The Design of the Personal Enemy - MIMLeBot as an Intelligent Agent in a Game-based Learning Environment

Kristijan V. Kuk

Academy of Criminalistic and Police Studies, Cara Dusana 196, 11080 Belgrade, Serbia, kristijan.kuk@kpa.edu.rs

Ivan Z. Milentijević

Faculty of Electronic Engineering, University of Nis, Aleksandra Medvedeva 14, 18000 Nis, Serbia, ivan.milentijevic@elfak.ni.ac.rs

Dragan M. Ranđelović, Brankica M. Popović, Petar Čisar

Academy of Criminalistic and Police Studies, Cara Dusana 196, 11080 Belgrade, Serbia, dragan.randjelovic@kpa.edu.rs, brankica.popovic@kpa.edu.rs, petar.cisar@kpa.edu.rs

Abstract: A bot is one of the main elements of all computer video games, frequently used for the creation of various opponent characters within a game. Opponent modeling is the problem of predicting the agent actions in a gaming environment. This paper proposes and describes the implementation of a bot as a personal opponent in a small educational game. In order to increase the efficiency when using such a small educational application/module, artificial intelligence was added in the form of a bot competing with the students. Pedagogical elements of the intelligent learning system are introduced through the pedagogical model and the student model. This paper demonstrates the use of the student model to present the player model built by the experience of a human teacher, with true/false questions incorporated with the bot strategy into the opponent model. The authors use the Monte Carlo approach in this implementation, known as artificial intelligence technique and a best-first search method used in most video games, but to the best of their knowledge, it has not been used for prediction in educational games based on bot strategy. The results highlight that the Monte Carlo approach presented via the BFTree classifier provides the best classification accuracy compared with other predictive models based on data mining classifiers. It was shown that the training data from the human player can help in creating a bot strategy for a personalized game-based learning system. The Help option can be used for the assessment of the students' current knowledge by counting the number of Help option accesses, the player relies on Help as a 'source of *knowledge' needed to complete the game task successfully. The obtained results show that the bot (personal opponent) stimulated players to replay the game multiple times, which may contribute to the increase of the students' knowledge.*

Keywords: knowledge personalization and customization; educational games; intelligent tutoring systems; personalized e-learning

1 Introduction

Depending on the type of game playing, video games may be associated with positive cognitive outcomes [1]. Simple games are mainly intended for one player. Therefore, the personal opponent developed as a game bot might have an important role in them. Game bots as 'automated programs with or without artificial intelligence that help players enhance, accelerate, or bypass some routines in the game' [4] are generally approved by the gaming community. Artificial intelligence (AI) depends both on knowledge about the world, and algorithms to intelligently process that knowledge. Three key understandings about the world, called 'models', used by intelligent systems in education, are the pedagogical model, the domain model, and the learner/student model. The application of artificial intelligence to education (AIED) has been the subject of many academic studies focusing on pedagogical agent realization [5, 6] or personalized educational game [7]. They are mostly based on a student model and teaching strategies implemented by agents technology including pedagogical features (pedagogical model) in an intelligent e-learning systems [8].

For the effective integration of educational games into the teaching material, Singh and Sivaswamy [2] proposed concentration on developing simple and small games instead of creating highly complex game systems where the student is then left alone to figure out the relations. Multimedia Interactive Modules for Learning – MIMLE is an E-learning system with many small education modules that comprises game elements [3]. It is a set of 2D interactive modules which showed significant success with teenage students. Through the Help window in the MIMLE system, the student is provided with textual messages in the form of a theorem or a definition, vital for successfully solving the problem given in the current module level. From a pedagogical aspect, the MIMLE system stimulates students to draw conclusions based on the accuracy of the achieved steps in solving the entire task on their own. Adding time as a game element contributing to the player's higher ranking in the game, also results in an increase of knowledge, as well as the improvement of the skills acquired through the easier levels. In order to keep the player as long as possible in the game, there is a need for some stimulation of his/her competitive spirit (in competition with themselves or with other players). The MIMLE system evaluates the player through his interaction with the system in the form of keeping count of the correct or incorrect answers and tracking the Help option activation. Appropriate changes are made in the student model based on the recorded player performance. The conversion of the student model with teaching strategies so as to create a player model with agent strategies is not an easy task in the process of designing and developing educational games. The bot or opponent in the given e-learning system should assume the role of an intelligent agent aiming not to demoralize the student as a player by playing better or worse than the player does. Instead, the bot should closely match the student's performance, thereby continuously forcing him to compete.

The bot as a personal opponent in the modules of the MIMLE system, follows the interaction of the student with the system, and subsequently compiles its own set of answers requested in the game. Therefore, the bot and the student compete with each other. Based on the detected player performance, the appropriate changes are then made in the student model. In order to achieve better results than his bot, the student will start the game, over and over again, thus getting a chance to open the Help option several times and, in that manner, increase his knowledge. In this paper, the authors propose to design a personal opponent approach for the intelligent agent – bot in an MIMLE system. Machine learning is a technique by which the computer 'learns' from the set of given training data, and then the set is able to predict the result of new data. The machine learning algorithms are designed to identify patterns based on different characteristics or 'features' and then make predictions about the new, unclassified data based on the patterns 'learned' earlier. Many classification techniques are implemented on the educational datasets used to build a player model in most video games. The Monte Carlo tree search is a method for making optimal decisions in numerous artificial intelligence problems, especially for the personal opponent approach into combinatorial games. The Monte Carlo approach presented as a form of data mining classifier can give very good classification accuracy in predictive models for designing the bot strategy.

The current state-of-the-art options in teaching strategies for evaluating students' knowledge and existing opponent techniques in playing are shown in Section 2. Section 3 describes the conceptual framework of small game-based modules, addressing the problem of a player modeling based on student model in small e-learning module [3]. The knowledge discovery techniques presented in Section 4 are used for developing the bot strategy based on the data analysis of the player's answers and player's interaction with the Help option. Also, the methodology of searching for hidden connections between the gaming data is presented in the form of data mining algorithms and classifiers to create the best bot strategy. The experimental results, presented in Section 5, confirmed the validity of the described bot strategy (called MIMLeBot) and the authors' policy in setting up possible states in the player model that the player/student could experience in a special class of educational games, justifying its implementation in such a type of e-learning system.

2 State of the Art

Exams are commonly used assessment and evaluation tools in universities and there are numerous types of exam questions, generally categorized into 7 types [9]. One of them is the true-false (T/F) question type. With this type there are only two options for answering: “True” and “False”. This question provides students with a 50% chance of guessing the correct answer. Due to this fact T/F questions are suited for evaluating students' knowledge of specific facts and concepts, therefore true/false games are usually used as educational games. The student, as a player, is offered a task sequence (as a picture or text) and is expected to select the T/F item. For a simple T/F item, each student has a 50/50 chance of correctly answering the item even without any knowledge of the item's content [10, 11]. The teaching strategy for game completion is created, by monitoring the student’s response to the T/F items. In this paper, a simple educational game is developed in the described T/F manner, where a sequence of rectangles, i.e. ‘boxes’ represents T/F items.

Intelligent tutoring systems and agent-based learning environments also provide students with individualized practice rather than static sets of tasks. Many Intelligent Virtual Teaching Environments (IVTEs) include pedagogical features that are based on the student model and teaching strategies [12]. A student model (which reflects the state of knowledge), is applicable if its present state can be utilized by a certain interpreter to simulate the behavior of the modeled student when the student is solving training problems. The three-tier architecture of a typical agent IVTE system consists of a domain model, a student model and a pedagogical model [13]. Teaching an opponent model can be approached as a pattern recognition task, where the model is taught based on the history of the players’ previous actions [14]. However, this paper proposes a different approach. The authors determined a posterior distribution (using Bayes’ rule) for two specific states when the student’s answer is an accidental hit or miss, to gain appropriate (defined) bot action for those states. With this approach for T/F items, the use of the Help option plays a significant role in successfully completing the game. It may increase the students' knowledge of specific facts and concepts. For example, if the first answer is correct, and the second answer incorrect, then the probability that third answer will be correct is > 50% if the student opts for using the Help option.

In the game proposed here the player selects a color box out of a choice of four so as to complete a task, in this case it is a trick-taking game with players selecting a card sequentially. In addition, in trick-taking games, the player has one optimal strategy and should play the cards in a specific order. In many games the optimal opponent strategy is based on different search methods. Monte Carlo Tree Search (MCTS) is a best-first search method that builds a search tree iteratively. MCTS has been used in several single player games [15] such as the puzzles SameGame or Morpion Solitaire. The authors in [16] applied MCTS in the context of minimax game trees, since they are encountered in incomplete information games such as Poker, where the opponent's actions cannot simply be predicted as value minimization actions. Using a back propagation strategy, they are evaluated as a part of a complete Poker bot (kind of MCTS bots). MCTS is a search algorithm based on random play-outs. It has been observed that MCTS is successful for trick-taking card games, but rather not suitable for poker-like card games. The question the present authors are set to answer is whether or not MCTS can be applied in an educational game, as well, e.g. by creating the bot strategy as a personal opponent.

3 Game-based Learning Modules for Computer Science

The authors created a module as special modules into the MIMLE system, which combined different types of the existing interactive multimedia environments for learning mathematics, physics and electronics. The game concept is based on two components: a) students have to obtain the course information through its interpretation in the game world; b) students have to see the results of this algorithm in a game context. Also, apart from placing a game interface into the learning environment, the authors also applied basic game elements, such as: result, time and difficulty levels. These new modules, named game-based modules [17], which include game elements, represent research multimedia learning applications and are intended for Computer Science students. This module was deployed as a learning environment for topics in the field of Computer Science. To increase the engagement and interest of students for this type of teaching material, the authors included game characteristics in this environment. Students need to have appropriate knowledge to successfully solve the tasks given as part of the game, and to advance from one game level to the next. The *MIMLE* system is presented as an e-learning system based small two-level game-based modules for a single player.

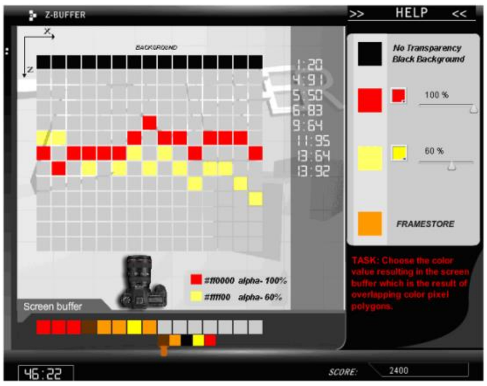
When experiencing any difficulties in task solving on a certain level, the Help option can accelerate the student's finding the right solution. Help option activation enables the students to learn the rules and apply them in practice on the presented examples. Therefore, the formulation of definitions and theorems within the Help is a crucial moment in designing the entire application.

3.1 Simple Module for Teaching Z-buffer Algorithm

The possibility of a visual representation of the task solving method for practicing the material in the Computer graphics module enabled their implementation as a small game-based module into the MIMLE system. The 'Z-buffer' module in the

MIMLE system was designed to help students learn basic terms referring to the principle of the Z-buffer algorithm operations and apply them practically through solving the given examples with a randomly generated content of buffer registers. This game contains interactive tasks implemented in the graphic environment that are visually directly associated with the learning material.

While analyzing the techniques for the first-class innovative testing select/recognize [18], the authors opted for giving the answers in MIMLE as a series of boxes to be clicked. The player must choose one out of 5 colors to represent a bit in the buffer registry the moment when the scanning line moves across the screen. Since the implemented buffer in the 'Z-buffer' algorithm uses 15 bits, the task of this module is to determine the value of each bit (i.e. contents of the registry) in various situational polygons shown on the screen. By theoretical definition, the frame buffer is used to store the intensity value of color value at each position (x, y). The goal of the game is for the player to test the z - depth of each surface (sequence of box in one color) so as to determine the closest (visible) surface (sequence of box in another color). The player determines the Z-value presented in the game as T/F items in the screen buffer registry in the moment by moving the scanning line. The complete buffer registry should display the one (pixel by pixel) that has the smallest value from the camera.



TASK: Choose the color value resulting in the screen buffer which is the result of overlapping color pixel polygons.

Figure 1
Interface of the 'Z-buffer' module

The correct answer which fills the content of one bit is one of the proposed answers presented to the students in the form of a box to be selected. These cubes (i.e. the offered answers), are presented in two ways. In Level 1 of the module, the answers are offered in the form of a 15-box column. When a certain buffer box is selected from the scanning line (moving from left to right), the box falls down and fills the content of the active bit (one sequence) in the buffer. Determining the buffer register contents is a task in Level 2, only during this time, does the player see polygons having some level of transparency. In comparison to Level 1, Level 2 is more challenging because it requires the player to determine the final color of the buffer cube which presumes additional knowledge of mixing two colors with different percentage of visibility.

3.2 Player Modeling Based on Teaching Strategies

A common characteristic of many intelligent tutoring e-learning systems is that they can recognize whether or not the students understand the given lessons and thus adjust the learning process in accordance to the students' needs. This kind of reasoning is also known as setting a diagnosis, revealing the students' level of knowledge [19]. The students' current level is represented by the student model. The teachers do not simply consider the students' answers in the exam, but also the sequence of the players' actions during the exam (e.g. the students make a mistake, then correct themselves and give the right answer). They remember the number and type of the students' mistakes, as well as the number of the questions or some other type of help used to move them away from a "deadlock". The teachers grade the students' knowledge based on all these details.

The player's success in the learning system is often seen as a sequence of the given answers [20]. Based on the sequence of correct and incorrect answers, the system can make the decision about whether or not the player was successful. In the 'Z-buffer' module the authors used a diagnostic technique called model tracing [21]. Monitoring the player's interaction with the system has been reduced to the player's last three steps and the system keeps track of the values of their combinations (i.e. the situations that result from the last three events are observed). As stated earlier in the paper, the player has only three options: the correct answer, the incorrect answer and asking for help. The states that the player can find within 'Z-buffer' module are shown in the last column of Table 1. Their determination is based on tracking the last two answers and the appearance of a Help window that was opened by the player before providing the second answer. The possible states of the player dependent on the events are shown in Table 1.

Table 1
Possible states of a player, depending on the events

Answer$_{t-1}$/ A$_{t-1}$	Answer$_t$/ A$_t$	Help$_t$ / H$_t$	State$_t$/ S$_t$
Ic	Ic	N	Does not know
Ic	C	N	Accidental hit
Ic	Ic	Y	Does not know
Ic	C	Y	Knows
C	Ic	N	Does not know
C	C	N	Knows
C	Ic	Y	Accidental miss
C	C	Y	Knows

Abbreviations used in Table 1 are: Correct – C, Incorrect – Ic, Yes – Y andNo – N

The intelligent agent – bot exclusively decides its own actions based on the action of the state in which a player is. Depending on his interaction with the 'Z-buffer' module (Figure 1), the player can find himself in one of the four states: knows, does not know, accidental hit, accidental miss, therefore, S = *(Knows, Does not know, Accidental hit, Accidental miss*).

4 Implementation of the Bot in Game-based Modules of the MIMLE System

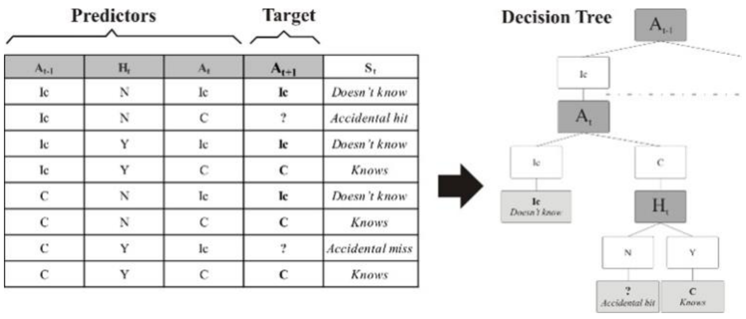
The authors chose modules of the MIMLE system as the environment where the bot will be implemented as a personal opponent. Possible decisions are presented, the bot's answers, as 'Correct' or 'Incorrect'. An MIMLE bot is a kind of a reflex agent which plays fully automatically according to the player model built by the human teacher (in the present case, by the authors of this paper). Playing against human players is the best way to test the player's knowledge. The first issue needed to model a strategy, is to obtain data from the games between human players. Intelligent agents in combination with fuzzy logic can help increase the quality and amount of interaction in a computer game.

The acquired knowledge is usually represented in the form of 'if-then' prediction rules. This representation is preferable for being a high-level, symbolic knowledge representation, contributing to the comprehensibility of the knowledge acquired. These decision rules can be placed in a decision queue (DQ) [22], which entails that they must be applied in a specific order. With this policy, the number of rules may be reduced because the rules could be one inside or another. The knowledge manager decides which representation provides the highest level of accuracy, while producing the smallest number of rules. DQ presents the following structure [23]:

If conditions ***then*** class ***Else if*** conditions ***then*** class ***Else if*** conditions ***then*** class ***Else*** 'unknown class' (1)

The discovered rules can be evaluated according to several criteria, such as the degree of confidence in the prediction, classification accuracy rate on unknown-class examples, comprehensibility, cost, etc. Since each feature, used as part of the classification procedure, can increase the cost and running time of a classifier system, as well as reduce the accuracy of the result, there is a strong motivation to design and implement systems using small feature sets.

A decision or classification tree can be described as 'a tool representing an algorithm in the form a graph or binary, tertiary or n-ary tree with each node and branch having a certain associated outcome, weight in terms of outcome, and probability' [24]. It is easy to implement, understand and customize. It is considered as one of the fastest in terms of learning and classification among machine learning techniques, where it can be seen as a predictive model which makes decisions on a branching series of Boolean tests. In computer logic terms, it can be viewed as a series of nested ‘if-else’.



A_{t-1}	H_t	A_t	A_{t+1}	S_t
Ic	N	Ic	**Ic**	*Doesn't know*
Ic	N	C	?	*Accidental hit*
Ic	Y	Ic	**Ic**	*Doesn't know*
Ic	Y	C	C	*Knows*
C	N	Ic	**Ic**	*Doesn't know*
C	N	C	C	*Knows*
C	Y	Ic	?	*Accidental miss*
C	Y	C	C	*Knows*

Figure 2
A decision tree for the bot’s answers problem

Figure 2 shows a sample decision tree of the bot’s answers. Here, the variable to be predicted or the ‘target variable’ is the Answer$_{t+1}$ –‘A_{t+1}’ variable. Given the set instances, each with same values for Answer$_{t-1}$ –‘A_{t-1}’, Answer$_t$ –‘A_t’ and Help$_t$ – ‘H_t’ and the resulting value is Knows, a tree is constructed as shown. Further, when given an instance with values for A_{t-1}, A_t and H_t, the machine learning system ought to be able to predict the value for the bot’s answer (Answer$_{t+1}$) variable by traversing through the tree, based on the conditions in the given instance. The target variable A_{t+1} may have accidental states with the same values of predictors. Usually, one accidental variable is identified as independent (x), and the other one as dependable accidental variable (y). The set of statistical methods that explores interlinks of statistical labels and appearances (direction, strength, shape) is called the theory of correlation, and the basic indicators of correlation links are equation of regression and correlation coefficient.

In the domain of machine learning, a classification problem involves machine learning attempting to predict to which class/group/category a new observation would most likely belong to, based on the training data set, which contains the already classified data. The classification of a given object is based on finding similarities with previously determined objects belonging to different classes, whereas the similarity of two objects is determined by analyzing their characteristics [25].

4.1 The Built Predictive Model

Classification is performed through supervised learning. After extracting the game variables, the lists were used along with Weka (a collection of machine learning algorithms for data mining tasks) to train different classifiers [26]. The classifiers were evaluated and it was determined which one was more suitable for this domain.

A total of 240 records were extracted from the 'Z-buffer' MIMLE database for analysis. The discarded records were related to the players who had only started the module, but then failed to perform any further activities. The analysis included 25 representative players and their activities. The original data vector contain 30 typical activities for each player (15 answers and 15 registered contacts with the Help option), and they represent attributes - independent input variables for a predictive model. By applying a discretional filter, numerical values of the A_{t+1} attribute were transformed to nominal and two intervals were determined ('*Correct*-C', '*Incorrect*-Ic'). A_{t+1} attribute was labeled as a class attribute, representing the dependent variable for a predictive model.

There are various techniques to test/estimate the performance of a predictive model. For the case described in this paper, the MIMLE dataset divides the training set into two parts (usually 1/3 and 2/3) where the larger part is used for training the model and the other one for validating it. Having the data ready, the next step is to use classifiers so as to learn the tactics presented on them. The following classifiers were used: Bayesian Networks (BayesNet), Best First Search Tree (BFTree), C4.5 Search Tree (J48), Multilayer Perceptron Neural Network (MultilayerPerceptron), Naive Bayes (NaiveBayes), Random Forest Search Tree (RandomForest) and SMO (SMO).

Table 2 presents the classification of the MIMLE dataset in which the BFTree, J48, Multilayer Perceptron, SMO algorithms show 75 % of correctly classified instances, followed by Bayes Net, Navie Bayes and Random Forest with 74.17% of correctly classified instances.

Table 2
Classification of MIMLE dataset using Weka

	Bayes Net	*Naïve Bayes*	*BFTree*	*J48*	*Random Forest*	*Multilayer Perceptron*	*SMO*
Correctly Classified Instances	178 (74.17%)	178 (74.17%)	180 (75%)	180 (75%)	178 (74.17%)	180 (75%)	180 (75%)
Incorrectly Classified Instances	62 (25.83%)	62 (25.83%)	60 (25%)	60 (25%)	62 (25.83%)	60 (25%)	60 (25%)
Kappa statistic	0.3393	0.3393	0.4143	0.4184	0.3342	0.4184	0.4184
Mean absolute error	0.3268	0.3377	0.3348	0.3594	0.3395	0.3383	0.25
Root mean squared error	0.4187	0.4163	0.4091	0.4239	0.4097	0.4106	0.5

Relative absolute error	74.90%	77.39%	76.72%	82.37%	77.80%	77.54%	57.30%
Root relative squared error	89.70%	89.19%	87.64%	90.81%	87.76%	87.95%	107.11%
Total Number of Instances	240	240	240	240	240	240	240
Confusion Matrix	a b 147 16 | 46 31 |	a b 147 16 | 46 31 |	a b 136 27 | 33 44 |	a b 135 28 | 32 45 |	a b 148 15 | 47 30 |	a b 135 28 | 32 45 |	a b 135 28 | 32 45 |

The measure of the dataset between the categorization of the predicted and observed is called Kappa Statistic. If the predicted and observed values are identical, then the Kappa Statistic value equals 1. The classifiers were evaluated through one of two error rates. The average values of Root mean squared error (RMSE) and Mean Absolute error (MAE) are determined in order to select the best predictive model. If both error values are higher, the accuracy is lower and vice versa. Kappa statistics for J48, Multilayer Perceptron and SMO showed the maximum. The MAE is low for SMO. The RMSE is low for BFTree. It is not possible to claim that MAE is a better indicator for model performance than RMSE because it is smaller. The chart in Figure 3 demonstrates the average error rate for each classifier.

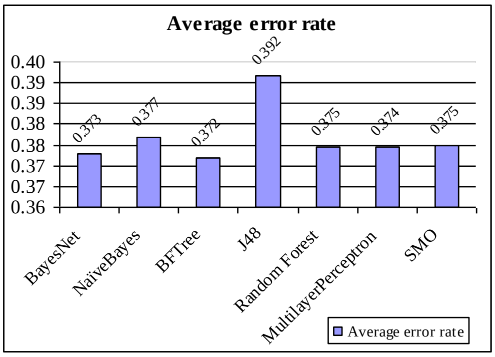


Figure 3

Average error rate

As can be seen both in Table 2 and Figure 3, the classifiers that performed better for this predictive model were the search trees, more specifically, the BFTrees with an average error of approximately 32.7%. The errors are relatively high, as was to be expected since the MIMLE bot tends to change tactic during the game, making it difficult to find a pattern with a small error.

4.2 Creating the Bot Strategy

In order to determine the likeliness of the possible states the player could fall into after three interactions with the game, the authors compared the results acquired through the predictive model with the parameters set up by the authors (on the basis of teaching experience) in Table 1. This paper shows that deploying classifiers for the prediction model design can be used as a method for detecting undefined players conditions, for which the target variables do not have values in the players model table ('Accidental miss/hit').

After training the selected predictive model on the basis of the game, with 240 players, the acquired results confirm the validity of the players' model (shown in Table 1) set up by the teachers. In the re-evaluation of the model with the data test set where all instances contained the values '*?*' for all possible players conditions (Table 3), the same values were obtained as assumed and set up for the class attribute – target during the modeling process by the authors .

Table 3

Predictions on test set

Instance#	Actual	Predicted	Probability	Distribution
0,0,n	?	2:Ic	0.359	*0.641
0,1,n	?	**1:C**	***0.563**	**0.438**
0,0,y	?	2:Ic	0.167	*0.833
0,1,y	?	1:C	*0.857	0.143
1,0,n	?	2:Ic	0.462	*0.538
1,1,n	?	1:C	*0.863	0.137
1,0,y	?	**1:C**	***0.5**	**0.5**
1,1,y	?	1:C	*1	0

Abbreviations used in Table 3 are: Correct – 1, Incorrect – 0, Yes – y andNo – n

Bot strategy creation is based on a set of *if-then-else* decision rules, as well as on decision tables. The question is what action the bot should perform if the target remains in an unknown state. Once the model was trained based on the BFTree classifier, it can be used to classify as yet unseen MIMLE data as part player model, presented in Figure 4. First, the file with the cases to be predicted must have the same structure as the file with the training set used to teach the model. Prior to training the classifiers, it was necessary to create an 'arff' file so that Weka can recognize the data. Assuming that one has trained the decision tree using the MIMLE datasets with 8 instances, as shown in the Table 1 column called State$_t$/S$_t$, one receives the result of predictions obtained on test data. The predicted column contains '*Correct'* or '*Incorrect'* for each of the lines in the test file. The instances with data which show player's states: '*Accidental miss'* and '*Accidental hit'*, predicted columns are marked as '*Correct'* value.

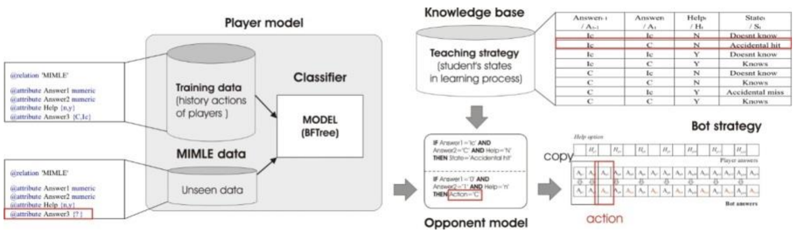


Figure 4

Generalizable opponent strategy approach in MIMLE modules

After teaching the tactics, the next stage is to implement the bot strategy for its answers in the game. The idea behind the strategy is that the bot will follow the combinations that the player accomplishes with the MIMLE module and thereby, will fill in the answers in the game. The first and second answer is copied from the player, while the third answer is filled in on the basis of the assumed position the player currently holds (Table 1). The subsequent set of the answers is filled in by moving one step back so that the fourth answer is copied from the player while the fifth answer is given on the basis of the player's condition caused by its third answer interpreted as the player's third answer, the player's fourth answer and option of asking for help during that time-frame. At the end of the game, the player can see the final set of answers given by the bot after each level (Figure 5). Further, the player can see the general success that he or she achieved, as portrayed by the accomplished scores (Figure 6).

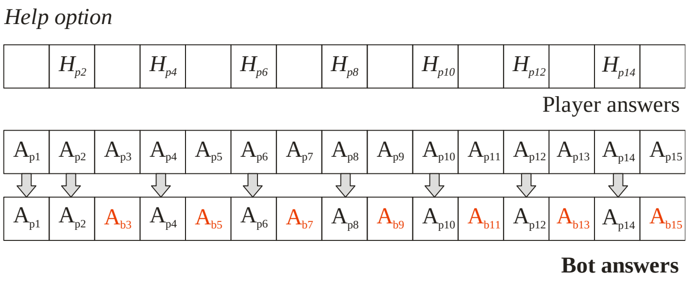

Help option
H_{p2} H_{p4} H_{p6} H_{p8} H_{p10} H_{p12} H_{p14}
Player answers
A_{p1} A_{p2} A_{p3} A_{p4} A_{p5} A_{p6} A_{p7} A_{p8} A_{p9} A_{p10} A_{p11} A_{p12} A_{p13} A_{p14} A_{p15}
A_{p1} A_{p2} A_{b3} A_{p4} A_{b5} A_{p6} A_{b7} A_{p8} A_{b9} A_{p10} A_{b11} A_{p12} A_{b13} A_{p14} A_{b15}
Bot answers

Figure 5

Bot strategy for the student's answers in modules

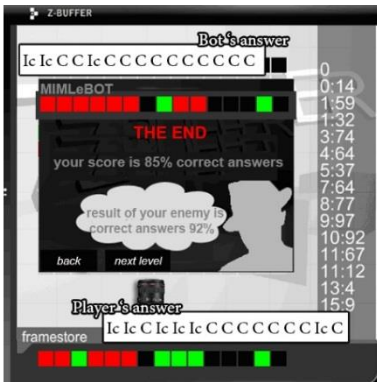

Z-BUFFER
Bot's answer
Ic Ic C C Ic C C C C C C C C C C
MIMLeBOT
THE END
your score is 85% correct answers
result of your enemy is correct answers 92%
back
next level
Player's answer
Ic Ic C Ic Ic Ic C C C C C C C Ic C
framestore
0
0:14
1:59
1:32
3:74
4:64
5:37
7:64
8:77
9:97
10:92
11:67
11:12
13:4
15:9

Figure 6
MIMLe Bot answers at game end of a level in the module

5 Results and Discussion

Decision trees are among the most popular classification techniques in data mining [27]. The classifier, whose average error value with MIMLE player data was shown as the lowest, was BFTrees. The performance of the BFTrees using different test options is presented and interpreted in Table 4. The performance is interpreted in terms of accuracy and error rate.

Table 4
Performance of BFTrees under different test options

Test Options	Accuracy	Error Rate	Kappa Statistic	MAE
UTS	75%	25%	0.4143	0.3348
CV (10 folds)	71.7%	28.3%	0.3454	0.3444
PS (66%)	74.4%	25.6%	0.357	0.3392

Table 4 demonstrates that the performance is the best when tested with "*use training set - UTS*" followed by "*cross validation-CV*" with 10 folds (it would apply training on the first 9 parts and testing on the last part), then with "*percentage split-PS*" option (random percentage split of the dataset is going to be 66% training data and 34% test data). In the BFTrees the selection of the best split is based on boosting algorithms [28] which are used to expand nodes in the best-first order instead of a fixed order. The results of this study confirmed that the Monte Carlo approach presented via BFTree classifiers provides the best classification performances in a small educational game, which supports the authors' assumption that it can be successfully used in these kinds of educational games.

The authors performed the analysis of the number of game activations with and without the bot. The efficiency of using the bot and its role in game-based modules by the *MIMLE* system was studied by analyzing the interaction of players with the system, as well as having an actual insight into their achievements at the end of the game. The analysis included 32 typical players for the Z-buffer module. The number of game activations was followed up for each player, as were their highest scores when the system did not contain a bot, and in the case when player competed with bot. It was established that the number of repetitions of the game increased in the case when the player competed with his personal opponent. The number of repetitions of the module the player performed with and without the use of the bot can be seen in Figure 7. The average value of module repetition without a bot was 3.78, while for the same player, the value increased to 6.47 for when he or she was playing against the personal opponent. It was also determined that the player made progress in his or her average value of the points.

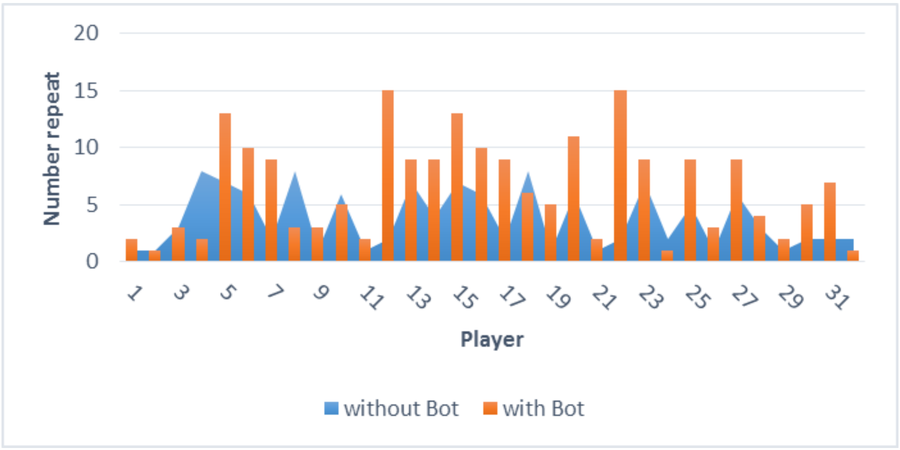


Figure 7

The interaction of players with the game-based module Z-buffer

In order to test whether or not a bot, personal opponent, has an impact on the number of game replays by the player, the authors formed two groups (experimental and control group), as shown in Figure 7, consisting of 32 players each. The results obtained using the (non-parametric) Mann-Whitney's test, by conventional criteria, indicate a statistically significant means difference in the number of game activations with and without the bot. The realized U-value is 309.5. The distribution is approximately normal, which indicated that the Z-value below should be used. The Z-Score is -2.71229. The p-value is .00672. The result is significant at $p < .01$. This result confirms the authors' right attitude regarding the importance of using a personal opponent in the game since its existence triggers a greater interest in the player, and motivates him or her to compete and achieve better results.

Conclusions

Many incorporated AIED and educational data mining (EDM) techniques are used for ‘tracking’ student’s behavior – for example, collecting data on class attendance and assignment submission in order to identify (and provide support for) students who are at risk of dropping out from their studies. Data mining methods provide automated predictions of the proposed solutions on the basis of well-known behavior patterns of the past, as well as the identification of previously unknown relationships, patterns and trends in very large databases.

This paper outlines a part of a bot strategy based on EDM, and its implementation in a small game-based learning system. Machine learning algorithms in EDM are applied to the task of detecting a bot’s strategy before it is executed and predicting when a player will perform strategic actions. As researches in gaming try to draw conclusions about player characteristics from their actions in open-ended gaming environments, understanding players’ goals can help provide an interpretive lens for those actions. The idea behind the bot strategy is that the bot follows the combinations that player accomplishes with the MIMLE module: correct answer, incorrect answer and asking for help. Based on the present authors’ analysis, it can be concluded that in the case of a small training data set, the BFTree algorithm provides good prediction results if the data are a combination of numerical (ANSWER1 and ANSWER2) and category types (Help), where a class attribute can have one of two class values (ANSWER3). The performance of the BFTree algorithm on this dataset was better than C4.5 algorithm.

Future work will focus on improving the precision of the bot’s strategy and implementing artificial intelligence in educational networking. For example, bots and students might be able compete with each other in social networks, thus maximizing the effectiveness of the learning process.

Acknowledgement

This work was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia under the project no. III 47016.

References

[1] T. Greitemeyer, D. Mügge: Video games do affect social outcomes a meta-analytic review of the effects of violent and prosocial video game play, *Personality and Social Psychology Bulletin*, Vol. 40, No. 5, pp. 578-589, 2014

[2] J. Singh, J. Sivaswamy: Creating Educational Game by Authoring Simulations, *Proceedings of the 17th International Conference on*

Computers in Education, Hong Kong, Asia-Pacific Society for Computers in Education, 2009.

[3] K. Kuk, I. Milentijević, D. Rančić, P. Spalević: Pedagogical agent in multimedia interactive modules for learning–MIMLE, *Expert Systems with Applications*, Vol. 39, No. 9, pp. 8051-8058, 2012.

[4] K. Chen, H. K. Pao, H. C. Chang: Game bot identification based on manifold learning, *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, ACM, pp. 21-26, 2008.

[5] C. Conati, A. Gertner, K. Vanlehn: Using Bayesian networks to manage uncertainty in student modeling. *User modeling and user-adapted interaction*, Vol. 12, No. 4, pp. 371-417, 2002.

[6] J. Sabourin, B. Mott, J. C. Lester: Modeling learner affect with theoretically grounded dynamic Bayesian networks, *International Conference on Affective Computing and Intelligent Interaction*, Springer, Berlin Heidelberg, pp. 286-295, 2011.

[7] M. M. Weng, I. Fakinlede, F. Lin, T. K. Shih, M. Chang: A conceptual design of multi-agent based personalized quiz game, *Advanced Learning Technologies (ICALT)*, 11th IEEE International Conference on Advanced Learning Technologies, pp. 19-21, 2011.

[8] W. L. Johnson, J. C. Lester: Face-to-Face Interaction with Pedagogical Agents, Twenty Years Later, *International Journal of Artificial Intelligence in Education*, Vol. 26, No. 1, pp. 25-36, 2016.

[9] D. M. Zimmaro: Writing good multiple-choice exams, Measurement and Evaluation Center, *University of Texas at Austin*, https://facultyinnovate.utexas.edu/sites/default/files/writing-good-multiple-choice-exams-04-28-10_0.pdf

[10] E. Ferrándiz, C. Puentes, P. J. Moreno, E. Flores: Engaging and assessing students through their electronic devices and real time quizzes, *Multidisciplinary Journal for Education, Social and Technological Sciences*, Vol. 3, No. 2, pp. 173-184, 2016.

[11] S. M. Čisar, D. Radosav, B. Markoski, R. Pinter, P. Čisar: Computer adaptive testing of student knowledge, *Acta Polytechnica Hungarica*, 7(4): pp. 139-152, 2010.

[12] M. A. Nunes, L. L. Dihl, L. M. Fraga, C. R. Woszezenki, L. Oliveira, D. J. Francisco, G. J. C. Machado, C. R. D. Nogueira, M. G. Notargiacomo: Animated pedagogical agent in the intelligent virtual teaching environment, *Digital Education Review*, Vol. 4, pp. 53-61, 2002.

[13] X. Luo, M. Spaniol, L. Wang, Q. Li, W. Nejdl, W. Zhang (eds.): Advances in Web-Based Learning-ICWL 2010, *9th International Conference,*

Shanghai, China, Proceedings Lecture Notes in Computer Science. Vol. 6483. Springer, 2010.

[14] M. J. V. Ponsen, G. Gerritsen, G. Chaslot: Integrating opponent models with Monte-Carlo tree search in poker, *Proc. Conf. Assoc. Adv. Artif. Intell. Inter. Decision Theory Game Theory Workshop*, pp. 37-42, 2010.

[15] S. Matsumoto, N. Hirosue, K. Itonaga, K. Yokoo, H. Futahashi: Evaluation of simulation strategy on single-player Monte-Carlo tree search and its discussion for a practical scheduling problem, *Proceedings of the International MultiConference of Engineers and Computer Scientists*, IMECS 2010, Hong Kong. Vol. 3, pp. 2086-2091, 2010.

[16] G. Van den Broeck, K. Driessens, J. Ramon: Monte-Carlo tree search in poker using expected reward distributions, *Asian Conference on Machine Learning*, ACML 2009, Nanjing, China, Springer Berlin Heidelberg, pp. 367-381, 2009.

[17] K. Kuk, I. Milentijević, D. Rančić, P. Spalević: Designing Intelligent Agent in Multilevel Game-Based Modules for E-Learning Computer Science Course, *E-Learning Paradigms and Applications*. Springer Berlin Heidelberg, pp. 39-63, 2014.

[18] R. Felder: Reaching the second tier: Learning and teaching styles in college science education, *College Science Teaching*, Vol.23, No.5, pp. 286-290, 1993.

[19] M. Chi, P. W. Jordan, K. VanLehn, M. Hall: Reinforcement Learning-based Feature Seleciton For Developing Pedagogically Effective Tutorial Dialogue Tactics, *Proceedings of 1st International Conference on Educational Data Mining*, EDM'08, Montreal, Quebec, Canada, pp. 258-265, 2008.

[20] I. Varlamis, S. Bersimis: Providing shortcuts to the learning process, Recent Progress in Computational Sciences and Engineering, Edited by Th. Simos & G. Maroulis, Lecture Series on Computer and Computational Sciences 7, 2006.

[21] K. Kuk: Artificial intelligence in process of collecting and analyzing data within police works, *Nauka, bezbednost, policija*, 20(3):131-48, 2015.

[22] J. C. Riquelme, J. S. Aguilar, M. Toro: A decision queue based on genetic algorithms: axis-paralle classifier versus rotated hyperboxes, *Computational Intelligence and Applications*, pp. 123-128, 1999.

[23] S. O. Danso: *An Exploration of Classification prediction techniques in data mining: the insurance domain*, Master Degree Thesis, Bournemouth University, 2006.

[24] C. Bhawe: Big data classification using decision trees on the cloud, Master's Projects Paper 317, http://scholarworks.sjsu.edu/etd_projects/317, 2013.

[25] G. Šimić, Z. Jeremić, E. Kajan, D. Randjelović, A. Presnall: A Framework for Delivering e-Government Support, *Acta Polytechnica Hungarica*, 11(1), pp. 79-96, 2014.

[26] S. M. Weiss, N. Indurkhya: *Predictive data mining: a practical guide,* Morgan Kaufmann, 1998.

[27] M. A. Hall, I. H. Witten, E. Frank: *Data Mining: Practical machine learning tools and techniques*, Kaufmann, Burlington, 2011.

[28] R. E. Schapire*: The boosting approach to machine learning: An overview*, In Denison DD, Hansen MH, Holmes C, Mallick B and Yu B, eds. Nonlinear estimation and classification, Springer New York, pp. 149-171, 2003