International Journal of Advanced Culture Technology Vol.9 No.4 274-281 (2021)
DOI https://doi.org/10.17703/IJACT.2021.9.4.274

IJACT 21-12-32

Proposal Self-Assessment System of AI Experience Way Education

[1]Kibbm Lee, [2]Seok-Jae Moon, [3]Jong-Yong Lee

[1]Graduate School of Smart Convergence KwangWoon University, Seoul, Korea
[2]Professor, Institute of Information Technology, Kwangwoon University, Seoul, Korea
[3]Ingenium College of Liberal Arts, KwangWoon University, Korea
{kibbmcc, msj8086, jyonglee}@kw.ac.kr

Abstract

In the field of artificial intelligence education, discussions on the direction of artificial intelligence education are actively underway, and it is necessary to establish a foundation for future information education. It is necessary to design a creative convergence teaching-learning and evaluation method. Although AI experience coding education has been applied, the evaluation stage is insufficient. In this paper, we propose an evaluation system that can verify the validity of the proposed education model to find a way to supplement the existing learning module. The core components of this proposed system are Assessment-Factor, Self-Diagnosis, Item Bank, and Evaluation Result modules, which are designed to enable system access according to the roles of administrator, instructors and learners. This system enables individualized learning through online and offline connection.

Keywords: *AI education, Experience education, Assessment System, AI literacy*

1. INTRODUCTION

Recently, various platforms and programs for experiencing AI are being distributed [1][2]. There are various block-based coding platforms such as Teachable Machine [4], Entry [5], Scratch-based ML4kids [6], and App Inventor [7], which are integrated into curricula for primary and secondary school students. Although the need for artificial intelligence education for elementary, middle and high school (K-12) is emerging, according to a report [3] conducted for AI education, there are currently insufficient instructors for K-12 students, and the concept of artificial intelligence for K-12 students is difficult to understand. There is also a limit to the unfamiliarity of new concepts, algorithms, and understanding of professional mathematics. Therefore, in this paper, we propose an evaluation system that can evaluate learners' understanding of the AI coding experience education curriculum [8] to enhance the digital competitiveness of instructors and learners. The proposed system consists of the Assessment-Factor module, Self-Diagnosis module, Item Bank module, and Evaluation Result module as core elements and the evaluation system is designed based on the item response theory. Each component reflected in the evaluation factors, which can be self-evaluated by leaners, and the results can be managed after evaluating education based on the Item bank. As a function of this system, it is designed so that the feedback received by the learner and the report shown to the learner and the instructor about the evaluation result are different. The learner is the result of the feedback on the participating Quiz, and the instructor is

Manuscript received: August 11, 2021 / revised: November 2, 2021 / accepted: December 2, 2021
Corresponding Author: msj8086@kw.ac.kr
Tel: + 82-01-4727-6815, Fax: +82-2-916-4751
Author's affiliation: Institute of Information Technology, Kwangwoon University, Korea

provided with a report of the evaluation result as feedback according to each individual learner's evaluation factors and the number of evaluations. It designed data tables for instructor education subjects, learner learning subjects, Quiz production, individual learner evaluation, and system login. Roles are divided into administrator, instructors and learners. The administrator manages the evaluation elements, the learner joins through the class, and the instructor manages only the learners of the class in charge of the subject. Through the responsive Web, the learner's courses, courses, and evaluation records are recorded from the data storage according to the request of the client, thereby providing feedback for instructors and learners to reach their goals. The structure of this paper is as follows. Chapter 1 describes the introduction, Chapter 2 describes the proposed system. Chapter 3 describes the comparison with other systems. Finally, Chapter 4 describes the conclusion.

2. PROPOSAL SYSTEM

2.1 System Overview

<Figure1> is the configuration of the system proposed in this paper.

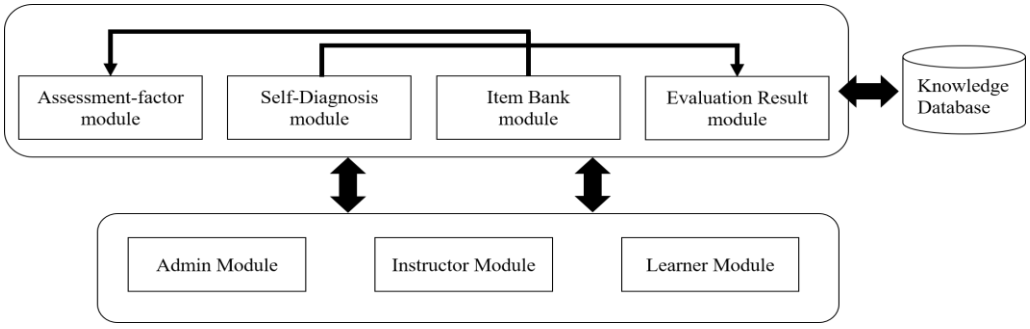

Figure 1. System configuration

- Assessment-Factor module: This module manages the assessment factors for AI questions. As a measure to check the response of the item, it is used to select the evaluation factor applied to the item when the question is created in the Item Bank module. The administrator can insert, modify or delete AI cognitive elements in the module. In this paper, we design and manage AI cognitive factors to be used as evaluation factors for questions.

- Self-Diagnosis module: This is a self-diagnosis module. The module is exposed as a menu that learners can select when they log in. The learners themselves can selectively proceed to access their understanding of the class. The results of performing the module are stored as the Evaluation Result module.

- Item Bank module: This module manages for a repository of test items. The items are of multiple-choice format. The instructor makes the quiz questions to be evaluated, and marks the correct answer. If necessary, an image can be added when creating an item. An evaluation system is constructed based on Item response theory for objective learner level evaluation. For each generated quiz question, it is managed by matching the evaluation elements managed by one Assessment-Factor module.

- Evaluation Result module: A module where evaluation results are stored and managed. The evaluation results will generate a report that allows learners and instructors to view the results through the module. The evaluation results of the learning rounds are accumulated, and the average of each evaluation element and the individual score are converted and managed.

2.2 Assessment-Factor

The Assessment-Factor module selects the AI recognition elements as the registration of the problem, analysis and evaluation of information, organization and creation of information. First, the ROP (Recognition of the problem) is the ability to understand the problem using ICT and to grasp the information necessary to solve it. Second, AEI (Analysis and evaluation of information) refers to the ability to determine the validity, accuracy, reliability, and usefulness of information needed to solve problems using ICT. Third, OCI (Organization and creation of information) evaluates the ability to create new information by systematically organizing, integrating, and designing collected information using ICT.

```
Algorithm1: assessmentFactor
1  Input: models instance reference
2  Output: models object
3  BEGIN:
4     from import models
5     class AiFactor(models.Model):
6        aifactor = models.CharField(max_length=100)
7        return aifactor
8  END:

9  data model creation cell command:
10 py manage.py makemigrations
11 py manage.py migrate
12 py manage.py shell

13 Input: afact[] = {'rop', 'aei', 'oci'}
14 Output: save in repository
15 BEGIN:
16    from assessmentai.models import AiFactor
17    strafact = input('aifactor element: ')
18    IF strafact == afact[0] THEN:
19       modelCreatequery = AiFactor(aifactor='rop')
20       modelCreatequery.save()
21    ELSEIF strafact == afact[1] THEN:
22       modelCreatequery = AiFactor(aifactor='aei')
23       modelCreatequery.save()
24    ELSEIF strafact == afact[2] THEN:
25       modelCreatequery = AiFactor(aifactor='oci')
26       modelCreatequery.save()
27 AiFactor.objects.all()
28 END:
```

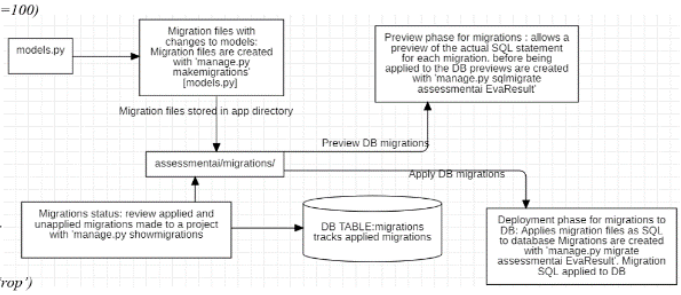

Figure 2. Assessment-Factor table design and field value input

Assessment Factor management is designed to be accessible to administrators and has the ability to add, edit and delete items. The evaluation elements designated by the administrator are designated as evaluation items for each quiz question when the instructor ask questions. < Figure 2> is an algorithm that creates tables and stores data so that the database can be used in the assessmentFactor model. Line 3 to 8 create a table by creating a model with aifactor property through a class called AiFactor. The workflow starts when you add or modify a model in the models.py file. In line 9~11, use makemigrations related command to create migrations. Running the command will scan the models.py file for all apps declared in installed_apps. When it detects a change to the models.py file, it creates a new migration file for the app. The makemigrations command creates files for performing table operations. Lines 13~28 receive the aifactor element value and execute a query to input it into the model, and the aifactor value is entered in the table. In this paper, ROP, AEI and OCI designed in evaluation factors are inserted and managed as field values of aifactor.

2.3 Feedback on correct answers to questions

The self-diagnosis module stores information based on the learner's learning evaluation in the diagnostic test module. Learners select a subject and conduct self-diagnosis in the instructor's class.

< Figure 3> is an algorithm that get the result of the correct answer to the current question. At the same time, it provides immediate feedback to the learner as to whether the answer is correct or wrong. In line 4~7, if the answer is the same as the id of the clicked option, the line 12~15 code displays green in the contes of the option that corresponds to the correct answer among the options. Line 8~10 are marked in red if the answer is incorrect.

This is designed to enable learning at the same time as Quiz because it can tell what the answer is right away when the question is correct or wrong. In addition, correct is marked on the indicator as many questions at the bottom. The indicator at the bottom indicates whether the question is incorrect or not in the entire question at the same location, even when the question changes. Lines 16 to 22 compare the ai-Factor of the item and increase the item by 1 and increase the attempt by 1. Lines 23~27 mark already-answered questions that have already been answered to prevent duplicate responses to the questions.

```
Algorithm2 : answer_indicator
1  Input: quiz list, option list, modelSelectquery, current_answer
2  Output: result of current attempt question
3  BEGIN
4   IF clicked option==answer THEN:
5     update: set the green color to the CORRECT option
6     update: add the indicator to the CORRECT mark
7     currect_answer = currect_answer + 1
8   ELSE:
9     update: set the red color to the WRONG option
10    update: add the indicator to the WRONG mark
11    wrong_answer = wrond_answer + 1
12   REPEAT:
13   IF option.length == answer THEN:
14     option[i].id to CORRECT mark
15   UNTIL i < option.length
16  IF modelSelectquery == afact [0] THEN:
17    rop = rop + 1
18  ELSEIF modelSelectquery == afact[1] THEN:
19    aei = aei + 1
20  ELSEIF modelSelectquery == afact[2] THEN:
21    oci = oci + 1
22  Attempt = Attempt
23  REPEAT:
24    update: option [i] to 'already-answered' mark
25    i = i + 1
26  UNTIL i < option.length
27  END
```

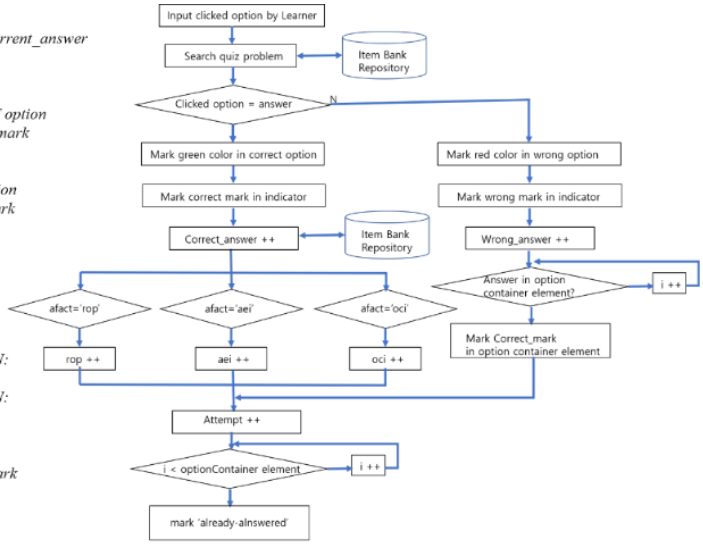

Figure 3. Feedback on correct answers to questions

2.4 Available Question

```
Algorithm3 : AvailableQuestion
1  Input: classId, question list, quiz list
2  Output: available questions list
3  BEGIN
4    classId = input('class_id: ')
5    IF modelSelectquery == classid THEN:
6    initialized all the variables
7      ...
8    totalQuestion = quiz.length
9      REPEAT:
10       IF questionCounter ≠ 0 THEN:
11         IF totalQuestion > quiz.length THEN:
12           REPEAT:
13             availableQuestions(quiz[i])
14             i = i + 1
15           UNTIL i < totalQuestion
16         ELSE:
17           return quizOver
18       ELSE:
19         return false
20     j = j + 1
21     UNTIL j < classId.count
22 END
```

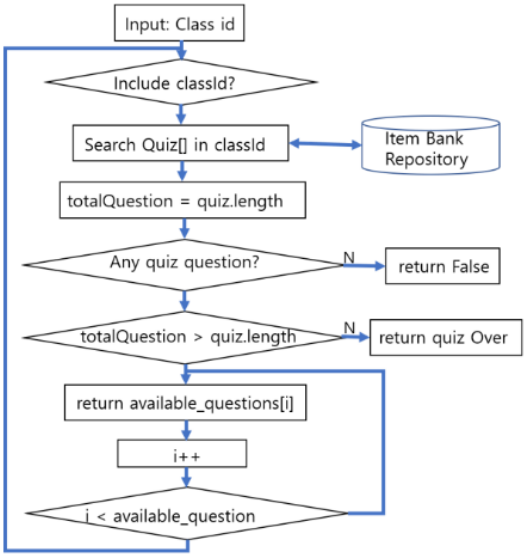

Figure 4. Get available question

< Figure 4> is an algorithm that brings available questions. Lines 3~5 receive the class id input by the user and search the list of quiz questions corresponding to the class condition as in lines 20~22 from the item bank

repository. Line6 initializes the necessary variables in the function. lines 8 to 19 compare the total number of questions with the number of questions if there are questions. If the total number of questions is less than the number of questions, quizOver is returned. As in line 11~15, if the total number of questions is larger than the number of questions issued, it can bring questions normally, so it brings available questions. The index of the questions in quiz array is incremented by 1, and the quiz values in the array are repeated in sequence. This loop is repeated until the length of totalQuestion is small.

2.5 Get new question and evaluation factor

```
Algorithm4 : newQuestion
1  Input: question_limit, available_question list, available_options list, evaluation_factor
2  Output: current_question, available_options list
3  BEGIN
4    questionLimit = input("questionLimit:")

5    IF questionCounter ≠ questionLimit THEN:
6       current_question = question.random()
7       index1 = available_question.index
8       remove: the 'question_index' from the available_question list
9    IF (img.length ≠ 0) THEN:
10      return img
11   evaluation_factor = modelSelectquery

12   optionLen = options.length
13   REPEAT
14      return available_options[i]
15      option_index = option.random()
16      index2 = availe_options.index
17      remove: the 'option_index' from the available_option Array
18      i = i + 1
19   UNTIL(i < optionLen)
20   question_counter = question_counter + 1
21 ELSE:
22   return quizOver
23 END
```

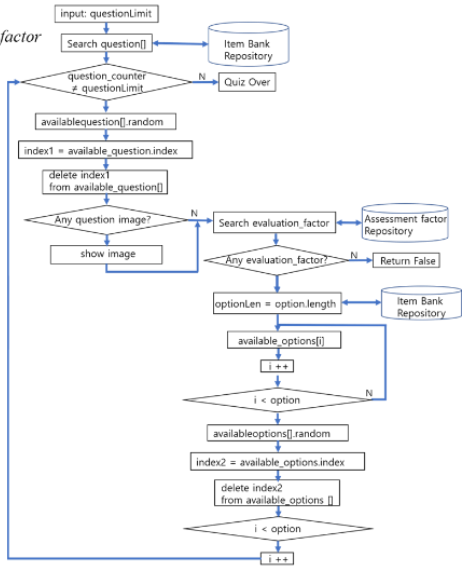

Figure 5. Get questions, options and evaluation factor

< Figure 5> is an algorithm for numbering problems and fetching problems and options. In line 4~5, input the number of questions to be asked from the instructor and compare it with the total number of questions. In line6, the available questions are randomly fetched from the available questions array. Lines7~8 delete the randomly imported index to prevent repeating the problem. Lines 9 to 10 are imported if there is an image in the question. In line11, the evaluation factor of the corresponding question is retrieved from the assessment factor repository.

Lines 12~15 get the number of item options. Select options one by one from the available_options array and repeats for the length. Randomly get the available options from the available_options array. Line 13~19 delete the randomly imported index to prevent repeating the question options. Repeat as many times as the number of corresponding item options. When line20 finishes the repetition, the question_counter variable is incremented by 1 to bring the next question. If the number of questions matches the number of questions entered by the instructor, the quiz ends.

2.6 Evaluation Results

< Figure 6> is an algorithm that saves the total score and incorrect answers for each item in the repository after self-diagnosis. Search the previous round records in the repository to see if there are any previous self-diagnostic records on line4. Lines 4 to 7 add 1 to the previous round if there is a round record, and initialize it to 1 if there is no previous record. Line 8 determines if the quiz is over, and if the quiz is over, run lines 9 to

15 to record the results of the corresponding rounds. Line 9 stores the questionLimit as the total number of questions, and line10 stores the number of questions attempted in attempt. In lines 11~12, the number of correct problems is stored in total_correct, and the number of incorrect problems is stored in total_wrong to calculate the next probability. In line 13, the result is stored in a variable by calculating (correctAnswers/questionLimit) * 100 to display a score out of 100. In addition, line 14 stores(correctAnswers/questionLimit) in a variable to know the total score, and in line 15 inserts the round and score, and whether or not each item has an incorrect answer in the repository.

Algorithm5 : quizResult

```
1  Input: round, questionLimit, elements after self-diagnosis
2  Output: save in repository
3  BEGIN
4   IF any previous round==TRUE THEN:
5     round = round + 1
6   ELSE:
7     round = 1
8   IF finish quiz==TRUE  THEN:
9     totalQuestion  =  questionLimit
10    Attempt = try quiz count
11    total_correct  = correct quiz count
12    total_wrong = wrong quiz count
13    percentage = (correctAnswers / questionLimit) * 100
14    total_score = (correctAnswers / questionLimit)
15   insert at repository <- round, score, total_correct, total_wrong
16  END:
```

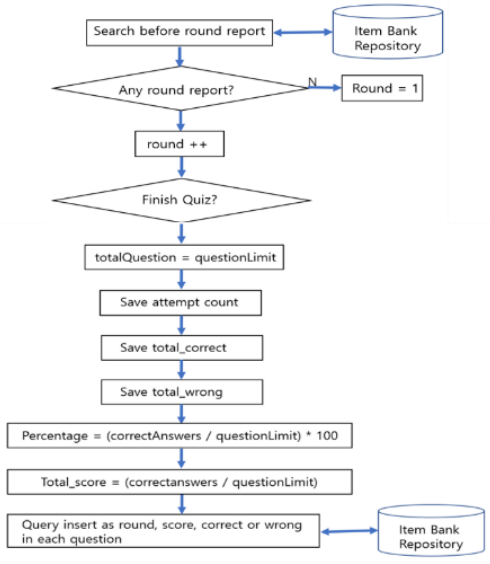

Figure 6. Save evaluation results

2.7 Evaluation Report

Algorithm6 : make Report

```
1 Input: reshandle[]={'perclass','perstu','perstuhistory'}
2 Output: class result object, student result object, round result object
3 BEGIN
2  Result agent(itemLists, handle) {
3    initalize all the variables
4    ...
5    handle = input('handling element:')
5    IF handle =
6      CASE 0:
7        erClass = ER xtract(XMDR, 0);
8        return erclass;
9        break;
10     CASE 1:
11       erEF = ER xtract(XMDR, 1);
12       return erEF;
13       break;
14     CASE 2:
15       erRound = ER xtract(XMDR, 2);
16       return erRound;
17       break;
18   }
19   }
20 END
```

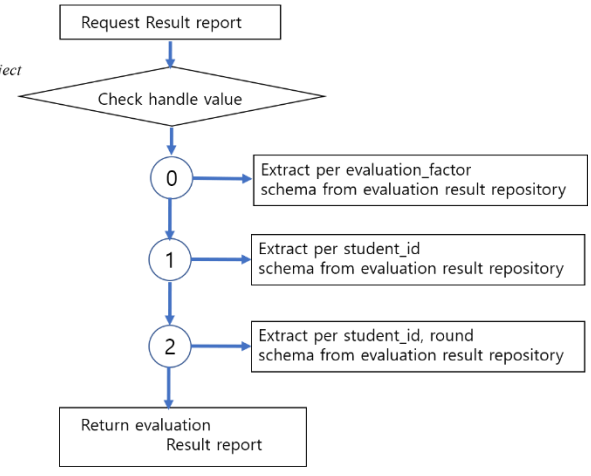

Figure 7. Make evaluation report

< Figure 7> is an algorithm that brings the results of class overall learners and the results of learners' individual analysis by area. Feedback is provided to the instructor so that the change and improvement in understanding of the entire class and individual learners in the class can be made history and inquired. Lines 5

to 20 are designed to select each report using the handle value. The first element is the full report of a class. This shows the overall average for each factor. The second element is the individual learner's report by factor. The third element is the history inquiry by learner.

3. COMPARISON WITH OTHER SYSTEMS

The following <Table 1> is a comparison with other quiz systems.

Table 1. Comparison Analysis with other Quiz App

	Kahoot	Plickers	Proposal System
For self-paced learning	Support	Not support	Support
Quiz format	Multiple choice	Multiple choice	Multiple choice
Limit of multiple choice questions	Variable within 1~4 choice	Variable within 1~4 choice	Variable within 1~N choice
Image upload	Support	Support	Support
Management of assessment elements	Not Support	Not Support	Support
Management of assessment round	Not Support	Not Support	Support
Realtime feedback for leaner	Support	Not Support	Support
Instance feedback for instructor	Incorrect answer and total score for each question	Incorrect answer and total score for each question	Incorrect answer and total score for each question, cumulative comparison

4. CONCLUSION

In this paper, we proposed an assessment system to evaluate learners' understanding of the AI coding experience education curriculum. Through this, individualization learning through online and offline connection is possible. Based on the item response theory, it is applied to the learning evaluation technique, and it is a system that is proposed to intuitively measure performance after learning for elementary and middle school learners. This system supports the ability to adjust the curriculum-based learning difficulty that occurs in the process of evaluating learning between learners and instructors and inquiring results. It is possible to learn at the same time as evaluation by providing correct answer feedback for each question for learners. And after evaluation, it supports the service that can be applied again by modifying the learning model developed to suit the learner's level. Future research, there remains a task to be expanded to apply the achievement evaluation by developing a learning module that combines the AI education model with a test model that includes ability factors other than image classification.

REFERENCES

[1] Ministry of Education_2021 Ministry of Education work plan.hwp (01.26 press release)
[2] Dataset analysis and utilization plan for elementary and middle school AI education, KERIS, 2020.
[3] Ministry of Science and ICT (2019). Artificial intelligence national strategy announced
[4] TeachableMachine: https://teachablemachine.withgoogle.com
[5] EntryLabs: https://playentry.org

[6] ML4kids https://machinelearningforkids.co.uk/
[7] AppInventer https://appinventor.mit.edu/
[8] Lee, Kibbm, and Seok-Jae Moon. "Experience Way of Artificial Intelligence PLAY Educational Model for Elementary School Students." International Journal of Internet, Broadcasting and Communication 12.4, pp232-237, 2020.