LIA: An Intelligent Advisor for e-Learning

Nicola Capuano[1,2], Matteo Gaeta[1], Sergio Miranda[1], Francesco Orciuoli[1,2], and Pierluigi Ritrovato[1]

[1] University of Salerno, Department of Information Engineering and Applied Mathematics, via Ponte don Melillo, 84084 Fisciano (SA), Italy
[2] CRMPA, Centro di Ricerca in Matematica Pura ed Applicata, via Ponte don Melillo, 84084 Fisciano (SA), Italy
ncapuano@unisa.it, {gaeta,smiranda,orciuoli, ritrovato}@diima.unisa.it

Abstract. Intelligent e-learning systems have revolutionized online education by providing individualized and personalized instruction for each learner. Nevertheless, till now very few systems were able to leave academic labs and be integrated in real commercial products. One of this few exceptions is the Learning Intelligent Advisor (LIA) described in this paper, built on results coming from several research projects and currently integrated in a complete e-learning solution named IWT. The purpose of this paper is to describe how LIA works and how it cooperates with IWT in the provisioning of an individualized and personalized e-learning experience. Results of experimentations with real users coming from IWT customers are also presented and discussed in order to demonstrate the benefits of LIA as an add-on in on-line learning.

Keywords: e-Learning, ITS, Knowledge Representation, Planning.

1 Introduction

The Learning Intelligent Advisor (LIA) is an intelligent tutoring engine capable of integrating, in "traditional" e-learning systems, "intelligent" features like learner modelling and learning experience individualisation. LIA was born from the cooperation of an high-tech company named MoMA with the Research Centre in Pure and Applied Mathematics (CRMPA) and the Information Engineering and Applied Mathematics Dept. of the University of Salerno. It is currently included in a complete solution for e-learning named Intelligent Web Teacher (IWT) [1].

LIA is based on a set of models able to represent the main entities involved in the process of teaching/learning and on a set of methodologies, leveraging on such models, for the generation of individualised learning experiences with respect to learning objectives, pre-existing knowledge and learning preferences. Models and methodologies behind LIA integrate and extend results coming from several researches on knowledge representation and intelligent tutoring systems made by the authors, some of which partially founded by the European Commission.

M.D. Lytras et al. (Eds.): WSKS 2008, LNAI 5288, pp. 187–196, 2008.

A first prototype of an intelligent system for learning, based on conceptual graphs and software agents and able to generate individualised learning courses was proposed by authors in [2]. Defined models and methodologies were then improved introducing description logic for knowledge representation, planning techniques for learning path generation and machine learning techniques for learning preferences discovery and adaptation leading to a new prototype described in [3].

Obtained results convinced the authors to start the development of a complete system for learning (IWT) also involving a spin-off company (MoMA). The obtained system, providing a comprehensive set of "traditional" e-learning features and including a component (LIA) offering "intelligent" features coming from research was described in [4].

This first version of the system had several limitations like: limiting domain model offering only a small and pre-defined set of relations between concepts, impossibility to specify any teaching preference connected to domain concepts, imprecise and computationally expensive algorithms for course generation applying no optimisation techniques, impossibility to consider global optimisation parameters like the total cost or duration of a learning experience, impossibility to deal with resources different from learning objects (i.e. lack of support for learning services), impossibility to revise the evaluation of concepts once they are considered as known by the system, lack of support for pre-test.

To overcome these limitations, models and methodologies applied by LIA have been fully reorganised and, in some cases, completely re-thought. The purpose of this paper is to describe improved models (chapter 2) as well as related methodologies and how they are used during the whole learning life-cycle (chapter 3). Results of a small-scale experimentations with real users (chapter 4) are also presented.

2 LIA Models

The first step needed to describe the whole process of teaching/learning is to formally represent main involved actors and objects by means of appropriate models. The next paragraphs describe the four modes adopted by LIA while the next chapter shows how they are used in the teaching/learning process.

2.1 The Domain Model

The domain model describes the knowledge that is object of teaching through a set of concepts (representing the topics to be taught) and a set of relations between concepts (representing connections among topics). Such structure can be formally represented with a ***concepts graph*** G (C, R_1,… R_n) where C is the set of nodes representing domain concepts and each R_i is a set of arcs corresponding to the i-th kind of relation.

Two categories of relations are supported: *hierarchical* relations are used to factorise high-level concepts in low-level concepts while *ordering* relations are used to impose partial orderings in the concept set. Without loss of generality in this paper we consider a concept graph G (C, BT, IRB, SO) with three relations BT, IRB and SO whose meaning is explained below (where a and b are two concepts of C):

- *BT* (*a, b*) means that the concept *a belongs to b* i.e. *b* is understood iif every *a* so that *a* belongs to *b* is understood (hierarchical relation);
- *IRB* (*a, b*) means that the concept *a is required by b* i.e. a necessary condition to study *b* is to have understood *a* (ordering relation);
- *SO* (*a, b*) means that the *suggested order* between *a* and *b* is that *a* precedes *b* i.e. to favour learning, it is desirable to study *a* before *b* (ordering relation).

Any number of additional relations may be introduced provided that they belong to one of the two categories above. The figure 1 shows a sample domain model in the didactics of artificial intelligence exploiting the relations defined above and stating that to understand "logics" means to understand "formal systems", "propositional logic" and "first order logic" but, before approaching any of these topics it is necessary to have an "outline of set theory" first. Moreover, "formal systems" must be taught before both "propositional logics" and "first order logic" while it is desirable (but not compulsory) to teach "propositional logics" before "first order logic".

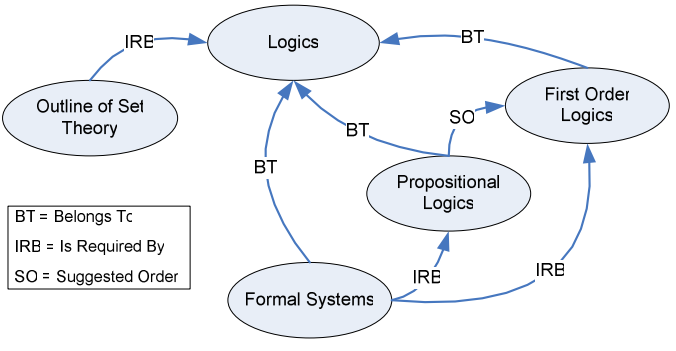

Fig. 1. A sample concepts graph

A set of ***teaching preferences*** may be added to the domain model to define feasible teaching strategies that may be applied for each available concept. Such preferences are represented as an application $TP\ (C \times Props \times PropVals) \rightarrow [0, 10]$ where *Props* is the set of didactical properties and *PropVals* is the set of feasible values for such properties. The table 1 provides some (non exhaustive) example of didactical property and associated feasible values. It is worth noting that *TP* is defined only for couples of *Props* and *PropVals* elements belonging to the same row in table 1.

Table 1. Example of didactical properties and feasible values

Properties	**Feasible values**
didactic method	deductive, inductive, etc.
activity type	text reading, video clip, simulation, discussion with a peer, discussion with the teacher, etc.
interactivity level	high, medium, low

2.2 The Learner Model

The learner is the main actor of the whole learning process and it is represented with a cognitive state and a set of learning preferences. The **cognitive state** represents the knowledge reached by a learner at a given time and it is represented as an application $CS\ (C) \rightarrow [0, 10]$ where C is the set of concepts of a given domain model. Given a concept c, $CS\ (c)$ indicates the degree of knowledge (or grade) reached by a given learner for c. If such grade is greater then a given "passing" threshold θ then c is considered as known, otherwise it is considered as unknown.

The ***learning preferences*** provide an evaluation of learning strategies that may be adopted for a given learner. They are represented as an application $LP\ (Props \times PropVals) \rightarrow [0, 10]$ where *Props* and *PropVals* are the same sets defined in 2.1. Differently from teaching preferences, learning preferences are not linked to a domain concept but refer to a specific learner.

The cognitive state of any learner is initially void (i.e. $CS\ (c) = 0$ for any c included in a given domain model) and may be initialized on a teaching domain with a pre-test. Learning preferences may be initialized by the teacher or directly by learners through a questionnaire capable of evaluating learners styles and transform them in suitable values for learning preferences. Both parts of the learner model are automatically updated during learning activities by a ***learner model updating algorithm*** (see 3.4).

2.3 The Learning Activity Model

A learning activity must be performed by a learner to acquire one or more domain concepts. According to IMS-LD [5], activities may relate to learning objects (e.g. textual lessons, presentations, videoclips, podcasts, simulations, exercises, etc.) or learning services (e.g. virtual labs, wikis, folksonomies, forums, etc.). The ***learning presentation generation algorithm*** (see 3.1) uses learning activities as bricks to generate learning experiences. In order to be effectively used in this way, a learning activity A is described through the following elements:

- a set of ***concepts*** C_A part of a given domain model, that are covered by in the learning activity;
- a set of ***didactical properties*** expressed as an application $DP_A\ (property) = value$ representing learning strategies applied by the learning activity;
- a set of ***cost properties*** expressed as an application $CP_A\ (property) = value$ that must be taken into account in the optimisation process connected with the *learning presentation generation algorithm.*

Didactical properties components have the same meaning with respect to teaching and learning preferences i.e. *property* and *value* may assume values from a closed vocabulary (see table 1). Differently from learning and teaching preferences, they are neither linked to a domain concept nor to a specific student but to a learning activity.

Cost properties are couples that may be optionally associated to learning activities, whose properties may assume values from the closed vocabulary {price, duration} and whose values are positive real numbers representing, respectively the eventual price of a single learning resource and its average duration in minutes.

Testing activities are learning activities used to verify the knowledge acquired by the learner. As learning activities, a testing activity T is connected to a set C_T of covered concepts indicating, in this case, the list of concepts verified by the activity. Once executed by a specific learner, they return an evaluation E_T belonging to the range [0, 10] indicating the degree of fulfilment of the test by that learner. Differently from other activities here DP_T and CP_T are not significant.

2.4 The Unit of Learning

An unit of learning represents a sequence of learning activities needed to a learner in order to understand a set of target concepts in a given domain with respect to a set of defined cost constraints. It is composed by the following elements:

- a set of ***target concepts*** *TC* part of a domain model, that have to be mastered by a given learner in order to successfully accomplish the unit of learning;
- a set of ***cost constraints*** *CC* (*property*) = *value* that must be taken into account in the optimisation process connected with the *learning presentation generation algorithm*;
- a ***learning path*** *LPath* (c_1, …, c_n) i.e. an ordered sequence of concepts that must be taught to a specific learner in order to let him master target concepts;
- a ***learning presentation*** *LPres* (a_1, …, a_m) i.e. an ordered sequence of learning activities that must be presented to a specific learner in order to let him/her master the target concepts.

While target concepts and cost constraints are defined by the course teacher, the learning path and the learning presentation are calculated and updated after each testing activity by specific generation algorithms (described in section 3). Concerning cost constraints, the *property* may assume values from the closed vocabulary {price, duration}. Feasible values are positive real numbers representing, respectively the maximum total price and the maximum total duration of the unit of learning.

3 The Learning Life-Cycle

After having seen how the main involved actors and objects are represented by means of appropriate models, it is necessary to see how LIA uses such models to automate some of the phases of the teaching/learning process. LIA sees the learning life-cycle as composed by five phases, each including activities that have to be performed by the actors involved in the process (namely the teacher and the learner) or by LIA itself.

In the **preparation phase** the teacher defines or selects a feasible domain model and prepares or selects a set of learning activities that may be used in the learning experience while learners may define their learning preferences.

In the **starting phase** the teacher initializes a unit of learning by setting target concepts and cost constraints and associates learners to it. Then LIA generates a personalised *learning path* for each learner through a *learning path generation algorithm* and introduces placeholders for testing activities (*milestones*) through a *milestone setting algorithm*.

In the **execution phase**, LIA selects a fragment of the learning path and generates the best learning presentation for each enrolled learner by applying the *learning presentation generation algorithm*. The learner then undertakes the learning and testing activities of the learning presentation until its end.

In the **evaluation phase**, when the learner ends a learning presentation fragment, his/her learner model is updated on the basis of the results of tests included in the fragment according to the *learner model updating algorithm* and a new execution phase starts by generating a new learning presentation fragment that will possibly include recovery activities for concepts that the student did not understand.

In the **closure phase**, once all concepts of the unit of learning are mastered by the learner, the system collects statistical information on the process that may be used by teachers to improve the domain model and/or learning and testing activities.

The following paragraphs describe in more details the algorithms exploited by LIA in the several phases of the learning life-cycle.

3.1 Learning Path Generation

The generation of the learning path is the first step to completely generate a unit of learning. Starting from a set of ***target concepts*** *TC* and from a ***domain model***, a feasible learning path must be generated taking into account the *concepts graph* $G(C, BT, IRB, SO)$ part of the domain model (with $TC \subseteq C$). The four steps of the learning path generation algorithm are summarizes below.

- The ***first step*** builds the graph $G'(C, BT, IRB', SO')$ by propagating ordering relations downward the hierarchical relation. IRB' and SO' are initially set to IRB and SO respectively and then modified by applying the following rule: for each arc $ab \in IRB' \cup SO'$ substitute it with arcs ac for all $c \in C$ such that there exist a path from c to b on the arcs from BT.
- The ***second step*** builds the graph $G''(C', R)$ where C' is the subset of C including all concept that must be taught according to TC i.e. C' is composed by all nodes of G' from which there is a ordered path in $BT \cup IRB'$ to concepts in TC (including target concepts themselves). R is initially set to $BT \cup IRB' \cup SO'$ but all arcs referring to concepts external to C' are removed.
- The ***third step*** finds a linear ordering of nodes of G'' by using depth-first search so by visiting the graph nodes along a path as deep as possible. The obtained list L will constitute a first approximation of the learning path.
- The ***fourth step*** generates the final learning path *LPath* by deleting from L all non-atomic concepts with respect to the graph G i.e. *LPath* will include any concept of L apart concepts b so that $ab \in BT$ for some a. This ensures that only leaf concepts will be part of *LPath*.

As an example we may consider the concept graph in figure 1 as G and the set {"First Order Logics"} as TC. The algorithm result on this input is:

$LPath$ = ("Outline of Set Theory", "Formal Systems", "First Order Logics").

3.2 Milestones Setting

Once the learning path is generated, it is necessary to insert in it placeholders for testing activities named ***milestones***. While concepts of the learning path will be

converted in learning activities different from tests by the presentation generation algorithm, milestones will be converted in testing activities by the same algorithm.

Milestones can be placed in the learning path directly by teachers or, alternatively, they can be placed basing on a list of percentages given by the teacher (e.g. the input list [0.2, 0.5, 0.7, 1] means that four milestones should be placed in the learning path approximately at 20%, 50%, 70% and at the end).

Each milestone covers all preceding concepts in the learning path until the beginning of the course apart concepts already known by the learner according to his/her cognitive state (i.e. any concept a so that $CS(a) \geq$ the "passing" threshold θ as defined in 2.2).

A milestone at 0% of the learning path is a special milestone meaning that a ***pre-test*** is necessary before entering in the unit of learning. Differently for other milestones, pre-test milestones may be of different kinds: a *pre-test on requirements* tests concepts of the learning path considered as known by the learner; a *pre-test on content* tests concepts of the learning path considered as unknown by the learner.

3.3 Learning Presentation Generation

The presentation generation algorithm is purposed to build a fragment of presentation, part of an unit of learning, suitable for a specific learner basing on a ***learning path*** *LPath'* that have to be covered, on a set of teaching preferences *TP* belonging to a ***domain model***, on a cognitive state *CS* and a set of learning preferences *LP* both part of the ***learner model*** associated to the target learner, on a set of optional ***cost constraints*** *CC* and on a set of available ***learning activities*** (including tests). The three steps of the presentation generation algorithm are summarizes below.

- The ***first step*** is to select the sub-list *L* of *LPath'* that have to be converted in a presentation. *L* is the sequence of all the concepts of *LPath'* not already known by the learner (i.e. any concept a so that $CS(a) < \theta$) from the beginning to the first milestone preceded by at least one concept not already known by the learner. If *L* is empty then the algorithm ends because the learner already knows all concepts of the learning path.
- The ***second step*** is to define the best sequence of learning activities *P*, selected from available learning activities (not including tests), covering *L* on the basis of *TP*, *LP* and *CC*. This requires the resolution of an optimisation problem that will be discussed later.
- The ***third step*** is to add testing activities at the end of *P* so obtaining the final learning presentation *Pres*. Testing activities are selected in order to cover all concepts of *L* without taking into account didactical and cost properties eventually linked to testing activities.

If *LPath* starts with a ***pre-test*** milestone, then *L* will be defined in order to include all concepts that should be tested according to the kind of pre-test milestone (as defined in 3.2), *P* is then settled to be void and only the third step is executed. Then the pre-test milestone is removed from *LPath*.

Let's give more details about the second step. Its purpose is to find the optimal set of learning activities *P* covering *L* on the basis of *TP*, *LP* and *CC*. First of all a measure of distance $d_{TP}(A, c)$ between an activity A and the set of preferences *TP* has to be

defined with respect to a concept c. In a similar way a measure of distance d_{LP} (A) basing on LP may be defined. A further measure d (A, c) shall be defined as a weighted sum of the two measures.

Once a feasible measure of distance is defined the problem of selecting the best set of activities P covering concepts of L becomes a *facility location problem* [6]. P must be built as the smallest set of activities covering all concepts of L with the minimum sum of distances between activities and covered concepts. To solve this problem it is possible to use a greedy algorithm to obtain a first feasible solution and, then, a local search algorithm to try to improve the initial solution. These algorithm are well know in the literature [6] so their description is out of the scope of this paper.

3.4 Learner Model Updating

For each testing activity T executed by the learner in the last learning presentation fragment, the test returns (as explained in 3.3) an evaluation E_T between 1 and 10 representing the degree of fulfilment of the test by the involved learner. For each concepts c belonging to C_T, the cognitive state of the learner is modified in this way: if CS (c) is not defined then CS (c) = E_T, otherwise CS (c) = (CS (c) + E_T) / 2. This is repeated for any executed testing activity T in $LPres$.

The evaluation of each concept is then propagated over required concepts in the concepts graph following backward the ordering relation IRB. Iterated failures to understand a set of concepts may in fact indicate a possible misconception in a common requirement. Such procedure helps to find these misconceptions and forces the learning presentation generation algorithm to introduce activities covering them in subsequent fragments by decreasing their grade in the learner cognitive state.

After a testing phase, learning preferences may be also modified in the following way: if the same learning activity A has been proposed n times to the learner (for a given n) without success (so bringing to a negative score in testing activities on related concepts), then the system decreases, for each didactical property DP_A (*property*) = *value* associated with A, learner preferences LP (*property*, *value*) of a given constant δ. Conversely learning activities that demonstrate to be successful (bringing to positive scores in testing activities) will increase learner preferences related to activities' didactical properties.

4 Experimental Results

As anticipated, LIA models and methodologies are integrated in a complete e-learning solution named Intelligent Web Teacher (IWT). IWT is currently used by about 30 Italian organisations including companies as well as University departments. IWT was also selected by the Italian Ministry of Education as the base technology for a project purposed to introduce e-learning in 550 Italian schools.

In such contexts a small-scale experimentation was performed to demonstrate the benefits of LIA as an add-on in on-line learning. Such experimentation involved a group of 28 voluntary learners belonging to 7 Small and Medium Enterprises dealing with vocational training on enterprise management. The group of learners was split in two separate sub-groups: a first sub-group composed of 20 learners was enabled to

use all IWT facilities except LIA while a second sub-group composed of 8 learners was enabled to access the whole IWT (including LIA).

All the voluntary learners were tested before and after a training phase on the same topics. In all the tests the learners' skills in the chosen domain were quantified using three ability ranges: low-level (0-3 scores), medium-level (4-7 scores) and high-level (8-10 scores). The figure 5 shows the performances of the two sub-groups. As it can be seen, the progress made by the second group of students is much sharper with respect to the first group.

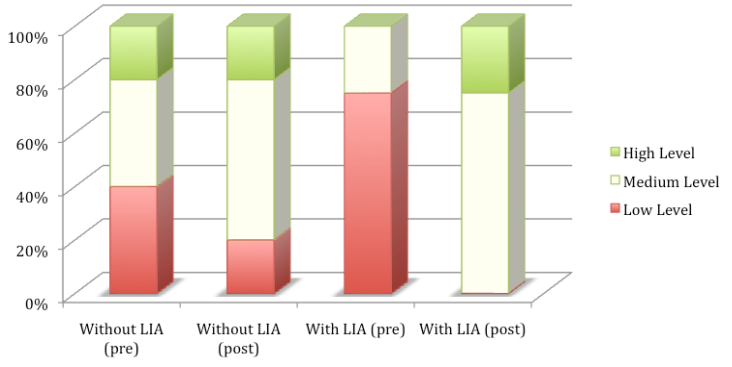

Fig. 2. Experimentation results on a group of 28 learners with and without LIA

5 Conclusions and Future Work

In this paper we have described an intelligent tutoring engine named LIA and how it cooperates with a complete system for e-learning named IWT in the provisioning of individualized and personalized e-learning experiences. Defined algorithms and underlying models have been described. Results of a small-scale experimentation with real users have been also presented and demonstrate the benefits of LIA as an add-on in on-line learning. A large-scale experimentation is still in course and results will be published in a future paper.

The approach carried out by LIA (together with IWT) has a strong linkage with the **Knowledge Society** perspective providing an effective mean for lifelong learning and knowledge spreading, capable of overcoming time and place barriers while adapting learning goals and strategies to virtually everyone starting point, context, perceptive capabilities and cognitive abilities.

Thanks to the innovative features provided by LIA, IWT is currently adopted by about 30 Italian organisations (with more then 40.000 users) and in November 2007 it won the prize "Best Practices for Innovation" from the General Confederation for Italian Industry. Despite that, the research on LIA is still active and further improvements are under study.

Among the other things, memetic algorithms for learning presentation generation are under study (preliminary results have been published in [7]) as well as algorithms and tools for the semi-automatic extraction of domain models starting from

pre-existing learning material. Further research work purposed to improve the support for didactic methods in the domain model [8] as well as the support for learning styles in the learner model [9] is in-progress. A visionary hypotheses of extension of LIA to fully support enterprise learning has been also recently proposed [10].

References

1. MoMA, s.r.l.: IWT Intelligent Web Teacher, http://www.momanet.it/english/iwt_eng.html
2. Capuano, N., Marsella, M., Salerno, S.: ABITS: An Agent Based Intelligent Tutoring System for Distance Learning. In: Proceedings of the International Workshop on Adaptive and Intelligent Web-Based Education Systems held in Conjunction with ITS 2000, Montreal, Canada, June 19-23, pp. 17–28 (2000)
3. Capuano, N., Gaeta, M., Micarelli, A., Sangineto, E.: An Integrated Architecture for Automatic Course Generation. In: Petrushin, R.V., Kommers, P., Kinshuk, Galeev, I. (eds.) Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT 2002), Kazan, Russia, September 9-12, pp. 322–326. IEEE Computer Society, Los Alamitos (2002)
4. Capuano, N., Gaeta, M., Micarelli, A.: IWT: Una Piattaforma Innovativa per la Didattica Intelligente su Web. AI*IA Notizie, year XVI (1), 57–61 (2003)
5. IMS Learning Design Specification, http://www.imsglobal.org/learningdesign
6. Shmoys, D., Tardos, E., Aardal, K.: Approximation Algorithms for Facility Location Problems. In: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, El Paso, Texas, United States, pp. 265–274 (1997)
7. Acanfora, G., Gaeta, M., Loia, V., Ritrovato, P., Salerno, S.: Optimizing Learning Path Selection through Memetic Algorithms. In: Proceedings of IEEE World Congress on Computational Intelligence, Hong Kong, June 1-6 (2008)
8. Albano, G., Gaeta, M., Ritrovato, P.: IWT: an innovative solution for AGS e-Learning model. Intl. Journal of Knowledge and Learning 3(2/3), 209–224 (2007)
9. Sangineto, E., Capuano, N., Gaeta, M., Micarelli, A.: Adaptive Course Generation through Learning Styles Representation. Universal Access in the Information Society International Journal 7(1/2), 1–23 (2008)
10. Capuano, N., Gaeta, M., Ritrovato, P., Salerno, S.: How to Integrate Technology Enhanced Learning with Business Process Management. Journal of Knowledge Management (to appear)