I.J. Information Technology and Computer Science, **2018, 9, 1-12**
Published Online September 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2018.09.01

Formalizing Logic Based Rules for Skills Classification and Recommendation of Learning Materials

Kennedy E Ehimwenma, Paul Crowther and Martin Beer
Sheffield Hallam University, Department of Computing, Sheffield, S1 1WB, United Kingdom
E-mail: letters4ken@gmail.com, P.Crowther@shu.ac.uk, mdb.shu@gmail.com

Received: 14 September 2017; Accepted: 07 August 2018; Published: 08 September 2018

Abstract—First-order logic based data structure have knowledge representations in Prolog-like syntax. In an agent based system where beliefs or knowledge are in FOL ground fact notation, such representation can form the basis of agent beliefs and inter-agent communication. This paper presents a formal model of classification rules in first-order logic syntax. In the paper, we show how the conjunction of *boolean [Passed, Failed] decision predicates* are modelled as *Passed(N)* or *Failed(N)* formulas as well as their implementation as knowledge in agent oriented programming for the classification of students' skills and recommendation of learning materials. The paper emphasizes logic based contextual reasoning for accurate diagnosis of students' skills after a number of prior skills assessment. The essence is to ensure that students attain requisite skill competences before progressing to a higher level of learning.

Index Terms—First order logic, skills classification rules, recommender systems, multi-agent systems, pre-learning assessment decisions, formative, education.

I. Introduction

The problem of students' skills evaluation whether in formative or summative learning systems is not uncommon in literature. But what is not common is the diagnosis and classification of skills of open-ended answer entries into a system for the purpose of assessment and recommendation of materials requisite to the level of students. Several systems have been designed over time to assist in learning, teaching and assessments (LTA). Whilst some LTA support career path recommendation after assessments e.g. [1], others categorise students for learning via skills assessment that are based on multiple choice question technique or item rating e.g. [2, 1]. Unlike existing systems that employs rating items or multiple-choice test techniques, the Pre-assessment System of this study that is agent based, takes as inputs open-ended answer entries and classifies students' skills for learning materials. As with natural reasoning in which humans engages the use of acquired facts to make decisions; logic based assertions, precisely, first order logic (FOL) are strong and reliable knowledge base representation models for the act of reasoning in intelligent systems [3]. However, the main challenges according to [3] in their use of FOL model in context-aware application are: how to model, represent and classify context, and then how to reason about it.

This paper is a description of a system of FOL based rules for the diagnosis and classification of skills, and the recommendation of learning path. The paper supports the application of the educational principle of *chunking* [4, 5] in the diagnosis of students' skills for learning material recommendation in the learning of SQL: a subject that has been described as difficult. The structure of this paper continues with related work in Section II, and Section III SQL learning, teaching and assessment systems. Section IV presents the hypothesis of our logic based classification model, and the strategies for supporting students to successful learning. In Section V is the implementation of the logic based rules and the data gathered so far, and Section VI conclusions and further work.

II. Related Works

Classification is *feature*, *instance* or *attribute* learning. It is when features (inputs or training set) that are symbolised in a system have corresponding class labels (i.e. outputs) to predict. These features can be continuous, categorical or boolean [6]. Classification consists of taking input vectors or data and deciding which *N* classes they belong after running them through a classifier(s) [7, 8]. Classification, involves one form of reasoning over some tasks. Reference [9] describes classification as a data mining technique that maps data input into predefined groups or classes. That the technique is a supervised learning approach which requires a set of training data to generate rules for classifying the test (unknown) data into predetermined groups or classes [9,10]. There are many models to classification, but choices are guided by the kind of data or problem at hand. For instance, in [9], the ID3/C4.5 technique was demonstrated in the classification of students' grades or scores into a *pass* or *fail* nodes in a decision tree on the

platform of Weka. In [11] a Neural Net (NN) model was developed for CPGA prediction that could support students to plan their further semester study. The NN model has two tasks: the prediction task that predicts student performances and a classification task that classifies students into groups. Case based reasoning (CBR) is also a type of classification technique that was combined with multi-agents in [12] to gather information about students and categorises them based on their knowledge level and learning preferences. CBR is a method in which concrete previous experience is applied to solve current and similar problem situations.

In contrast to CBR approaches where a current problem is interpreted as a previous one based on similarities or differences (classification CBR), or where a new solution is adapted based on the past, stored or existing solutions (problem CBR) [13]; the approach taken in this work is a model of FOL rule-based approach to reasoning by a *classifier* agent. This is where domain specific rules are specified as antecedents for a body of conclusions that is applied in a classification process [14, 7, 8]. This is because, we believe that the rule-based approach is more decisive to address the errors that are liable to be made by students in their responses to questions from the system that will in the end make accurate recommendation for their learning gap. In addition, because the answer inputs to the system are open ended, as such answers submitted by students to the pre-assessment system may also not be similar. While most classification systems are the CBR, decision trees and support vector machine; this paper considers an agent based classifier in students' learning. The act of classification in the study was not about the grouping of nodes in an ontology tree. But the collection of information about the skills status of students and recommendation of appropriate or a set of appropriate learning materials based on the available skills and related information to the system. The decision process in which students are categorised is through *condition-action* rules.

A. Condition-Action Rule

In a classification system, decision rules are the fundamental knowledge that are compared and matched with available information or known facts, and subsequently utilised by the system to perform the act of classification or conclusions. Rules of this nature have two component parts: the left-hand side known as the antecedent, condition, premise or situation, and the right-hand side part referred to as the consequent, action, conclusions, response, or prediction e.g. [14]. This is the logical structure of a rule based system where a classification system is given a reasoning task about some available knowledge or concepts in order to draw conclusions about some incoming data. In [15] such methods can be used for learning *concepts*: In AI (artificial intelligence), a concept is treated as a formal definition or predicate. For most of these systems to work, [15] states that in a learning system the following assumptions (that we have elaborated) are valid:

- *Conditions which are basic predicates for testing a state must be specified in advance:* This is preparing rules that must be satisfied as pre-conditions for the system or a component of the system.
- *The predicates are the essential part of the language or formalism for task representation:* All the variables in the environment should be gathered for adequate representation in the system.
- *There must be something—set of rules—to learn:* For a system to make decisions, a set of rules must be specified according to the environment and variables in the problem.
- *The training set is clean or devoid of noisy relations:* In that case, the data used for preparing the rules for the system must be unambiguous to be suitable to match the incoming unknown data or information.
- *The training set should contain counter-examples:* All examples (or facts) that may be available to a system may not be similar. Some may be positive and others negative. Rules should be stated to cover both positive and negative facts.
- *Basic predicates can be partitioned into independent group:* Different variables that are related can be grouped in one rule.
- *Within each group, the predicates are mutually exclusive and cover all cases:* No case of classification must be missed. Otherwise, this would result in the misclassification of an object.

A rule based system have ***IF*** <*conditions*> ***THEN*** <*actions*> rules, where the set of <*conditions*> are needed to be matched and satisfied before the <*actions*> part is triggered. In a system that supports teaching and learning, performance observation <*conditions*> are therefore the prelude to determine learning path <*actions*>.

Over some given tasks in learning, recommender systems for formative assessment needs to have a suitable collection of predictable performances of student skills in order to suggest accurate learning path. With the boolean logic *passed(N)* or *failed(N)* predicates where *N* represents the decisions made on any leafnode after pre-assessment, our agent based system adapts to the changing knowledge levels of students in a specific domain of SQL using FOL rules.

III. SQL Learning, Teaching & Assessment Systems

A database is a repository of information organised in such a way that it can be accessed, managed and updated easily. A database is created, stored and maintained on a database management system (DBMS). SQL (Structured Query Language) is the dominant database language [16]. In [17] SQL is a formal declarative database programming language that comprise data manipulation keywords such as select, from, where, delete, insert, into, update, set, on, and join to mention a few. The skills in

SQL are challenging and students have many difficulties learning them [18].

In the perspective of [19] learning and mastering of these skills is a difficult process that requires considerable practice and effort on the part of students. One of the challenges faced by students is mapping a statement of problem given in natural language into the information that is required from the database in an appropriate SQL statement; this [19] further stated is not easy. Another difficulty is students' misunderstanding of the basic elements of SQL and first order logic and the relational data model in general [20]. Recent studies have also shown the factors affecting students' failure in programming. As one of the factors, [4] in their descriptive study identified inadequacy in the time dedicated to programming courses, that this consequently heralds to students' inability to match the degree of difficulty required in programming skills. Other factors attributed are lack of higher order thinking on the part of students and incomplete or inappropriate teaching processes [21], as well as the problem of selecting the correct teaching method or tool [22, 4].

To support students with the learning of SQL and determine individual students' SQL query formulation skills, systems such as the AssesSQL test software [19, 23] have been developed. Their research examined the difficulty faced in the assessment of students' SQL query skills, and encourage students to use structured query language as software professions. For assessment, the system present questions to student, expects students to enter query solution to the questions. The AssesSQL query content covers only the SELECT statements.

In the LEARN-SQL tool, [16] implemented a strategy that objectively allows the evaluation of the correctness of the solution to a question given by a student by providing automatic correction to queries by comparing the students' solution to all existing valid solutions in the system. The system, tests, feedback and grade students in their learning of SQL. The LEARN-SQL was developed and comprised statements such as the SELECT and UPDATE queries. This is from the backdrop of previously development SQL systems whose content only covered the SELECT statements [16]. There also exists a number of sites that provide tutorials to students on SQL learning. Examples are "w3schools.com/sql" [24], "Beginner SQL Tutorial" [25] and "SQLCourse.com" [26] that have lots of modules from which a student would make a choice in order to start learning; and the "SQLzoo.net" [27] that provide supports through multiple choice (objective type) quizzes. While these systems provide ability for students to run queries or take quizzes, they do not provide assistance for errors or incorrect queries. Besides, for students to learn some higher level skill, relative prerequisite ought to have learned. While [21] anchored their prerequisites on general problem solving skills and logical thinking, our prerequisite is based on previously learned knowledge or lower level concepts to a higher concept in the same subject of SQL. Thus our agent based Pre-assessment System, pre-assesses students, feedback to them, keep students' SQL queries to questions, timestamp every activity, and finally make recommendation for learning materials. The system carries out diagnosis on the learning gap of in students' learning between what wants to learn (called the *desired_Concept D*) and some prior learning (called *prerequisites C*). The SQL query modules covered in the system are SELECT, INSERT, DELETE, UPDATE, JOIN and Union.

A. Recommender Systems for E-Learning

From the preceding section, SQL is a difficulty subject that poses challenges to students' learning. Thus one perceived approach to tackling this difficulty is through recommendation for learning materials after some prior skills test and assessment by a formative assessment system. In that regards, there are already existing intelligent tutoring systems (ITS) that provides support for a given level of adaptability. Reference [28] states that such system must be able to present materials at a level of difficulty and detail suited to the state of knowledge of the student, and to do so, the system must know and follow the student's changing knowledge. This can be achieved by a set of carefully planned rules [15] where a set of outputs are provided for some given set of inputs. This amounts to integrating supervised classification technique into ITS development, aim at making accurate class predictions that suits an individual student's need and level of knowledge.

Recommender systems for adaptive learning propose and prescribe content and items that centres around the learning needs of students. This is quite different from recommender systems for buying products; this is because learning is an effort intensive task that requires more time and interaction on the part of students compared to commercial transactions [29]. Furthermore, learners rarely achieve a final end state based on the fact that there are levels in learning. Instead of buying a product and owning it, learners achieve different levels of competences that have various levels in different domains. In such situation, what is important is identifying the relevant learning goals and supporting learners to achieving them (p.6).

In the views of [30] adaptive or personalised learning tends to model learners' learning path, activities and educational resource. To this end, several e-learning recommender systems have been proposed. In [30] for instance, a standalone quasi-summative assessment model was proposed to boost instruction process and customisation of learning path. In the model, students are graded based on some learning activities using a model of equation, and the adaption on the students' preferences and effort spent on course. Should a learner fail an activity, it means the competence needed has not been completely acquired; and this could hinder further learning.

Reference [31] also proposed a recommender system that can recommend the most appropriate content for learning. The system architecture comprises four interactive modules, namely: i) data collection part that is based on users' profiles and interest; ii) information

processing unit for the learning model, user classification and content classification; iii) recommendation module; and iv) log file component for the recommended classes meant for use in future reclassification. The system matches users' interests with content categories and classify users according to e.g. content submitted, subjects, and item ratings, respectively.

Like the proposed recommender system in [31], several classification systems employ the use of multiple components with different functions in order to fulfil the task of classification or recommendation. Multi-components in a recommender system draws similarity with multiagents to solve a problem. However, the aforementioned proposed system is not the kind that would assess students' skills before making recommendation. This is also similar to the recommender system proposed in [2] in which the system supports users to tick through a set of checkboxes such as *Completed Courses* or *Not Completed Courses* so as to classify users whether they possess the requisite skills for a given job. Though the system is geared towards employability skills classification, it can also assist users in recognising their areas of skills limitation and then focus on the desirable skills. The system does not provide any form of skills assessment.

One other assessment and learning tool is the PAT Tutor [32] which is an ITS for teaching introductory algebra. In PAT, learning tasks and exercises are arranged in sections at different skills level as specified in a standard mathematics curriculum. When students demonstrate mastery of a section (by achieving a level of competence on all underlying skills), the PAT system promotes the student to a new section, which includes some new skills [32]. In this strategy, the student's knowledge is assessed before moving to a higher level. Which means that the system can ascertain that a set of competences have been achieved before promotion to other skills. But the strategy for assessment and promotion to a new level in the PAT system was not presented.

B. Logic Based Models

This is the model that uses symbolic representation for modelling agent behaviour and reasoning. It involves the definition of agent capability using logic based semantics for expression of: rules, reasoning, knowledge preferences to react to several alternative choices of actions, and retrieval of information for users' best interest [33]. Reference [34] asserted that logical formulas are used to represent agent beliefs, and from the deductions made from the logical formulas, agent behaviours are derived. That the deductions from the formulas are through a set of rules whose predicates or antecedents correspond to executable actions.

First order logic rules are used in capturing *context*. Where the term *context* implies the situation or any information that can be used to represent the state of an entity or individual [35]. Reference [36] ascertained that FOL for modelling context allows complex rules to be written and automated. As logic based models, FOL enables automated inductive and deductive reasoning to be done on contextual information, and their strong formalisation has led to the verification and validation of context models [3, 36].

Thus FOL has been used in modelling rules for applications within and outside the scope of learning, and career choices. Reference [36] presented a model of logic for context-aware application. The model described FOL rules for context information and the type of operation that is performed on the rules that involves parameters such as: people, location, and temperature. In the model *existential* $\exists$ (*for some or at-least one*) and *universal* $\forall$ (*for all*) quantifiers were used to indicate that the context which follows the quantifiers is true *for some* and true *for all*, respectively. The system takes query inputs and return answers such as the *location* predicate and its instantiated variables. Reference [37] proposed and implemented the KMCD (knowledge-based major choosing decision) – a decision system that provides support to students in choosing their majors (or specialisation). The KMCD system used first-order formulas in the modelling of the predefined set of attributes e.g. marks, course, student score in a course, and the major. In the circumstance that an incoming student's attributes are matched with the predefined set, some specialisation are suggested that is hoped would help students graduate successfully.

In FOL based representation systems, the knowledge that is required for diagnosis and classification must be expressed using the chosen language valid syntax and rules or conjunctions of rules before execution. In the following section IV we present the reasoning-based approach in which one of the agents in the multi-agent pre-assessment system classifies students for learning materials based on the decisions received about the student's assessment from the *sending* agent. This is after the agent based system have taken the students' *desired_Concept* (i.e. the student goal of learning), and pre-assessed the student on some prerequisite concepts.

IV. A Hypothesis for First Order Logic Based Rules

Firstly, let us begin with the popular syllogism:

1. *All men are mortal.*
2. *Socrates is a man.*
3. *Socrates is mortal.*

In first-order logic (FOL), the *premise 1* states mortal(men). In *premise 2*, the concept of man is instantiated as man ≡ socrates; and the *conclusion* 3 states mortal(socrates). This, completely translates to a valid logical statement or axiom:

$$\forall X\ mortal(X) \rightarrow mortal(Socrates) \qquad (1)$$

which is a deductive reasoning approach or top-down logic. Likewise, given the above analysis, we present a hypothesis in Fig. 1 for the purpose of deriving logic

based classification and recommendation rules for a classifier agent.

Any failed prerequisite leafnode is learned.

A prerequisite leafnode is failed.

Therefore, the leafnode is learned.

Otherwise the desired_Concept is learned.

Fig.1. Hypothesis for derivation of logic based rules

The logical axioms from the stated hypothesis (Fig. 1) are

$$\exists N\ failed(N) \rightarrow learn\big(failed(N)\big) \quad (2)$$

and

$$\forall N\ passed(N) \rightarrow learn(desired\_Concept). \quad (3)$$

The Fig. 2 presents an abstract domain of concepts in a subsumption relationship. In the Fig. 2, a set of elements N_x are subclassed by a set of classes C_k. Logically, we state this relationship as

$$\forall N\ \forall C \epsilon D\ node(N_x, C_{k-1}) \wedge \quad node(C_{k-1}, C_k) \rightarrow node(N_x, C_k) \quad (4)$$

where C_{k-1} represents a *prerequisite* class node and C_k a *higher* class concept such that $C \in D$, and literally $D \equiv C$ where *D* is a desired learning concept of a student. In (4), the transitivity of the property (or relation) is also depicted.

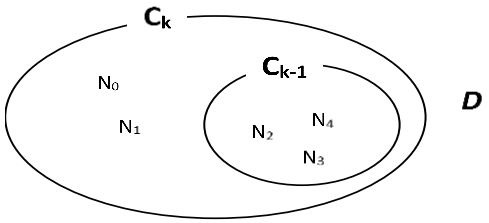

Fig.2. The set of concept subsumption: $D \equiv C \epsilon \{C_k\}$ and $N \sqsubseteq C$

Now for our analysis, let C_k be the *desired_Concept D* with the prerequisite C_{k-1}. From the hypothesis, two logical axioms can be derived with the *existential* ∃ and *universal* ∀ quantifiers:

$$\exists N\ failed(N) \rightarrow learn(failed(C_{k-1}.\{N\})) \quad (5)$$

which states that for some leafnode *N*, if the leafnode *N* is *failed* then that leafnode *N* is learned; or that,

$$\forall N\ passed(N) \rightarrow learn(C_k.\{N\}) \quad (6)$$

which states, for all leafnodes *N*, if all the leafnodes *N* are *passed*, then the *desired_Concept D* is learned. But our greatest interest is to fill-in the gaps in learning. So for every *failed* concept, the conclusion of the rule must prove that that concept is learned. As stated earlier, *D* is any parent class and $D \equiv C$ which is entered at the system interface by a student, and *N* the leafnode(s) of a *prerequisite C* parent class and *C* is a prerequisite to a *desired_Concept D*. As such $D \equiv C \sqsupseteq N$. Given the boolean predicates *[Passed, Failed]*, we represent any *passed leafnode N* in FOL formula as $\exists N\ passed(N)$ and any *failed leafnode N* as $\exists N\ failed(N)$, respectively; where the quantifier $\exists$ is a unary operator and *N* the childnode(s) or instance(s) of *C*.

A. The Student Model

In [38] a s*tudent model* involves the method used in representing the knowledge of students. As given in [39], modelling a system for learning purposes involves the use of interactive components and attributes of the learner (i.e, the student). Software agents are designed to observe their environment. The environment that was observed in this work are not the natural environment. Rather a student environment which is part of a software system [40]. Reference [41] called this environment a *partially observable* environment. In this work, for agents to observe the student environment, the environment needs to be modelled with the parameters that can elicit and represent the inherent knowledge attributes of students with regards to identifying gaps in their learning. The student model as specified and detailed below extends the work of [42, 43] so as to formalise the rule based classification process. In a tuple, the model M = <D, C, P, F, V> where

- <D>: The *desired_Concept* is the set $D = \{c_0, c_1, \ldots, c_{k-1}, c_k\}$ of observable parent classes in an ontology tree.
- <C>: The set of *prerequisites* $C = \{C_1\}; C = \{C_1, C_2\}$; or $C = \{c_1, c_2, \ldots, c_{k-1}, c_k\}$ parent classes underneath a *desired_Concept D*. For instance, let C_0 be a *desired_Concept*, then any of the set *C* can be *prerequisites* to C_0, respectively.
- <P>: The set of *passed* predicates $P = \{p_1, p_2, \ldots, p_{k-1}, p_k\}$ over some leafnodes *N* of the *prerequisites C*, to a *desired_Concept D*. The first order logic (FOL) form is $p(N_x)$ where *x* in N_x corresponds to the number of individual leafnode *N* per C_k. The $p(N_x)$ formula symbolises knowledge gain after pre-assessment on leafnode N_x.
- <F>: The set of *failed* predicates $F = \{f_1, f_2, \ldots, f_{k-1}, f_k\}$ over some leafnodes *N* of the *prerequisites C* with respect to a *desired_Concept D*. In FOL formula this is given as $f(N_x)$ where *x* in N_x also corresponds to the number of individual leafnode *N* per C_k. The $f(N_x)$ formula symbolises knowledge gap after pre-assessment on leafnode N_x.
- <V>: The set of *observable inputs* e.g. SQL answer queries $V = \{v_1, v_2, \ldots, v_{k-1}, v_k\}$ from students over the leafnodes *N* of the *prerequisite C* to a *desired_Concept D*. For every correct answer input that is assessed, the atomic formula $p(N_x)$ as

the corresponding decision statement is taken and communicated; for every incorrect answer input, the corresponding predicate $f(N_x)$ decision statement is taken and communicated for appropriate classification.

From the model M, the following outlines the purpose of the modelled parameters in the Pre-assessment System:

- *To fetch and communicate observed percepts (inputs) from the environment:* Consider <*D*> or *desired_Concept* as any topic or concept a human tutor, for instance, wants to teach. The agent based Pre-assessment System, like the tutor wants to know whether students are prepared for <*D*>. Then the system pre-assesses students on some prerequisites <*C*>.
- *To construct classification rules for agent:* To classify students for appropriate learning material, the classifier agent *agModelling* gets messages from the pre-assessment agent *agSupport* with a *tell performative.* This messages are the decisions reached after every pre-assessment. The decisions statements that are communicated are logic based formulas with <*P*> and <*F*> as predicates. After aggregating the messages, the plan context that is matched in the agent *agModelling* is triggered, and the body of the plan (sequence of instruction) execute the recommended materials.

In our study with the agent based Pre-assessment System, the entering of a desired_concept $D \equiv C$ signifies the desired learning goal of students. But the question is: *has the student achieved the competency level of the prerequisites subsumed by D,* when $C \sqsubseteq D$?

B. Logic Based Rules Formalism for Classification

Previous studies have shown that there are challenges and difficulties in the learning of SQL [44, 45, 19, 18]. To ensure considerable practice as proposed by [19], a formative assessment system should pre-assess students' prior learning before the start of a new learning, and where there are gaps, recommend materials for more practice. Learning is an effort intensive process [29]. Thus in a subject like SQL programming, skills diagnosis for learning material recommendation should be organised to remove any form of fatigue or information overload on students. In that perspective, two strategies of pre-assessments are proposed: *i) Pre-assessment By Immediate Prerequisite Class* and *ii) Pre-assessment By Multiple Prerequisite Classes*.

Rules for Pre-assessment By Immediate Prerequisite Class: To derive the FOL rules for this strategy of pre-assessment, we present in Fig. 3 a *regular ontology* [43] tree structure of equal number of leafnodes (N_x) per parent class node (C_k). The tree is a directed graph that shows parent classes and their subclasses. It also illustrates the process of navigation between concepts. In this classification process, our strategy of *pre-assessment by immediate prerequisite class* is supported.

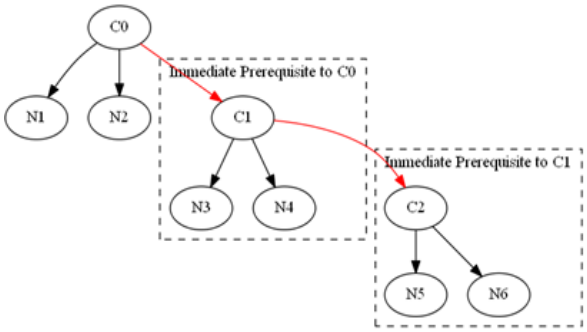

Fig.3. A digraph of a regular ontology tree.

Given that C_0 is a *desired_Concept*, a pre-assessment would be on the leafnodes N_3 and N_4; and for C_1 as a *desired_Concept*, pre-assessment would be on leafnodes N_5 and N_6. Firstly, let C_0 be the desired concept, and if leafnodes N_3 and N_4 are *passed*, the student learns the leafnodes N_1 and N_2 that are leafnodes (or childnodes) of the *desired_Concept*. Otherwise, the *failed* leafnodes N_3 or N_4 or both (N_3 *and* N_4) are learned in accordance with the hypothesis that is given in Fig. 1. Similarly, let C_1 be the *desired_Concept*, such that when leafnodes N_5 and N_6 are *passed*, the student learns the leafnodes N_3 and N_4 that are the leafnodes (or childnodes) of the *desired_Concept* C_1. Otherwise, the *failed* leafnodes N_5 or N_6 or both are learned. Thus, applying a conjunction of FOL formulas, we present (below) the logic based rules in (7) to (14) below for the classification of students' learning and recommendation of learning in the form of $F : X \rightarrow Y$:

$$\forall D(C_0)\ \forall(N_3)\ \forall(N_4)$$
$$[$$
$$: \exists D(C_0) \wedge \exists p(N_3) \wedge \exists f(N_4) \rightarrow f(N_4) \quad (7)$$
$$: \exists D(C_0) \wedge \exists f(N_3) \wedge \exists p(N_4) \rightarrow f(N_3) \quad (8)$$
$$: \exists D(C_0) \wedge \exists f(N_3) \wedge \exists f(N_4) \rightarrow f(N_3), f(N_4) \quad (9)$$
$$: \exists D(C_0) \wedge \exists p(N_3) \wedge \exists p(N_4) \rightarrow D(C_0).\{N_1, N_2\} \quad (10)$$
$$]$$
$$\forall D(C_1)\ \forall(N_5)\ \forall(N_6)$$
$$[$$
$$: \exists D(C_1) \wedge \exists p(N_5) \wedge \exists f(N_6) \rightarrow f(N_6) \quad (11)$$
$$: \exists D(C_1) \wedge \exists f(N_5) \wedge \exists p(N_6) \rightarrow f(N_5) \quad (12)$$
$$: \exists D(C_1) \wedge \exists f(N_5) \wedge \exists f(N_6) \rightarrow f(N_5, N_6) \quad (13)$$
$$: \exists D(C_1) \wedge \exists p(N_5) \wedge \exists p(N_6) \rightarrow D(C_1).\{N_3, N_4\} \quad (14)$$
$$].$$

The N_x in the $p(N_x)$ and $f(N_x)$ logic formulas are the decision statements about a student's performance on the ontology leafnodes after pre-assessment on that given node. The stated axioms are rules-based reasoning where each axiom represents a case or a category in the pre-

assessment of the leafnodes N_3 and N_4, and N_5 and N_6 respectively, before a student learns a desired concept. The rules which are 8 in number: i) defines the condition for the pre-assessment of immediate prerequisite leafnodes, and ii) presents the rule structure for a two leafnodes per parent class node in a *regular ontology* (Fig. 3). Each rule is a parameter combination of the <P> and <F> predicates in combination with the *desired_Concept* <D>. The <D> parameter represents the concept entered by a student which is also part of the conditions in the *classifier* agent plan context as implemented in Jason agent language [8]. From the foregoing, rule (7):

$$\forall D(C_0)\ \forall(N_3)\ \forall(N_4) : \exists D(C_0) \wedge \exists p(N_3) \wedge \exists f(N_4) \rightarrow f(N_4)$$

states that *for all* ∀ *desired_Concept* that is C_0, and *for all* leafnodes N_3 and N_4 of the immediate prerequisites C_1, such that, there *exists* ∃ in the agent beliefs the *desired_Concept* C_0 and there *exists* a *passed* pre-assessment of the leafnode N_3 and a *failed* pre-assessment of the leafnode N_4, then the conclusion and recommendation for learning shall be the failed leafnode N_4. While (10):

$$\forall D(C_0)\ \forall(N_3)\ \forall(N_4) : \exists D(C_0) \wedge \exists p(N_3) \wedge \exists p(N_4) \rightarrow D(C_0).\{N_1, N_2\}$$

states that *for all* ∀ *desired_Concept* that is C_0, *for all* leafnodes N_3 and N_4 of the prerequisite C_1, such that, there *exists* ∃ in the agent beliefs the *desired_Concept* C_0 and there *exists* some *passed* pre-assessments of the leafnode N_3 and the leafnode N_4, then the conclusion and recommendation for learning shall be the leafnodes N_1 and N_2 of the *desired_Concept* C_0 which is the intended concept of learning submitted by the student. This rule formation system also applies to the parent class node C_1 whose pre-assessment would be the leafnodes N_5 and N_6. While rules (7) to (9) and (11) to (13) corresponds to the axiom (5), (10) and (14) corresponds to the axiom (6). With regards to Fig. 3 tree structure, there are four rule axioms per parent class node *if and only if* the immediate class prerequisite to a desired concept is considered for pre-assessment. This type of learning strategy supports and implements *chunking* [46, 5] that prescribes the breaking down of skills and learning materials into smaller and more manageable units in order for students to succeed.

Rules for Pre-assessment By Multiple Prerequisite Classes: This is the second strategy where pre-assessment could extend from one prerequisite *C* to another prerequisite *C* under a *desired_Concept*. In this type of arrangement, the more the number of leafnodes *N* under a given *desired_Concept*, the more the complexity in the rule representation process. This complexity may extend to students in managing their learning gaps as well from having to deal with large amount of recommended URL materials. This is particularly so when there is a large amount of incorrect responses to pre-assessment quizzes. The Fig. 4 which is a *non-regular ontology* [47] illustrates the rule formation process using an ontology of five leafnodes *N*. In contrast to a regular ontology that has equal number of leafnodes N_x across all parent class nodes, a non-regular ontology is a tree with varying number of leafnodes across its parent class C_x node (Fig. 4).

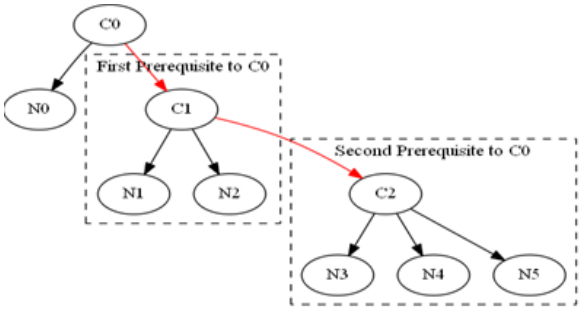

Fig.4. A digraph of non-regular ontology tree. A model where all the prerequisite classes under a given parent class, in this case C_0, are being considered for pre-assessment.

The parent classes in the tree are $C_0, C_1,$ and C_2. C_0 has a sub-parent class C_1 that has two leafnodes N_1 and N_2, and a sub-parent class C_2 that has three leafnodes N_3, N_4 and N_5. To consider all leafnodes N_1, N_2, N_3, N_4 and N_5 for pre-assessment under the parent class C_0 as the *desired_Concept*, the logic based rules for classification are given as follows in rules (15) to (46):

$$\forall D(C_0)\ \forall(N_1)\ \forall(N_2)\ \forall(N_3)\ \forall(N_4)\ \forall(N_5)$$
$$[$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists p(N_2) \wedge \exists p(N_3) \wedge \exists p(N_4) \wedge \exists p(N_5) \rightarrow D(C_0).\{N_0\} \quad (15)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists p(N_2) \wedge \exists p(N_3) \wedge \exists p(N_4) \wedge \exists f(N_5) \rightarrow f(N_5) \quad (16)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists p(N_2) \wedge \exists p(N_3) \wedge \exists f(N_4) \wedge \exists p(N_5) \rightarrow f(N_4) \quad (17)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists p(N_2) \wedge \exists f(N_3) \wedge \exists p(N_4) \wedge \exists p(N_5) \rightarrow f(N_3) \quad (18)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists f(N_2) \wedge \exists p(N_3) \wedge \exists p(N_4) \wedge \exists p(N_5) \rightarrow f(N_2) \quad (19)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists p(N_2) \wedge \exists p(N_3) \wedge \exists p(N_4) \wedge \exists p(N_5) \rightarrow f(N_1) \quad (20)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists p(N_2) \wedge \exists p(N_3) \wedge \exists f(N_4) \wedge \exists f(N_5) \rightarrow f(N_4), f(N_5) \quad (21)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists p(N_2) \wedge \exists f(N_3) \wedge \exists p(N_4) \wedge \exists f(N_5) \rightarrow f(N_3), f(N_5) \quad (22)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists p(N_2) \wedge \exists f(N_3) \wedge \exists f(N_4) \wedge \exists p(N_5) \rightarrow f(N_3), f(N_4) \quad (23)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists p(N_2) \wedge \exists f(N_3) \wedge \exists f(N_4) \wedge \exists f(N_5) \rightarrow f(N_3), f(N_4), f(N_5) \quad (24)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists p(N_2) \wedge \exists p(N_3) \wedge \exists p(N_4) \wedge \exists f(N_5) \rightarrow f(N_1), f(N_5) \quad (25)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists p(N_2) \wedge \exists p(N_3) \wedge \exists f(N_4) \wedge \exists p(N_5) \rightarrow f(N_1), f(N_4) \quad (26)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists p(N_2) \wedge \exists p(N_3) \wedge \exists f(N_4) \wedge \exists f(N_5) \rightarrow f(N_1), f(N_4), f(N_5) \quad (27)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists p(N_2) \wedge \exists f(N_3) \wedge \exists p(N_4) \wedge \exists p(N_5) \rightarrow f(N_1), f(N_3) \quad (28)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists p(N_2) \wedge \exists f(N_3) \wedge \exists p(N_4) \wedge \exists f(N_5) \rightarrow f(N_1), f(N_3), f(N_5) \quad (29)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists p(N_2) \wedge \exists f(N_3) \wedge \exists f(N_4) \wedge \exists p(N_5) \rightarrow f(N_1), f(N_3), f(N_4) \quad (30)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists p(N_2) \wedge \exists f(N_3) \wedge \exists f(N_4) \wedge \exists f(N_5) \rightarrow f(N_1), f(N_3), f(N_4), f(N_5) \quad (31)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists f(N_2) \wedge \exists p(N_3) \wedge \exists p(N_4) \wedge \exists f(N_5) \rightarrow f(N_2), f(N_5) \quad (32)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists f(N_2) \wedge \exists p(N_3) \wedge \exists f(N_4) \wedge \exists p(N_5) \rightarrow f(N_2), f(N_4) \quad (33)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists f(N_2) \wedge \exists p(N_3) \wedge \exists f(N_4) \wedge \exists f(N_5) \rightarrow f(N_2), f(N_4), f(N_5) \quad (34)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists f(N_2) \wedge \exists f(N_3) \wedge \exists p(N_4) \wedge \exists p(N_5) \rightarrow f(N_2), f(N_3) \quad (35)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists f(N_2) \wedge \exists f(N_3) \wedge \exists p(N_4) \wedge \exists f(N_5) \rightarrow f(N_2), f(N_3), f(N_5) \quad (36)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists f(N_2) \wedge \exists f(N_3) \wedge \exists f(N_4) \wedge \exists p(N_5) \rightarrow f(N_2), f(N_3), f(N_4) \quad (37)$$

$$: \exists D(C_0) \wedge \exists p(N_1) \wedge \exists f(N_2) \wedge \exists f(N_3) \wedge \exists f(N_4) \wedge \exists f(N_5) \rightarrow f(N_2), f(N_3), f(N_4), f(N_5) \quad (38)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists f(N_2) \wedge \exists p(N_3) \wedge \exists p(N_4) \wedge \exists p(N_5) \rightarrow f(N_1), f(N_2) \quad (39)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists f(N_2) \wedge \exists p(N_3) \wedge \exists p(N_4) \wedge \exists f(N_5) \rightarrow f(N_1), f(N_2), f(N_5) \quad (40)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists f(N_2) \wedge \exists p(N_3) \wedge \exists f(N_4) \wedge \exists p(N_5) \rightarrow f(N_1), f(N_2), f(N_4) \quad (41)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists f(N_2) \wedge \exists p(N_3) \wedge \exists f(N_4) \wedge \exists f(N_5) \rightarrow f(N_1), f(N_2), f(N_4), f(N_5) \quad (42)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists f(N_2) \wedge \exists f(N_3) \wedge \exists p(N_4) \wedge \exists p(N_5) \rightarrow f(N_1), f(N_2), f(N_3) \quad (43)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists f(N_2) \wedge \exists f(N_3) \wedge \exists p(N_4) \wedge \exists f(N_5) \rightarrow f(N_1), f(N_2), f(N_3), f(N_5) \quad (44)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists f(N_2) \wedge \exists f(N_3) \wedge \exists f(N_4) \wedge \exists p(N_5) \rightarrow f(N_1), f(N_2), f(N_3), f(N_4) \quad (45)$$

$$: \exists D(C_0) \wedge \exists f(N_1) \wedge \exists f(N_2) \wedge \exists f(N_3) \wedge \exists f(N_4) \wedge \exists f(N_5) \rightarrow f(N_1), f(N_2), f(N_3), f(N_4), f(N_5) \quad (46)$$

].

Note that, for the space limitation, the 32 rules from the 5 leafnodes *N* have been reduced to just 8. As per the five leafnodes N_1, N_2, N_3, N_4 and N_5, the number of classification rules to code for the *classifier* agent is 32 (i.e. rules 15 to 46) for all cases that must be accurately captured. As established in literature and from the preceding section, for a technical subject such as SQL considering a large number of leafnodes under a given *desired_Concept*, would presents large materials to students as shown from the rules generated. For illustration, (46) states that *for all* $\forall$ *desired_Concept* that is C_0, and *for all* leafnodes N_1, N_2, N_3, N_4 and N_5 of the prerequisite classes, such that, there *exists* $\exists$ in the agent beliefs the *desired_Concept* C_0 and there *exists* some *failed* pre-assessment of the leafnodes N_1, N_2, N_3, N_4 and N_5, then the conclusion and recommendation for learning shall be the leafnodes N_1, N_2, N_3, N_4 and N_5 underneath the *desired_Concept* C_0. For a 5 leafnode of pre-assessment, the number of rules can also be estimated beforehand with the equation proposed in [20]:

$$R = CT^N + 1. \quad (47)$$

Because all leafnodes *N* are prerequisites to the desired concept C_0 (in this illustration), then *C* in this equation takes a unit value of 1. Thus from (47), we have

$$R = 1 * 2 ** 5 + 1$$
$$R = 1 * 32 + 1$$
$$R = 32 + 1$$
$$R = 33$$

Note that, the constant *1* in (47) always depict the default rule for the first or simplest parent class node in an ontology. This class node, in the hierarchy of concepts is at the bottom of the ontology tree and has no prerequisites other than its leafnodes. At the point of submission of this class node by a student to the pre-assessment system, the associated learning material(s) to the leafnode(s) of this class node is presented to the student without pre-assessments.

Chunking [46, 5] would not support the strategy of *pre-assessment by multiple prerequisite classes* because the strategy may involve the linking or connection of a large amount of successive prerequisite class nodes *C*. Thus may result in task-overloading and fatigue on the part of

students. While the strategy of *pre-assessment by immediate prerequisite class* supports *chunking* with a bearable number of leafnodes *N* for students' pre-assessment and skills progression, it would allow students to stay on tasks, get classification results quickly and then focus on the technical details in their SQL query constructs. In fact, at the completion of a range of pre-assessment exercises on a set of skills in *chunks* and the subsequent learning of the materials recommended, a student can again choose another *desired_Concept D* for a different skills set learning. However, the number of leafnodes *N* to a parent class *C* is left to be determined by the knowledge engineer and the domain of interest.

C. Generating Classification Rules

As shown with the derived rules, each leafnode *N* has two boolean predicates [*passed* or *failed*] upon which a student is pre-assessed and classified. For a large number of leafnodes, say leafnode $N \geq 4$ under a *desired_Concept D*, the process of classification rules through parameter combination can be tedious in which errors in combination may lead to inaccurate classification or a miss-classification. Thus to combine the [*passed* or *failed*] parameters for accurate classification with respect to the leafnodes *N*, the Fig. 5 presents the algorithm for generating the classification rules.

Algorithm for Generating Classification Rules

1. Initialise T = [P, F] */** pass or fail boolean parameter */*
2. $1 \leq x \leq k$
3. While x != k
   a. $N \leftarrow N_1, \ldots, N_{x+1}$ */** number of leafnodes */*
   b. Initial_Rule = T * (N_x) */** leafnode(s) and parameter mapping */*
   c. Current_Rule ← Current_Rule * Initial_Rule */** rule formation */*
4. Output Current_Rule

Fig.5. Classification rules generation algorithm

In the algorithm, a number of leafnodes *N* under a *desired_Concept D* is given. Firstly, the first leafnode N_1 is mapped to the two boolean predicate <*P*> and <*F*> parameters (i.e. *passed* and *failed*): an operation that generates the first two rules. Subsequently, to obtain further rules (if there exists some leafnodes *N*), the outcome of the previous mapping is mapped to the outcome of a current mapping to produce the new classification rules. This process is graphically shown in Fig. 6.

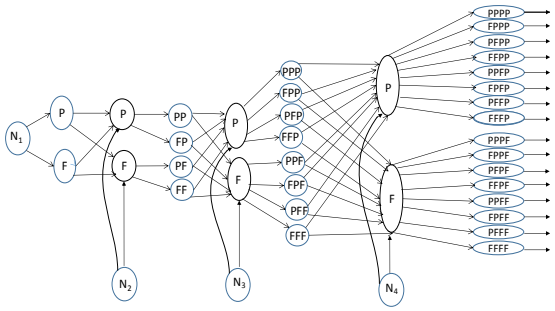

Fig.6. Classification rules formation process

V. Implementation

Applying the theory of *chunking* [46, 5], in Fig. 7 we presents an ontology of concepts in our SQL domain of interest in which the leafnodes *N* of every parent class are organised in smaller (chunk) units. The class concepts in the tree that has directed red-arrow lines are the SQL modules. The concepts are arranged in such a way that a lower concept is a prerequisite to a higher concept. In Fig. 7, the parent class *C* node and their respective leafnodes *N* have been designed to support both strategies of *prerequisite by immediate prerequisite class* and *prerequisite by multiple prerequisite classes* (with limited number of classes *C* and leafnodes *N*).

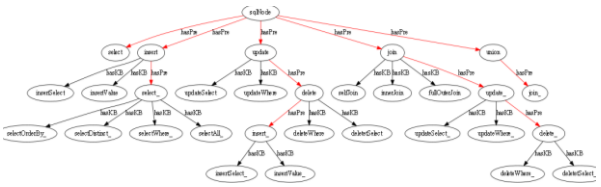

Fig.7. A non-regular ontology in the domain of SQL

To show a concrete implementation of the FOL classification and recommendation model, let us consider the Fig. 8 which is a cross-section of Fig. 7. In the Fig. 8, suppose Join is the *desired_Concept D* (with SelfJoin, FullOuterJoin and InnerJoin as its immediate leafnodes i.e. the unit of lessons). As shown, Join has 2 prerequisite classes *C* which are: Update and Delete, and Update and Delete, each, further has immediate leafnodes $N = 2$, namely: UpdateSelect and UpdateWhere; DeleteSelect and DeleteWhere, respectively. Therefore, the total prerequisite leafnodes to the Join *Desired_Concept is* $N = 4$.

When the Join class is entered by a student, the student is pre-assessed on the UpdateSelect, UpdateWhere, DeleteSelect and DeleteWhere. Given that leafnode $N = 4$, the number of classification rules $R = 16$ (excluding the default rule of 1). In Jason agent code, an example of implementation of 1 out of the 16 classification rules is presented in Fig. 9.

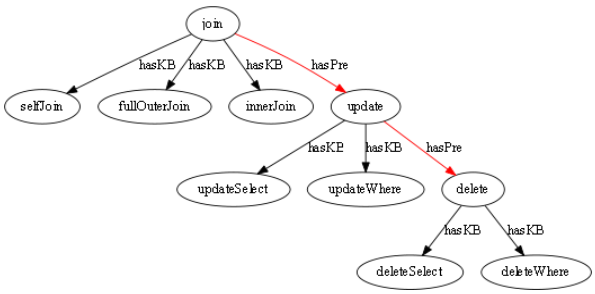

Fig.8. Semantic relations of a total of 4 prerequisite leafnode of 2 prerequisites parent classes under Join.

If all prerequisite leafnodes *N* are failed i.e. *failed(N)* as implemented in the *condition* part of Jason programming code (in agent plan), all of the *failed* leafnodes would be learned as contained in the *action* part of the plan. The corresponding logic based rule for the implemented plan in Fig. 9 is

$$\forall D(C_0)\ \forall(N_1)\ \forall(N_2)\ \forall(N_3)\ \forall(N_4)$$
$$[$$
$$: \exists D(C_0)\ \wedge\ \exists f(N_1) \wedge\ \exists f(N_2)\ \wedge\ \exists f(N_3)\ \wedge\ \exists f(N_4)$$
$$\rightarrow\ f(N_1), f(N_2), f(N_3), f(N_4)$$
$$].$$

The implementation in Fig. 9 is a prove of the axiom (5) that states *if some leafnode N is failed then that leafnode N is learned*. Conversely, all leafnodes *N* that are *passed* would have their logic formula as

$$\forall D(C_0)\ \forall(N_1)\ \forall(N_2)\ \forall(N_3)\ \forall(N_4)$$
$$[$$
$$: \exists D(C_0)\ \wedge\ \exists p(N_1) \wedge\ \exists p(N_2)\ \wedge\ \exists p(N_3)\ \wedge\ \exists p(N_4)$$
$$\rightarrow\ D(C_0).\{N_{x-2}, N_{x-1}, N_x\}$$
$$]$$

and the implementation in Jason [48] is as shown in Fig. 10 where the agent *agMaterial* recommends the 3 leafnode instances of the Join concept *(from Fig. 8).*

```
@joinRule16
+!recommendMaterial[source(agSupport)] :  desired_Concept("JOIN")[source(agSupport)]
    & failed("The student has NOT passed the UPDATE with SELECT question.")
    & failed("The student has NOT passed the UPDATE with WHERE question.")
    & failed("The student has NOT passed the DELETE with SELECT question.")
    & failed("The student has NOT passed the DELETE with WHERE question.")
    <- .send(agMaterial, achieve, has_KB(X, update_select));
       .send(agMaterial, achieve, has_KB(X, update_where));
       .send(agMaterial, achieve, has_KB(X, delete_select));
       .send(agMaterial, achieve, has_KB(X, delete_where)).
```

Fig.9. A classification and recommendation rule implementation.

The plan (sequence of instruction) (Fig. 9) of the agent *agModelling* also contained annotations such as *[source(agSupport)]* which specifies the sending agent (the source) of the decisions. Also, another agent as shown in the implementation is the agent *agMaterial;* this agent receives the classified learning materials that is being recommended.

```
@joinRule1
+!recommendMaterial[source(agSupport)] :  desired_Concept("JOIN")[source(agSupport)]
    & passed("The student has passed the UPDATE with SELECT question.")
    & passed("The student has passed the UPDATE with WHERE question.")
    & passed("The student has passed the DELETE with SELECT question.")
    & passed("The student has passed the DELETE with WHERE question.")
    <- .send(agMaterial, achieve, hasPrerequisite(join, X)).
```

Fig.10. The implementation of the classification and recommendation of all passed prerequisite leafnodes N.

By implementation, Fig. 10 also satisfies the axiom (6) above, which states that *if all prerequisite leafnodes N are passed then (the instance(s) of) the desired_Concept are learned.*

A. Data Collection and Visualization

Our system has been evaluated based on the *Strategy of Pre-assessment by Immediate Prerequisite Class* on the basis of a *regular ontology*. The data gathered from the system is as presented in Table 1 after students' use of the system in which learning materials were recommended based on their pre-assessed performances. The data shows the boolean *classification* (1 or 0 i.e. *passed* or *failed*) performance vs. *timespent* (on task by students). From the data, *time* length has not had any positive influence on students' performance on SQL query tasks. The boolean classification of students' performance also reveals that the students are thus faced with challenges in SQL. Furthermore, in Fig. 11 our data is visualized after splitting the data into training and test sets. This is towards the use of the gathered data to make future predictions for performance classification of skills.

Table 1. Time-Independent Variant Students' Performance Analysis

Time spent (mm.ss)	Boolean classification	Time spent (mm.ss)	Boolean classification
3.31	0	2.33	0
0.05	0	0.39	0
0.06	1	2.24	0
0.44	1	0.08	0
4.44	0	0.39	0
0.19	0	1.55	0
2.21	0	2.21	0
1.02	0	3.01	0
16.56	0	1.45	0
1.29	1	1.10	1
0.33	0	0.54	1

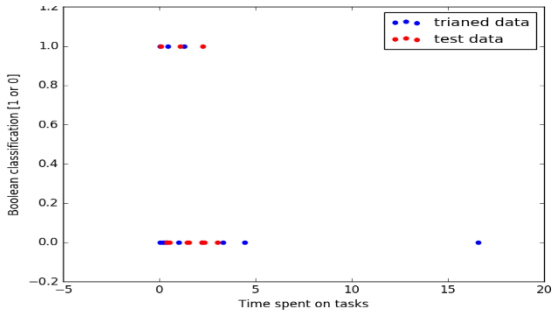

Fig.11. Time-Independent Variant Student Performance Analysis

VI. Conclusions & Further Work

This paper has presented a logic-based skills classification method that recommends learning materials to students after a set of pre-skill assessments in the domain of SQL. The position of the paper is that in a difficult subject like SQL, presentation of learning materials to students in *smaller units* called *chunking* would help students to overcome their challenges compared to a large amount of learning materials recommendation in one block. This is to avail students enough time and affordable learning space to master the materials recommended. As a result, the paper presented two kinds of pre-assessment strategies and demonstrated the classification and recommendation procedure using first order logic syntax. Then, the system implementation in Jason agent language. From the data obtained from the system, it is evident that the students are faced with challenges in their learning of SQL. The paper also presented a model of first order logic for validating the specification and categorization of skills attribute for multi-agent system decision making. This is based on

open-ended answer percepts that are entered by students as their response to quizzes. While the pre-assessment and classification of skills allows students to self-diagnose their skills status, the recommendation of materials ensures that students are engaged in requisite skills learning through considerable practices. This work has been carried out on the platform of a multi-agent system, but the aspect of inter-agent communication has been excluded from this presentation. The next stage of this work is to gather more data with the use of the pre-assessment system based on the two *strategies* already discussed. Then, in furtherance, compare the results between these *strategies*, and measure how accurate the predictions from the linear regression models conform with real time students' performance.

REFERENCES

[1] T. R. Razak, M. A. Hashim, N. M. Noor., I. H. A. Halim, and N. F. F. Shamsul, "Career path recommendation system for UiTM Perlis students using fuzzy logic," in IEEE *5th International Conference on Intelligent and Advanced Systems (ICIAS)*, 2014, pp. 1-5.

[2] D. Bañeres and J. Conesa, "A life-long learning recommender system to promote employability," *International Journal of Emerging Technologies in Learning*, vol. 12(6), 2017.

[3] G. J. Nalepa and S. Bobek, "Rule-based solution for context-aware reasoning on mobile devices," *Computer Science and Information Systems*, vol. 11(1), pp. 171-193, 2014.

[4] R, P. Bringula, A. D. Aviles, M. Y. C. Batalla and M. T. F. Borebor, "Factors affecting failing the programming skill examination of computing students," International Journal of Modern Education and Computer science (IJMECS), 5, 1-8, 2017, doi: 10.5815/ijmecs.2017.05.01.

[5] T. Anderson, "The theory and practice of online learning," *Athabasca University Press*, 2008. http://biblioteca.ucv.cl/site/colecciones/manuales_u/99Z_Anderson_2008-Theory_and_Practice_of_Online_Learning.pdf (accessed: 06.06.2017).

[6] S. B. Kotsiantis, I. Zaharakis and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging Artificial Intelligence Applications in Computer Engineering*, I. Maglogianis et al, Eds. IOS Press, 2007, pp. 3-24.

[7] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of machine learning research*, *5*(Jan), pp. 101-141, 2004.

[8] S. Marsland, "*Machine learning: an algorithmic perspective*," vol. 10, no. 1, CRC press, 2014.

[9] K. Adhatrao, A. Gaykar, A. Dhawan, R. Jha and V. Honrao, "Predicting students' performance using ID3 and C4. 5 classification algorithms," *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, Vol.3, No.5, September 2013, pp. 39-52, *arXiv preprint arXiv:1310.2071*.

[10] M. H. Dunham, "Data Mining: Introductory and Advanced Topics," *Pearson Education Inc., Indian*, 2003.

[11] N. Chanamarn & K. Tamee, "Enhancing Efficient Study Plan for Student with Machine Learning Techniques," *International Journal of Modern Education and Computer Science*, vol. 9(3), 1, 2017.

[12] C. González, J. C. Burguillo and M. Llamas, "Case-Based student modeling in multi-agent learning environment," in *Multi-Agent Systems and Applications IV*, Springer Berlin Heidelberg, 2005, pp. 72-81.

[13] R. L. de Mantaras, "Case-based reasoning," in *Machine Learning and Its Applications*, Springer Berlin Heidelberg, 2001, pp. 127-145.

[14] D. Patterson, "*Introduction to artificial intelligence and expert systems*," Prentice-Hall, Inc. 1990.

[15] A. Hutchinson, "*Algorithmic learning,"* Oxford University Press, Inc. 1994.

[16] A. Abelló, M. E. Rodríguez, T. Urpí, X. Burgués, M. J. Casany, C. Martín and C. Quer, "Learn-sql: automatic assessment of sql based on IMS QTI specification," in *Eighth IEEE International Conference on Advanced Learning Technologies ICALT'08*, pp. 592-593, 2008.

[17] C. Kenny and C, Pahl, "Automated tutoring for a database skills training environment," ACM, vol. 37, No. 1, pp. 58-62, 2005.

[18] A. Mitrovic, "Learning sql with a computerized tutor," in *SIGCSE'98*, 1998, pp. 307 – 311.

[19] J. C. Prior, "Online assessment of SQL query formulation skills," in *Proceedings of the fifth Australasian conference on Computing education*, Australian Computer Society, Inc vol. 20, pp. 247-256, January 2003.

[20] S. Dekeyser, M. de Raadt and T. Y. Lee, "Computer assisted assessment of SQL query skills," in *Proceedings of the eighteenth conference on Australasian database*, Australian Computer Society, Inc., Vol. 63, pp. 53-62, March 2007.

[21] Ö. Korkmaz, "The effects of scratch-based game activities on students' attitudes, self-efficacy and academic achievement," *International Journal of Modern Education and Computer Science (IJMECS)*, vol. *8*(1), 16, 2016.

[22] C. Rogerson and E. Scott, "The fear factor: how it affects students learning to program in a tertiary environment," *Journal of Information Technology Education*, vol. 9, 2010.

[23] J. C. Prior and R. Lister, "The backwash effect on SQL skills grading," *ACM SIGCSE Bulletin*, vol. 36(3), pp. 32-36, 2004.

[24] w3Schools.com, "SQL tutorial," *https://www.w3schools.com/sql/sql_intro.asp* (Accessed: 2nd June 2018).

[25] Beginner SQL Tutorial, "Learn sql programming," *http://beginner-sql-tutorial.com/sql.htm* (Accessed: 2nd July 2017).

[26] SQLCourse.com, "Interactive online sql training," *http://www.sqlcourse.com/table.html* (Accessed: 2nd June 2018)

[27] SQLzoo, "Sql zoo," *https://sqlzoo.net/wiki/SELECT_Quiz* (Accessed: 22nd May 2018)

[28] R. S. Michalski, J. G. Carbonell and T. M. Mitchell, "A comparative review of selected methods for learning from examples,"in "*Machine learning: An artificial intelligence approach"*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Eds. Springer Science & Business Media, pp. 4-81, 2013.

[29] N. Manouselis, H. Drachsler, R. Vuorikari, H. Hummel and R. Koper, "Recommender systems in technology enhanced learning," *Recommender systems handbook*, 2011, pp. 387-415.

[30] D. Bañeres, "A personalized summative model based on learner's effort," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 12(06), pp. 4-21, 2017.

[31] M. El Mabrouk, S. Gaou and M. K. Rtili, "Towards an intelligent hybrid recommendation system for e-learning platforms using data mining," *International Journal of*

Emerging Technologies in Learning (iJET), vol. 12(06), pp. 52-76, 2017.
[32] S. Ritter, J. Anderson, M. Cytrynowicz and O. Medvedeva, "Authoring content in the published pat algebra tutor," *Journal of Interactive Media in Education*, no. 2, 1988, https://jime.open.ac.uk/98/9.
[33] P. Dell'Acqua, F. Sadri, F. Toni and F. S. F. Toni, "*Communicating agents*," Citeseer, 1999.
[34] L. De Silva, "Planning in BDI agent systems," *PhD Thesis RMIT University*, Australia, 2009.
[35] A. K. Dey, "Providing architectural support for building context-aware applications," *Ph.D. thesis*, Atlanta, GA, USA, 2000.
[36] A. Ranganathan and R. H. Campbell, "An infrastructure for context-awareness based on first order logic," *Personal and Ubiquitous Computing*, vol. *7*(6), pp. 353-364, 2003.
[37] F. Yu, Q. Zhou, X. Lu and S. Zhao, "A first-order logic framework of major choosing decision making with an uncertain reasoning function," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016.
[38] P. T. Baffes, "Learning to model students: using theory refinement to detect misconceptions," *Technical Report, Artificial Intelligence*, University of Texas at Austin, 1994.
[39] I. Padayachee, "Intelligent tutoring systems: architecture and characteristics," in *Proceedings of the 32nd Annual SACLA Conference*, 2002, pp. 1-8.
[40] A. Ricci, M. Piunti and M. Viroli, "Environment programming in multi-agent systems: an artifact-based perspective," *Autonomous Agents and Multi-Agent Systems*, vol. 23(2), pp. 158-192, 2011.
[41] F. Wang, "POMDP framework for building an intelligent tutoring system," *Computer Supported Education (CSEDU2014)*, SCITEPRESS, pp. 233-240, 2014.
[42] K. E. Ehimwenma, M. Beer and P. Crowther, "Adaptive multiagent system for learning gap identification through semantic communication and classified rules learning," *7th International Conference on Computer Supported Education* (CSEDU), Doctoral Consortium, SCITEPRESS, pp. 33-38, 2015.
[43] K. E. Ehimwenma, M. Beer, and P. Crowther, "Student modelling and classification rules learning for educational resource prediction in a multiagent system," in *IEEE 7th Computer Science and Electronic Engineering Conference (CEEC), 2015*, pp. 59-64.
[44] X. Chen, A. Mitrovic and M. Mathews, "Does adaptive provision of learning activities improve learning in sql-tutor?," in *International Conference on Artificial Intelligence in Education*, Springer, Cham, June 2017, pp. 476-479.
[45] D. Lavbič, T. Matek and A. Zrnec, "Recommender system for learning SQL using hints," *Interactive Learning Environments*, vol. 25(8), pp. 1048-1064, 2017.
[46] C. Casteel, "Effects of chunked reading among learning disabled students: an experimental comparison of computer and traditional chunked passages," *Journal of Educational Technology Systems*, vol. 17(2), pp.115-21, 1988.
[47] K. E. Ehimwenma, P. Crowther and M. Beer, "A system of serial computation for classified rules prediction in non-regular ontology trees," *International Journal of Artificial Intelligence and Applications (IJAIA), vol.* 7(2), pp.21-33, 2016.
[48] R. H. Bordini, J. F. Hübner and M. Wooldridge, "*Programming multi-agent systems in AgentSpeak using Jason*, John Wiley & Sons, UK, Vol. 8, 2007.

Authors' Profiles

Kennedy E. Ehimwenma completed his PhD in multi-agent systems and knowledge representation for eLearning application development in 2017 at the Sheffield Hallam University, United Kingdom. His research interests include intelligent agents, semantic ontology and decision systems. Dr. Ehimwenma is a member of the BCS and the IEEE.

Paul Crowther has a PhD in Computer Science. He is the Deputy Head of the Department of Computing, Sheffield Hallam University, United Kingdom. His expertise is in database systems with research interest in knowledge base systems, agents, and mobile learning. Dr. Crowther is a Fellow of the British Computer Society (BCS) and SFHEA.

Martin Beer has a PhD in Computational Chemistry. Until his retirement in 2017, he was a Principle Lecturer in the Department of computing, Sheffield Hallam University, UK. His research interest includes multi-agents, semantic web technology, and mobile learning. Dr. Beer is a Fellow of the BCS.

How to cite this paper: Kennedy E Ehimwenma, Paul Crowther, Martin Beer, "Formalizing Logic Based Rules for Skills Classification and Recommendation of Learning Materials", International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.9, pp.1-12, 2018. DOI: 10.5815/ijitcs.2018.09.01

