the examples that are classified correctly. All generated models are stored and used for classification combined by voting.

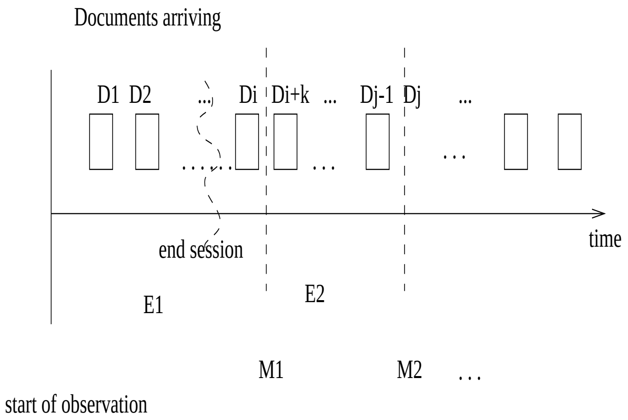

Figure 1: Illustration of the data collection process while the user is browsing the Web. Text documents D_1, D_2, ... are arriving over time as request by the user.

There are different questions that should be addressed, for instance, what information to include in the training examples, how to represent documents, how to reduce the high number of features, how to generate new background knowledge, how to include background knowledge in the model generation. Some of them are addressed in [5].

4 Learning a single model

In order to get a set of models, we have to decide what is the target function, we want the model to approximate. We have identified two interesting problems (target functions): predicting clicked hyperlinks and predicting interesting hyperlinks. In this paper the first problem is considered that is predicting the hyperlinks that the user is going to click on. For the simplicity, we will refer to this hyperlinks as **clicked hyperlinks**. The rest of this section briefly describes the both problems.

4.1 Predicting clicked hyperlink

For the problem of predicting clicked hyperlink, the simplest solution would be to have a binary-valued class. In this way some existing text-learning methods can be used. In this case all the hyperlinks clicked by the user are considered positive examples and all other hyperlinks that were shown to the user (but remain un-clicked) are negative examples. This partially already captures user interests. However, in many cases user didn't click on a hyperlink just because the lack of time and also probably didn't find interesting all the visited documents.

This approach could be adapted for the problem of suggesting interesting hyperlinks, if we are prepared to accept that all the documents visited by the user were interesting to the user and that un-clicked (or random) documents are uninteresting. Of course, the simple solution to that would be to ask the user for document rating but we do not want to put additional work to the user.

4.2 Predicting interesting hyperlink

Predicting interesting hyperlinks can be performed by predicting clicked hyperlinks as described in Section 4.1. Noise in the class value can be reduced by learning from the clicked hyperlinks (positive examples) only. This would in our case require solving at least three additional problems: (1) finding an appropriate learning algorithm, (2) finding an appropriate feature selection method, and (3) proposing a suitable result evaluation. (1) There is not much work on learning from positive examples only. The best known approach is the association rules induction that is widely used on large databases. On the other hand, some of the existing learning algorithms such as k-Nearest Neighbors can be also used, if a threshold is introduced defining the maximum distance from positive examples in order for a new example to be classified as positive. (2) Most feature scoring methods used for feature selection in learning over text documents assume that there are at least two different class values. One very simple method that does not have this requirement is a word frequency. (3) In order to evaluate the result of learning (the induced knowledge model), we need some method for the model evaluation. Evaluation methods that are commonly used in information retrieval (eg., precision, recall) or in machine learning (classification accuracy, information score) are all based on the assumption that there are at least two different class values. We are currently not aware of some general method designed for the evaluation of models in domains with only positive examples. The possible way to evaluate a model that returns probability for each hyperlink would be to maximize the average predicted probability of clicked hyperlinks. In general, the method would try to maximize the average predicted probability of positive examples.

5 Experiments on predicting clicked hyperlinks

5.1 Model of user interests

Our experiments are performed using our personal browsing assistant named `Personal WebWatcher`. 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".

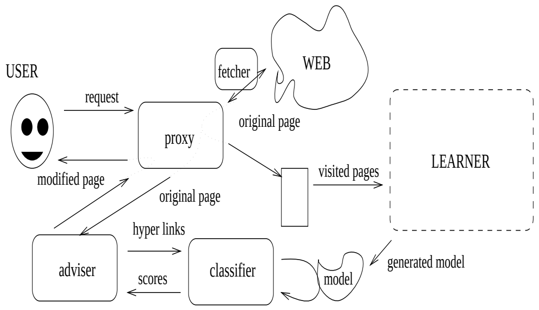

Figure 2: Structure of browsing assistant Personal WebWatcher.

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.

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 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 paper we use the first approach that requires less time to collect data and results in domain containing less features (smaller vocabulary is used).

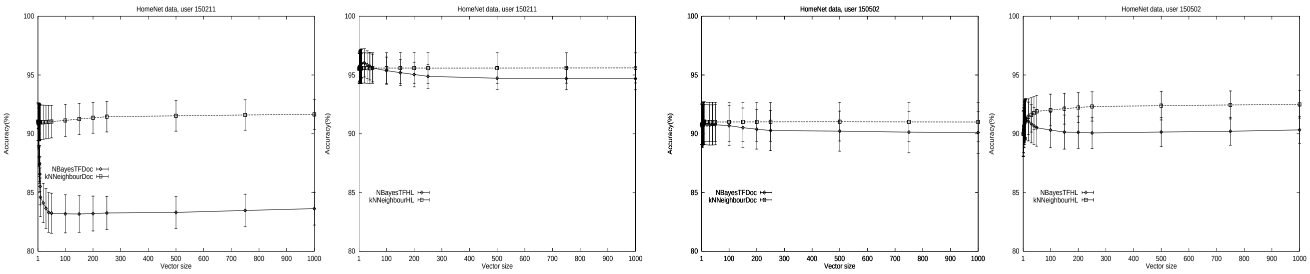

Figure 3: Influence of the number of selected features to classification accuracy of model based on documents (DOC) and hyperlinks (HL) for the HomeNet project user 150211 (left two) and user 150502 (right two). We show mean value and standard deviation of 10-fold cross validation.

5.2 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 and the k-Nearest Neighbor algorithm. We use the Naive Bayesian classifier on text as proposed in [3] where the feature independence assumption is used for different words as well as for different occurrences of the same word. In this approach documents are represented with bag-of-words (unigram model of text is used) where a feature is defined for each word position in the document having word at that position as the feature value. Recall that in the bag-of-words representation only words, and not their positions, are considered. Thus a document can be seen as a set of words, where each word is assigned a frequency - the number of times a word occurred in the document. k-Nearest Neighbor was applied on frequency vectors using Euclidean distance between examples and summing class probabilities predicted by k-neighbors as used in [8]. We tested both algorithms on data for each approach $User_{DOC}$ and $User_{HL}$ (see Section 5.1). Documents are currently represented using the bag-of-words representation and feature selection is performed using information gain [7]. Our experiments are performed on data collected for four users participating in the HomeNet project [2] with the data characteristics given in Table 1.

In all experiments k-Nearest Neighbor achieved slightly higher classification accuracy than the Naive Bayesian classifier (see Figure 3), but the difference is significant only in one out of the eight experiments (the first graph). 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 [5].

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.

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 in. Tom Mitchell's group.

References

[1] Breiman, L. (1996). Pasting bites together for prediction in large data sets and on-line. Berkeley University, November.

[2] 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), pp. 55-63.

[3] Mitchell, T.M. (1997). Machine Learning. The McGraw-Hill Companies, Inc..

[4] Mladenić, D. (1999). Text-learning and related intelligent agents. IEEE EXPERT, Special Issue on Applications of Intelligent Information Retrieval, May-June 1999.

[5] Mladenić, D. & Grobelnik, M. (1999). Feature selection for unbalanced class distribution and Naive Bayes, *Proceedings of the 16th International Conference on Machine Learning ICML-99*, Morgan Kaufmann Publishers, pp. 258-267.

[6] Pazzani, M., & Billsus, D. (1997). Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning, 27*, 313-331. Kluwer Academic Publishers.

[7] Quinlan, J.R. (1993). Constructing Decision Tree. In *C4.5: Programs for Machine Learning.* Morgan Kaufman Publishers. pp. 17-26.

[8] Yang, Y. (1994). Expert Network: Effective and Efficient Learning form Human Decisions in Text Categorization and Retrieval. *Proc. of the 7th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, ACM Press, New York, pp. 13-22.