An Adaptive Quiz Generation System for Moodle using moosh and LLM

Toshiro KITA[a*], Rwitajit MAJUMDAR[a], Izumi HORIKOSHI[b] & Hiroaki OGATA[c]
[a]*Kumamoto University, Japan*
[b]*Uchidayoko Institute for Education Research, Japan*
[c]*Kyoto University, Japan*
*kita@rcis.kumamoto-u.ac.jp

Abstract: This is a system designed to automate the creation of adaptive quizzes within the Moodle Learning Management System (LMS). Leveraging the moosh command-line tool for Moodle administration and the OpenAI API for content generation, the system provides personalized learning pathways. It consists of an initialization script to set up the Moodle course and initial, and a periodically run script that analyzes learner performance on existing quizzes to generate new, tailored quizzes focusing on areas where the learner demonstrated weakness. Helper functions are encapsulated in a separate module for clarity and reusability. This approach aims to enhance the learning experience by automatically providing remedial or supplementary exercises based on individual learner needs.

Keywords: LMS, Automatic question generation, Personalized learning, Generative AI

1. Introduction

Adaptive learning systems tailor educational content and activities to the specific needs and performance levels of individual learners. Within Learning Management Systems like Moodle, quizzes are a common tool for assessment. However, manually creating personalized follow-up quizzes for each student based on their performance can be time-consuming and impractical, especially in large courses. Various attempts have been made to automatic question generation (Kurdi, et al., 2020), but none have yet been implemented on a general LMS.
This is a system implemented in Python that automates the process of creating personalized follow-up quizzes. It utilizes moosh (Muras, n.d.), a command-line tool that allows scripting of Moodle actions, to interact with the Moodle environment.

For generating new, relevant quiz questions, the system integrates with a Large Language Model (LLM) like OpenAI GPT-4o-mini or GPT-4o. The system operates in two main phases:

1. **Initialization**: Setting up a new Moodle course structure and populating it with initial quizzes.
2. **Adaptive Update Cycle**: Periodically analyzing quiz results and generating supplementary quizzes targeted at the weaknesses identified for each learner (represented by Moodle course sections).

2. System Architecture

All files that make up this system are publicly available in the author's GitHub repository (Kita, n.d).

The system comprises three Python scripts and one XML data file:

- `adaptive_quiz_moosh1.py`: The initialization script.
- `adaptive_quiz_moosh2.py`: The script responsible for the adaptive update cycle.

- `adaptive_quiz_moosh_mod1.py`: A module containing shared functions used by the other scripts.
- `moodle-quest1.xml`: An example Moodle quiz XML file containing initial questions.

2.1 Initialization (adaptive_quiz_moosh1.py)

This script is intended to be run once at the beginning. Its primary functions are:

1. **Course Creation:** It uses `moosh course-create` to create a new Moodle course (e.g., "Demo Course 1") with a specified number of sections (`numsection`). Each section of the Moodle course is intended to have access restricted so that it is visible only to specific users. This allows us to virtually provide personalized content for each individual user.
2. **Initial Quiz Population:** For each section created, it uses `moosh activity-add` to create a new quiz activity (e.g., "Quiz s1", "Quiz s2").
3. **Question Import:** It imports questions from a predefined Moodle XML file (`moodle-quest1.xml`) into each newly created quiz using `moosh question-import`. This ensures all learners start with the same baseline quiz content within their respective sections.

Figure 1. show an example of the result after the Initialization phase.

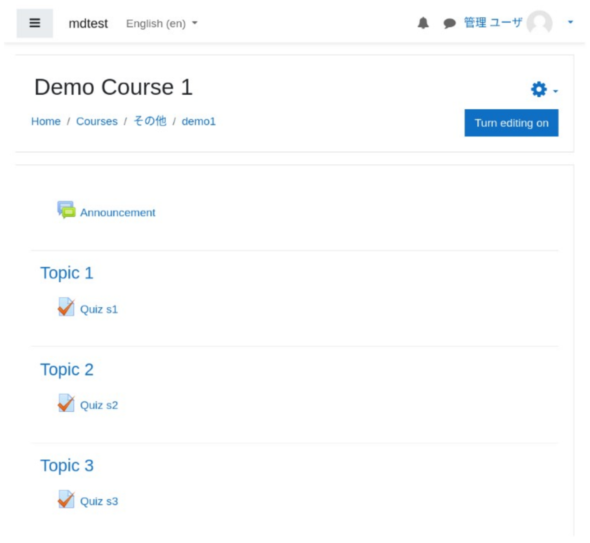


Figure 1. A newly created Moodle course and the quizzes by the script

2.2 Adaptive Update Cycle (adaptive_quiz_moosh2.py)

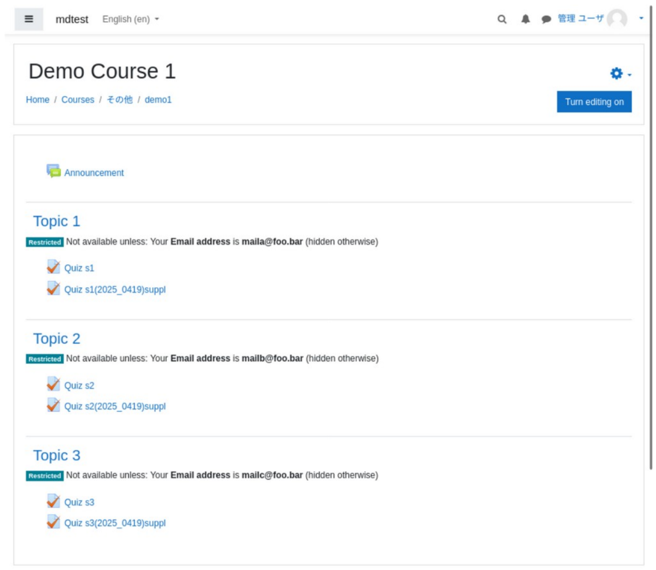
This script is designed to be run periodically (e.g., daily via a cron job). It implements the core adaptive logic:

1. **Identify Target Quizzes:** Uses `moosh gradeitem-list` to retrieve a list of all grade items (including quizzes) in the course. The helper function `get_quizids` (from `adaptive_quiz_moosh_mod1.py`) filters this list to find the IDs of the *original* quizzes (excluding previously generated supplementary quizzes, identified by "suppl" in their name).
2. **Analyze Performance:** For each original quiz ID:
    - It fetches the gradebook report for that specific quiz using `moosh gradebook-export`.
    - The `get_quiz_score` function parses the CSV output to extract scores for each participant.

- The `top_score` function (or alternatively, `calculate_average_score`) determines a representative score for the quiz attempt(s). *Note: The current implementation uses* `top_score`*.*
- The `quiz_result_comment` function analyzes this score. **Crucially, it interprets the score as a set of bit flags**, where each bit position corresponds to a question (bit 0 for Q1, bit 1 for Q2, etc.). A '1' indicates a correct answer, and a '0' indicates an incorrect answer. It generates a textual feedback comment listing the 1-based indices of the incorrectly answered questions (e.g., "Question No.1, Question No.3 are incorrect..."). This assumes a maximum number of questions (`max_questions=4` in the code) and that the total score reflects this bit pattern (e.g., answering Q1 and Q3 correctly out of 4 might yield a score of 1*1* + 0*2* + 1*4* + 0*8* = 5, assuming the `defaultgrade` values 1, 2, 4, 8 seen in the XML).

3. **Generate New Quiz:**
    - The `create_question_xml` function takes the generated feedback comment. It uses this comment, along with the content of the original `moodle-quest1.xml`, as input for the OpenAI API.
    - A carefully constructed prompt asks the LLM to generate a new set of 4 multiple-choice questions in Moodle XML format, specifically targeting the concepts related to the incorrectly answered questions mentioned in the feedback.
    - A second prompt refines the LLM's output to ensure strict adherence to Moodle XML formatting requirements (correct tags, CDATA sections, question naming conventions, general feedback inclusion, MathJax usage, `defaultgrade` values, etc.).
    - The validated XML content is saved to a temporary file.
4. **Add New Quiz to Moodle:**
    - The script determines the correct section ID for the learner using `sectionid_from_quizid` (which parses the section number from the original quiz name like "Quiz s1").
    - It creates a *new* quiz activity in the corresponding section using `moosh activity-add`, naming it distinctively (e.g., "Quiz s1(YYYY_MMDD)suppl").
    - It imports the questions from the newly generated temporary XML file into this new quiz using `moosh question-import`.

Figure 2. show an example of the result after the first adaptive update cycle.



Demo Course 1

Topic 1

Restricted Not available unless: Your **Email address** is **maila@foo.bar** (hidden otherwise)

Quiz s1

Quiz s1(2025_0419)suppl

Topic 2

Restricted Not available unless: Your **Email address** is **mailb@foo.bar** (hidden otherwise)

Quiz s2

Quiz s2(2025_0419)suppl

Topic 3

Restricted Not available unless: Your **Email address** is **mailc@foo.bar** (hidden otherwise)

Quiz s3

Quiz s3(2025_0419)suppl

Figure 2. Supplementary quizzes automatically added to each section

3. Workflow and Usage

1. Configure the `MOODLE_DIR` constant in `adaptive_quiz_moosh_mod1.py` and place the OpenAI API key in `../keys/openai-key.txt`.
2. Prepare an initial Moodle quiz XML file (`moodle-quest1.xml`).
3. Run `adaptive_quiz_moosh1.py` once to create the Moodle course and initial quizzes for each section/learner.
4. Set up `adaptive_quiz_moosh2.py` to run periodically (e.g., using `cron`) to analyze results and generate supplementary quizzes. Learners will find these new quizzes appearing in their respective course sections over time, focusing on areas they need to practice.

4. Discussion

This system provides an automated framework for basic adaptive quizzing in Moodle. By identifying incorrectly answered questions (based on a bitwise score interpretation) and prompting an LLM to generate related questions, it offers a degree of personalization.

Strengths:

- **Automation:** Significantly reduces the manual effort required to create personalized follow-up quizzes.
- **Personalization:** Tailors supplementary content to address specific areas of weakness for each learner.
- **Leverages Existing Infrastructure:** Works within the standard Moodle environment using the `moosh` tool.
- **AI-Powered Content:** Utilizes powerful LLMs for generating relevant new questions, potentially covering a wide range of topics and difficulty levels based on the initial XML and feedback.

Limitations and Potential Enhancements:

- **Adaptation Logic:** The current adaptation is based solely on identifying incorrect answers via bit flags derived from the score. This requires careful setup of `defaultgrade` values (e.g., 1, 2, 4, 8...) in the initial XML and assumes the final score directly reflects the bitmask. This is a fragile assumption. More robust methods could involve analyzing individual question attempts if Moodle's reporting allows, or using more sophisticated scoring analysis.
- **Scoring Interpretation:** Using `top_score` might not accurately reflect overall understanding if multiple attempts are allowed. `calculate_average_score` is provided as an alternative but might also not capture the nuances of improvement over time.
- **AI Reliability:** LLM output quality can vary. Generated questions might be inaccurate, irrelevant, or improperly formatted despite the refinement prompt. Error handling for XML validation and content relevance could be added.
- **Fixed Number of Questions:** The system currently generates a fixed number (4) of new questions. This could be made dynamic.

Future work could involve implementing more sophisticated adaptive algorithms, improving the robustness of score analysis, and implementing better error checking for the AI-generated content.

5. Conclusion

The described system presents a practical approach to implementing adaptive quizzes in Moodle by integrating the `moosh` command-line tool with the generative capabilities of the

OpenAI API. While the current adaptation logic based on bitwise score interpretation is relatively simple, the framework successfully automates the process of analyzing learner performance and generating targeted supplementary quiz content. This demonstrates the potential of combining existing LMS tools with AI to create more personalized and potentially more effective learning experiences.

References

Kurdi, G., Leo, J., Parsia, B., Sattler, U., & Al-Emari, S. (2020). *A systematic review of automatic question generation for educational purposes.* International Journal of Artificial Intelligence in Education, 30, 121-204. doi: https://doi.org/10.1007/s40593-019-00186-y

Muras, T. (n.d.). *moosh*. https://moosh-online.com/

Kita, T. (n.d.). *moodle-quiz-autoupdate.* https://github.com/kita-toshihiro/moodle-quiz-autoupdate