# Machine Learning Approaches for Inducing Student Models

Oriana Licchelli, Teresa M.A. Basile, Nicola Di Mauro, Floriana Esposito, Giovanni Semeraro, and Stefano Ferilli

Dipartimento di Informatica - Università degli Studi di Bari
via E. Orabona, 4 – 70125 Bari - Italia
{licchelli, basile, nicodimauro, esposito, semeraro, ferilli}@di.uniba.it

**Abstract.** The main issue in e-learning is student modelling, i.e. the analysis of a student's behaviour and prediction of his/her future behaviour and learning performance. Indeed, it is difficult to monitor the students' learning behaviours. A solution is the exploitation of automatic tools for the generation and discovery of user profiles, to obtain a simple student model based on his/her learning performance and communication preferences, that in turn allows to create a personalized education environment. This paper focuses on Machine Learning approaches for inducing student profiles, respectively based on Inductive Logic Programming (the INTHELEX system) and on methods using numeric algorithms (the Profile Extractor system), to be exploited in this environment. Moreover, an experimental session has been carried out, comparing the effectiveness of these methods along with an evaluation of their efficiency in order to decide how to best exploit them in the induction of student profiles.

## 1 Introduction and Motivations

In all areas of the e-era, personalization plays an important role. Specifically, in e-learning a main issue is student modelling since it is not easy to monitor students' learning behaviours. Adaptive personalized e-learning systems could accelerate the learning process by revealing the strengths and weaknesses of each student. They could dynamically plan lessons and personalize the communication and didactic strategy.

Artificial Intelligence (AI) offers powerful methods, which are useful in the development of adaptive systems. In the past, several intelligent techniques have been experimented in the ITS (Intelligent Tutoring Systems) development: in particular, AI techniques were exploited for the representation of pedagogical knowledge, the construction of the knowledge bases related both to the subject domain and to the didactic strategies and, finally, the student model generation, based on explicit knowledge of the student behaviour or on the analysis of the student mistakes and misunderstandings. Using AI techniques, Computer-Assisted Instruction systems can be adapted, during the interaction, to the student personality, characteristics and learning performance.

However, still today, many teaching systems based on the Web have not capitalized such experience and are often not capable to personalize the instruction material they supply in order to satisfy the needs of each single student. Anyway, a lot of attention has been given to user modelling in e-learning systems: for instance, EUROHELP [2] was devised to provide tools and methods for developing Intelligent Help Systems; InterBook [3] provided a user model based on *stereotype*, which represented the user's knowledge levels about every domain concept, and was modified as the user moved through the information space. Other projects used specific criteria to define a user ability model (e.g. MANIC [12], an online courseware system, determines user typology through heuristics, such as which slides the student has seen and which quizzes he/she has done).

In this paper we have focused our attention on two different Machine Learning approaches in order to discover the user-student preferences, needs and interests and to generate simple student models based on the learning performance and the communication preferences. In particular, we have used two different systems for inducing student profiles, respectively INTHELEX (based on Inductive Logic Programming) and the Profile Extractor system (that exploits algorithms based on numeric methods).
Assuming to have a first set of students and to be able to group them in a number of classes, each of which represents a student category, it is possible, by means of inductive methods of Machine Learning, to infer the concepts, i.e. the intensional definitions of student classes, which represent the student models. The training set from which to infer the conceptual user-student models (profiles) is made up of data concerning each student. Such data were initially collected through preliminary tests to estimate the students' background knowledge and to gather information concerning their educational goals and motivations, the preferred modalities of communication etc. Then, they were enriched by the logs of the successive interactions.

After illustrating both systems, we will provide a comparison of the effectiveness of these methods along with an evaluation of their efficiency in such an environment, in order to better understand how they could be exploited in an e-learning platform.

## 2 The Profile Extractor System

The Profile Extractor System, developed at LACAM (Knowledge Acquisition and Machine Learning Laboratory of the University of Bari) is a system to generate user profiles automatically [1]. It is a highly reusable module that allows the classification of users through the analysis of their past interaction with the system and employs supervised learning techniques.
Figure 1 shows the complete system architecture, that is further subdivided into four modules: Profile Rules Extractor, Profile Manager, Usage Patterns Extractor and XML I/O Wrapper.

The Profile Manager and the Profile Rules Extractor are the modules mainly involved in the profile generation process; the Usage Patterns Extractor groups dialogue sessions in order to infer some usage patterns that can be exploited for understanding user trends and for grouping single users, who share the same interests and preferences, into user communities [8]. The XML I/O Wrapper is the layer responsible for the integration of the inner modules with external data sources (using the XML protocol) and for the extraction of the data required for the learning process.

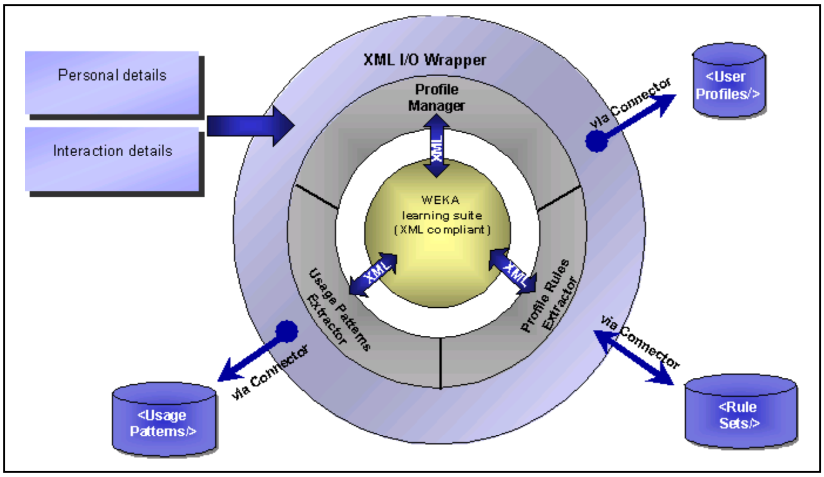

**Fig. 1.** The architecture of the Profile Extractor

The input to the Profile Extractor is represented by the XML file that contains the personal and interaction data of the user. This information is arranged into a set of unclassified instances, where each instance represents a single user, from the XML I/O Wrapper. The subset of the instances chosen to train the learning system has to be preclassified by a domain expert (each user is associated with a subset of the categories): this is the actual input to the Profile Rules Extractor, which will infer classification rule sets. The actual user profile generation process is performed by the Profile Manager, on the grounds of the user data and the set of rules induced by the Profile Rules Extractor. When the need to generate/update user profiles arises, the user data are arranged into a set of instances which represents the input to the Profile Manager. On the basis of the classification rule sets inferred, the classifier predicts the user behaviour in a system.

For the purpose of extracting user profiles, we focused on supervised machine learning techniques. Starting from preclassified examples of some target concepts, these techniques induce rules useful for predicting the classification of further unclassified examples. For this reason the core of the Profile Extractor is WEKA [13], a machine learning tool developed at the University of Waikato (New Zealand), that provides a uniform interface to many learning algorithms, along with methods for pre/post-processing and for the evaluation of the results of learning schemes, when applied to any given dataset. To integrate WEKA in the Profile Extractor we developed XWEKA, an XML compliant version of WEKA, that is able to represent input and output in XML format. The learning algorithm adopted in the profile generation process is based on PART [5], a rule-based learner that produces rules from pruned partial decision trees, built using C4.5's heuristics [9]. The antecedent, or precondition, of a rule is a series of tests, just like the tests at nodes in the classification path of a decision tree, while the consequent, or conclusion, gives the class that applies to instances covered by that rule. The main advantage of this method is not performance but simplicity: it produces good rule sets without any need for global optimization.

Extensive experimentation of the system proposed for the automatic extraction of the user profile has been carried out in a field not far from that of e-learning: digital libraries. We experimented the Profile Extractor System in digital libraries in several contexts like e-Commerce [1] and contemporary European cultural documents [6].

## 3 The Symbolic Learning System: INTHELEX

INTHELEX[1] (INcremental THEory Learner from EXamples) [4] is a symbolic learning system for the induction of *hierarchical* first-order logic theories from positive and negative examples. It can focus the search for definitions by exploiting the *Object Identity* bias (according to which terms denoted by different names within a formula must refer to different objects) [11]. Such a system is able to learn simultaneously *multiple concepts* (i.e., definitions for different concepts), possibly related to each other; it guarantees validity of the theories on all the processed examples; it uses feedback on performance to activate the theory revision phase; in addition to the possibility of refining a previously generated version of the theory, learning can also start from an empty theory. It exploits a previous version of the theory (if any), a graph describing the dependence relationships among concepts, and an historical memory of all the past examples that led to the current theory.

Incremental learning is necessary when either incomplete information is available at the time of initial theory generation, or the nature of the concepts evolves dynamically, which are unnegligible issues for the task of learning student profiles. Indeed, the classical batch models, that perform learning in one step and hence require the whole set of observations to be available from the beginning, are not able to handle such cases, that are very frequent in real-world situations. Thus, the need for incremental models to complete and support the batch ones.

The learning cycle performed by INTHELEX can be described as follows. A set of examples of the concepts to be learned, possibly selected by an expert, is provided by the environment. This set can be subdivided into three subsets, namely training, tuning, and test examples, according to the way in which examples are exploited during the learning process. Specifically, training examples, previously classified by the expert, are stored in the base of processed examples, and are exploited to obtain a theory that is able to explain them. Such an initial theory can also be provided by the expert, or even be empty. Subsequently, the validity of the theory against new available examples, also stored in the example base, is checked by taking the set of inductive hypotheses and a tuning/test example as input and producing a decision that is compared to the correct one. In the case of incorrectness on a tuning example, the cause of the wrong decision can be located and the proper kind of correction chosen, firing the theory revision process. In this way, tuning examples are exploited

[1] INTHELEX is currently available in binary format for i586 DOS-based platforms (`http://lacam.di.uniba.it:8000/systems/inthelex/`).

incrementally to modify incorrect (too weak or too strong) hypotheses according to a data-driven strategy. Test examples are exploited just to check the predictive capabilities of the theory, intended as the behavior of the theory on new observations, without causing a refinement of the theory in the case of incorrectness.

Another peculiarity of INTHELEX is the integration of multistrategy operators that may help to solve the theory revision problem by pre-processing the incoming information [7]. The purpose of induction is to infer, from a certain number of significant observations, regularities and laws that may be valid for the whole population. INTHELEX incorporates two inductive refinement operators, one for generalizing hypotheses that reject positive examples, and the other for specializing hypotheses that explain negative examples. Exceptions are exploited as a last resort when no correct refinement can be found. Deduction is exploited to fill observations with information that is not explicitly stated, but is implicit in their description, and hence refers to the possibility of better representing the examples and, consequently, the inferred theories. Indeed, since the system is able to handle a hierarchy of concepts, some combinations of predicates might identify higher level concepts that are worth adding to the descriptions in order to raise their semantic level. The concepts hierarchy contains all the dependencies among the concepts to be learned; if there are no expressed relations, the system will assume the concepts to be isolated. Such relations are expressed as a set of clauses like the following:

*bicycle(X) :- wheel(X), mechanic(X,Y)*
*mechanic(X,Y):-bicycle_chain(X,Y,Z),front_gear(X,Y),rear_gear(X,Y)*

whose interpretation is, respectively: "concept *bicycle* depends on concepts *wheel* and *mechanic*"; and "concept *mechanic* depends on concepts *bicycle_chain*, *front_gear* and *rear_gear*". In the graph variables are used as placeholders to indicate the concepts arity.

INTHELEX exploits deduction to recognize such concepts and explicitly add them to the example description. The system can be provided with a Background Knowledge containing complete or partial definitions expressed as the theory rules. Abduction aims at completing possibly partial information in the examples, adding more details. Its role in INTHELEX is helping to manage situations where not only the set of all observations is partially known, but each observation could also be incomplete. Abducibles are the predicates on which assumptions (abductions) can be made; integrity constraints provide indirect information on them. The proof procedure implemented in INTHELEX corresponds, intuitively, to the standard Logic Programming derivation suitably extended in order to consider abducibles and integrity constraints. Lastly, Abstraction removes superfluous details from the description of both the examples and the theory. The exploitation of abstraction in INTHELEX concerns the shift from the language in which the theory is described to a higher level one. An abstraction theory contains information on the operators according to which the shift is to be performed. INTHELEX automatically applies it to the learning problem at hand before processing the examples. The implemented abstraction operators allow the system to replace a number of components with a compound object, to decrease the granularity of a set of values, to ignore whole objects or just part of their features, and to neglect the number of occurrences of a certain kind of object.

## 4 Experimental Sessions

CAMPUS ONE is an e-Learning Project of the University of Bari, (`http://www.campusone.uniba.it`), for providing courses on Fundamentals of Computer Science for all types of degree (human degree, science degree, etc.). In this project, each student for each kind of degree must attend the first two modules (*Module 1 Fundamentals Computer Science, Module 2 Management Computer And File*). An experiment was performed for each module, by identifying three classes for each of them on the ground of the final student performance evaluation: good, sufficient or insufficient. The information on each student were gathered from the log file of an e-learning platform.

### 4.1 Design of the Experiments

The experimental dataset was made up of information on 140 students. The students'profiles were classified, by a domain expert, as *Good*, *Sufficient*, or *Insufficient* in the two modules above mentioned. The distribution of students into the three classes for both the considered modules are as follows. For module 1: 45% Good – 23,57% Sufficient – 31,43% Insufficient; for module 2: 45% – 21,43% – 33,57% respectively.

```
class_good(user_id) :-
  average_duration_acces(user_id,19),
  curriculum_name(user_id,ecdlprevenzioneinambientedilav),
  job_description(user_id,tecnico_della_prevenzione),
  number_access(user_id,41),
  initial_score_mod1sec1(user_id,25),
  final_score_mod1_sec1(user_id,100),
  initial_score_mod1_sec2(user_id,30),
  final_score_mod1_sec2(user_id,81),
  initial_score_mod2_sec1(user_id,35),
  final_score_mod2_sec1(user_id,48),
  initial_score_mod2_sec2(user_id,15),
  final_score_mod2_sec2(user_id,92),
  initial_score_mod3_sec1(user_id,45),
  final_score_mod3_sec1(user_id,56),
  initial_score_mod3_sec2(user_id,12),
  final_score_mod3_sec2(user_id,65), ...
```

**Fig. 2.** Example of student's description

The descriptions of the students, exploited as examples to induce classification rules, were made up of personal data of the students, such as background culture and current job (e.g., *curriculum_name* and *job_description*), and enriched by logs of their successive interactions with the e-learning platform, such as initial and final score that the student obtained for each section in each module, or number of accesses and duration of each access. Figure 2 shows a part of the example description for a student.

The goodness of profiles induced by the two approaches was evaluated according to a 10-fold cross-validation methodology, that is the 90% of the whole dataset was used for the training process and the remaining 10% for the testing phase. This process was made ten different times assuring that there was no intersections between the ten test sets. Several metrics were used in the testing phase and classification effectiveness has been measured in terms of the classical Information Retrieval notions – Precision (Pr) and Recall (Re) – and predictive accuracy [10].

More in detail, let the classes be $\{d_1 = \text{Good}, d_2 = \text{Sufficient}, d_3 = \text{Insufficient}\}$, for each value $d_i$, TP (True Positive) is the number of test users that both the system and the domain expert assigned to class $d_i$ in the selected experiment. TN (True Negative) is the number of users that both the system and the domain expert did not classify as $d_i$. FP (False Positive) is the number of test users that the system classified as $d_i$ in the selected experiment, differently from the domain expert classification (not $d_i$) in the same experiment. FN (False Negative) is the number of users that the system did not classify as $d_i$ while the domain expert classified them as $d_i$.

Then, Recall, Precision and Accuracy are computed as follows:

$$\text{Re} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{Pr} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TOT}}$$

where TOT indicates the total number of test users.
We also used F-measure, which is a combination of Precision and Recall:

$$\text{F} = \frac{2 \times \text{Re} \times \text{Pr}}{\text{Pr} + \text{Re}}$$

## 4.2 Discussion

For each class (Good, Sufficient and Insufficient), the systems were trained to infer proper classification rules, on the basis of an instance set representing different students (training set).

Figure 3 shows the classification rules describing the Good class of the Profile Extractor system (where the rule sets may be expressed as disjunctions of conditions) for the experiment set up on the first module, *Module 1 Fundamentals of Computer Science*, on the ground of logs containing interaction and student features, while Figure 4 shows an example of rule induced by INTHELEX for the same experiment.

The rule learned by INTHELEX for the class Good, shown in Figure 4, says, among other things, that a *UserA* is a good student for module 1 if he didn't obtain an high result (score) at the beginning of the first section (section 1) of such a module, but he has filled his gap before the end of the module; in fact, he has obtained a high result (score) in the final test of the last section (section 2) of such a module.

The rule learned by INTHELEX for the class Good, shown in Figure 4, says, among other things, that a *UserA* is a good student for module 1 if he didn't obtain an high result (score) at the beginning of the first section (section 1) of such a module, but he has filled his gap before the end of the module; in fact, he has obtained a high result (score) in the final test of the last section (section 2) of such a module.

```
The Rules Extracted for the Class GOOD are 5:

1. If NUMBER_ACCESS <= 15.0
   Then Class: no

2. If AVERAGE_DURATION_ACCESS > 15.0 And
      FINAL_SCORE_MODULE_1_SECTION_1        >        4.0
   Then Class: yes

3. If AVERAGE_DURATION_ACCESS <= 16.0
   Then Class: no

4. If INITIAL_SCORE_MODULE_1_SECTION_1 <= 21.0
   Then Class: yes

5. Otherwise Class: no
```

**Fig. 3.** An example of classification rules returned by the Profile Extractor system

```
class_good(UserA) :-
   high_final_score_module_1_section_2(UserA),
   low_final_score_module_2_section_1(UserA),
   low_final_score_module_2_section_2(UserA),
   low_final_score_module_3_section_3(UserA),
   low_final_score_module_3_section_5(UserA),
   curriculum_name(UserA, Type_curriculum),
   ecdl_scienze_biosanitarie(Type_curriculum),
   not(high_medium_initial_score_module_1_section_1(UserA)).
```

**Fig. 4.** An example of classification rules returned by the INTHELEX system

Table 1 reports the experimental results concerning the classification effectiveness for both the experiments: *Module 1 Fundamentals Computer Science* and *Module 2 Management Computer And File*.

Both the experiments concerned the two modules that the students must attend. As shown in Table 1, the performance of the Profile Extractor system is greater than that of INTHELEX for all the evaluated metrics. An in-deep analysis of such a result suggests that the motivation of this behaviour is located in the inborn nature of the two systems: the Profile Extractor was built to *extract profiles* from a set of data that have an attribute-value representation, whereas the INTHELEX learning capabilities are more suitable in discovering relationships among objects in the representation. Therefore, since the dataset representing the student features is a set of attribute-value couples, the performance of the Profile Extractor system is more effective. Nevertheless, while the rules induced by the Profile Extractor give poor information about student data, the rules discovered by INTHELEX provide a lot of details both on the student's personal information and on their learning capability.

**Table 1.** Results for the "*Module1*" experiment.

| | | | **Re** | **Pr** | **F-mea** | **Accuracy** |
|---|---|---|---|---|---|---|
| **Module 1** | **Good** | **Profile Extractor** | 0,97 | 0,97 | 0,96 | 0,97 |
| | | **INTHELEX** | 0.48 | 0.60 | 0.51 | 0.62 |
| | **Sufficient** | **Profile Extractor** | 0,97 | 0,97 | 0,96 | 0,97 |
| | | **INTHELEX** | 0.35 | 0.54 | 0.41 | 0.57 |
| | **Insufficient** | **Profile Extractor** | 0,91 | 0,92 | 0,90 | 0,94 |
| | | **INTHELEX** | 0.61 | 0.53 | 0.52 | 0.69 |
| **Module 2** | **Good** | **Profile Extractor** | 0,97 | 0,92 | 0,94 | 0,95 |
| | | **INTHELEX** | 0.39 | 0.52 | 0.42 | 0.54 |
| | **Sufficient** | **Profile Extractor** | 0,97 | 0,96 | 0,96 | 0,98 |
| | | **INTHELEX** | 0.43 | 0.61 | 0.41 | 0.78 |
| | **Insufficient** | **Profile Extractor** | 0,88 | 0,90 | 0,88 | 0,91 |
| | | **INTHELEX** | 0.67 | 0.64 | 0.60 | 0.74 |

Thus, a conclusion of this analysis is that a combination of both approaches could be effectively applied in the e-learning environment in order to adapt the platform, during the interaction, to the student personality, characteristics and learning performances. In fact, a co-operation of both methods could consist in, firstly, discover a profile with an high accuracy, by means of the Profile Extractor, in order to perform a phase of adaptation of the platform to a coarser grain-size level, and successively to fine-tune the personalization of the platform exploiting the more detailed profiles induced by INTHELEX.

## 5 Conclusion and Future Work

E-learning environments give users a high degree of freedom in following a preferred educational path, together with a control to explore effective paths. This freedom and control is beneficial for the students, resulting in a deeper understanding of the instructional material. Sometimes, this type of e-learning environment is problematic, since some students are not able to explore it effectively. One way to address this problem is to augment the environments with personalized support. Indeed, it is possible to adapt an e-learning environment planning a personalized path for each user-student, based on his needs, goals and characteristics, with the aim of improving the learning process.

Paper focused on student modelling, and presented two systems for automatically generating the profiles of an e-learning user. We have evaluated the effectiveness of both systems in such an environment and we have observed that they seem to be complementary for improving the personalization of the e-learning platform.
Future work concerns the exploitation of the induced profiles in order to plan a personalized educational path.

**Acknowledgement.** The authors would like to thank Professor T. Roselli for the CAMPUS ONE dataset used in this experimentation.

## References

1. Abbattista, F., Degemmis, M., Licchelli, O., Lops, P., Semeraro, G., Zambetta, F.: Improving the usability of an e-commerce web site through personalization. In: Ricci, F., Smyth, B. (Eds.): Recommendation and Personalization in Ecommerce. Proc. RPeC'02, Malaga, Spain (2002) 20-29
2. 2 Breuker, J. editor: EUROHELP: Developing Intelligent Help Systems. Conceptual model of intelligent help system. EC, Copenhagen (1990) 41-67
3. Brusilovsky, P., Eklund, J.: A Study of User Model Based Link Annotation. Educational Hypermedia. Journal of Universal Computer Science 4, 4 (1998) 429-448
4. Esposito, F., Ferilli, S., Fanizzi, N., Basile, T.M.A., Di Mauro, N.: Incremental Multistrategy Learning for Document Processing. Applied Artificial Intelligence Journal, 17(8/9), Taylor & Francis, London (2003) 859-883
5. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. Proc. of the 15th International Conference on Machine Learning. Morgan Kaufmann (1998) 144-151
6. Licchelli, O., Lops, P., Semeraro, G., Bordoni, L., Poggi, F.: Learning preferences of users accessing digital libraries. In: Cha, J., Jardim-Gonçalves, R., Steiger-Garção, A. (Eds.): Concurrent Engineering – Advanced design, production and management systems. Proceedings CE'03, Madeira, Portugal (2003) 457-465
7. Michalski, R. S.: Inferential Theory of Learning. Developing Foundations for Multistrategy Learning. In: Michalski, R. S. Tecuci, G. (Eds.): Machine Learning. A Multistrategy Approach, volume IV, Morgan Kaufmann, San Mateo, CA. 3–61
8. Paliouras, G., Papatheodorou, C., Karakaletsis, V., Spyropoulos, C., Malaveta, V.: Learning User Communities for Improving the Service of Information Providers. LNCS 1513, Springer (1998) 367-384
9. Quinlan, J.R. : C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA (1993)
10. Sebastiani, F.: Machine Learning in Automated Text Categorization. ACM Computing Surveys, 34 (1), (2002) 1–47
11. Semeraro, G., Esposito, F., Malerba, D., Fanizzi, N., Ferilli, S.: A logic framework for the incremental inductive synthesis of datalog theories. In: Fuchs, N. E. (Ed.): Logic Program Synthesis and Transformation, LNCS 1463, Springer-Verlag (1998) 300-321
12. Stern, M., Woolf, B.P., Kurose, J.F.: Intelligence on the Web? Proc. of the 8th World Conference of the AIED Society, Kobe, Giappone (1997).
13. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (2000).