Interaction Profiling in Digital Libraries through Learning Tools

G. Semeraro, F. Esposito, N. Fanizzi, and S. Ferilli

Dipartimento di Informatica
Università di Bari
Via Orabona, 4 I-70125 Bari
{semeraro,esposito,fanizzi,ferilli}@di.uniba.it

Abstract. We present improvements to a learning module, the *Learning Server*, to be exploited in a digital library system for supporting document management tasks as well as for providing a form of user interface adaptivity based on user classification. Indeed, our system is equipped with a web-based environment endowed with visual tools that are thought for improving the interaction of inexperienced users and for supporting experienced users in an effective accomplishment of their retrieval tasks. By logging user interaction, the Learning Server is able to suggest the most suitable interaction tools for each user.

1 Introduction

Digital libraries research focuses on the support for functions allowing access, retrieval, and organization of information, in order to make rapid decisions on what is relevant and which patterns exist among objects contained in the library. Thus, it becomes fundamental to decide which interface to assign to each user.

Among the characterizing features of CDL (Corporate Digital Library), a digital library prototype [11, 4], there is the exploitation of machine learning techniques for *document analysis*, *classification*, and *understanding* [5, 6]. Besides, the interaction environment is endowed with representations of some meta-information concerning document content, that provide users with proper cues for locating the desired data. Visual techniques are exploited, whose main advantage is the capability of shifting load from user's cognitive system to his perceptual system [10, 3].

We further improved the user-profiling module of the CDL visual environment [3], designed for providing each user with the most suitable interface that complies with his skills and knowledge about the content of a digital library. This feature is achieved by automatically classifying the user by means of machine learning techniques [10]. The Learning Server has been enriched with ultimate tools for the induction of decision trees [13].

The paper is organized as follows. Section 2 gives an overview of the architecture and the visual interaction environment of CDL. WEKA, a novel suite of machine learning and data mining tools, is presented in Section 3, while Section 4 shows experiments on interface adaptivity achieved by using the new learning component. Section 5 concludes the paper and outlines future work.

2 Overview

CDL was developed as an evolution of IDL (Intelligent Digital Library) [11, 4] in order to provide a common infrastructure supporting the creation, dissemination, manipulation, storage, integration and reuse of information. The adjective *corporate* designates a shared mechanism for searching information, updating content, controlling user access, charging users, etc., independently of the meaning and internal representation of information. The CDL project focuses on the development of effective middleware services for digital libraries, and on their interoperability across heterogeneous hardware and software platforms [2].

In CDL the typical client/server architecture of a hypertextual service on the Internet has been implemented. Since a thin-client stateful architecture [7] was adopted, it suffices a Web browser to run the application on the client-side. Indeed, there is no need of storing data locally: the DBMS runs on the server-side and its services are accessed by means of servlets. Furthermore, the architecture is characterized by a *Learning Server* that provides all the services related to document management. Besides, it is able to infer interaction profiles concerning the different types of users from data collected in *log files*. Advantages and a complete description of the architecture have been discussed in [11, 6, 3].

The main features of CDL are strictly related to the library functions. Supervised learning systems are used to overcome the problem of cheaply and effectively setting information items free of the physical medium on which they are stored (*information capture*). These learning tools are also used to automatically perform the tasks of *document classification* and *document understanding* (recognition of the logical structure of a document), that are necessary steps to index information items according to their content (*semantic indexing*). Finally, CDL provides different users with distinct interaction tools. It helps novice users to understand the content and the organization of a digital library through a suitable visual environment, and supports users that are familiar with the digital library content in easily and fast retrieving desired information items by means of an appropriate interface modality. A form of adaptivity is achieved through an automated user classification process based on machine learning techniques.

Three different roles have been identified for the interaction with CDL [4]: the *Library Administrator* runs the service and creates/deletes libraries; the *Librarians* manage specific digital libraries; and the *(Generic) User* requests CDL services. A user can consult the available libraries and query or browse them to retrieve documents of interest. Then, he may display/obtain the digital version of the documents found. The user interaction environment of CDL [3] is adapted according to the user class automatically recognized.

The *form-based* interface, more powerful and flexible, is appropriate for users who are already acquainted with the library structure and know about its content. Casual users often perform queries whose result is null, just because they do not have any idea of the kind of documents stored in the library. Other visual tools are better suited for novice users to grasp the nature of the information stored more intuitively and the possible patterns among the objects, so that they can make rapid decisions about what they really need and how to get it.

The *topic map* gives a global view of the semantic content of a collection of documents by exploiting a geographic metaphor [14]. A collection of items (topics or documents) is considered as a territory containing resources (the items themselves); regions, cities, and roads are used to convey an idea of the underlying semantic relationships: a region represents a set of items, and its size reflects the number of items in that region. Similarly, the distance between two cities reflects the similarity relationship between them: if two cities are close to each other, then the items are strongly related (e.g., topics are related to each other).

The *tree-based* interface provides another visual modality to both browse CDL and perform queries. The user navigates into CDL along a tree structure, starting from the root and expanding the tree step by step, so that at each node he can decide whether further exploring that path. By selecting a node, one can choose to expand it or to ask for an explanation of its meaning. Recently, this interface has been improved, giving the possibility of specializing or generalizing the currently explored search-index by means of an ontology.

3 A Learning Component Based on WEKA

WEKA (Waikato Environment for Knowledge Analysis) [13] is a system, written in Java, providing a uniform interface to different learning algorithms (*classifiers*), along with pre and post processing tools for evaluating their outcomes. Classifiers can be applied to datasets, supplied in a proper format (*arff*), made up of pre-classified instances of some class. Tools for preprocessing the data (*filters*) can be used to delete specific attributes from a dataset and to perform manual attribute selection. The learning schemes most important to our purposes are illustrated in the following.

J4.8 implements a recent improved version of C4.5 decision tree learner [9]. It outputs a decision tree in textual form (that might have been automatically pruned for achieving a better predictive accuracy) and an estimate of the tree predictive performance over both training and test datasets, generated by WEKA's evaluation module (see Figure 1). Nodes in a decision tree involve a test on a particular attribute; leaf nodes give a classification. To classify an unknown instance, a path in the tree is found starting from the root node, according to the values of the attributes tested in successive nodes, until a leaf node is reached. Classification rules represent an alternative to decision trees: the *antecedent* is a series of tests ANDed together; the *consequent* gives the class that applies to instances covered by that rule. Decision rules are generated directly off a decision tree, one for each leaf. They are usually pruned to remove redundant tests and to achieve a better predictive accuracy on further datasets. In the following we will use decision trees and rules as synonyms.

PART produces rules from partially pruned decision trees built using C4.5 heuristics. NAIVEBAYES implements the probabilistic Naive Bayesian classifier; DECISIONTABLE produces a decision table exploiting a good subset of attributes; IBK is an implementation of the k-nearest neighbors classifier (*instance-based*

```
J48 pruned tree
------------------
Freq_of_Tree_Based_choices <= 0.4
|   Freq_of_Topic_Map_choices <= 0.285714
|   |   Num_of_Parsing_Error_on_Form_Based <= 2
|   |   |   Num_of_connections <= 1: Novice (3.0/1.0)
|   |   |   Num_of_connections > 1
|   |   |   |   Num_of_Sort_on_Paper_of_class_Springer_of_DB_AI_in_DL <= 0
|   |   |   |   |   Num_of_Query_on_DB_AI_in_DL <= 14: Teacher (31.0)
|   |   |   |   |   Num_of_Query_on_DB_AI_in_DL > 14
|   |   |   |   |   |   Num_of_Query_on_Title_of_class_Springer_of_DB_AI_in_DL <= 0: Teacher (3.0)
|   |   |   |   |   |   Num_of_Query_on_Title_of_class_Springer_of_DB_AI_in_DL > 0: Novice (3.0)
|   |   |   |   Num_of_Sort_on_Paper_of_class_Springer_of_DB_AI_in_DL > 0: Novice (2.0/1.0)
|   |   Num_of_Parsing_Error_on_Form_Based > 2: Novice (6.0/1.0)
|   Freq_of_Topic_Map_choices > 0.285714
|   |   Num_of_Query_on_Authors_of_class_Icml_of_DB_AI_in_DL <= 1
|   |   |   Num_of_Query_on_Paper_of_class_Icml_of_DB_AI_in_DL <= 0
|   |   |   |   Freq_of_Sort_on_Authors_of_class_Icml_of_DB_AI_in_DL <= 0.111111: Novice (70.0/6.0)
|   |   |   |   Freq_of_Sort_on_Authors_of_class_Icml_of_DB_AI_in_DL > 0.111111
|   |   |   |   |   Num_of_Query_on_Abstract_of_class_Icml_of_DB_AI_in_DL <= 0: Teacher (3.0)
|   |   |   |   |   Num_of_Query_on_Abstract_of_class_Icml_of_DB_AI_in_DL > 0: Novice (3.0)
|   |   |   Num_of_Query_on_Paper_of_class_Icml_of_DB_AI_in_DL > 0: Teacher (3.0/1.0)
|   |   Num_of_Query_on_Authors_of_class_Icml_of_DB_AI_in_DL > 1
|   |   |   Freq_of_NullQuery_on_class_Springer_of_DB_AI_in_DL <= 0.076923: Teacher (3.0)
|   |   |   Freq_of_NullQuery_on_class_Springer_of_DB_AI_in_DL > 0.076923: Novice (2.0/1.0)
Freq_of_Tree_Based_choices > 0.4
|   Freq_of_Form_Based_choices <= 0.444444
|   |   Num_of_Help_Request_of_Form_Based <= 0
|   |   |   Num_of_Sort_on_Abstract_of_class_Icml_of_DB_AI_in_DL <= 0
|   |   |   |   Num_of_Sort_on_Affiliation_of_class_Springer_of_DB_AI_in_DL <= 0: Expert (23.0)
|   |   |   |   Num_of_Sort_on_Affiliation_of_class_Springer_of_DB_AI_in_DL > 0
|   |   |   |   |   Num_of_Query_on_Title_of_class_Icml_of_DB_AI_in_DL <= 1: Novice (2.0)
|   |   |   |   |   Num_of_Query_on_Title_of_class_Icml_of_DB_AI_in_DL > 1: Expert (4.0)
|   |   |   Num_of_Sort_on_Abstract_of_class_Icml_of_DB_AI_in_DL > 0
|   |   |   |   Num_of_Query_on_Keywords_of_class_Pami_of_DB_AI_in_DL <= 0: Expert (2.0)
|   |   |   |   Num_of_Query_on_Keywords_of_class_Pami_of_DB_AI_in_DL > 0: Novice (2.0)
|   |   Num_of_Help_Request_of_Form_Based > 0
|   |   |   Num_of_Sort_on_Title_of_class_Icml_of_DB_AI_in_DL <= 0
|   |   |   |   Freq_of_Query_on_Keywords_of_class_Pami_of_DB_AI_in_DL <= 0.066667: Novice (8.0/1.0)
|   |   |   |   Freq_of_Query_on_Keywords_of_class_Pami_of_DB_AI_in_DL > 0.066667: Expert (2.0)
|   |   |   Num_of_Sort_on_Title_of_class_Icml_of_DB_AI_in_DL > 0: Expert (2.0)
|   Freq_of_Form_Based_choices > 0.444444: Teacher (2.0)

Number of Leaves  :     21

Size of the tree :  41


=== Error on training data ===

Correctly Classified Instances         167               93.2961 %
Incorrectly Classified Instances        12                6.7039 %
Mean absolute error                      0.0715
Root mean squared error                  0.1891
Total Number of Instances              179

=== Confusion Matrix ===

  a  b  c   <-- classified as
 90  0  1 |  a = Novice
  3 33  0 |  b = Expert
  8  0 44 |  c = Teacher

=== Error on test data ===

Correctly Classified Instances          12               57.1429 %
Incorrectly Classified Instances         9               42.8571 %
Mean absolute error                      0.2815
Root mean squared error                  0.5092
Total Number of Instances               21

=== Confusion Matrix ===

 a b c   <-- classified as
 5 0 5 | a = Novice
 2 4 0 | b = Expert
 2 0 3 | c = Teacher
```

Fig. 1. A decision tree induced by J4.8

learning); M5' implements an induction algorithm generating *model trees*, that predict the class value using a linear regression model stored at the leaf nodes.

WEKA also includes implementations of algorithms for *clustering* and for learning *association rules*. Clustering is used to group instances that seem to naturally fall together according to some similarity measure. The aim of building clusters of the training instances is to discover classes underlying the dataset in order to be able to assign further instances to them.

WEKA can be used for inducing profiles of the quality of user interaction with CDL, to build an extractor of permanent attributes (a *profile*) from logs containing session related features together with their relationships. This can be cast as a problem of supervised learning, aiming at producing rules for classifying users on the ground of their interaction. WEKA supervised learning techniques can be used for analyzing instances of interaction logs that describe features such as user daily number of connections, queries on a given search-index, etc. One of WEKA's learning schemes can be used to induce a set of rules, provided a dataset of training instances together with their classes. Then, users accessing the digital library service can be categorized on the basis of their logs, so that the most suitable interface can be chosen for them.

A different perspective regards exploiting the logs to generate more general *usage patterns* that characterize groups of users (abstracting away from individual differences) and can be used to generate general assumptions about new users. Since classes are not given *a priori*, WEKA's unsupervised learning methods could be used to extract regularities from the logs in order to find out relevant rule patterns for identifying communities of users. If the descriptions are likely to change in time, incremental approaches could be best suited.

4 The Learning Server: Interaction Profiling

CDL Learning Server can be defined as a suite of learning systems that can be exploited concurrently for performing tasks such as the mentioned document analysis, classification and understanding as well as the inference of user interaction profiles. In this section, we address this last task by means of supervised learning methods. Indeed, when developing a system accessed by several users like a service on the World-Wide Web, a fundamental problem to cope with is ensuring effective answers to the needs of various user classes. In fact, each user may have special capabilities, skills, knowledge, preferences and goals, thus he will behave differently from other users. Being hard to recognize each single user, we take into account classes of users that may interact similarly.

In the architecture of CDL, an intelligent component of the Learning Server is able to automatically assign a user to a specific class in order to improve the system usability. This component should help users at accomplishing their goal easier (through contextual helps, explanations, suitable interaction modalities etc.). As a consequence, one of the main problems concerns the definition of meaningful user classes, and the identification of the features that properly describe each of them and characterize the corresponding kind of interaction.

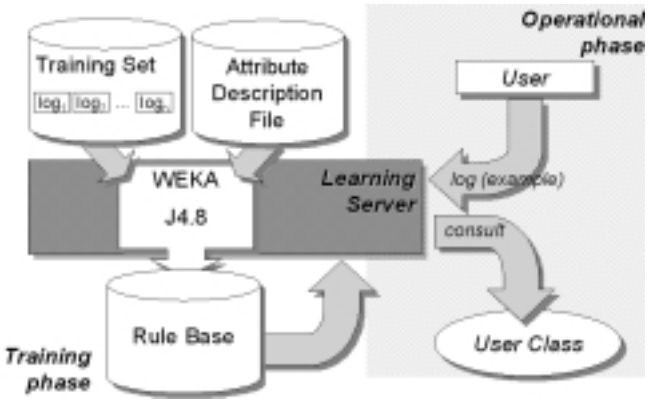

Fig. 2. Assigning user classes through the CDL Learning Server

4.1 Assigning User Classes

The main function required is to automatically assign each user to one of some predefined classes on the ground of information drawn from real interaction sessions (*interaction modeling*) [1]. Our approach relies on machine learning methods [8], since interaction modeling can be cast as a supervised learning problem by considering some user interactions with CDL as training instances for a learning system, whose goal is to induce rules for classifying CDL users (see Figure 2). Such a classification is used to associate each user class with an interface that is adequate to the user's degree of familiarity with the service. This aims at speeding up the process of understanding the organization and the content of the digital libraries of interest for an effective retrieval of the desired information.

We defined three possible classes of CDL users, namely `Novice`, `Expert` and `Teacher`. Since, during the time, the user may become familiar with the system usage, potential changes of user class should be tracked. This requires user registration and identification. Each new user has to fill in a form with personal data, then he receives an identity code - User ID - that he will use whenever he enters CDL again. Correspondingly, a log file is associated to each User ID.

By examining the log file, it is possible to extract some characteristics that are useful for recognizing the user class. We observed that most of the characteristics identified turned out to be application dependent, while only few showed to be system dependent. For instance, relevant characteristics are those concerning the way users exploit the capabilities of CDL search engine, such as date and time of session beginning, document class or search indexes chosen, criterion for sorting the search results, number of documents obtained as results of the search, types of errors made during the interaction.

Log files are then exploited to train a learning system in order to induce a decision tree and a set of rules that are used by the system to autonomously classify users. Figure 3 reports a portion of an *arff* file, containing the attributes and the corresponding values coming from the users log files used for our purposes.

```
@RELATION userdata

@ATTRIBUTE Num_of_connections REAL
@ATTRIBUTE Average_daily_connections REAL
@ATTRIBUTE Num_of_Query_on_DB_AI_in_DL REAL
@ATTRIBUTE Freq_of_Query_on_DB_AI_in_DL REAL
@ATTRIBUTE Num_of_Query_on_Title_of_class_Springer_of_DB_AI_in_DL  REAL
@ATTRIBUTE Freq_of_Query_on_Title_of_class_Springer_of_DB_AI_in_DL REAL
@ATTRIBUTE Num_of_Sort_on_Title_of_class_Springer_of_DB_AI_in_DL  REAL
@ATTRIBUTE Freq_of_Sort_on_Title_of_class_Springer_of_DB_AI_in_DL REAL
...
@ATTRIBUTE Num_of_Query_on_Authors_of_class_Springer_of_DB_AI_in_DL  REAL
@ATTRIBUTE Freq_of_Query_on_Authors_of_class_Springer_of_DB_AI_in_DL  REAL
@ATTRIBUTE Num_of_Sort_on_Authors_of_class_Springer_of_DB_AI_in_DL REAL
@ATTRIBUTE Freq_of_Sort_on_Authors_of_class_Springer_of_DB_AI_in_DL  REAL
.
.
.
@ATTRIBUTE Num_of_Sort_on_Authors_of_class_Icml_of_DB_AI_in_DL REAL
@ATTRIBUTE Freq_of_Sort_on_Authors_of_class_Icml_of_DB_AI_in_DL REAL
@ATTRIBUTE Num_of_Query_on_All_of_DB_AI_in_DL REAL
@ATTRIBUTE Freq_of_Query_on_All_of_DB_AI_in_DL REAL
@ATTRIBUTE Num_of_Sort_on_Title_of_All_of_DB_AI_in_DL  REAL
@ATTRIBUTE Freq_of_Sort_on_Title_of_All_of_DB_AI_in_DL  REAL
...
@ATTRIBUTE Num_of_Sort_on_Authors_of_All_of_DB_AI_in_DL  REAL
@ATTRIBUTE Freq_of_Sort_on_Authors_of_All_of_DB_AI_in_DL  REAL
@ATTRIBUTE Num_of_NullQuery_on_class_Springer_of_DB_AI_in_DL  REAL
@ATTRIBUTE Freq_of_NullQuery_on_class_Springer_of_DB_AI_in_DL  REAL
...
@ATTRIBUTE Num_of_NullQuery_on_class_Icml_of_DB_AI_in_DL  REAL
@ATTRIBUTE Freq_of_NullQuery_on_class_Icml_of_DB_AI_in_DL  REAL
@ATTRIBUTE Num_of_NullQuery_on_All_of_DB_AI_in_DL  REAL
@ATTRIBUTE Freq_of_NullQuery_on_All_of_DB_AI_in_DL  REAL
@ATTRIBUTE Num_of_Topic_Map_choices  REAL
@ATTRIBUTE Freq_of_Topic_Map_choices  REAL
...
@ATTRIBUTE Freq_of_Tree_Based_choices  REAL
@ATTRIBUTE Num_of_Help_Request_of_Topic_Map REAL
@ATTRIBUTE Freq_of_Help_Request_of_Topic_Map  REAL
...
@ATTRIBUTE Freq_of_Help_Request_of_Tree_Based  REAL
@ATTRIBUTE Num_of_Parsing_Error_on_Topic_Map REAL
@ATTRIBUTE Freq_of_Parsing_Error_on_Topic_Map  REAL
...
@ATTRIBUTE Freq_of_Parsing_Error_on_Tree_Based  REAL
@ATTRIBUTE Num_of_Error_on_DB_choice REAL
@ATTRIBUTE Freq_of_Error_on_DB_choice  REAL
@ATTRIBUTE class {Novice,Expert,Teacher}

@DATA
3,1.0,9,1.0,0,0.0,0,0,0.0,0,0.0,0,0,0.0,0,0.0,0,0,0.0,0,0.0,0,0,0.0,0,0.0,0,0.0,0,0.0,0,0.0,2,0.22222222,0,0.0,
1,0.11111111,0,0.0,0,0.0,1,0.25,0,0.0,1,0.25,0,0.0,0,0.0,0,0.0,0,0.0,0,0.0,0,0.0,0,0.0,1,0.11111111,0,0.0,1,0.11111111,
0,0.0,0,0.0,0,0.0,1,0.11111111,0,0.0,0,0.0,0,0.0,0,0.0,0,0.0,0,0.0,1,0.11111111,0,0.0,3,0.33333334,0,0.0,1,0.25,0,0.0,
0,0.0,1,0.25,0,0.0,1,0.11111111,1,0.11111111,0,0.0,1,0.25,2,0.5,1,0.25,0,0.0,1,1.0,0,0.0,0,0.0,4,1.0,0,0.0,0,0.0,Teacher

2,1.0,14,1.0,1,0.071428575,0,0.0,0,0.0,0,0,0.0,0,0.0,0,0,0.0,0,0.0,0,0,0.0,0,0.0,0,0,0.0,0,0.0,0,0.0,0,0.0,0,0.0,3,0.21428572,
1,0.14285715,1,0.071428575,0,0.0,0,0.0,1,0.14285715,0,0.0,0,0.0,1,0.071428575,0,0.0,0,0.0,0,0.0,0,0.0,0,0.0,0,0.0,0,0.0,
1,0.14285715,1,0.071428575,1,0.14285715,2,0.14285715,0,0.0,0,0.0,1,0.14285715,0,0.0,0,0.0,0,0.0,0,0.0,1,0.071428575,
0,0.0,2,0.14285715,0,0.0,0,0.0,0,0.0,0,0.0,2,0.2857143,0,0.0,2,0.14285715,1,0.071428575,1,0.071428575,1,0.2,2,0.4,
2,0.4,0,0.0,1,1.0,0,0.0,0,0.0,2,1.0,0,0.0,2,0.33333334,Teacher

.
.
.
```

Fig. 3. A portion of the *arff* file used for user classification

4.2 Experimental Results

Following the preliminary experiment described in [3], both C4.5 [9] and the WEKA classifier J4.8 [13] have been customized in order to infer the classification rules in a batch way. Another experimentation has been set up with ITI 2.0 [12], a system which supports the incremental construction of decision trees by revising the current tree in response to each newly observed training instance. ITI can operate in two different incremental ways. In the normal operation mode, it first updates the frequency counts associated to each node of the tree as soon as a new instance is received, then it restructures the decision tree according to the updated frequency counts. In the Error Correction (EC) mode, the frequency counts are updated only in case of misclassification of a new instance. The main difference is that the normal operation mode guarantees the building of the same decision tree independently of the order in which the examples are presented, while the error-correction mode does not.

The new experiment on classification of CDL users involved 200 instances of interaction, 90% of which were drawn as training set, while the rest was used as test set. Each log file was used to draw 128 attribute values that describe user interactions and it was associated to one of the three classes mentioned. Then, the classifiers have been run in parallel on the same training and test sets for a comparison of the outcomes. The experiment has been replicated ten times by growing the decision trees on the training set, then pruning them. The performance of both unpruned and pruned trees has been evaluated by using them for classifying instances in the test set.

Table 1 reports the results of the experiment, in terms of predictive accuracy on the test set. A first comparison between the results obtained by C4.5 and J4.8 reveals that the latter outperforms the former in half of the runs, while the opposite happens in only 3 cases (2 runs are even), both for the pruned and for the unpruned case. In fact, such a predominance is confirmed by the mean predictive accuracy only for the pruned case, probably due to the very good performance reached in run #1 by C4.5 unpruned. On the other hand, comparing such systems with the incremental ones results in a clear success of the latter, whose predictive accuracy is 2.4% better in the worst case (ITI unpruned vs. C4.5 unpruned), reaching a 4.4% improvement in the best case (ITI EC pruned vs. C4.5 pruned). In particular, as regards ITI, the error correction modality outperforms by 0.4% the normal one when no pruning is allowed, while there is no clear difference between the two in the pruned case.

After the training phase, whenever any user accesses CDL through a client, the log file generated during the interaction session is exploited to provide a new example that the Learning Server will classify on the ground of the inferred rules. The way in which rules are consulted by the Learning Server (and the existence of the classification rules itself) is completely transparent to the user. On the ground of the classification, the Learning Server selects a distinct type of interface, regarded as the proper one for that user class. Specifically, the Learning Server suggests to prompt any user recognized as a member of the class `Novice`

Table 1. Outcomes of the experimentation (Predictive Accuracy)

	0	1	2	3	4	5	6	7	8	9	mean
C4.5 unpruned	47.6	95.2	66.7	71.4	66.7	61.9	66.7	71.4	47.4	66.7	66.2
J4.8 unpruned	66.7	66.7	66.7	81.0	71.4	47.6	66.7	81.0	36.8	76.2	66.1
ITI unpruned	85.7	71.4	66.7	66.7	66.7	61.9	66.7	66.7	52.6	76.2	68.1
ITI EC unpruned	71.4	81.0	61.9	85.7	71.4	47.6	66.7	71.4	47.4	81.0	68.6
C4.5 pruned	47.6	95.2	66.7	76.2	66.7	61.9	66.7	76.2	42.1	76.2	67.6
J4.8 pruned	57.1	85.7	71.4	81.0	71.4	52.4	66.7	81.0	36.8	76.2	68.0
ITI pruned	85.7	76.2	76.2	76.2	71.4	57.1	57.1	81.0	52.6	85.7	71.9
ITI EC pruned	76.2	90.5	42.9	81.0	76.2	71.4	52.4	85.7	57.9	85.7	72.0

with the topic map interface, the tree-based interface is proposed to a user in the class `Expert` and `Teacher` users have the form-based interface as a default.

The main idea underlying the mapping between user classes and types of interfaces is that users being unfamiliar with the system need an environment that allows them to preliminarily understand the content of a digital library. More skilled users are supposed to be already acquainted with both the organization and the content of the digital library, thus they need tools that allow to speed up the retrieval process. Independently of the decision made by the system classifier, any user is free to switch to another interface at any moment.

The system showed the ability to track user class evolution. Indeed, after a number of interactions, CDL may propose a different interface to the same user, as soon as more information are collected in his log.

5 Conclusions and Future Work

We presented a component of the Learning Server exploited in our digital library CDL for performing a form of user profiling. Indeed, CDL is equipped with a web-based visual environment, offering different solutions to differently skilled users. An effective user classification as that provided by a learning component may be fundamental for providing a form of interface adaptivity. Both systems that operate in a batch way and incremental ones have been tested on such a task.

We intend further study the use of incremental learning schemes, that avoid the drawback of starting over the learning process from scratch each time new log examples become available, and hence are more suitable for the nature of Digital Library applications. Besides, being the WEKA based on Java technology, it seems suitable for an exploitation for online learning. Furthermore, also unsupervised learning methods could be exploited to cluster instances together to form *usage patterns*, thus inferring also the number of classes directly from the available data. Finally, we plan to apply learning tools to the inference of the ontologies mentioned in Section 2. This would make the search through the tree-based interface more knowledge-intensive.

Acknowledgments. The authors would like to thank Francesca Lioce who performed the experiments with the CDL Learning Server.
This work was partially funded by the IST Project COGITO 1999-13347.

References

[1] D. Banyon and D. Murray. Applying user modeling to human-computer interaction design. *Artificial Intelligence Review*, 7:199–225, 1993.

[2] P. A. Bernstein. Middleware: A model for distributed system services. *Communications of the ACM*, 39(2):86–98, 1996.

[3] M.F. Costabile, F. Esposito, G. Semeraro, and N. Fanizzi. An adaptive visual environment for digital libraries. *International Journal of Digital Libraries*, 2(2+3):124–143, 1999.

[4] M.F. Costabile, F. Esposito, G. Semeraro, N. Fanizzi, and S. Ferilli. Interacting with IDL: The adaptive visual interface. In *Research and Advanced Technology for Digital Libraries. Proceedings of the 2nd European Conference - ECDL98*, volume 1513 of *Lecture Notes in Computer Science*, pages 515–534. Springer, 1998.

[5] F. Esposito, D. Malerba, and G. Semeraro. Multistrategy learning for document recognition. *Applied Artificial Intelligence*, 8(1):33–84, 1994.

[6] F. Esposito, D. Malerba, G. Semeraro, N. Fanizzi, and S. Ferilli. Adding machine learning and knowledge intensive techniques to a digital library service. *International Journal of Digital Libraries*, 2(1):3–19, 1998.

[7] M.C. McChesney. Banking in cyberspace: An investment in itself. *IEEE Spectrum*, pages 54–59, February 1997.

[8] V.S. Moustakis and J. Herrmann. Where do machine learning and human-computer interaction meet? *Applied Artificial Intelligence*, 11:595–609, 1997.

[9] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[10] G. Semeraro, M. F. Costabile, F. Esposito, N. Fanizzi, and S. Ferilli. A learning server for inducing user classification rules in a digital library service. In Z.W. Raś and A. Skowron, editors, *Proceedings of the 11th International Symposium on Methodologies for Intelligent Systems*, volume 1609 of *Lecture Notes on Artificial Intelligence*, pages 208–216. Springer, 1999.

[11] G. Semeraro, F. Esposito, D. Malerba, N. Fanizzi, and S. Ferilli. Machine learning + on-line libraries = IDL. In C. Peters and C. Thanos, editors, *Research and Advanced Technology for Digital Libraries. First European Conference - ECDL97*, volume 1324 of *Lecture Notes in Computer Science*, pages 195–214. Springer, 1997.

[12] P.E. Utgoff, N. Berkman, and J.A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29:5–44, 1997.

[13] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[14] M. Zizi and M. Beaudouin-Lafon. Hypermedia exploration with interactive dynamic maps. *International Journal on Human-Computer Studies*, 43:441–464, 1995.