AUTHORING THE KNOWLEDGE OF INTELLIGENT TUTORING SYSTEMS

Simone Riccucci
University of Bologna
via Mura Anteo Zamboni, 40127, Bologna, Italy
riccucci@cs.unibo.it

ABSTRACT
The e-learning systems are becoming more and more important in Internet society. Many of the systems currently in use allow for a limited interaction with the user sometimes limiting itself only to the substitution of the traditional text books. In order to improve the interaction with these systems a knowledge base must be used in some way in order to adapt the system behavior to the user input. This kind of systems, called Intelligent Tutoring Systems (ITS), is difficult to construct because of the need for competences in knowledge engineering and artificial intelligence to implement the knowledge base on the domain of interest. In this paper we describe an on going wok about a method to partially automate the knowledge acquisition task of the ITS authoring process.

KEYWORDS
E-learning, Intelligent Tutoring System, Knowledge acquisition.

1. INTRODUCTION

Intelligent Tutoring Systems (ITSs) are very useful tools to support and enhance the learning process in many fields. This kind of systems includes the necessary information for a real simulation of teaching activity: nowadays, most of the systems used in learning support, are a slightly improvements of automated textbook. Furthermore, they do not embody any particular instructional approach, theory, or philosophy, other than the instructional approach that exist in the textbook on which the system is based.

ITSs can adapt their behavior on the base of the domain and student models, approaching the benefits of one-on-one instruction. Many ITSs have been proved highly effective. PAT Algebra Tutor [Koedinger and Anderson 1997] was developed for use in High-School setting and is based on the ACT theory [ACT]. The main purpose of the system is teaching to apply mathematics learned at school to real world problems. An experiment was conducted on 470 students using this system and the experimental classes outperformed students in comparison classes by 15% on standardized tests and 100% on tests targeting the ITS objectives (real life problems). LISP Tutor [Anderson and Reiser 1985], a system for tutoring on LISP programming language, enabled students to cover more exercises in the same amount of time compared to students that did not use the tutor, which subsequently gave them an advantage. SQL-Tutor [Mitrovic et al 2001] is a knowledge-based teaching system which supports students learning SQL. It is based on Ohlsson’s theory of learning from performance errors. In an evaluation studies SQL-Tutor has been proved to be effective in performance increasing to solve problems.

Despite these instruments have been demonstrated very useful in the learning process, they suffer from the disadvantage of being difficult to construct especially from people who do not have notions of knowledge engineering and/or of computer programming. In fact nearly all these systems need for the presence of an expert in ITSs design and construction assisted by the expert of the domain being implemented. However, many efforts have been made in order to facilitate the authoring process of the ITSs like WETAS-CAS [Suraweera et al 2004, Suraweera et al 2005] for the ITS based on constraint model and CTAT [Koedinger et al 2004] for the ITS based on ACT Cognitive Theory.

In this paper we define an algorithm for acquisition of knowledge in form of existentially quantified rules that can be used in a constraint based tutor. The process of knowledge acquisition is based on multi relational
data mining techniques. In the next section we describe the main ideas behind the algorithm then we show a simple run of a simple domain and finally we draw some conclusions.

2. KNOWLEDGE ACQUISITION

We formulate the problem of knowledge acquisition in the following way: *given a set of n problems having each m, alternate solutions, finds the set of exact rules the holds in the given examples*. The found rules are existentially quantified first order formula restricted to Datalog formalism (which is Prolog language restricted to function free Horn clauses). The context is described in [Riccucci et al 2005] where it is shown the general architecture for knowledge base acquisition which is made of three phases: ontology building, creation of a set of parser rules and construction of constraint base. We will focus on automating the construction of constraint base even if there are some methods that can be used to automate the other two tasks.

The problem of finding a general rule that must be satisfied in a domain, can be formulated as an Inductive Logic Programming (ILP) [Lavra, N. and Deroski, S., 1994] problem. In a general setting an ILP problem is formulated as follow:

Given a set of training positive and negative examples $E$ and background knowledge $B$, where $B$ is a set of rules or facts, find a hypothesis $H$, expressed in some concept description language $L$, such that $H$ is complete and consistent with respect to the background knowledge $B$ and the examples $E$.

In our case both the examples, background knowledge and hypothesis are expressed in Prolog language. The basic concepts related to data mining task performed in the context of our systems is described in [Dehaspe L. and De Raedt, 1997]. We can view the data mining task as a subclass of ILP general base formulation and use also first query mining to find suitable hypothesis that fits the given examples in the form of association rules.

In our setting teacher gives only positive examples as a set of solutions to the same problem. Such solutions are then processed by the Prolog representation producer module in order to obtain a suitable representation to be processed in an ILP setting. The background knowledge is represented by the ontology opportunely transformed in Prolog terms and rules and it is used to map the solution in Prolog terms.

This information are processed by another component called “ILP problem generator” that produces a set of data that can be directly processed by an external ILP engine that in our case is FARMER [Nijssen, S. and Kok, J.N., 2003] augmented with an algorithm to extract disjunctive association rules that we call RuleMinator. RuleMinator will produce a new set of rules representing the constraints related to the provided examples starting from the frequent queries found by FARMER. We extend the notion of **multi relational association rule** [Dehaspe and De Raedt 1997] with the more general **disjunctive multi relational association rule** having disjunction of conjunctions in its consequent. Such set of rules must be validated before it can be added to the constraints base. The whole process can be repeated until all necessary constraints are found. Currently, constraints base has to be generated from scratch each time the teacher adds new problem but in the future we plan to implement a mechanism to modify the added constraint.

Once the frequent association rules are found they must be transformed in a suitable form for being used in the ITS. Until now the produced rules have been generated starting from the examples given from the teacher but for being usable by the system they have to be generated taking into account that the given examples have to be considered in a new space defined by the cartesian product of the answers to the same problem. In this way each couple of solutions is treated as if contains respectively ideal solution and student solution obtaining rules that express in the premise characteristic pertaining to the teacher solution and in the consequent the facts that must hold in order for the rule to be satisfied.

Once the rules have been generated they have to be listed to the user in comprehensible way and ordered by their importance. Since the obtained rules are all exact rules (i.e. 100% confident rules) heuristics, other than confidence, are needed in order to decide how to order rules. Currently those used are based on the length of the rule defined on the number of atoms (the more the rule are simple the more are probable it is interesting), number of disjunctions in the body of the rule (the few are the number of disjunctions the more the rule are interesting), intersection of the support of the disjunctions.
3. SIMPLE RUN

We test the algorithm with a problem on simple domain to see if the rules founded by the system are plausible. Suppose we have a problem which allows the following solutions:

\[3 \times x - 4 = 2;\]
\[3 \times x - 2 = 4;\]
\[3 \times x = 4 + 2;\]

The representation of a solution in Prolog terms can be obtained starting from an ontology that contains the concepts involved in this domain such as “left hand side”, “right hand side”, “operator”, “integer”, “identifier”, etc... and relation correlating these concepts. For example a relation “assocOp” can relate “operator” with “integer”. If we represent a concept as a one parameter prolog term and a relation as a two parameter prolog term, we obtain the following prolog expression as a representation of the first equation:

\[\text{solution}(V2N0), \text{has}(V2N0,V2N1), \text{lhs}(V2N1), \text{has}(V2N0,V2N2), \text{rhs}(V2N2),\]
\[\text{has}(V2N1,V2N3), \text{times}(V2N3), \text{has}(V2N1,V2N4), \text{identifier}(V2N4),\]
\[\text{has}(V2N1,V2N5), \text{integer}(V2N5), \text{has}(V2N2,V2N6), \text{integer}(V2N6),\]
\[\text{hasValue}(V2N5,3), \text{assocOp}(V2N3,V2N4), \text{assocOp}(V2N5,V2N3),\]

After running the system on this simple set of solution it presents the following rules as the most important (we report only a few to show that system found a set of plausible rules):

RULE 1
IF IS solution(V2N0), has(V2N0,V2N1), lhs(V2N1) THEN SS solution(V2N0), has(V2N0,V2N1), lhs(V2N1)

RULE 2
IF IS solution(V2N0), has(V2N0,V2N1), lhs(V2N1),
has(V2N1,V2N2), integer(V2N2) THEN SS solution(V2N0), has(V2N0,V2N1), lhs(V2N1),
has(V2N1,V2N2), integer(V2N2)

RULE 3
IF IS solution(V2N0), has(V2N0,V2N1), lhs(V2N1),
has(V2N1,V2N2), times(V2N2), has(V2N1,V2N3), identifier(V2N3), assocOp(V2N2,V2N3) AND SS solution(V2N0), has(V2N0,V2N1), lhs(V2N1),
has(V2N1,V2N2), times(V2N2), has(V2N1,V2N3), identifier(V2N3),
THEN SS assocOp(V2N2,V2N3)

RULE 4
IF IS solution(V2N0), has(V2N0,V2N1), lhs(V2N1),
has(V2N1,V2N2), minus(V2N2) THEN SS solution(V2N0), has(V2N0,V2N1), lhs(V2N1),
has(V2N1,V2N2), minus(V2N2)
OR SS solution(V2N0), has(V2N0,V2N1), rhs(V2N1),
has(V2N1,V2N2), plus(V2N2)

RULE 5
IF IS solution(V2N0), has(V2N0,V2N1), rhs(V2N1),
has(V2N1,V2N2), integer(V2N2), hasValue(V2N2,X) THEN SS solution(V2N0), has(V2N0,V2N1), rhs(V2N1),
has(V2N1,V2N2), integer(V2N2), hasValue(V2N2,X)
OR SS solution(V2N0), has(V2N0,V2N1), lhs(V2N1),
Rules 1 states that if an ideal solution has a left hand side then the student solution must have a left hand side. Rules 2 state that if right hand side of equation has an integer then student solution must have an integer on the right hand side. The second rule is plausible, even if not always true in general, because it holds in all the given examples. Rule 3 states that if there are an integer and a times operator related by an “assocOp” relation in the ideal solution and there are also an integer and a times operator in the student solution they have to be related by “assocOp” relation too. Rule 4 states that if in the left hand side of equation there is a minus operator then the student solution must have a minus on the left side or a plus on the right side. The rule 5 states that if in the right side of the equation there is an integer with value X then the student solution must have an integer with the same value in one of the side.

4. CONCLUSION

We have presented a framework for knowledge acquisition in an Intelligent Tutoring System based on first order data mining that allows to acquire disjunctive multi-relational association rules. This method can be applied in constraint based tutor and with some modification to the initial representation to pseudo-cognitive tutor.

The main advantage respect to other framework is that this approach is based on a strong theoretical bases of first order data mining and can improve its efficacy as the research on this fields goes on. Once we have a representation of solution for an ILP engine, we do not need to develop an ad hoc mechanism in order to find the constraints rule. Currently we suppose that both a parser and ontology are given by someone else but there are some mechanisms that can partially automate the acquisition process of ontology and parser rules.

We plan to implement a tutor by means of this framework for “C” programming language and try it out in a first year academic course of computer science faculty.

REFERENCES

Act-r: a cognitive theory about human cognition [http://act-r.psy.cmu.edu/].