The Intelligent Tutoring System AI-VT with Case-Based Reasoning and Real Time Recommender Models

Daniel Soto-Forero[1 [0000−0003−0753−4673]], Simha Ackermann[1 [0009−0006−0732−8326]], Marie-Laure Betbeder[1 [0000−0002−8103−4098]], and Julien Henriet[1 [0000−0002−7671−4574]]

DISC, Université de Franche-Comté, CNRS, institut FEMTO-ST, 16 Route de Gray, Besançon, F-25000, France.
{daniel.soto_forero, marie-laure.betbeder, julien.henriet}@univ-fcomte.fr, sloubaton@gmail.com

Abstract. This paper presents a recommendation model coupled on an existing CBR system model through a new modular architecture designed to integrate multiple services in a learning system called AI-VT (Artificial Intelligence Training System). The recommendation model provides a semi-automatic review of the CBR, two variants of the recommendation model have been implemented: deterministic and stochastic. The model has been tested with 1000 simulated learners, and compared with an original CBR system and BKT (Bayesian Knowledge Tracing) recommender system. The results show that the proposed model identifies learners' weaknesses correctly and revises the content of the ITS (Intelligent Tutoring System) better than the original ITS with CBR. Compared to BKT, the results at each level of complexity are variable, but overall the proposed stochastic model obtains better results.

Keywords: Real Time Revision · Intelligent Training System · Thompson Sampling · Case-Based Reasoning · Bayesian Knowledge Tracing · Automatic Adaptation

1 Introduction

The AI-VT system is a generic ITS that aims to accompany learners by proposing sheets of exercises called sessions. Inside each session, the expected abilities are divided into skills themselves broken down into subskills. The learner chooses a skill to work on, and the system generates a session composed of exercises associated with several subskills of the chosen skill. The system offers a list of exercises at the beginning of a session using the case-based reasoning paradigm with a database of questions [8].

Actually, the AI-VT system has a database of questions. Each of them is associated with a context, the text of a question, and a complexity level. The questions belong to a subskill level, and the subskills belong to a skill level. The

teacher and the learner are the principal actors in the system. The teacher has the capacity to configure the whole system, number of skills, subskills in a skill, number of questions, complexity of each of them, number of complexity levels, and time per session. The learner can start the series of a specific subskill, access complementary support resources, and answer the test questions in the sessions proposed by the system.

Using the CBR philosophy, the global AI-VT system assumes that there are learners with similar learning performances, needs, and abilities. It is then possible to group them and thus improve the general learning process for all of them. The case base comprises exercises associated with multiple skills.

CBR phases in AI-VT are used as described below. The retrieve phase involves finding exercises previously proposed for the same skill to other learners. The reuse phase modifies this list according to the sessions previously proposed to the learner. The revision phase, presented in this paper, involves modifying this list of exercises in real time. The retain phase maintains this new session [9]. To improve the functionality of AI-VT, two real-time recommendation models have been developed and the necessary architecture has been designed to integrate all the elements without significantly affecting the internal structure of the original system.

The two main types of software architecture to integrate modules and functionalities are monolithic and modular. In monolithic architecture, the software system is considered as a single unit with only one code source, one database, and one deployment for the whole system. This type of system is simple to develop and test but is not good for updating and scaling because of its rigidity. Modular architecture divides the system into independent modules that can communicate with each other. Each module then contains everything needed to work. Actually, there are many software systems designed with modular architecture because of the multiple advantages it offers [3] [19].

The recommendation systems in ITS generally pretend to find the weakness of learners and try to help them to improve the knowledge adapting the system. A common recommender algorithm is Bayesian Knowledge Tracing (BKT). This algorithm use four parameters for each learner to estimate the learned probability. Then with a defined threshold, it is possible to adapt a specific system to personal calculated knowledge [12]. Formally, the model defines four parameters $P(k)$ is the parameter to estimate the knowledge probability in a specific skill. $P(w)$, is the probability of the learner demonstrating knowledge. $P(s)$, is the probability the learner makes a mistake. $P(g)$, is the probability that the learner guessed a response. For each response given by the learner, the $P(k)$ value is updated as shown in Equations 1, 2 and 3. With these equations, it is possible to see the knowledge progression in a skill [16].

$$P(k_{t-1}|Correct_t) = \frac{P(k_{t-1})(1 - P(s))}{P(k_{t-1})(1 - P(s)) + (1 - P(k_{t-1}))P(g)} \quad (1)$$

$$P(k_{t-1}|Incorrect_t) = \frac{P(k_{t-1})P(s)}{P(k_{t-1})(P(s)) + (1 - P(k_{t-1}))(1 - P(g))} \quad (2)$$

$$P(k_t) = P(k_{t-1}|evidence_t) + (1 - P(k_{t-1}|evidence_t))P(w) \quad (3)$$

The *evidence* represents the response to a specific question. If the response is *correct* then the equation 1 is used, but if the response is *incorrect* the equation 2 is used. The BKT equations calculate the values for each step of time t sequentially.

This work focuses on the CBR revise step. Indeed, in our proposition a training session is proposed to the learner after the reuse step. This initial training session is adapted to the learner level *a priori*. Then, the new module described in this paper analyses the answers given by the learner in real-time during the training session and proposes some modifications in real-time. For this reason, we consider the contribution of this paper is situated in the revise phase of the classic CBR-cycle. The recommendation module is integrated into the AI-VT CBR system by means of a modular architecture.

This paper presents two main contributions: (i) a modular distributed architecture that allows integration of multiple artificial intelligence models in CBR in an asynchronous manner, (ii) two recommender models based on rules and reinforcement learning to improve the revision CBR system for learner needs. These models have been compared to common BKT algorithm used in others ITS.

The structure of this paper is as follows: Section II presents related works on recommender systems, Bayesian Knowledge Tracing, Case-Based Reasoning and adaptation methods. The proposed model is explained in Section III. Section IV describes the experiments and results. Finally, Section V presents the conclusions.

2 Related Works

In the field of ITS, recommender systems have been shown to have a positive impact on learners as they aid in finding relevant resources and maintaining motivation. Additionally, the application of artificial intelligence techniques to personalize recommendations has made these systems more efficient. These effects were validated by Huang *et al.*[10], who measured the difference between preliminary tests and tests taken after the completion of a course, with and without recommender system. In intelligent tutoring systems, the architecture

is also important to define the actors, services, functionalities, and scenarios while taking into account the information corresponding to each learner, such as the uncertainty of progress, specific needs, and defined objectives. Xu and Zhao [15] propose a modular architecture for adjusting an educational system with artificial intelligence, where the authors show the flow of request and answer information between modules and actors to perform a specific task, as well as the layers and their functionalities.

The work of Bradác *et al.*[4], designs a system that allows for adaptation through the collection of complementary information from the learner throughout the learning process. The modules have been defined to obtain information from the learner and after processing and analyzing it, to transmit it to other modules whose purpose is to adapt the learning objects.

Recommender systems require information about the learner and their requirements so that a module can define a profile and perform a set of algorithms to produce a recommendation. The work of Lalitha and Sreeja [11] uses the KNN algorithm to identify the common characteristics between the learners and to extract the adapted resources from the web using Random Forest. There are various methods for creating and suggesting personalized resources and paths. For instance, in Zhao *et al.* [17], learner data is collected and classified into groups with similar characteristics to determine the performance of each learner using the data envelopment analysis (DEA) method.

The recommendation may suggest resources, themes, or exercises. In Zhou and Wang [18] an exercise recommendation system for learning English was developed. The system includes a main module that represents learners as vectors based on the DINA model. For instance, if in the vector $k_1 = 1$, the learner has mastered knowledge point k_1. On the other hand, if $k_2 = 0$, the learner needs to study knowledge point k_2 as they have not yet acquired mastery in that area.

Some recommendation and personalization works consider complementary variables to the grades, as in Ezaldeen *et al.* [7], which combines the analysis of the learner's behavior and semantic analysis. The first step is to collect the necessary data to create a learner profile. The profile is assigned to a set of pre-defined learning categories according to their preferences and historical data. Then, the system generates a guide to obtain resources that the system can recommend on the web.

In Zhang and Yao [16] a variant of the BKT algorithm for a specific type of problem with three possible answers is proposed. Basically, the work divides the learning process into three states by changing the equations and defining new parameters to obtain the learning state with more precision. Using the AUC, RMSE and standard deviation, the BKT for three answers show better results

compared with original BKT.

The work of Xu *et al.*[14] uses the BKT model to recommend personalized training materials according to learner progress in a system of educational safety behaviors in the construction industry. The model was tested with real participants using the results of several quizzes performed throughout the training as a metric. The model help to acquire a higher level of mastery of the knowledge associated with a specific competency and the students present better results than the students who used the system without a recommendation system.

Other techniques and algorithms were applied to improve the ITS as a CBR. The use of CBR in ITS shows positive results as indicated by the work of Supic [13] whose model follows the traditional cycle of CBR by combining traditional and digital learning models. The main contributions are the representation of cases and the recommendation of personalized learning paths according to other learners' information. To demonstrate the effectiveness of the model, an initial case base was created to recommend courses to 120 learners. The results were obtained with the help of exams before and after following the course recommended by the model.

Recommendation systems are generally used in commerce and on the market to recommend products to customers, but these techniques may be used in other fields to recommend training exercises, planning paths, study resources, etc. These recommendations are based on historical data and user feedback. In Eide *et al.* [6], a Recurrent Neural Network as a recommender system is proposed for a marketplace. The network had been tested in combination with several variants of the Thompson Sampling (TS) algorithm because it allows maximization of the explorative opportunities and ensures a machine learning process.

The recommendation of products for e-commerce with an extension of the Bernoulli Thompson Sampling algorithm is proposed in Brodén *et al.* [5], where combinations of TS with sleeping bandit and dynamic partitioning are tested with information belonging to a single user in the same session. The results of metrics precision and recall show high values of accuracy for three different types of products.

Another application of TS appears in Akerblom *et al.* [1], where a variant of TS is implemented to find the minimax paths in a network with stochastic weights and partial knowledge. The work with three different test scenarios demonstrates the algorithm effectiveness despite the complexity and lack of information.

3 Proposed Model

3.1 Proposed architecture

The proposed model[1] belongs to an architecture shown in Figure 1, where the solid lines represent a bidirectional information flow, the solid lines with an arrow represent the unidirectional flow and the dashed lines represent the information dependency between the modules. The external devices that can be used by the modules to execute their functionalities and the labels that indicate what kind of information the module sends to the central system are also shown, as well as some of the artificial intelligence algorithms implemented in each module and the stage of development each one of them is in. Some devices are necessary for the execution of modules requiring getting data from external sources. The NAO robot, sensors, video camera, and microphone are represented in the architecture diagram. This kind of architecture preserves the original AI-VT CBR system functionality, and thus, the system can make use of complementary functions simply by activating the corresponding module by sending and receiving the necessary information for its operation, which extends the original global functionality of the system and facilitates integration with the designed modules and even new modules. The modular design also facilitates code maintenance, development and integration of new extensions, and configuration and adaptation of the system to different scenarios all reducing risks and costs. The modularity also enables the functionalities of each module to be executed asynchronously in parallel or distributed mode if required.

The architecture consists of two main elements: the central system (AI-VT CBR System) and the functional modules. The central system manages all the learning processes; generates and starts the sessions; stores the data for skills, questions, resources, learners, and responses; contains the controls and general interface; manages the flow of information; and activates the necessary modules. Modules are a set of independent functionalities implemented with artificial intelligence algorithms that receive and send data from the central component. Each module operates according to specific criteria related to its own particular purpose. The modules are grouped in layers according to their functionality: automatic correction, identification, adaptation, revision, and testing. The teacher and learner do not use the modules directly. The modules are used by the system to complement some functionalities.

The automatic correction layer (LC) corresponds to the modules in charge of receiving the learner's answers. According to the defined algorithms and criteria, they establish a grade consistent with a reference answer to a specific question. In this layer, the router module (LC0) is in charge of identifying the type of correction needed and instantiating the appropriate module to execute

[1] https://disc.univ-fcomte.fr/gitlab/daniel.soto_forero/ai-vt-recommender-system/tree/main

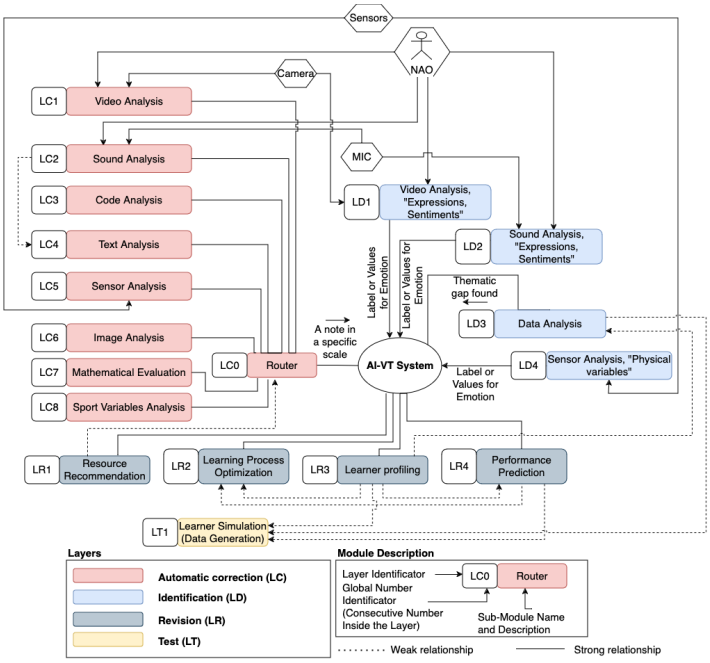

Fig. 1. Proposed Architecture to Integrate AI-VT CBR with AI Algorithms

the specific task. The identification layer (LD) contains the modules that identify the learner's weaknesses or external variables when performing the exercises proposed by the system or after analyzing the results. These modules help to customize the learner's processes according to the results obtained from the analyses. The revision layer (LR) includes the modules that take the data from the results obtained in the LC layer and the results of the analysis in the LD layer to modify the learner's path by trying to reinforce the learning in the detected weaknesses. The modules that obtain information from the learners and try to predict their results according to different skills and levels of complexity are in the revision layer (LR).

Specifically, the recommender model belongs to the revision layer (LR1, LR2), where the modules use the information generated by the automatic correction and identification layer modules, as well as the learner's complementary information stored in the system's database. All this information helps establish the best way to guide the learner towards better understanding by overcoming the weaknesses and gaps that have been identified. In this case, the recommender model attempts to change the list of exercises using CBR and according to the learner's partial results. The architecture with modular design also facilitates code maintenance, development and integration of new extensions, and configuration and adaptation of the system to different scenarios all reducing risks

and costs. The modularity also enables the functionalities of each module to be executed asynchronously in parallel or distributed mode if required.

3.2 Proposed models of recommendation

For adaptation and revision modules, the model requires the grades, time of answers, and complexity of questions to perform the recommendation for the learner, this information is represented as a numeric vector. The model variables and parameters are detailed in Table 1. The model is proposed in two versions, one deterministic and the other stochastic.

The proposed deterministic model calculates the mastery rate with Equations 4, 5 and 6. The mastery rate helps to guide the learner within the complexity levels using historical grades and to recommend the complexity level closest to the current one where weaknesses have been detected. Mastery rate m_c is a value on a scale of 0 to 100 that reflects the level of proficiency attained by the learner in a subskill. It serves to condense the learner's scores for each subskill. It is calculated based on the (latest) scores and weighted by the complexity of the exercises. The mastery rate must take into account the complexity of the exercises as follows: a 100% mastery rate cannot be achieved solely with low-complexity exercises; conversely, it can be achieved with maximum-complexity exercises alone. We then propose a tiered system. With c_n levels of complexity, exercises of complexity c should enable the attainment of a maximum mastery rate of $c * 100/c_n$. Finally, the mastery rate is calculated recursively.

$$f(g) =< g > - \left(< g > *\lambda * \frac{< t_c >}{t_m} \right) \tag{4}$$

$$m_1 = \frac{10}{c_n} * f(g_1)_{n_q} \tag{5}$$

$$m_c = max \left(m_{c-1} + \frac{10}{c_n} * f(g_c)_{n_q}, \frac{c * 10}{c_n} * f(g_c)_{n_q} \right) \tag{6}$$

With the calculation of the mastery rate for the last level of complexity it is possible to recommend an adapted complexity level using a predefined strategy. Equation 7 uses an indicator function I_c to determine the adapted complexity level using a table $v_{s,nv}$ of predetermined intervals.

$$ncl = I_c(inf(v_{s,nv}) <= m_c <= sup(v_{s,nv})),\ \forall nv \tag{7}$$

The proposed stochastic model uses the Beta family of distribution of probability to define dynamically the new complexity level inspired by the Thompson Sampling algorithm. This model version allows recommendation of non-contiguous complexity levels, but the priority is to recommend levels where faults have been detected. The initial parametrization of all distributions of probability can force the model to recommend contiguous complexity levels.

Table 1. Variables and parameters of the proposed model

ID	Description	Domain
c_n	Complexity levels	$\mathbb{N} \mid c_n > 0$
g_m	Max value into the scale of grades	$\mathbb{N} \mid g_m > 0$
g_t	Grade threshold	$(0, g_m) \in \mathbb{R}$
s	Number of defined paths	$\mathbb{N} \mid s > 0$
s_c	Defined fixed current path	$[1, s] \in \mathbb{N}$
Δs	Step for beta distribution parameters in path s	$(0, 1) \in \mathbb{R}$
t_m	Max value in time answer	$\mathbb{R} \mid t_m > 0$
nv	Number of intervals for each path	$\mathbb{N} \mid nv > 0$
g_c	Grade for a learner for a question with complexity c	$[0, g_m] \in \mathbb{R}$
ng_c	Grade of learner with time penalization	$[0, g_m] \in \mathbb{R}$
t_c	Time to answer a question with complexity c	$[0, t_m] \in \mathbb{R}$
m_c	Mastery rate of learner for a complexity c	$[0, 100] \in \mathbb{R}$
$v_{s,nv}$	Limits for each nv interval	$[0, 100]$
n_q	Number of questions to consider of history	$\mathbb{N}$
$f(g)$	Function to calculate the mean of grades g	$\mathbb{R}_+$
ncl	New calculated complexity level	$\mathbb{N}$
α_c	Value for α in complexity c	$\mathbb{R} \mid \alpha_c > 0$
β_c	Value for β in c complexity	$\mathbb{R} \mid \beta_c > 0$
$\Delta\beta$	Step of beta parameter initialization	$\mathbb{N} \mid \Delta\beta > 0$
λ	Weight of time penalization	$(0, 1) \in \mathbb{R}$
G_c	Set of d grades in complexity level c	$\mathbb{R}^d, d \in \mathbb{N} \mid d > 0$
x_c	Normalized average grades	$[0, 1] \in \mathbb{R}$
n_c	Number of total questions in a session	$\mathbb{N} \mid n_c > 0$
ny_c	Number of questions in complexity level c	$\mathbb{N} \mid 0 < ny_c \leq n_c$
y_c	Proportion of questions in complexity level c	$[0, 1] \in \mathbb{R}$
r	Total value for adaptability defined metric	$[0, c_n] \in \mathbb{R}$
sc	Total value for cosine similarity metric	$[-1, 1] \in \mathbb{R}$
th_1	First threshold for BKT system	$[0, 1] \in \mathbb{R}$
th_2	Second threshold for BKT system	$[0, 1] \in \mathbb{R}$

In this case, the grade threshold variable g_t is necessary to determine the variability of distribution of probability for each complexity level. Equations 8 and 9 show the correlated update rules. These rules modify the values by inverse reward. Each complexity level has an associated Beta distribution of probability with predefined initial values for the parameters α and β.

$$ng_c = g_c - \left(g_c * \lambda * \frac{t_c}{t_m} \right) \tag{8}$$

$$ng_c \geq g_t \rightarrow \begin{cases} \beta_c = \beta_c + \Delta_s \\ \beta_{c-1} = \beta_{c-1} + \frac{\Delta_s}{2} \\ \alpha_{c+1} = \alpha_{c+1} + \frac{\Delta_s}{2} \end{cases} \quad ng_c < g_t \rightarrow \begin{cases} \alpha_c = \alpha_c + \Delta_s \\ \alpha_{c-1} = \alpha_{c-1} + \frac{\Delta_s}{2} \\ \beta_{c+1} = \beta_{c+1} + \frac{\Delta_s}{2} \end{cases} \tag{9}$$

The new complexity level is the maximum random value index for all the complexity levels (Equation 10).

$$ncl = max_x(\mathbb{E}[Beta(\alpha_x, \beta_x)]), 0 <= x <= c_n \tag{10}$$

4 Evaluation

The model was tested with a generated dataset that contains the grades and response times of 1000 learners for five different levels of complexity. The approximation of each learner's grades are generated with the logit-normal distribution of probability because it is experimentally the best representation model [2], and response times with normal distribution of probability. The generated dataset is a simulation of learners grades for answers to fifteen questions at each of the five levels of complexity. The dataset simulates a weakness in each level of complexity for 70% of learners in the first ten questions. The difficulty of the complexity is also simulated by reducing the average score and increasing the variance.

The defined general testing parameters for the model are detailed in Table 2. The specific parameters for all models have been estimated based on *Optuna* hyperparameter optimization for Python, to get the best possible global configuration. The parameters for the deterministic variant are in Table 3. Using these parameters, the model is formally written as Equations 11, 12, 13, 14 and 15. Table 4 shows the definition v of intervals. The stochastic variant parameter values are in Table 5, and parameters with an unique subjective configuration for BKT are in table 6.

Table 2. Values for tested scenarios

ID	c_n	g_m	t_m	s	s_c	λ
Value	5	10	120	3	2	0.25

Table 3. Initial values for the deterministic recommendation model

ID	nv_1	nv_2	nv_3	n_q	$v_{s,nv}$
Value	8	7	7	3	Table 4

$$m_1 = \frac{10}{5} * f(g_1)_3 \tag{11}$$

$$m_2 = max\left(m_1 + \frac{10}{5} * f(g_2)_3, \frac{20}{5} * f(g_2)_3\right) \tag{12}$$

$$m_3 = max\left(m_2 + \frac{10}{5} * f(g_3)_3, \frac{30}{5} * f(g_3)_3\right) \tag{13}$$

Table 4. Table v with the values for three paths ($s = 3$), seven to eight intervals ($nv_1 = 8, nv_2 = 7$ *and* $nv_3 = 7$) and five CL (Complexity Levels)

Path 1		Path 2		Path 3	
Interval	CL	Interval	CL	Interval	CL
$[0, 20]$	0	$[0, 15]$	0	$[0, 10]$	0
$[21, 30]$	1	$[16, 25]$	1	$[11, 20]$	1
$[31, 45]$	1	$[26, 35]$	2	$[21, 30]$	2
$[46, 50]$	2	$[36, 42]$	2	$[31, 36]$	2
$[51, 65]$	2	$[43, 50]$	3	$[37, 43]$	3
$[66, 75]$	3	$[51, 75]$	3	$[44, 65]$	3
$[76, 90]$	3	$[76, 100]$	4	$[66, 100]$	4
$[91, 100]$	4				

Table 5. Initial values for the stochastic recommendation model (x represents all learners, y represents all complexity levels greater than 1)

ID	g_t	$\alpha_{x,1}$	$\alpha_{x,y}$	$\beta_{x,1}$	$\Delta\beta_{x,y}$	Δ_1	Δ_2	Δ_3
Value	6	2	1	1	1	0.3	0.5	0.7

Table 6. Initial values for the BKT recommendation model

ID	p_k	p_{wl}	p_s	p_g	th_1	th_2	g_t
Value	0.3	0.2	0.4	0.2	0.5	0.95	6

$$m_4 = max\left(m_3 + \frac{10}{5} * f(g_4)_3, \frac{40}{5} * f(g_4)_3\right) \tag{14}$$

$$m_5 = max\left(m_4 + \frac{10}{5} * f(g_5)_3, \frac{50}{5} * f(g_5)_3\right) \tag{15}$$

To compare the models, three scenarios has been defined, where the progression of learners are simulated. The first scenario is the recommendation without initial data (cold start). The second scenario is the learner data for the first complexity level and the third scenario contains grades for all learners in the first and second complexity levels.

Equation 16 describes a quality metric to compare the original system, two recommendation model variants and BKT numerically. The goal of this metric is to measure the system's ability to adapt in dynamically in real-time the navigation between levels of complexity. The metric calculates a value for each complexity level according to grade averages and the number of recommended questions in that complexity level. The purpose of this metric is to give a high score to the recommender systems that propose more exercises at the complexity level where the learner has registered a lower average grade with the idea of reinforcing the knowledge at that complexity level. If they propose fewer exercises at complexity levels where the average grade is high it is assumed that the student has already acquired sufficient knowledge at those complexity levels. Low scores

are assigned to systems that recommend few exercises at complexity levels with low average grades and, conversely, if they propose many exercises at complexity levels with high average grades.

$$r = \sum_{c=0}^{c_n-1} e^{-2(x_c+y_c-1)^2};\ x_c = \frac{< g_c >_{G_c}}{g_m};\ y_c = \frac{ny_c}{n_c} \tag{16}$$

In Equation 16, x_c is the normalized average of grades in complexity level c, and y_c is the normalized number of answered questions in complexity level c. The global equation for the metric rp is inside the domain of two variables x_c and y_c. The maximum value for r in a specific complexity level is 1. The global maximum value for the tested scenarios is 5; therefore a good recommender system should have a high r value.

5 Results and Discussion

A first comparison between the recommender models (deterministic, stochastic and BKT) is shown in Figure 2 where it is possible to see the different state transitions for the test scenarios, with five states corresponding to the levels of complexity.

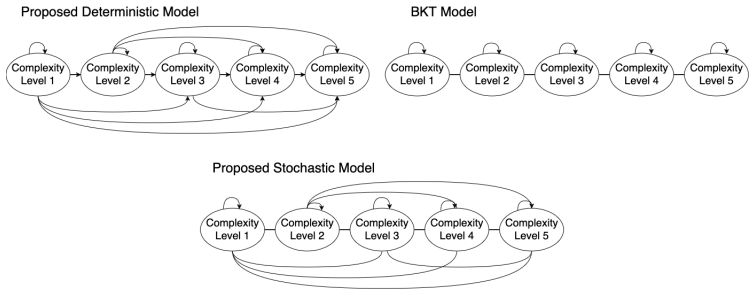

Fig. 2. Possible transitions for the three different recommender models (lines without arrow mean that the flow can be bi-directional)

The results of the comparison without historical data (cold start) between the two versions of the proposed model, BKT and the original system (CBR) are shown in Figure 3, where different transition numbers and scales appear. The original system does not have transitions, this behavior motivates the development of a real-time adaptation module, as it allows to adjust the system more efficiently to the needs of each individual learner. All the learners are evaluated inside complexity level 0. Grades obtained during the session are not considered. The deterministic model generates four big transitions with a large number of learners in Questions 5, 6, 8 and 12 all of them between contiguous complexity

levels. The trends are downward for levels 0, 1, and 2 after the eighth question and upward for levels 1 and 3. The mean after 20 executions for stochastic model show that it starts by proposing all the possible levels of complexity but focuses on level 0. The transitions are constants, but for a small number of learners, the trends after the tenth question are downward for levels 0 and 4 and upward for levels 1, 2 and 3. The BKT model starts stable for the first 5 questions. After that, it is less stable and propose big changes between complexity levels. In the cold start the trend is very similar to the deterministic model, but the progression is faster. In the first session almost 800 learners pass to complexity level 1 and 100 to level 4, that behavior can be explained by the values of the configuration parameters. Then it is possible to see that Deterministic and BKT systems are very sensible to small changes in the value of grades and that can difficult the precision in the weakness identification. The results of quality

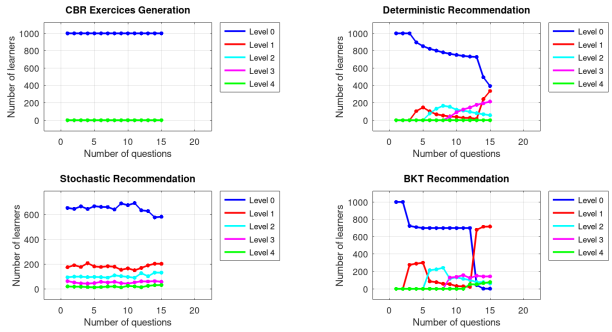

Fig. 3. Recommended complexity levels with two version models, first session case without historical grades data (cold start)

metric calculations for the original system, BKT, and the two proposed models in the three defined scenarios are shown in Table 7. According to these results, it can be said that each model can be used in specific situations, BKT obtains good results for each level of complexity in the cold start scenario, but only allows advancement between contiguous levels of complexity. The deterministic model allows passing between different levels of complexity, but since the defined scales are closed intervals only advancement is possible, which allows the model to guarantee level passage. The stochastic model obtains better overall results in scenarios 1 and 2. In addition it provides transitions between different levels forward and backward generating greater diversity in the proposed exercises.

6 Conclusion

The proposed architecture is based on concepts and patterns commonly used to design complex systems that use artificial intelligence algorithms and tools in

Table 7. Results Metric Table (CBR - System without recommendation model, DM - Deterministic Model, SM - Stochastic Model)

	c_0	c_1	c_2	c_3	c_4	Total (r)	Total (%)
Scenario 1							
CBR	0.5388	-	-	-	-	0.5388	10.776
DM	0.8821	0.7282	0.9072	0.8759	-	3.3934	67.868
SM	0.9463	0.8790	0.7782	0.7108	0.6482	3.9625	79.25
BKT	**0.9996**	**0.9981**	**0.9262**	**0.8769**	**0.8023**	**4.6031**	**92.062**
Scenario 2							
CBR	0.9445	**0.9991**	-	-	-	1.9436	38.872
DM	-	0.9443	0.8208	**0.9623**	-	2.7274	54.548
SM	**0.9688**	0.9861	0.8067	0.7161	0.6214	**4.0991**	**81.982**
BKT	-	0.9954	**0.9910**	0.893	**0.8233**	3.7027	74.054
Scenario 3							
CBR	-	0.8559	0.7377	-	-	1.5936	31.872
DM	-	-	0.5538	0.7980	-	1.3518	27.036
SM	**0.9089**	0.9072	0.9339	0.7382	0.6544	**4.1426**	**82.852**
BKT	-	**0.9988**	**0.9882**	**0.9102**	**0.8478**	3.745	74.9

some way. This kind of design allows the implementation of a functional system with the adaptability necessary for the execution of one of the main requirements of intelligent learning systems. Moreover, as the referenced works indicate, modular architecture allows more flexible implementation and gives the system the possibility to scale quickly and even add complementary functionalities.

The recommendation module implemented tries to identify learners' weaknesses in order to propose coherent revisions, the algorithms in each of the cases tend to use few subjective and individual parameters for each learner unlike in the case of with the BKT model. The results show that it is indeed possible to identify the weaknesses of the learners and to revise the content of the training session in a personalized way for each learner. Compared to the global unique best BKT subjective parameter settings, the results are differents depending on the level of complexity and the initial situation. In some configurations, the best results have been obtained globally with the stochastic model. Therefore, it is determined that the proposed model can be complementary and used according to the specific situation of each learner.

Future work includes an analysis of architecture with other parameter values and other configurations, a complementary module to verify the changes proposed by the recommender model before sending them to the learner and analyze the specific pedagogical situations.

References

1. Akerblom, N., Hoseini, F.S., Haghir Chehreghani, M.: Online learning of network bottlenecks via minimax paths **122** (2023). https://doi.org/https://doi.org/10.1007/s10994-022-06270-0, https://doi.org/10.1007/s10994-022-06270-0
2. Arthurs, N., Stenhaug, B., Karayev, S., Piech, C.: Grades are not normal: Improving exam score models using the logit-normal distribution. In: International Conference on Educational Data Mining (EDM). p. 6 (2019), https://eric.ed.gov/?id=ED599204
3. Auer, F., Lenarduzzi, V., Felderer, M., Taibi, D.: From monolithic systems to microservices: An assessment framework. Information and Software Technology **137**, 106600 (2021). https://doi.org/https://doi.org/10.1016/j.infsof.2021.106600, https://www.sciencedirect.com/science/article/pii/S0950584921000793
4. Bradáč, V., Smolka, P., Kotyrba, M., Průdek, T.: Design of an intelligent tutoring system to create a personalized study plan using expert systems. Applied Sciences **12**(12) (2022). https://doi.org/10.3390/app12126236, https://www.mdpi.com/2076-3417/12/12/6236
5. Brodén, B., Hammar, M., Nilsson, B.J., Paraschakis, D.: Ensemble recommendations via thompson sampling: An experimental study within e-commerce. In: 23rd International Conference on Intelligent User Interfaces. p. 19–29. IUI '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3172944.3172967, https://doi.org/10.1145/3172944.3172967
6. Eide, S., Leslie, D.S., Frigessi, A.: Dynamic slate recommendation with gated recurrent units and thompson sampling **36** (2022). https://doi.org/https://doi.org/10.1007/s10618-022-00849-w, https://doi.org/10.1007/s10618-022-00849-w
7. Ezaldeen, H., Misra, R., Bisoy, S.K., Alatrash, R., Priyadarshini, R.: A hybrid e-learning recommendation integrating adaptive profiling and sentiment analysis. Journal of Web Semantics **72**, 100700 (2022). https://doi.org/https://doi.org/10.1016/j.websem.2021.100700, https://www.sciencedirect.com/science/article/pii/S1570826821000664
8. Henriet, J., Christophe, L., Laurent, P.: Artificial intelligence-virtual trainer: An educative system based on artificial intelligence and designed to produce varied and consistent training lessons. Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology **231**(2), 110–124 (2017). https://doi.org/10.1177/1754337116651013, https://doi.org/10.1177/1754337116651013
9. Henriet, J., Greffier, F.: Ai-vt: An example of cbr that generates a variety of solutions to the same problem. In: Case-Based Reasoning Research and Development: 26th International Conference, ICCBR 2018, Stockholm, Sweden, July 9-12, 2018, Proceedings 26. pp. 124–139. Springer (2018)
10. Huang, A.Y., Lu, O.H., Yang, S.J.: Effects of artificial intelligence–enabled personalized recommendations on learners' learning engagement, motivation, and outcomes in a flipped classroom. Computers and Education **194**, 104684 (2023). https://doi.org/https://doi.org/10.1016/j.compedu.2022.104684, https://www.sciencedirect.com/science/article/pii/S036013152200255X
11. Lalitha, T.B., Sreeja, P.S.: Personalised self-directed learning recommendation system. Procedia Computer Science **171**, 583–592

(2020). https://doi.org/https://doi.org/10.1016/j.procs.2020.04.063, https://www.sciencedirect.com/science/article/pii/S1877050920310309, third International Conference on Computing and Network Communications (CoCoNet'19)
12. Sun, J., Wei, M., Feng, J., Yu, F., Li, Q., Zou, R.: Progressive knowledge tracing: Modeling learning process from abstract to concrete. Expert Systems with Applications **238**, 122280 (2024). https://doi.org/https://doi.org/10.1016/j.eswa.2023.122280, https://www.sciencedirect.com/science/article/pii/S0957417423027823
13. Supic, H.: Case-based reasoning model for personalized learning path recommendation in example-based learning activities. In: 2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). pp. 175–178 (2018). https://doi.org/10.1109/WETICE.2018.00040
14. Xu, S., Sun, M., Fang, W., Chen, K., Luo, H., Zou, P.X.: A bayesian-based knowledge tracing model for improving safety training outcomes in construction: An adaptive learning framework. Developments in the Built Environment **13**, 100111 (2023). https://doi.org/https://doi.org/10.1016/j.dibe.2022.100111, https://www.sciencedirect.com/science/article/pii/S266616592200045X
15. Xu, X., Zhao, H.: Artificial intelligence education system based on feedback-adjusted differential evolution algorithm. Soft Computing (2023). https://doi.org/10.1007/s00500-023-08828-z, https://doi.org/10.1007/s00500-023-08828-z
16. Zhang, K., Yao, Y.: A three learning states bayesian knowledge tracing model. Knowledge-Based Systems **148**, 189–201 (2018). https://doi.org/https://doi.org/10.1016/j.knosys.2018.03.001, https://www.sciencedirect.com/science/article/pii/S0950705118301199
17. Zhao, L.T., Wang, D.S., Liang, F.Y., Chen, J.: A recommendation system for effective learning strategies: An integrated approach using context-dependent dea. Expert Systems with Applications **211**, 118535 (2023). https://doi.org/https://doi.org/10.1016/j.eswa.2022.118535, https://www.sciencedirect.com/science/article/pii/S0957417422016104
18. Zhou, L., Wang, C.: Research on recommendation of personalized exercises in english learning based on data mining. Scientific Programming **2021**, 5042286 (Dec 2021). https://doi.org/10.1155/2021/5042286, https://doi.org/10.1155/2021/5042286
19. Zuluaga, C.A., Aristizábal, L.M., Rúa, S., Franco, D.A., Osorio, D.A., Vásquez, R.E.: Development of a modular software architecture for underwater vehicles using systems engineering. Journal of Marine Science and Engineering **10**(4) (2022). https://doi.org/10.3390/jmse10040464, https://www.mdpi.com/2077-1312/10/4/464