Kashihara, A. et al. (Eds.) (2024). Proceedings of the 32nd International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

PERS: A Personalized Recommender System for Student-Generated Questions in Programming Courses

Pham-Duc THO*
International School, Vietnam National University, Hanoi, Vietnam
*thopd@vnu.edu.vn

Abstract: This study introduces PERS (Personalized Exercise Recommender System), integrating Student-Generated Questions (SGQ) with a recommender system for programming courses. Implemented in a 10-week introductory programming course with 395 undergraduates (200 using PERS, 195 as control), PERS uses collaborative filtering to suggest personalized exercises. This study addresses the impact of PERS on student engagement and learning outcomes, the effectiveness of its recommender system, and students' perceptions of the system. Results show significant improvements in exercise completion rates, assignment scores, and final exam performance for PERS users compared to the control group. The system demonstrated high recommendation relevance and user satisfaction, suggesting its potential to enhance personalized learning in programming education.

Keywords: programming education, student-generated questions, recommender systems, personalized learning, collaborative filtering

1. Introduction

Computer programming courses are fundamental to computer science education, yet they often face challenges of high dropout rates and student disengagement. To address these issues, educators have explored various pedagogical strategies, including Student Question-Generation (SGQ) exercises. While SGQ has shown promise in enhancing student learning and engagement, existing systems often lack personalization, potentially leading to suboptimal learning experiences for students of varying skill levels and interests.

PERS builds upon the cognitive apprenticeship model (Collins et al., 1991) and adaptive learning theory (Aleven et al., 2016), integrating SGQ with collaborative filtering to create a dynamic, personalized learning environment. This study addresses the following research questions:

1. How does PERS impact student engagement and learning outcomes in programming courses?
2. How effectively does the recommender system match students with appropriate exercises?
3. What are students' perceptions of the PERS system?

2. Related work

2.1 Student-Generated Questions in Programming Education

Student-Generated Questions (SGQ) have gained traction as an effective pedagogical strategy in various disciplines, including computer science. Lai et al. (2017) demonstrated that SGQ activities in programming courses can enhance students' algorithmic thinking and problem-solving skills. Hsu and Wang (2018) further showed that integrating SGQ with game mechanics can promote engagement and motivation in learning programming concepts.

In the context of programming education, systems like CodeWrite (Denny et al., 2011) and PIPLS (Lai & Tho, 2016) have been developed to support SGQ activities. These platforms allow students to create and answer coding-related questions, fostering active learning and peer assessment. However, most existing systems lack personalization features, potentially limiting their effectiveness for diverse student populations.

2.2 Recommender Systems in E-Learning

Recommender systems have been widely adopted in e-learning environments to enhance personalized learning experiences. Manouselis et al. (2011) provided a comprehensive overview of recommender systems in technology-enhanced learning, highlighting their potential to support self-directed learning and improve student engagement.

In recent years, collaborative filtering approaches have shown promise in educational contexts. Dwivedi and Bharadwaj (2013) demonstrated the effectiveness of a hybrid recommender system for personalizing learning object recommendations. More recently, Liu et al. (2022) reviewed deep learning-based recommender systems in e-learning, emphasizing their ability to capture complex patterns in student behavior and preferences.

Despite these advancements, the application of recommender systems specifically to SGQ in programming education remains largely unexplored. This gap presents an opportunity to leverage the benefits of both SGQ and personalized recommendations to enhance programming education.

Recent advancements in deep learning-based recommender systems have shown promise for e-learning applications. Liu et al. (2022) demonstrated improved personalization using neural collaborative filtering in educational contexts, while Zhang et al. (2023) proposed a knowledge-aware attentive neural network specifically for programming exercise recommendations. These developments suggest potential avenues for enhancing PERS in future iterations.

2.3 PERS: Bridging SGQ and Recommender Systems

Our previous work introduced PERS as a novel approach to combine SGQ with a recommender system tailored for programming courses. PERS builds upon the collaborative filtering principles outlined by Bobadilla et al. (2013), adapting them to the unique context of programming education and SGQ exercises.

The current study extends previous work by implementing and evaluating PERS in a real educational setting. This approach aligns with recent trends in educational technology research, which emphasize the importance of field studies to assess the effectiveness of new learning tools (Crogman & Crogman, 2018).

By integrating SGQ, recommender systems, and rigorous evaluation in an actual course environment, our work aims to contribute to the ongoing efforts to enhance personalized learning experiences in programming education. The findings from this study will inform future research on adaptive learning systems and guide the development of more effective tools for programming instruction.

3. PERS System and Study Design

3.1 PERS Implementation

PERS is designed as a web-based application that integrates student-generated questions (SGQ) with a personalized recommender system. The system architecture consists of three main components:

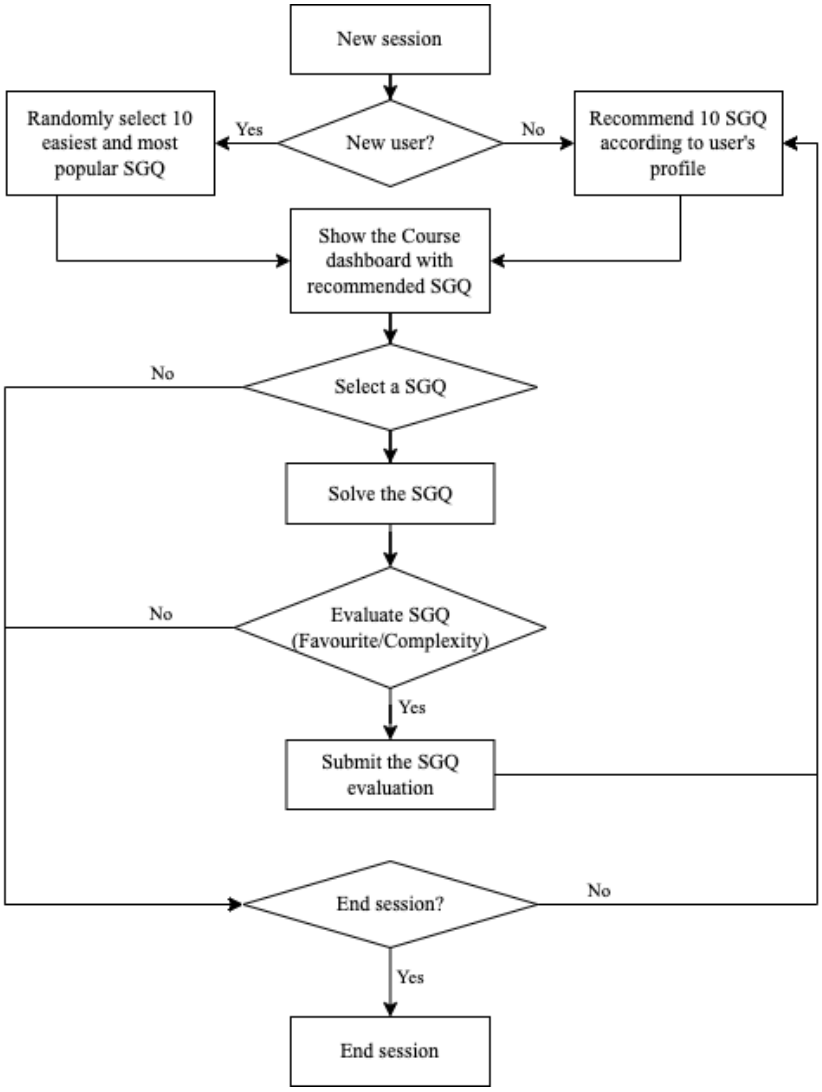

Figure 1. The main workflow of the interaction between the student and the proposed system

1. User Interface: Provides separate interfaces for students and instructors. Students can create, attempt, and rate SGQ exercises, while instructors can monitor progress and manage the exercise bank.
2. SGQ Exercise Bank: Stores all student-generated exercises along with metadata such as difficulty level, topics covered, and user ratings.
3. Recommender Engine: Utilizes collaborative filtering to suggest appropriate exercises to students based on their profile and interaction history.

The core of PERS is its collaborative filtering algorithm, which operates as follows:

1. User-Item Matrix: Constructs a matrix where rows represent students and columns represent SGQ exercises. Each cell contains the student's rating for an exercise (if attempted) or is empty otherwise. To construct the CF model, this study begin with a set of n students denoted as $\{u_1, u_2, u_3, \dots, u_n\}$, alongside a set of *m* SGQ exercises represented as $\{e_1, e_2, e_3, e_4, \dots, e_m\}$. Each student's evaluation of an SGQ exercise is based on two distinct criteria:
    - Favourite ($I \in \{1, 2, 3, 4, 5\}$): This quantifies the degree of interest the student had in the exercise. A score close to 1 signifies minimal interest, while a score of 5 indicates significant interest.

- Complexity ($C \in \{1, 2, 3, 4, 5\}$): This indicates the perceived complexity level of the exercise from the student's standpoint. Specifically, a value of 1 corresponds to low complexity, and 5 to high complexity.

These two criteria, favourite (I) and complexity (C), serve as the foundation for assessing students' responses to the SGQ exercises.

2. Similarity Computation: Calculates the similarity between students using the Pearson correlation coefficient based on their exercise ratings and interaction patterns.
3. Neighborhood Formation: Identifies a set of similar students (neighbors) for each active user.
4. Rating Prediction: Predicts ratings for unattended exercises using the weighted average of ratings from similar users.
5. Recommendation Generation: Suggests the top N exercises with the highest predicted ratings to each student.

3.2 Study Setup

To evaluate PERS, we conducted a quasi-experimental study with the following parameters:

- Participants: A total of 395 undergraduate students were involved in this study.
    - Intervention Group: 200 students enrolled in the current year's introductory programming course, using PERS.
    - Control Group: Historical data from 195 students who took the same course in the previous year without PERS.
- Duration: One 10-week academic quarter.
- Course Structure: Both groups were taught by the same instructor using identical course materials, excluding the PERS intervention.
- SGQ Exercise Bank: Initially seeded with 100 exercises, growing to over 500 by the end of the study as students contributed new questions. To ensure quality, all student-generated questions were reviewed by teaching assistants before being added to the exercise bank. Questions were tagged with relevant topics and difficulty levels, which were then used by the recommender system to match exercises to students' current skill levels and learning goals.

We conducted pre-tests to ensure no significant differences in initial programming knowledge between the groups ($t(393) = 1.24$, $p = 0.216$). Students in the intervention group were introduced to PERS in the first week of the course and encouraged to use the system throughout the quarter as a supplement to regular coursework. They were instructed on how to create, attempt, and rate SGQ exercises. Student data was anonymized before analysis, and the recommender system used encrypted user IDs to protect privacy. Participants provided informed consent and could opt out of data collection while still using PERS.

3.3 Data Collection and Evaluation Metrics

We collected data through the PERS platform and additional surveys. The following metrics were used to assess the system's effectiveness:

1. Student Engagement:
- Number of exercises attempted per week
- Time spent on the platform (tracked through system logs)
- Self-reported engagement levels (collected via mid-term and end-of-term surveys)

2. Learning Outcomes:
- Performance on weekly programming assignments
- Scores on mid-term and final exams
- Self-assessment of skill improvement (collected via surveys)

3. System Performance:
- Recommendation accuracy (measured by student ratings of recommended exercises)
- Computational efficiency (time to generate recommendations)

Additionally, we conducted five focus groups, each with 4 students, to gather qualitative feedback on their experience with PERS.

To establish a baseline for comparison, we used historical data from the same course taught in the previous year without PERS. This allowed us to compare learning outcomes and engagement levels between PERS users and non-PERS users.

Throughout the study, we adhered to ethical guidelines for educational research, obtaining informed consent from all participants and ensuring data privacy and confidentiality.

4. Results

The implementation of PERS led to significant improvements in student engagement and learning outcomes:

1. Exercise Completion: PERS users attempted significantly more exercises per week ($M = 7.3$, $SD = 2.1$) compared to the control group ($M = 4.2$, $SD = 1.8$), $t(393) = 14.62$, $p < 0.001$, representing a 73.8% increase.

Figure 2 illustrates the weekly exercise completion rates for PERS users compared to non-PERS users throughout the 10-week course. As shown, PERS users consistently completed more exercises each week, with the gap widening as the course progressed.

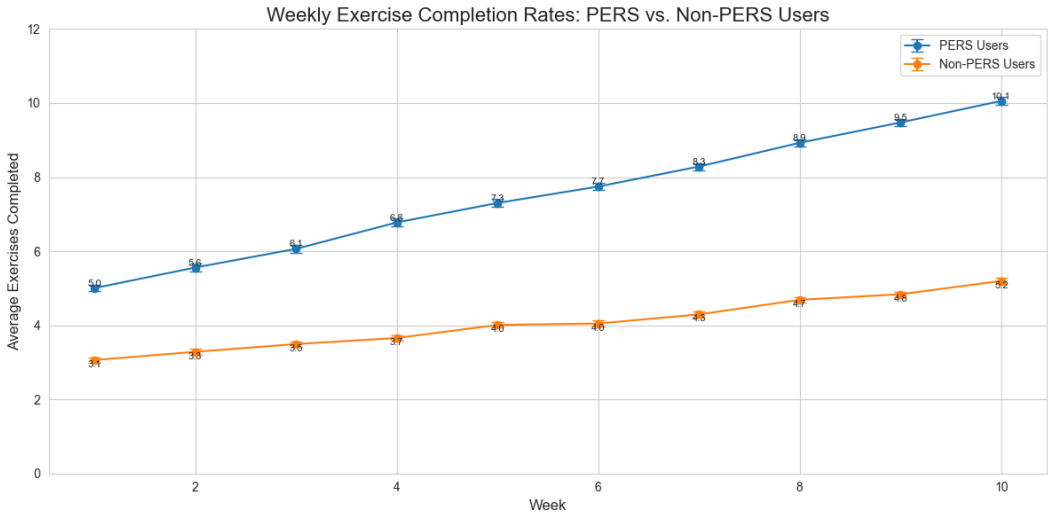

Figure 2. Weekly exercise completion rates for PERS users vs. non-PERS users

The graph clearly demonstrates the sustained engagement of PERS users, with their exercise completion rates remaining high throughout the course, while the control group's rates tended to decline over time.

2. Time Spent: By the final week, PERS users were spending an average of 5.2 hours per week ($SD = 1.8$) interacting with the system, in addition to their regular coursework.

3. Self-reported Engagement: In the end-of-term survey, 78% of students reported feeling "more engaged" or "much more engaged" with the course material compared to their other programming courses.

4. Learning Outcomes:
   - Weekly Programming Assignments: PERS users scored an average of 85% ($SD = 7.5$) on weekly assignments, compared to 76% ($SD = 8.2$) for the control group, $t(393) = 10.95$, $p < 0.001$, Cohen's $d = 1.14$.
   - Final Exam Performance: As illustrated in Figure 3, the average score on the final exam was significantly higher for PERS users ($M = 78.4\%$, $SD = 8.2$)

compared to the control group (M = 73.0%, SD = 9.1), t(393) = 5.87, p < 0.001, Cohen's d = 0.59, indicating a medium effect size.

To visualize the difference in final exam performance between PERS users and the control group, Figure 3 presents a comparison of their scores.

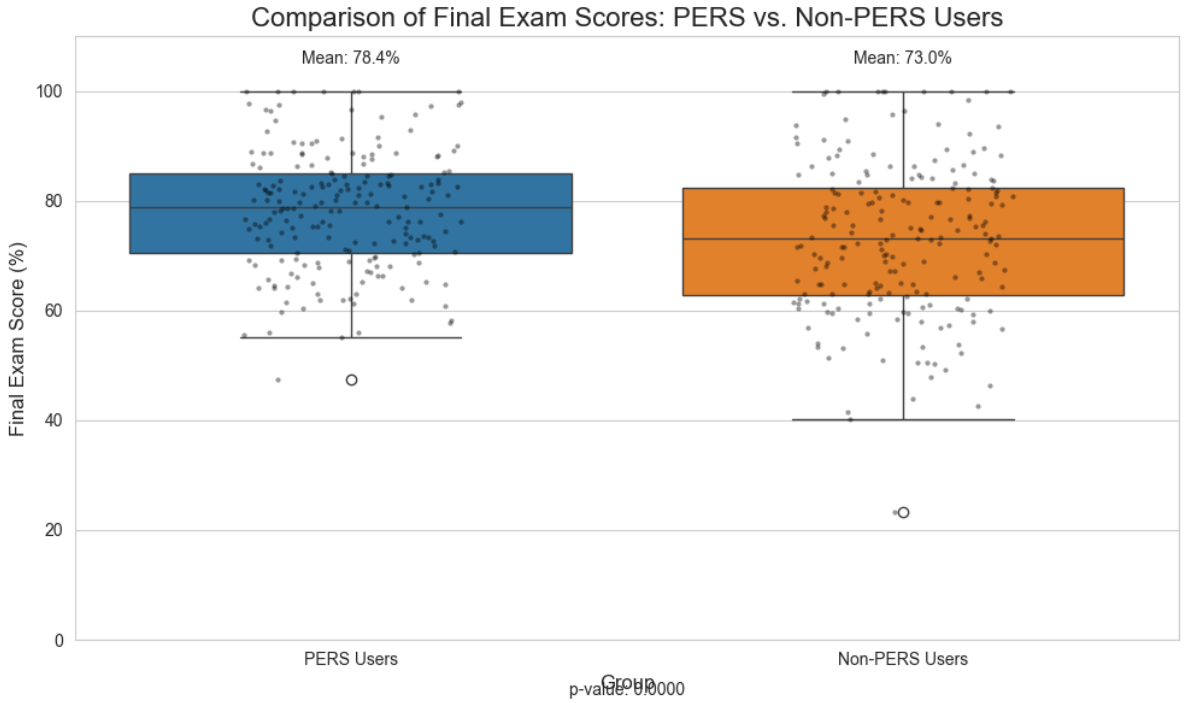

Figure 3. Comparison of final exam scores (PERS vs. non-PERS)

As evident from Figure 3, PERS users achieved higher scores on the final exam, with a greater proportion scoring in the upper ranges. This visual representation supports the statistical findings of improved learning outcomes for PERS users.

5. Skill Improvement: 82% of students reported feeling more confident in their programming skills by the end of the course. This correlated strongly with the number of SGQ exercises attempted (r = 0.73, p < 0.001).

6. System Performance:
   - Recommendation Relevance: On average, students rated the relevance of recommended exercises as 4.2 out of 5 (SD = 0.8).
   - Exercise Completion Rate: 73% of recommended exercises were attempted by students.
   - Computational Efficiency: The average time to generate a set of recommendations for a student was 0.8 seconds (SD = 0.2).

5. Discussion

Results show PERS's potential to enhance student engagement and learning outcomes in programming courses. Increased engagement levels suggest personalized SGQ recommendations effectively motivate more frequent programming practice. Improved learning outcomes, especially for lower-performing students, align with previous research on adaptive learning systems.

Qualitative feedback revealed appreciation for exercise variety, personalized challenge, and increased motivation. One student noted, "It felt like the system really understood my skill level and pushed me just the right amount."

Limitations include potential confounding factors from historical control data, possible Hawthorne effect, limited generalizability, and self-report bias. Future research should address these through randomized controlled trials and more objective measures. Some students

(12%) felt overwhelmed, suggesting future iterations should consider individual workload preferences.

6. Conclusion and Future Work

This study introduced PERS, a Personalized Exercise Recommender System for student-generated questions in programming courses. Key findings include increased engagement, improved learning outcomes, successful personalized learning experiences, and support for struggling students.

Results highlight the potential of combining student-generated questions with personalized recommendations to enhance programming education. Future work should focus on long-term retention studies, integration with existing systems, improved recommendation techniques, scalability testing, and addressing ethical considerations.

Additional areas for exploration include adaptive difficulty scaling, peer learning features, and advanced analytics tools for instructors. As AI-driven systems like PERS evolve, careful consideration of their ethical implications and impact on educational equity will be crucial.

Acknowledgements

This research is funded by International School, Vietnam National University, Ha Noi, Viet Nam

References

Aleven, V., McLaughlin, E. A., Glenn, R. A., & Koedinger, K. R. (2016). Instruction based on adaptive learning technologies. *Handbook of research on learning and instruction*, 2, 522-560.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109-132.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2), 77-101.

Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American educator*, 15(3), 6-11.

Crogman, H., & Crogman, M. T. (2018, May). Modified generated question learning, and its classroom implementation and assessment [Article]. *Cogent Education, 5*(1), 1-41, Article 1459340.

Denny, P., Luxton Reilly, A., Tempero, E., & Hendrickx, J. (2011). CodeWrite: supporting student-driven practice of java. 42nd ACM technical symposium on Computer science education,

Dwivedi, P., & Bharadwaj, K. K. (2013). Effective trust-aware e-learning recommender system based on learning styles and knowledge levels. *Journal of Educational Technology & Society, 16*(4), 201-216.

Hsu, C. C., & Wang, T. I. (2018, 2018/06/01/). Applying game mechanics and student-generated questions to an online puzzle-based game learning system to promote algorithmic thinking skills. *Computers & Education, 121*, 73-88. https://doi.org/10.1016/j.compedu.2018.02.002

Lai, C.-H., Tho, P.-D., & Liang, J.-S. (2017). Design and evaluation of question-generated programming learning system. 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI),

Lai, C. H., & Tho, P.-D. (2016). Development of a Programming Learning System Based on a Question Generated strategy. 24th International Conference on Computers in Education, India.

Liu, T., Wu, Q., Chang, L., & Gu, T. (2022, 2022/12/01). A review of deep learning-based recommender system in e-learning environments. *Artificial Intelligence Review, 55*(8), 5953-5980. https://doi.org/10.1007/s10462-022-10135-2

Manouselis, N., Drachsler, H., Vuorikari, R., Hummel, H., & Koper, R. (2011). Recommender systems in technology enhanced learning. *Recommender systems handbook*, 387-415.

Zhang, Y., Yin, H., Chen, T., & Chen, H. (2023). KAPN: Knowledge-Aware Attentive Programming Exercise Recommendation. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (pp. 2415-2424).