# WebClass: A Web Mining Tool

**GIULIO DE LUISE**
**DONATO MALERBA**
Dipartimento di Informatica
Università degli Studi di Bari
via Orabona, 4
70126 Bari
{malerba@lacam.uniba.it}

**GABRIELLA CONVERTINO**
**LUIGI DI PACE**
**PIETRO LEO**
**ANGELO MAFFIONE**
Java Technology Center - IBM SEMEA Sud
via Tridente, 42/14
70125 Bari
{crbari@it.ibm.com}

## 1. Introduction

*Web mining* is the application of specific algorithms for extracting patterns from *resources* (documents and services) distributed in the Web. The goal of Web mining seems similar to that of data mining, here intended as the application of specific algorithms for extracting patterns (models) from databases. Indeed, some attempts have been made to transform the Web into a multiple layered database where several data mining techniques can be used [14]. Nevertheless, this approach raises two problems: The manual construction of a data model for the layer-0 of the database and the updating of this level due to the dynamic aspect of the Web. It has been argued that information on the Web is sufficiently structured to facilitate effective Web mining [4]. Under this assumption it is possible to organize Web mining into three distinct subtasks:

1. *Resource discovery*, that is locating documents and services on the Web.
2. *Information extraction*, that is extracting information from the discovered resources. For textual information it is often necessary to take into account the structure of the texts and not only the words as usually done in information retrieval.
3. *Generalization*, that is learning about the Web itself.

To make use of information distributed in the Web, users need ways to locate information of interest. Several tools have been created and have gained wide popular acceptance in the Internet [12]. They are based either on the *organizing/browsing* paradigm (e.g., WAIS and Gopher), or on the *searching* paradigm (e.g. Archie and AltaVista) or on a hybrid approach (e.g., Yahoo and Excite) [3]. Current resource discovery tools that adopts the searching paradigm are based on Web robots that scan millions of Web documents and store an index of the document words. Often indices built by such robots include irrelevant or outdated information. More powerful resource discovery systems should make use of automatic text categorization technology to classify those Web pages whose relevant information is textual.

In this work we are interested to the problem of locating textual information on the Web by means of text categorization techniques [1]. In order to derive the classification rules that can be used for general document categorization we intend to exploit machine learning methods. The main challenge set by typical text categorization problems to machine learning methods resides in the very large number of categories. In fact, very few studies have tackled this issue, most of which concern the problem of recognizing thousands of chinese characters [13]. In text categorization the problem is even more complicated since categories may overlap. Fortunately, a Web user may be interested to a limited number of categories, for instance those identified by the names of the folders used to organize her/his browser bookmarks. In this context, it is possible to apply several machine learning techniques in order to learn users' interests, more precisely a *profile* of user topics. The user profile can then be used either to suggest which links of the current Web page a user would be interested in exploring or to autonomously browse the Web in order to find pages that would interest a user.

Several systems that learn Internet user interests have been presented in the literature. *NewsWeeder* [5] is a netnews-filtering system that addresses this problem by letting the user rate her/his interest level for each article being read, and then by learning a user profile based on these ratings. *WebWatcher* [2] interactively helps users locating desired information while browsing on the World Wide Web. This is done by taking keywords from the user, suggesting hyperlinks and receiving evaluation. The probability that a user will select a certain link given the current page and her/his interests is estimated using either a *k*-nearest neighbor approach or reinforcement learning [7]. *Letizia* [6] is another Web browser assistant that infers user likes and dislikes from actions and then recommends pages for a user to visit starting from the current web page (resource-limited breadth-first search). *Syskill & Webert* [9] collects ratings (cold or hot) of the explored Web pages to learn a profile of user topics (Biomedical, Movies and Protein). The authors compare six different learning algorithms and argues that the naive Bayesian classifier offers several advantages over the others. Finally, Queck [10] investigates several learning methods that take into account structural information in Web pages. Again, the goal is that of classifying web pages according to some predetermined ontology (categories).

In this paper we present the design of WebClass, a workbench for the experimentation of statistical and machine learning techniques on the problem of Web page classification. We focus in particular on the system architecture and the Web pages processing steps currently implemented. On-going and future activities are sketched in the conclusions.

## 2. WebClass: System architecture

Recently several Web Assistant (WA) architectures have been proposed both from academics and industrials, such as Imana Siteseer (www.imana.com), Forté Inc. Agent (www.forteinc.com), Surflogic Softbot (www.surflogic.com), CMU Personal WebWatcher (www.cs.cmu.edu/~TextLearning), an so on. In our case we are adopting a "suit" style to build our system, this means that we are designing a WA as a set of integrated modules which can supply also a "scaleable" solution responding to users' needs. We started to power the IBM WA architecture, WBI (Web Browser Intelligence) (www.networking.ibm.com/wbi/wbisoft.htm), so to integrate learning abilities in it that supply the WA with the capability to automatically classify Web pages.

WBI's architecture is built on top of a Web/HTTP Proxy server extending its capabilities. The architecture is structured in such a way that any request to the proxy server can be handled by a number of plugin MEGs (Monitors/Editors/Generators) for each request. In particular, Monitors track the data stream between the Web and learn patterns and trends. Editors intercept the communication stream and are able to modify what is presented to the user. Finally, Generators can be used to add new content or pages dynamically. MEGs run to answer a request are determined by a simple configuration/rule language. In this general architecture we are inserting an intelligent component, WebClass, that the user can manage through the MEGs mechanisms.

Currently, WBI supplies WA services such as: 1) testing the speed of web links and graphically displaying whether links are fast, slow or unavailable; 2) remembering everywhere a user has been on the Web handling a local user's visited links database; 3) letting users look back paths traversed in the past; 4) automatically checking favorite Web pages for changes since the user's last visit. Through WebClass, a WBI user can supply a set of training Web pages on a set of predetermined topics and then ask the system to induce his/her user profile on those topics. The user profile can then be used either to graphically display the topic of links of the current Web page or to perform autonomous Web exploration with the aim to find interesting Web pages on a specific topic.

## 3. WebClass: Learning

In building a user profile there is the twofold problem of determining what information is relevant to the user and how this decision can be automatically taken. As to the first problem, we assume that the relevance of information to user's interests depends on the topic of the document, so that external factors to the textual information, such as the novelty of the document, are excluded. Moreover, we assume that the user is able to provide the system with a set of classes of documents, which correspond to as many topics of interest. WebClass is used in two distinct steps: a *training* phase in which the user browses the Web and adds HTML documents to folders corresponding to predetermined classes, and an *operational* phase in which the system actually assists the user in browsing or performs an autonomous search. The decision when to stop the training phase and invoke assistance is taken by the user itself.

As to the second problem, it is important to define both a representation language for HTML pages and a statistical or machine learning algorithm that is able to classify new pages on the grounds of their representation. WebClass adopts an attribute-value representation of Web pages, where each single attribute corresponds to some word extracted from the training documents. For instance, to describe documents concerning astronomy, music, economy and AIDS, WebClass may consider the following words (or attributes): solar, earth, stars, band, music, sound, economic, finance, economics, aids, hiv and health. Thus, each document can be described by a vector of twelve numbers, each of which represents the relative frequency of the term in the page (term frequency, TF).

The attributes are determined by means of a complex preprocessing phase. All training documents are initially tokenized, and tokens shorter than three characters are removed. The set of tokens (words) is filtered in order to remove numbers and stopwords, such as articles, prepositions and conjunctions. Stopwords have been taken from Glimpse (glimpse.cs.arizona.edu), a tool used to index files by means of words. No stemming algorithm is applied, so that both "planet" and "planets" may be selected attributes. Moreover, the structural information of HTML is not taken into account, so that WebClass currently makes no difference between words in the title and words in the body of the page.

Given the *j*th training document of the *i*th class, for each token *t* we compute the frequency of the token in the document TF(i,j,t). Then we find TF(i,t), the maximum of the TF(i,j,t) on all documents of the *i*th class, and PF(i,t), the percentage of documents of class *i* in which the token *t* occurs. The union of sets of tokens extracted from Web pages of the *i*th class defines a sort of "empirical" dictionary used for documents of class *i*.

It is possible to order each class dictionary according to the TF-$PF^2$ (Term Frequency - Square Page Frequency) measure, computed as $TF(i,t)*PF(i,t)^2$. In this way, common words used in documents of a given class will appear in the first entries of the corresponding class dictionary. Some of these words are actually specific of that class, while others are simply common English words, such as "information" or "time", and should be considered as quasi-stopwords. In order to move quasi-stopwords down in the dictionary we multiply the TF-$PF^2$ of each term by a factor 1/CF(t), where CF(t) (category frequency) is the percentage of class dictionaries in which the word *t* occurs. In this way, the sorted dictionary will have the most representative words of each class in the first entries, so that it will be enough to choose the first N words per dictionary in order to define the set of attributes. For instance, by selecting sixty HTML pages classified according to Yahoo ontology as documents concerning astronomy music, economy and AIDS, WebClass selects the following attributes (*N*=8):

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **astronomy** | *solar* | *earth* | *stars* | *sun* | *space* | *telescope* | *light* | *moon* |
| **music** | *band* | *music* | *sound* | *album* | *albums* | *james* | *click* | *records* |
| **economy** | *economic* | *finance* | *economics* | *financial* | *market* | *price* | *business* | *research* |
| **AIDS** | *aids* | *hiv* | *health* | *infected* | *clinical* | *risk* | *trials* | *services* |

Once the attributes are determined, it is possible to describe each training/test document as a feature vector whose numeric values correspond to the relative frequency of the corresponding word in the document.

WebClass has two ways for assigning a Web page to a class. The first way consists in applying some classifiers to the document description. Classifiers are multivariate decision trees that use non-parallel axis hyperplanes to partition a dataset. In Figure 1 a simple multivariate decision tree is showed. It classifies examples of astronomy documents by means of a unique multivariate test involving the linear combination of values taken by some attributes. A univariate decision tree learned from the same training set is reported in Figure 1. The reason for the choice of multivariate decision trees is that when there are several numerical features, they are generally more effective than univariate decision trees. They are induced from training documents by means of the learning system OC1 [8], which is tuned especially for domains in which the attributes are numeric. Optionally, the user can also ask OC1 to induce univariate decision trees.

A characteristic of OC1 is the combination of deterministic and randomization procedures to search for a good hyperplane that discriminate positive from negative examples. As a result, it is possible to generate different decision trees from the very same input data. This offers the possibility to build *k* decision trees for the same training set, and then classify each test example by the majority vote of *k* trees.

Finally, it is worthwhile to point out that WebClass generates a multivariate decision tree for each class. This is done by considering training examples of a class as positive, taking all examples of the remaining classes as negative. In this way the Web assistant can take into account multiple classifications of Web pages.

As an alternative to decision tree induction, WebClass can classify a document on the grounds of the similarity to the prototypes defined for each class. Prototypes are computed as the centroids of the classes represented by training examples. For instance, below we give a partial definition of the four prototypes computed for the sixty examples of astronomy, music, economy and AIDS :

| | **solar** | **earth** | … | **band** | **music** | … | **business** | **research** | … | **aids** | **hiv** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **astronomy** | 0.0005 | 0.0003 | | 0 | 0 | | 0 | 0 | | 0 | 0 |
| **music** | 0 | 0 | | 0.0006 | 0.0007 | | 0 | 0 | | 0 | 0 |
| **economy** | 0 | 0 | | 0 | 0 | | 0.0003 | 0.0003 | | 0 | 0 |
| **AIDS** | 0 | 0 | | 0 | 0 | | 0 | 0.0002 | | 0.0024 | 0.0022 |

In order to classify a document, we identify the prototype that is most similar to the document description. The similarity adopted by WebClass is the cosine of the angle spanned by two vectors (the document description and the prototype description) in the feature space.

## 4. Conclusions and future work

WebClass is a workbench for the experimentation of statistical and machine learning techniques on the problem of Web page classification. It has been implemented both in Java (preprocessor, user interface and Web page classifier) and in C (the learning system OC1) under Windows 95/NT. We are still working on the integration of WebClass into WBI in order to provide the user with two different types of services (assistance while browsing and autonomous exploration of the Web space). WebClass has been preliminarily tested on a small set of sixty HTML pages classified according to Yahoo's ontology as members of the classes astronomy, music, economy and AIDS. As future work we plan a wider experimentation in order to test the system on conceptually similar classes (e.g., rock music and jazz music) and increasing number of training examples. We also want to test the effect of considering only part of the HTML document, say the first four kilobytes, on the performance of the system. Other experiments can be made by taking into account the structure of the HTML, for instance, by assigning a weight to each tag HTML (10 for TITLE, 9 for H1, 8 for H2, and so on).

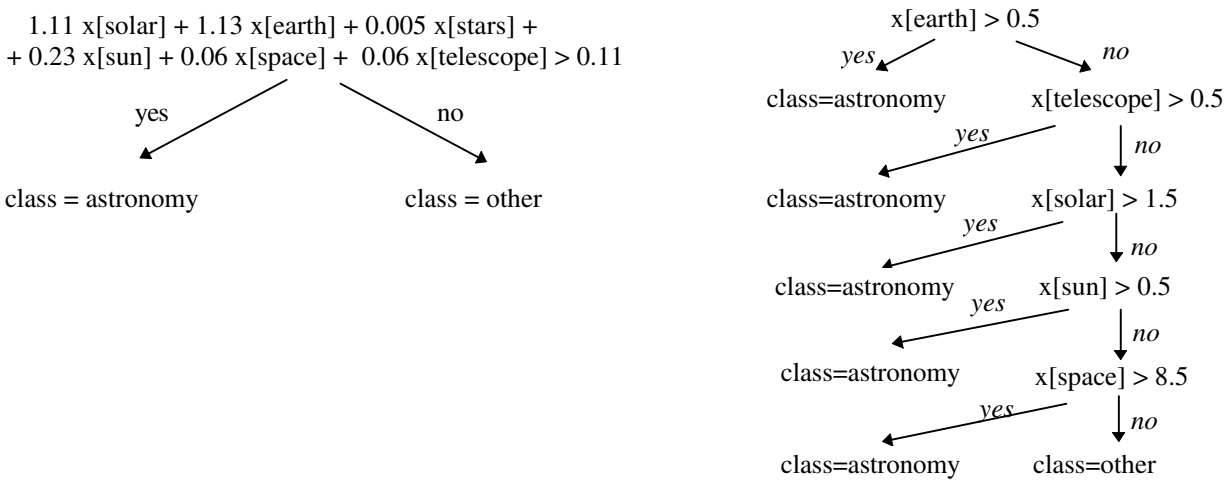


**Figure 1.** An example of multivariate decision tree (left) and univariate decision tree (right).

Web mining is revealing a source of new interesting and challenging problems for machine learning and statistics. One of the main issues is the automatic selection of features to describe the document. This is part of the preprocessing step, whose result strongly affects the subsequent learning phases. In the paper we presented a heuristic approach to feature selection, based on the combined effects of three statistics, the term frequency, the page frequency and the category frequency. This is a variant of the well known tf-idf measure used in the context of information retrieval [11]. Another important issue is the dinamicity of user's interests. The addition of a new category of documents leads to a representation change, since attributes selected in the preprocessing phase depends on the initial set of categories. To solve the problem we can either try to use learning algorithms that can take into account such representation changes or try to adopt a category-independent criterion for the selection of attributes. Both solutions are not at hand, yet.

**References**


[1] ACM Transactions on Information Systems - Special Issue on Information Extraction, 12(3), 1994

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

[3] Bowman, C.M., Danzig, P.B., Manber, U., & Schwartz, M.F., Scaleable Internet resource discovery: Research problems and approaches. *Communications of the ACM*, 37(8), 98-107, 1994.

[4] Etzioni, O., The World-Wide Web: Quagmire or Gold Mine? *Communications of the ACM*, 39(1), pp. 65-68, 1996.

[5] Lang, K., Learning to Filter Netnews. *Proc. of the 12th International Conference on Machine Learning*, pp. 331-339, 1995

[6] Lieberman, H., Letizia: An Agent That Assist Web Browsing. *Proc. of the 14th International Joint Conference on Artificial Intelligence*, pp. 924-929, 1995

[7] Joachims,T., Freitag, D., Mitchell, T., WebWatcher: A tour Guide for the World Wide Web. *Proc. of the 15th International Joint Conference on Artificial Intelligence*, pp. 770-775, 1997

[8] Murthy, S., Kasif, S., Salzberg, S., & Beigel, R., OC1: Randomized induction of oblique decision trees. *Proc. of the Eleventh National Conference on Artificial Intelligence AAAI 93*, pp. 322-327, 1993.

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

[10] Quek, C.Y., Classification of World Wide Web Documents. Technical Report, Carnegie Mellon University, Pittsburgh, PA, 1996.

[11] Salton, G., Developments in automatic text retrieval, *Science*, 253, pp. 974-980, 1991.

[12] Schwartz, M.F., Emtage, A., Kahle, B., & Neuman, B.C., A comparison of Internet resource discovery approaches. *Computer Systems*, 5(4), 461-493, 1992.

[13] Wang, Q.R., & Suen, C.Y., Large tree classifier with heuristic search and global training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(1), pp. 91-102, 1987.

[14] Zaiane, O.R., & Jiawei, H. Resource and knowledge discovery in global information systems: A preliminary design and experiment. *Proc. of Knowledge Database Discovery '95*, pp. 331-336, 1995.