# Student Modelling and Classification Rules Learning for Educational Resource Prediction in a Multiagent System

Kennedy E. Ehimwenma, Martin Beer & Paul Crowther
Communication and Computing Research Institute
Department of Computing
Sheffield Hallam Univerisity, United Kingdom
Email: Kennedy.K.Ehimwenma@student.shu.ac.uk, M.Beer@shu.ac.uk, P.Crowther@shu.ac.uk

***Abstract*—To model support for human learning, rules (i.e. triggering_event-conditions-actions) can be classified to encompass any state of student learning activity enroute to appropriate learning material prediction. In an agent based system, each component of an adaptive multiagent system can be represented as agents having individual autonomy and responsibility to realise the overall goal of the system. In this paper, we present an extended work on a multiagent based Pre-assessment System in which a modelling agent employs the technique of *One v All Multiple Classification* rules to make predictions for learning materials after some prerequisite assessment facts to a desired concept or topic are communicated by the pre-assessment agent. Using SQL ontology tree structure as the domain of learning content, a learning algorithm is described as a process for estimating the total number of classified rules required for the pre-assessment system. This estimate is proven to be dependent on: 1) two binary state values, 2) the number of leaf-nodes in the ontology tree, and 3) the number of prerequisite concept(s) to a desired concept. In addition, is the learning algorithm with which a modelling agent can increment or decrement its classified number of rules.**

***Keywords:*** agent classification learning, classified rules, student modelling, multiagent, pre-assessment, ontology, SQL, semantics, Pre-assessment, artificial intelligence

## I. INTRODUCTION

One of the attributes of a good teacher is the ability to adapt to a specific student needs [4]. As tutors in a face-to-face classroom context may perform a pre-learning or diagnostic assessment concerning a particular knowledge concept before teaching a higher level concept, so should intelligent tutoring systems (ITS) be modelled to assist a student [7]. To adapt a system and model performances for improved learning, classification rules (using triggering_event-conditions-actions [1]) can be classified to encompass any state of a student learning activities enroute to appropriate learning prediction. In the multiagent based Pre-assessment System [7], the parameters to assist improved learning and making prediction for learning materials are communicated to the modelling and model agents after assessment by the designated pre-assessment agent to meet specific student needs — that might not be known to be needed by the student. This paper is a presentation of the use of classified production rules based on some given parameters in a modelling agent for learning predictions after the pre-assessment of relative knowledge to a desired concept. In addition, we present a learning algorithm in which a modelling agent can increment or decrement its classified production-rules given a set of learned parameters. This work is on the platform of Jason [1]. This paper continues with related works in Section I. In Section II, we present our student model and classification learning for agents. Section III is agent learning algorithm. Section IV discussions, and Section V is Conclusions and Further Work.

### A. Related Work: Student Modelling

Various developmental approaches have been used in the modelling of ITS and these approaches have influenced the views ascribed to student modelling or models. For instance, student modelling as described by [10] is the task of building dynamic models of student ability. And student model as the history of student interaction with a system that is structured in terms of knowledge elements and processed history [11], [4]. From both definition, a student model keep track-records of student cognitive and learning activity for the system or human tutor's support towards improved student learning experiences. Nonetheless, our perspective on student model is that of a dynamic representation of learning activities plus the ability, partial ability or inability of the student for the purpose of diagnosing the student state of readiness to learn a desired concept. According to Brusilovskiy [4], different types of student models exists which reflects the students state and level of knowledge, and they are: Scalar Model which estimates the level of knowledge by means of integral estimate using number ranging from 1 to 5; Overlay Model that assigns a Boolean estimate – 1 or 0 – to each element, indicating whether the student knows or does not know; the Buggy Model for defining the cause of incorrect student behaviour or Perturbation (the most studied form of Error) Model which assumes that one or several error elements and perturbations exists for each element of expert knowledge; then the Genetic Graph (or generic student) Model that reflects the genesis of student knowledge from simple to complex. The genetic graph nodes function as the elements of procedural knowledge (production rules) with relations representing the interaction between nodes, where such genetic graph can be treated as a



978-1-4673-9481-9/15/$31.00 ©2015 IEEE

development of purely overlay model and error model which provides the exact reflection of the student state of knowledge [4]. For humans to dispense an effective training, three types of special components are engaged: knowledge representation, student model and teaching strategy [4]. For ITSs, knowledge representation is the data structure of the subject matter content that machines can understand and communicate. The student model (in addition to its foregoing description) represents the student state and subject matter knowledge which are updated based on the analysis of the student's action and assessment; and the teaching strategy — the interface between the student and learning. For ITS to provide adaptive support to learning, it could have: i) a diagnostic function (exact determination of student state of knowledge), ii) a predictive function (to determine the probable response of the student to a teaching action), and iii) a function for assessing the performance of students, etc. [14],[4]. While these functions have been considered in the design of our Pre-assessment System, the model of the system has leaned towards the integration of both the Overlay and Genetic Graph Models which has been described in [8] with five cooperative agents, namely: agents agInterface, agModelling, agModel, agSupport and agMaterial.

## II. THE STUDENT MODEL

To model a student activity that is reflective of his ability or inability, the model agent named agent *student* was modelled to keep four parameter-information persistently about a given student in its (belief base) BB after pre-assessments by the learning support agent *agSupport*. To preserve the student assessed and processed information, the agent student uses the TextPersistentBB class. In a tuple M = <D, P, F, V> [8] where

**M**: is the model
**D**: a set of desired concepts i.e. start state
**P**: a set of passed pre-assessment i.e. positive ability (gains in learning)
**F**: a set of failed pre-assessment i.e. negative ability (gaps in learning)
**V**: the set of SQL statements

### A. Classification Learning for Agent

Supervised learning, decision trees, rule induction, data mining and AI approaches are the basis for intelligent systems development and deployment thus forming a common ground for researches on system application in both academics and commerce. These approaches are projected for systems to recognise concepts, learn features, take appropriate decisions and/or make predictions. In supervised learning, classification is a common technique used for systems to learn from data. Classification predicts the target of a categorical attribute (the class) based on pre-defined parameters, values or features (the prediction attributes) [12]. Inductive learning is learning from examples [2]. Inductive learning or rule induction is a system in which IF-THEN production rules are extracted from a set of observations [5]. Rules are simple and easily comprehensible way to represent [reasoning with] knowledge [12]. Classification as a method of categorising and making prediction has been used by several researches e.g. [3] who used the J48 Weka classification algorithms to survey undergraduate students' choices for continuing education, and [13] where classification was used to learn the density of agents in an environment based on the measurement of six proximity sensors. In this study, while the set of modelled information <D, P, F, V> formed the history of learning activity in which the tutor and knowledge engineers may need to assist students to improve learning and to make changes to the content of the system respectively, the set <D, P, F> becomes the set of parameters used as pre-conditions in the agent agModelling plans that are needed to be learned, satisfied and further used to classify a student for an appropriate learning route(s). The classified production rules models any student case of pre-assessment. This classification technique is a case-based reasoning process that is comparable in analogy to the Multiple Classification Ripple Down Rule (MCRDR) inference rules methodology for medical diagnoses [9], [6] where patient healthcare-cases were diagnosed. The classification rule format for agent agModelling to learn the dynamic user attributes after they are communicated from the agent agSuport is represented in Jason agent plan structure is given in [8] as:

```
+!recommend_material
    : set_of_profile_parameters
    <- recommended_material
```

where $+!recommend\_material$ is the message passed from a sending agent, $set\_of\_profile\_parameters$ are parameters being learned from every updated beliefs about the student; and $recommended\_material$ is the prediction for learning material in a message being sent to the ontology agent agMaterial to achieve.
To ensure that no case of individual student is left out or unclassified by a knowledge engineer for a system of large number of rules due to big hierarchy of learning concepts and leaf-nodes in an ontological learning structure, we now define and prove (from Section III) the learning formula for a determinant number of classified production rules based on a two possible binary state predicates—pass or fail—which is $T = 2$.

## III. AGENT LEARNING ALGORITHM ESTIMATION

According to [8], for the Pre-assessment System rules classification learning, let $C$ be the number of prerequisite concept(s) to a desired concept $D$, $T$ the two-state values for student pre-assessment and $N$ the equal number of leaf-nodes across each parent node, then the total number of classified production rules $R$ for a given ontology tree is determined by:

$$R = CT^N + 1$$

where

$$C = \{0, 1, 2, 3, ..., k\}$$

### A. Proof

To demonstrate the process of estimating the number of classified rules

$$R = CT^N + 1$$

for a class concept with-respect-to its prerequisite class concept(s) i.e. class concepts below its hierarchy in the ontology tree (Figure 1). See tables I, II, and III for a brief summary of this process with particular attention to the number of leaf-nodes $N$ and prerequisites $C$.

TABLE I. No.of RULES FOR ONE LEAF-NODE ONTOLOGY.

| C | T | N | R = $CT^N$ +1 | No. of classified rules |
|---|---|---|---|---|
| 0 | 2 | 1 | 1 | One |
| 1 | 2 | 1 | 3 | Three |
| 2 | 2 | 1 | 5 | Five |
| 3 | 2 | 1 | 7 | Seven |
| ... | 2 | 1 | ... | ... |
| K | 2 | 1 | ... | ... |

TABLE II. No. of RULES FOR TWO LEAF-NODE ONTOLOGY.

| C | T | N | R = $CT^N$ + 1 | No. of Classified Rules |
|---|---|---|---|---|
| 0 | 2 | 2 | 1 | One |
| 1 | 2 | 2 | 5 | Five |
| 2 | 2 | 2 | 9 | Nine |
| 3 | 2 | 2 | 13 | Thirteen |
| ... | 2 | 2 | ... | ... |
| k | 2 | 2 | ... | ... |

TABLE III. No. of RULES FOR THREE LEAF-NODE ONTOLOGY.

| C | T | N | R = $CT^N$ +1 | No. of classified rules |
|---|---|---|---|---|
| 0 | 2 | 3 | 1 | One |
| 1 | 2 | 3 | 9 | Nine |
| 2 | 2 | 3 | 17 | Seventeen |
| 3 | 2 | 3 | 25 | Twenty-five |
| ... | 2 | 3 | ... | ... |
| k | 2 | 3 | ... | ... |

### B. Learning Algorithm

In general, the above classified rule estimation process can be represented in algorithms (1 & 2 below) as a sequential process of learning for the agent. For any production-rule set (i.e. number of agent plans for some leaf-nodes equivalent to $T^N$) that would need to be added to the array of classified production rules, an agent can then *increment* the number of classified rules on the system; conversely, for any existing rule set whose concept is not accessed or desired by students, the agent through its *decremental* algorithm can remove such classified rules.

**Algorithm 1: Agent Incremental Rule Learning**

1. initialise $C \leftarrow 0$
2. initialise $N : 0 < N \geq 1$, where N is the no. of leaf-node per parent node
3. $T = 2$: two possible state predicate values of either pass or fail
4. $R = CT^N + 1$
5. $N = N + 1$
6. $R' = R + CT^{N-1}$
7. end

**Algorithm 2: Agent Decremental Rule Learning**

1. $T = 2$: two possible state predicate values of either pass or fail
2. get N: $0 < N \geq 1$, where N is the no. of leaf-node per parent node
3. get C: where $C \neq 0$
4. $R = CT^N + 1$, i.e. number of rules built in the system
5. $R' = R - [CT^N/2]$
6. $N = N - 1$
7. **IF** $N = 1$ **THEN** stop
8. end

## IV. DISCUSSIONS

The purpose of modelling student for adaptive or individualised learning is for an intelligent system and the course tutor to give optimum support for improved performances. Different modelling approaches and parameters are being used for developing ITS. In this multiagent based pre-assessment system, we have used parameters that keeps track of student SQL queries (both topics and answer statements) including the predicated—pass or fail—semantic evaluation literals which are further used in forming an array of classified rules that model students to appropriate learning materials. The following snippet of Jason code gives a picture of some classified rules that makes prediction for the multiple learning path. Thus, if $T^N = 2^2$ then the number of classified rules equals 4 for each class concept. The code is from the classifier agent—agModelling—plan library for pre-assessing the INSERT prerequisite when DELETE concept is received as the desired_Concept [8]:

```
/** Classified rules for assessing the prerequisite to DELETE concept
and semantic literal communication for learning material prediction */
...
@d1
+recommendMaterial[source(agSupport)] : desired_Concept("DELETE")
& passed("The student passed the INSERT with SELECT question")
& passed("The student passed the INSERT with VALUE question.")
<- .broadcast(tell, hasPrerequisite(insert, X)).
@d2
+recommendMaterial[source(agSupport)] : desired_Concept("DELETE")
& passed("The student passed the INSERT with SELECT question")
& failed("The student has NOT passed INSERT with VALUE question.")
< - .send(agMaterial, achieve, has_KB(X, insert value)).
@d3
+recommendMaterial[source(agSupport)] : desired_Concept("DELETE")
& failed("The student has NOT passed INSERT with SELECT question.")
& passed("The student passed the INSERT with VALUE question.")
< -.send(agMaterial, achieve, has_KB(X, insert select)).
@d4
+recommendMaterial[source(agSupport)] : desired_Concept("DELETE")
```

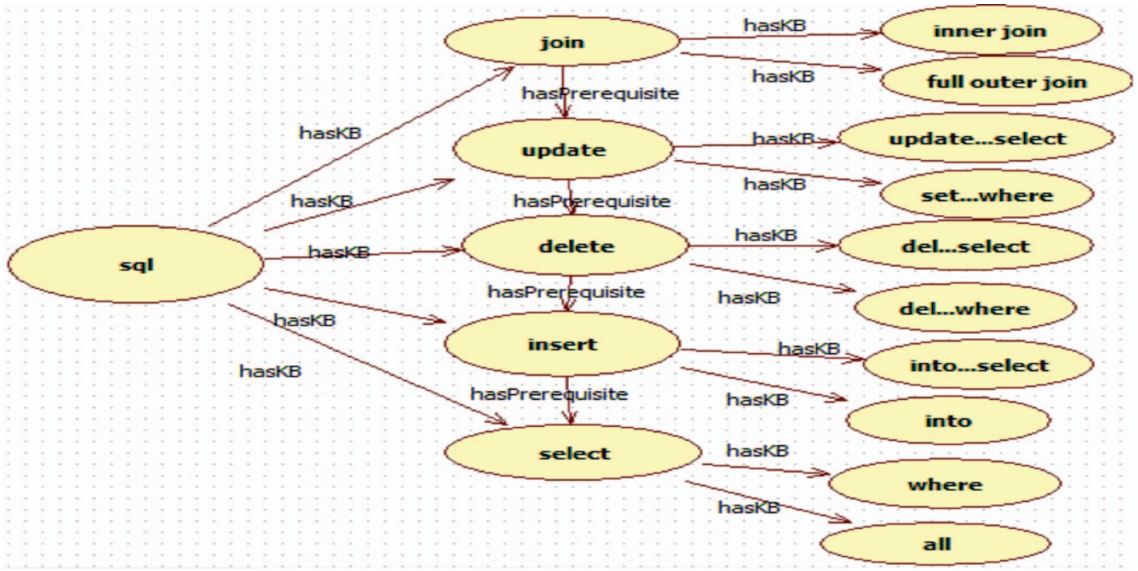


Fig 1. Class and Subclass Relationship in SQL Regular Ontology.

& failed("The student has NOT passed INSERT with SELECT question.")
& failed("The student has NOT passed INSERT with VALUE question.")
< -.send(agMaterial, achieve, hasPrerequisite(delete, X)).
...

As shown in the learning algorithm, the notation $T^N$ is a permutation series where $T = 2$ which is a constant of two possible state values and $N$ an integer value of the leaf-nodes to a parent class in the ontology tree in which $N$ could be 1, 2, 3, 4, 5, etc. as may be decided by the knowledge engineer for a given ontology. It is worthy of note to state that $N$ must be equal across all terminal nodes. As given in Figure 1, $N = 2$, and $N$ determines the number of questions for a class concept or topic. In this work, the leaf-node(s) are subconcept(s) upon which pre-assessment questions have been created. A question each for every leaf-node, and two leaf-nodes means two pre-assessment questions. So for the notation $2^N$, there are a total of $2^N$ category of classified rule that a student could fall within after pre-assessment about his desired_Concept. In Figure 1, there are six SQL concepts that a student may desire to learn (which are predicated as desired_Concept in this system), and two leaf-nodes each to all class concepts. For illustration, if SELECT is a student's desired_Concept; as the least concept in the hierarchy of learning (Fig. 1), it means that there will be no prerequisite and no pre-assessment as such. Therefore, $C = 0$ and $N = 0$ by implication: which conforms to the least step outlined in the Pre-asssessment Mechanism proposed in [7]. But in this ontology, $N$ cannot be zero as there is no blank node; because a student needs to be pre-assessed on every leaf-node. So, the number of rule required at this level of $C = 0$ and $N = 0$, is $R = 1$ — the DEFAULT rule: This we have shown for computational convenience as:

$$R = CT^N + 1.....(1)$$

$$R = 0 * 2^0 + 1$$

$$R = 1.....example(i)$$

So the student will receive the learning material URL of the SELECT concept. Suppose JOIN is the desired_Concept and JOIN has four prerequisite concepts, namely: UPDATE, DELETE, INSERT and SELECT (see Figure 1). The total number of classified rules under JOIN is calculated as:

$$R = CT^N + 1$$

$$R = 4 * 2^2 + 1$$

$$R = 17.....example(ii)$$

Thus a total of seventeen classified rules will be produced covering all the class concepts that are *prerequisited* under the JOIN concept. $R = CT^N + 1$ is therefore a production rule process for predicting deterministic number of classified rules for prerequisites with regards to a given topic or desired class concept. For the algorithms 1 and 2, the equation $R = CT^N + 1$ is first and foremost used as the *initialisation* factor to obtain the initial rule $R$ at the level where $C = 0$ — the least concept in the hierarchy that has no prerequisite nor pre-assessment questions. But for other higher class nodes, $R' = R + CT^{N-1}$ then becomes the iterative factor to determine the next total number of classified rules under a class concept depending on the leaf-nodes $N$ or prerequisite class concept(s) $C$ in the ontology tree. That is, from:

$$R = CT^N + 1$$

we get

$$R' = R + CT^{N-1}.....(2)$$

From example (ii) above where $R = 17$ for instance, the number of required rules for one additional leaf-node across the ontology will be $R = 33$, which could be determined either

by iterating equation (1) or applying the *incremental* algorithm in equation (2):

$$R' = R + CT^{N-1}$$

$$R = 17 + 4 * 2^{3-1}$$

$$R = 33.....example(iii)$$

Example (ii) implies that 17 classified rules are needed to be constructed or generated at the hierarchy of the UPDATE concept which is fourth in the class structure (Fig. 1), and example (iii) would accurately predict the needed rules R for an additional leaf-node; where $N-1$ in eqn (2) evaluates to give the immediate previous leaf-node state upon which the next number of rules R is dependent. TABLE I, II and III depicts the sequential build-up process. For the *decremental* learning algorithm, the reverse $R' = R - [CT^N/2]$ is the case to the calculation shown in equation (2) above.

To test for the scalability of the algorithms, we have the graph-plots of $C$ v $R$, and $N$ v $R$ in Figures 2 and 3, respectively using Python programming. Figure 2 shows the behavior of $R = CT^N + 1$ — the *Initialisation* formula when $N = 2$ as C increases sequentially. The Initialisation formula as tested also has the capability to function independently in predicting the number of rules $R$ for any regular ontology — ontology with equal number of leaf-nodes — when the required values are given. In Figure 3 we show the convergence of the *Initialisation* formula, *Incremental* and *Decremental* algorithm at data points: $C = 5$, $N = 5$, and $R = 161$.

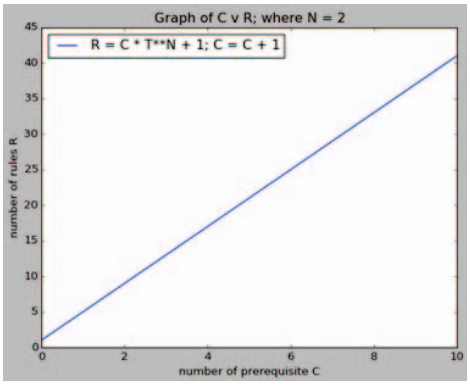


Fig 2. Perfomance test of the Initialisation algorithm.

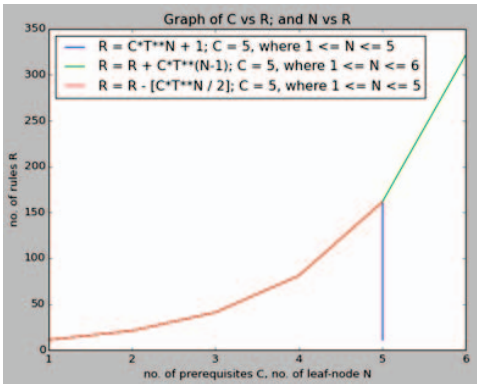


Fig 3. Convergence of all three algorithms at R = 161.

## V. CONCLUSIONS

In this paper we have described the development of student model in a multi-agent based system using Jason AgentSpeak. We uniquely adopted a set of parameters that includes a given student's desired_Concept and his measure of current knowledge status that are meant to: capture learning targets, learned and/or unlearned knowledge ever before the commencement of learning. This is to unravel the area of technical difficulties. In furtherance, we showed how the given parameters are used in building knowledge base (KB) rules and estimating the total number of classified rules to a given class concept; so that no case of classification is left out by the knowledge engineer. Then in the classified rules code, we in-turn showed how the prediction for learning material are made through semantical literal calls or messages to the ontology agent. To automate the process of self-maintenance, we have also proved and tested the algorithms for incremental and decremental rule estimation process. The aim is for the MAS to learn its operation of classified rules, and be able to make amends to its classified KB rules.

### A. Further Work

The next stage of this work is to automate the process of classified rule estimation through the use of the proposed algorithms for the MAS to self-learn its operation: to add to the classified KB rules or remove un-used KB rules based on the algorithms given some constraints.

## REFERENCES

[1] Bordini, R. H., Hubner, J. F., & Wooldridge, M. (2007). Programming multi-agent systems in AgentSpeak using Jason (Vol. 8). John Wiley & Sons.

[2] Bratko, I. (2001). Prolog programming for artificial intelligence. Pearson education.

[3] Bresfelean, V. P. (2007, June). Analysis and predictions on students' behavior using decision trees in Weka environment. In Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on (pp. 51-56). IEEE.

[4] Brusilovskiy, P. L. (1994). The construction and application of student models in intelligent tutoring systems. Journal of computer and systems sciences international, 32(1), 70-89.

[5] Calvo-Flores, M. D., Galindo, E. G., Jimnez, M. P., & Pineiro, O. P. (2006). Predicting students marks from Moodle logs using neural network models. Current Developments in Technology-Assisted Education, 1, 586-590.

[6] Compton, P., Peters, L., Edwards, G., & Lavers, T. G. (2006). Experience with ripple-down rules. Knowledge-Based Systems, 19(5), 356-362.

[7] Ehimwenma, K., Beer, M., & Crowther, P. (2014, July). Pre-assessment and Learning Recommendation Mechanism for a Multi-agent System. In Advanced Learning Technologies (ICALT), 2014 IEEE 14th International Conference on (pp. 122-123). IEEE.

[8] Ehimwenma, K. E., Beer, M., & Crowther, P. (2015). Adaptive Multi-agent System for Learning Gap Identification Through Semantic Communication and Classified Rules Learning. 7th International Conference on Computer Supported Education. In Doctoral Consortium (CSEDU), pp. 33-38. SCITEPRESS.

[9] Kang, B., Compton, P. and Preston, P. (1995). Multiple classification ripple down rules: evaluation and possibilities. The 9th knowledge acquisition for knowledge based systems workshop.

[10] Katz, S., Lesgold, A., Eggan, G., & Gordin, M. (1994). Modeling the student in Sherlock II. In Student Modelling: The Key to Individualized Knowledge-Based Instruction (pp. 99-125). Springer Berlin Heidelberg.

[11] Laurillard, D. (1988). The pedagogical limitations of generative student models. Instructional Science, 17(3), 235-250.

[12] Romero, C., Espejo, P. G., Zafra, A., Romero, J. R., & Ventura, S. (2013). Web usage mining for predicting final marks of students that use Moodle courses. Computer Applications in Engineering Education, 21(1), 135-146.
[13] Salem, Z., & Schmickl, T. (2014). The efficiency of the RULES-4 classification learning algorithm in predicting the density of agents. Cogent Engineering, 1(1), 986262.
[14] Self, J. A. (1994). Formal approaches to student modelling. In Student modelling: The key to individualized knowledge-based instruction (pp. 295-352). Springer Berlin Heidelberg.

978-1-4673-9481-9/15/$31.00 ©2015 IEEE