Research on Object-Oriented Programming Teaching Evaluation Method Based on Few-Shot Prompt Learning

Yubin Qu[1,2], Fang Li[1], Xianzhen Dou[1], Chao Feng[1], Binyong Li[2]*

[1] School of Information Engineering, Jiangsu College of Engineering and Technical, China
[2] Advanced Cryptography and System Security Key Laboratory of Sichuan Province, Chengdu University of Information Technology, China
quyubin@hotmail.com, lf@jcet.edu.cn, douxianzhen@163.com, 1102917308@qq.com, lby@cuit.edu.cn

Abstract

This paper proposes a programming education evaluation method based on few-shot prompt learning for object-oriented programming education. The method consists of three core components: establishing multi-dimensional evaluation criteria, designing a few-shot learning-based example code library, and developing intelligent evaluation prompt templates. Through a 16-week controlled experiment conducted in a university computer science program, results demonstrate significant improvements in both learning outcomes and teaching efficiency: the experimental group showed a 9.2% to 15.4% improvement in key metrics including final exam scores and code quality, while reducing grading time by 80% and shortening feedback time from 48 hours to 0.5 hours. This research provides a new paradigm for educational evaluation in computer science education.

Keywords: Object-oriented programming design, Few-shot learning, Teaching evaluation, Large language models

1 Introduction

With the rapid advancement of artificial intelligence, large language models (LLMs) demonstrate significant potential in the education sector [1-3]. Object-oriented programming, a cornerstone of computer science education, is inherently abstract and conceptually complex, posing considerable challenges for students. Traditional assessment methods often fail to address individual learning needs and provide timely, effective feedback or guidance. Given the rapid development of LLMs, their powerful natural language understanding and generation capabilities present an opportunity to enhance educational evaluation systems. Exploring how these models can be leveraged to improve teaching and assessment methods is a critical area of research [4-5].

Currently, the assessment of programming courses primarily relies on traditional metrics such as exam scores and assignment completion rates. However, this approach has clear limitations [5-6]. Firstly, these methods often focus solely on final outcomes, neglecting the development of critical thinking and skills during the learning process. Secondly, traditional evaluations fail to effectively capture students' progress in developing programming-related thinking. Lastly, existing assessment frameworks have not yet fully leveraged the potential of LLMs in prompt-based learning [7], and they lack systematic evaluation of students' learning effectiveness when using prompts [8-10]. This challenge becomes even more pronounced in few-shot prompt-based learning scenarios [11-13]. Designing a robust evaluation system that comprehensively assesses both the learning process and outcomes remains an urgent issue to address [14-15].

This study introduces an intelligent evaluation system for programming assignments, leveraging few-shot prompt learning [11-13]. The proposed system consists of three key components: 1. Comprehensive Evaluation Criteria: The system establishes evaluation standards covering multiple dimensions, including object-oriented feature application, code structure design, functionality implementation, and coding norms. 2. Few-Shot Learning Code Repository: A repository is developed containing exemplary code samples and evaluation guidelines to support the assessment process. 3. Adaptive Prompt Templates: Intelligent prompt templates are designed to quickly adapt to various programming assignment evaluation tasks with minimal examples. The system's effectiveness was validated through practical teaching applications. It demonstrated high evaluation accuracy, significantly improved efficiency (faster than manual grading), and consistent evaluation results.

The structure of this paper is arranged as follows: Section 2 provides a review of the related work on programming education assessment and few-shot learning; Section 3 introduces the design methodology of the evaluation system in detail, including the construction of evaluation dimensions, the design of the example library, and the implementation of the evaluation process; Section 4 validates the effectiveness of the evaluation system through experiments, and Section 5 analyzes and discusses the experimental results; finally, Section 6 summarizes the research findings and outlines future research directions.

*Corresponding Author: Binyong Li; Email: lby@cuit.edu.cn
DOI: https://doi.org/10.70003/160792642025122607005

2 Related Work

2.1 Teaching Evaluation

Traditional teaching evaluation systems predominantly rely on standardized exams, student surveys, teacher self-assessments, and expert reviews [16]. While these methods have played a significant role in educational practices, they exhibit inherent limitations such as subjectivity, delayed feedback, low efficiency, and difficulty in accommodating innovative teaching approaches. With advancements in educational technology, particularly the application of prompt engineering based on large language models in higher education, these traditional methods are increasingly inadequate for addressing the needs of modern teaching scenarios [17]. Emerging learning approaches demand evaluation systems capable of real-time assessment of learners' prompt design skills, evaluation of their interactions with AI systems, analysis of cognitive development during the learning process, and adoption of new paradigms for teaching evaluation using large language models. Researchers are now exploring innovative methods to leverage large language models for teaching assessments. For instance, Yuan et al. [16] demonstrated that with appropriate fine-tuning and prompt engineering, large language models can serve as effective tools for course evaluation. Similarly, Cohn et al. [17] proposed a chain-of-thought prompting method to evaluate students' formative assessment responses. This approach has shown promising results, particularly in assessing open-ended questions within the context of science education.

2.2 Object-Oriented Programming Teaching Method Based on Prompt Engineering

The integration of LLMs into object-oriented programming (OOP) education has introduced transformative changes to teaching methodologies. Research indicates that well-designed prompt strategies can significantly enhance students' learning outcomes and programming skills [18]. However, traditional text-based assignments often lead students to overly rely on LLMs, neglecting the development of critical problem-solving and programming design skills. To address this issue, Cipriano et al. [19] proposed an innovative approach combining diagrams and videos to create OOP assignments. This method not only fosters problem-solving abilities but also discourages students from merely copying LLM-generated solutions. Specifically, they developed a new symbolic system for representing OOP assignments, incorporating structural and behavioral requirements. This approach was evaluated over the course of a semester, yielding promising results. The study found that students responded positively to this diagram-and-video-based teaching method. Notably, video-based assignments were better received than diagram-based tasks. This approach encouraged students to invest more effort in understanding design diagrams and demonstrated higher engagement with video projects. Importantly, participants reported reduced reliance on LLM-generated code during these exercises. This suggests that well-structured prompts and thoughtfully designed assignments can leverage the advantages of LLMs while mitigating potential drawbacks [18]. This successful implementation provides valuable insights into effectively integrating LLMs into OOP education, offering a practical framework for balancing their benefits with the need to cultivate essential programming skills.

3 Methodology

3.1 Method Overview

In the rapidly evolving field of artificial intelligence and LLMs, teaching object-oriented programming presents both opportunities and challenges [18]. This study introduces a two-layered teaching evaluation framework, aiming to improve teaching effectiveness and evaluation efficiency through structured prompt engineering and automated assessment mechanisms. The proposed method offers the following key innovations: development of standardized teaching prompt templates, design of an automated evaluation mechanism based on few-shot learning, and quantifiable assessment of the teaching process.

3.2 Prompt Template Design for Object-Oriented Programming

This paper addresses the challenges in object-oriented programming education through a structured prompt-based approach. We propose a comprehensive template framework

$$T = \{C, M, E, R, A\} \quad (1)$$

, which integrates five essential components: context, methodology, examples, requirements, and assessment. The context component establishes the programming scenario and problem background, while the methodology component outlines specific instructional objectives and pedagogical strategies. To ensure practical understanding, the example component incorporates carefully curated sample code and best practices that demonstrate optimal implementation patterns. The requirements component defines clear implementation criteria aligned with object-oriented principles, focusing on encapsulation, inheritance, and polymorphism. Finally, the assessment component provides standardized evaluation metrics that objectively measure both code quality and concept mastery.

This framework transforms traditional teaching evaluation into a systematic process, bridging the gap between abstract concepts and practical implementation. The template guides instructors in creating structured learning materials and provides students with clear pathways for understanding and applying object-oriented concepts. Each component works cohesively to support both theoretical understanding and practical skill development, with built-in mechanisms for objective assessment and feedback. The framework's flexibility allows for customization across different programming

assignments while maintaining consistency in evaluation standards.

The scenario description should adhere to the following characteristics: Authenticity: Based on real-world application scenarios. Completeness: Provides sufficient background information. Progressiveness: Gradually increases in complexity.

Instructional objectives should:

- Clearly identify the object-oriented concepts to be mastered.
- Define specific skill requirements.
- Offer guidance on learning pathways.

Sample code should:

- Demonstrate standard programming practices.
- Include implementations of key concepts.
- Provide comments and explanations for clarity.

We demonstrate our proposed T = {C, M, E, R, A} structure for teaching Python List concepts through a practical bookstore inventory management scenario. By integrating context-based learning (bookstore inventory), clear methodological guidance (list operations), concrete examples (inventory management code), specific requirements (implementation criteria), and objective assessment standards (quantifiable metrics), the template effectively addresses the challenges in traditional programming education. It bridges the theory-practice gap by providing students with a real-world application context while ensuring standardized evaluation, making abstract list concepts more accessible and practical for learners.

Python List Teaching Template

C (Context)

You are developing an inventory management system for a small bookstore (...).

M (Method)

The teaching method is structured around clear learning objectives and key programming concepts. Students will progress through understanding the fundamental concepts of Python lists as ordered collections, mastering basic list operations and methods, and applying these skills in practical scenarios (...).

E (Example)

Creating a book inventory list

book_inventory = [“Python Basics”, “Data Science 101”, “Web Development Guide”] (...)

R (Requirements)

The implementation requirements focus on three key aspects of code development. First, code organization emphasizes the use of meaningful variable names, appropriate comments, and adherence to Python PEP 8 style guidelines (...).

A (Assessment)

The assessment framework consists of four components weighted according to their importance: (...).

3.3 Automated Evaluation Mechanism Based on Few-shot Learning

The evaluation mechanism consists of the following components: E (Examples): A repository of standard examples; C (Criteria): A set of evaluation standards; M (Metrics): A system of quantifiable metrics; (Feedback): A mechanism for generating feedback; R (Results): A collection of evaluation outcomes.

The evaluation system employs a multi-dimensional scoring approach, as detailed below in Table 1:

Table 1. Evaluation dimensions and weight distribution

Dimension	Weight	Evaluation focus
Code correctness	40%	Completeness of functionality, runtime correctness
Object-oriented design	30%	Application of encapsulation, inheritance, and polymorphism
Code quality	20%	Naming conventions, completeness of comments, code style
Program scalability	10%	Architecture design, interface definitions, maintainability

The scoring process uses a weighted calculation formula:

$$Score = \sum_{i=1}^{n} w_i \cdot m_i \quad (2)$$

Where, w_i: weight assigned to the ith dimension; m_i: score obtained in the ith dimension; n: total number of evaluation dimensions.

The system follows these steps to perform evaluations: (1) Example Matching: compare student code with standard implementations in the example repository; (2) Feature Extraction: analyze the structural characteristics and implementation patterns of the code; (3) Quality Assessment: conduct multi-dimensional scoring based on predefined standards; (4) Feedback Generation: produce detailed evaluation reports and suggestions for improvement.

4 Experimental Design

4.1 Overview of Experimental Design

This study conducted a 16-week controlled experiment to evaluate the effectiveness of an object-oriented programming assessment method based on Prompt templates and Few-shot Learning. A combination of quantitative and qualitative research methods was used. By strictly controlling experimental conditions, data

was systematically collected and analyzed to ensure the scientific rigor and reliability of the findings. While maintaining teaching quality, the experiment primarily focused on assessing the impact of the new evaluation method on students' learning outcomes and its potential to enhance teaching efficiency.

4.2 Class Design for the Experiment

The experiment was conducted with computer science students at a university. To ensure representativeness and comparability of the sample, two classes were randomly selected from six classes in the same grade. 30 students were assigned as the experimental group, while the other (30 students) served as the control group. All participants had completed a foundational programming course and possessed basic programming knowledge. An analysis of prior course grades confirmed no significant difference in programming proficiency between the experimental and control groups, providing a solid foundation for the study. To ensure reliability, strict group control measures were implemented. First, data such as students' grades from previous programming courses and college entrance examination scores were collected and analyzed. Variance analysis confirmed no significant differences in baseline levels between groups. Second, all classes were taught by the same instructor with over five years of teaching experience, using identical textbooks and syllabi to maintain consistency in content and pace. Lastly, external factors such as teaching environments were carefully controlled. The same classroom, equipment, and network settings were used to minimize external interference during the experiment.

The experimental group and the control group were subjected to a 16-week teaching experiment using different evaluation approaches. The experimental group employed a teaching method based on Prompt templates and an automated evaluation system leveraging Few-shot Learning. This allowed students to receive real-time feedback and personalized guidance. In contrast, the control group followed traditional teaching methods and manual grading, adhering to conventional teaching plans. Both groups completed the same teaching tasks and assessments but differed in aspects such as assignment submission and feedback mechanisms.

A comprehensive data collection framework was designed for the control group, incorporating three key metrics: learning outcomes, process metrics, and feedback metrics. Learning outcomes were assessed through midterm exams (30%), final exams (40%), and regular assignments (30%), providing a holistic view of student performance. Process metrics were captured by an automated system, tracking weekly programming assignments, code quality scores, and course participation. Feedback metrics were gathered via surveys and interviews, focusing on student satisfaction with the teaching method and teacher feedback on their experience. Data collection occurred at three intervals: at the beginning of the semester (T1) to establish baseline data, including initial student competency tests and attitude surveys; at Week 8 (T2) to gather midterm data, emphasizing learning progress and interim results; and at Week 16 (T3) to collect final data for comprehensive evaluation of learning outcomes and achievement of teaching goals. Each data collection phase adhered to standardized procedures to ensure accuracy and completeness.

4.3 Statistical Validation Methods

This study proposed two research hypotheses. The primary hypothesis examined the impact of the new evaluation method on student learning outcomes. The null hypothesis (H0) posited no significant effect, while the alternative hypothesis (H1) suggested the new method significantly improved learning outcomes. A secondary hypothesis investigated the influence of the method on teaching efficiency, particularly its impact on teachers' workload. To validate these hypotheses, rigorous statistical methods were applied to ensure reliable conclusions. First, independent sample *t-tests* were used to compare differences between the experimental and control groups across all metrics, with a significance level set at 0.05. Effect sizes were calculated using Cohen's d [20] to assess the practical impact of the teaching methods. Additionally, Pearson correlation analysis and multiple regression analysis were conducted to explore relationships among variables. Repeated measures ANOVA was used to analyze dynamic changes in learning outcomes over time.

4.4 Quality Control Measures

To ensure the validity of the experiment, several control measures were implemented. For internal validity, we minimized confounding factors by randomly assigning participants, strictly controlling extraneous variables, and standardizing experimental procedures. To enhance external validity, we selected representative samples and maintained the authenticity of the experimental environment, thereby increasing the generalizability of the findings. Data quality control was emphasized throughout the experiment. During data collection, a standardized process was established, leveraging automated tools to record learning process data. A multi-level verification mechanism was also employed to ensure data accuracy. In the data processing stage, a comprehensive data cleaning strategy was implemented, including outlier detection, handling of missing values, and consistency checks, ensuring the reliability of subsequent data analysis.

5 Experimental Results and Data Analysis

5.1 Analysis of Learning Outcomes

This study employed a controlled experiment involving two parallel classes from an object-oriented programming course. The experimental group employed a dual-layer evaluation method, while the control group followed traditional teaching methods. To ensure scientific rigor, we randomly assigned students to groups, used a unified syllabus and course content, and maintained identical teaching hours and environments for both groups. After data cleaning and preprocessing, all 30 samples from each

group were valid, achieving 100% data completeness.

The analysis of the experimental data shown in Table 2 reveals several key findings:

1. Overall Performance Improvement: the experimental group significantly outperformed the control group across all evaluation metrics, with differences being statistically significant.

2. Enhanced Code Quality: the experimental group achieved a higher average code quality score (87.5 ± 6.1) compared to the control group (75.8 ± 7.3), an improvement of 11.7 points or 15.4%. This difference is particularly notable (t=4.12).

3. Steady Learning Progress: the experimental group showed consistent improvement from the midterm exam (85.6 points) to the final exam (88.3 points), with an average increase of 2.7 points, indicating sustained learning effectiveness.

4. Higher Assignment Quality: the experimental group achieved significantly higher scores on assignments (90.2 ± 5.4) compared to the control group (82.1 ± 6.7). This demonstrates that the prompt-based teaching method effectively enhances students' practical skills.

To ensure the reliability of these results, we conducted independent sample t-tests:

• All evaluation metrics yielded p-values below 0.01, confirming the high statistical significance of the findings.

• The t-value for code quality (4.12) was the highest, indicating the most pronounced improvement in this dimension.

• The relatively small standard deviations across metrics suggest consistent teaching effectiveness.

The improvement achieved by the experimental group relative to the control group can be quantified using the following formula:

$$Improvement\ Ratio = \frac{Mean\ of\ Experimental\ Group - Mean\ of\ Control\ Group}{Mean\ of\ Control\ Group} \times 100\% \quad (3)$$

The calculated improvement rates are as follows: a 9.2% increase for the midterm exam, a 11.5% increase for the final exam, a 9.9% increase for assignments, and a 15.4% increase in code quality. These results demonstrate that the dual-layer evaluation framework significantly enhances the effectiveness of object-oriented programming education, with particularly strong improvements in code quality and adherence to programming standards.

Table 2. Comparison of learning outcomes between experimental and control groups

Evaluation metric	Experimental group (n=30)	Control group (n=30)	t-value
Midterm exam scores	85.6 ± 7.2	78.4 ± 8.1	3.42
Final exam scores	88.3 ± 6.8	79.2 ± 7.9	3.89
Homework scores	90.2 ± 5.4	82.1 ± 6.7	3.65
Code quality scores	87.5 ± 6.1	75.8 ± 7.3	4.12

5.2 Analysis of Understanding Programming Language Concepts

An evaluation of students' understanding of core concepts in object-oriented programming reveals that the experimental group demonstrated a deeper comprehension. This is evident in the following aspects: 1. Understanding of Classes and Objects: the experimental group achieved a correct response rate of 92.3% on related questions, compared to 78.6% in the control group, reflecting an improvement of 13.7 percentage points. 2. Application of Inheritance and Polymorphism: in practical exercises, 88.5% of the experimental group effectively applied inheritance and polymorphism, significantly higher than the 71.2% observed in the control group. 3. Mastery of Design Patterns: the experimental group correctly identified and applied basic design patterns in 85.7% of cases, while the control group achieved only 65.3%.

5.3 Analysis of Teaching Efficiency

The adoption of a new evaluation method resulted in notable improvements in teaching efficiency: 1. Assignment Grading Time: the average grading time per assignment decreased from 15 minutes to 3 minutes, representing an 80% efficiency gain. 2. Feedback Timeliness: the average feedback time for assignments in the experimental group was 0.5 hours, compared to 48 hours in the control group, showing a significant improvement in responsiveness. 3. Consistency in Scoring: the automated evaluation system based on Few-shot Learning reduced scoring discrepancies, with deviations decreasing from ±8% to ±2%.

An analysis of students' learning behavior data further highlights the experimental group's enhanced engagement: 1. Course Participation Rate: the experimental group achieved an average attendance rate of 96.8%, 8.5 percentage points higher than the control group. 2. Assignment Submission Rate: the on-time submission rate for assignments in the experimental group was 94.5%, compared to 85.2% for the control group. 3. Interaction Frequency: students in the experimental group participated in classroom discussions and online interactions 1.8 times more frequently than those in the control group.

5.4 Analysis of Student Feedback

At the conclusion of the 16-week teaching experiment, a comprehensive satisfaction survey was conducted among students in both groups. The survey employed a 5-point Likert scale for quantitative assessment [21], with scores defined as follows: 5 (very satisfied), 4 (satisfied), 3 (neutral), 2 (dissatisfied), and 1 (very dissatisfied). Additionally, open-ended questions were used to gather

qualitative feedback, and in-depth interviews were held with selected students to obtain more detailed insights into their learning experiences.

The survey results shown in Table 3 reveal significant differences between the experimental and control groups across all evaluated dimensions. In terms of teaching methods, the experimental group achieved an average score of 4.6, outperforming the control group's score of 3.8 by 0.8 points, representing a 21.1% improvement. This indicates the effectiveness of the new teaching approach. For perceived learning outcomes, the experimental group scored an impressive 4.5 compared to the control group's 3.7, reflecting a 21.6% enhancement in learning effectiveness. The most notable improvement was observed in the timeliness of feedback, where the experimental group scored a high 4.8, a remarkable 37.1% increase over the control group, highlighting the advantages of the automated evaluation system. Overall satisfaction was also significantly higher in the experimental group, with a score of 4.7 compared to the control group's 3.7, marking a comprehensive improvement of 27.0%. These results validate the success of the teaching reform.

Table 3. Analysis of teaching satisfaction survey results

Evaluation dimension	Experimental group	Control group	Improvement
Teaching methods	4.6 ± 0.3	3.8 ± 0.4	21.1%
Learning outcomes	4.5 ± 0.3	3.7 ± 0.5	21.6%
Timeliness of feedback	4.8 ± 0.2	3.5 ± 0.5	37.1%
Overall satisfaction	4.7 ± 0.3	3.7 ± 0.4	27.0%

Analysis of open-ended feedback further highlights the strengths of the new teaching model. First, the use of prompt templates provided students with a clear learning pathway, while the progressive task design and comprehensive coverage of knowledge points helped establish a structured learning framework. Second, the immediate feedback mechanism significantly improved learning efficiency; the automated evaluation system ensured objective and fair grading, with transparent scoring standards earning widespread student approval. Additionally, the new teaching approach reduced learning anxiety, increased motivation, and fostered independent learning skills. Students also offered constructive suggestions for improvement, such as incorporating more real-world application cases and supplementing content with industry best practices.

Based on student feedback, we have developed a phased improvement plan. In the short term (1-2 months), we will focus on expanding the repository of typical application cases, refining error feedback templates, and providing more detailed code optimization suggestions. For the mid-term (3-6 months), the plan includes developing personalized learning paths, improving the evaluation metrics system, and building a knowledge graph navigation tool. In the long term (beyond 6 months), we aim to establish an adaptive learning system, introduce intelligent tutoring features, and develop a collaborative learning platform to further enhance teaching effectiveness. These improvements will be implemented step-by-step to ensure continuous optimization of teaching quality. We will regularly collect student feedback and adjust our plans accordingly to maintain consistent progress in teaching outcomes. By establishing a dynamic feedback mechanism, we aim to refine the teaching model continuously and provide students with a superior learning experience.

5.5 Limitations and Contextual Constraints

While our framework provides a structured approach to OOP education, we acknowledge several key limitations. The implementation requires substantial initial resources, including teaching material development and instructor training. The framework's effectiveness may vary across different institutional contexts and student populations. Additionally, the structured nature might initially constrain creative problem-solving, and scalability challenges emerge in large class settings with multiple instructors.

6 Conclusion and Future Work

This study proposes a teaching evaluation method for object-oriented programming based on few-shot prompt learning. A 16-week controlled experiment demonstrated the effectiveness of this approach. The results indicate that the proposed method, leveraging prompt templates and few-shot learning, significantly outperforms traditional teaching methods across various dimensions. In terms of learning outcomes, the experimental group performed better in final exams (88.3 points), routine assignments (90.2 points), and code quality (87.5 points), with improvements ranging from 9.2% to 15.4% compared to the control group. Regarding conceptual understanding, the experimental group exhibited a marked advantage in grasping and applying core object-oriented concepts, with accuracy improvements of 13.7 to 20.4 percentage points. In teaching efficiency, the new method reduced assignment grading time by 80%, shortened feedback delivery from 48 hours to 0.5 hours, and significantly improved grading consistency, reducing scoring deviations from ±8% to ±2%. The primary innovation of this study lies in constructing a standardized prompt template system, developing a few-shot learning-based intelligent evaluation mechanism, and designing a multidimensional evaluation metric system, providing a replicable framework for object-oriented programming education.

Future research will further expand upon these findings. In terms of evaluation models, we plan to incorporate

deep learning techniques to enhance code comprehension, develop more sophisticated grading algorithms, and improve handling of unstructured feedback. For application scenarios, we aim to extend this evaluation method to other programming courses, create modules supporting various programming languages, and explore applications in online education contexts. Additionally, we are committed to developing adaptive learning path recommendation systems, intelligent learning diagnostic tools, and personalized feedback mechanisms. We also plan to design evaluation systems based on collective intelligence, support team-based assessments, and build platforms for knowledge sharing and collaboration. These research directions will contribute to further advancements in teaching quality and effectiveness.

Acknowledgements

This work was supported by China Vocational Education Society Huawei Technologies Co., Ltd. 2024 Annual Industry-Education Integration Special Topic (No. XHHWCJRH2024-02-01-02), 2023 Higher Education Scientific Research Planning Project of China Society of Higher Education (No. 23PG0408), 2023 Philosophy and Social Science Research Programs in Jiangsu Province (No. 2023SJSZ0993), Nantong Science and Technology Project (No. JC2023070), Key Project of Jiangsu Province Education Science 14th Five-Year Plan (Grant No. B-b/2024/02/41) and the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province (Grant No. SKLACSS-202407). This work is sponsored by the Cultivation of Young and Middle-aged Academic Leaders in Qing Lan Project of Jiangsu Province.

References

[1] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, A. Mian, A comprehensive overview of large language models, *arXiv preprint*, arXiv:2307.06435, July 2023. https://arxiv.org/abs/2307.06435

[2] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. P. Hou, Y. Q. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J. Nie, J. Wen, A survey of large language models, *arXiv preprint*, arXiv:2303.18223, March, 2023. https://arxiv.org/abs/2303.18223

[3] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, J. Gao, Large language models: A survey, *arXiv preprint*, arXiv:2402.06196, February 2024. https://arxiv.org/abs/2402.06196

[4] C. Leung, Classroom-Based Assessment Issues for Language Teacher Education, in: A. J. Kunnan (Ed.), *The concise companion to language assessment*, WILEY Blackwell, 2024.

[5] K. J. Topping, Digital peer assessment in school teacher education and development: A systematic review, *Research papers in education*, Vol. 38, No. 3, pp. 472–498, 2023. https://doi.org/10.1080/02671522.2021.1961301

[6] C. Tovani, *So what do they really know?*, Routledge, 2023.

[7] F. Li, Y. Qu, C. Feng, X. Dou, L. Li, A fine-grained emotion prediction method based on large language models, *2024 IEEE 24th International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*. Cambridge, United Kingdom, 2024, pp. 101–110. https://doi.org/10.1109/QRS-C63300.2024.00024

[8] J. Wang, Z. Liu, L. Zhao, Z. Wu, C. Ma, S. Yu, H. Dai, Q. Yang, Y. Liu, S. Zhang, E. Shi, Y. Pan, T. Zhang, D. Zhu, X. Li, X. Jiang, B. Ge, Y. Yuan, D. Shen, T. Liu, S. Zhang, Review of large vision models and visual prompt engineering, *arXiv preprint*, arXiv:2307.00855, July, 2023. https://arxiv.org/abs/2307.00855

[9] L. Giray, Prompt engineering with chatgpt: a guide for academic writers, *Annals of biomedical engineering*, Vol. 51, No. 12, pp. 2629–2633, December, 2023. https://doi.org/10.1007/s10439-023-03272-4

[10] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, D. C. Schmidt, A prompt pattern catalog to enhance prompt engineering with chatgpt, *arXiv preprint*, arXiv:2302.11382, February, 2023. https://arxiv.org/abs/2302.11382

[11] Y. Wang, Q. Yao, J. T. Kwok, L. M. Ni, Generalizing from a few examples: A survey on few-shot learning, *ACM computing surveys*, Vol. 53, No. 3, pp. 1–34, May, 2021. https://doi.org/10.1145/3386252

[12] A. Parnami, M. Lee, Learning from few examples: A summary of approaches to few-shot learning, *arXiv preprint*, arXiv:2203.04291, March, 2022. https://arxiv.org/abs/2203.04291

[13] Y. Song, T. Wang, P. Cai, S. K. Mondal, J. P. Sahoo, A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities, *ACM Computing Surveys*, Vol. 55, No. 13s, pp. 1–40, December, 2023. https://doi.org/10.1145/3582688

[14] Y. Qu, F. Li, L. Li, X. Dou, H. Wang, Can we predict student performance based on tabular and textual data?, *IEEE Access*, Vol. 10, pp. 86008–86019, August, 2022. https://doi.org/10.1109/ACCESS.2022.3198682

[15] Y. Qu, T. Bao, X. Chen, L. Li, X. Dou, M. Yuan, H. Wang, Do we need to pay technical debt in blockchain software systems? *Connection Science*, Vol. 34, No. 1, pp. 2026–2047, June, 2022. https://doi.org/10.1080/09540091.2022.2067125

[16] B. Yuan, J. Hu, An exploration of higher education course evaluation by large language models, *arXiv preprint*, arXiv:2411.02455, November, 2024. https://arxiv.org/abs/2411.02455

[17] C. Cohn, N. Hutchins, T. Le, G. Biswas, A chain-of-thought prompting approach with llms for evaluating students' formative assessment responses in science, *arXiv preprint*, arXiv:2403.14565, March, 2024. https://arxiv.org/abs/2403.14565

[18] T. Wang, N. Zhou, Z. Chen, Enhancing computer programming education with llms: A study on effective prompt engineering for python code generation, *arXiv preprint*, arXiv:2407.05437, July, 2024. https://arxiv.org/abs/2407.05437

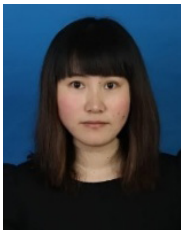
[19] B. P. Cipriano, P. Alves, P. Denny, A picture is worth a thousand words: Exploring diagram and video-based oop exercises to counter llm over-reliance, *arXiv preprint*, arXiv:2403.08396, March, 2024. https://arxiv.org/abs/2403.08396

[20] H. F. Ponce-Renova, Comparing Effect Sizes and their Confidence Intervals: A Primer on Equivalence Testing in Educational Research, *Journal of New Approaches in Educational Research*, Vol. 11, No. 2, pp. 209–225, July, 2022.
https://doi.org/10.7821/naer.2022.7.930

[21] M. Shardlow, M. Cooper, M. Zampieri, Complex: A new corpus for lexical complexity prediction from likert scale data, *arXiv preprint*, arXiv:2003.07008, June, 2020.
https://arxiv.org/abs/2003.07008

Biographies

Yubin Qu is an Associate Professor at Jiangsu College of Engineering and Technology. His primary research interests lie at the intersection of Machine Learning, Software Engineering, Software Security, and Software Testing & Quality Assessment.



Fang Li is an Associate Professor at Jiangsu College of Engineering and Technology. Her research expertise centers on Network-based Ideological and Political Education and Higher Education.

Xianzhen Dou received the M.S. degree in School of Electronics and Information from Nantong University in China in 2013. Since 2019, he has been a lecture with Information Engineering Institute, Jiangsu College of Engineering and Technology. His research interests include software engineering, and machine learning.

Chao Feng received the MSc degree from Sheffield School of Computing, UK, in 2022. Since 2024, he has been teaching in the School of Information Engineering, Jiangsu Institute of Engineering and Technology. His research interests include software engineering and big data analytics.

Binyong Li is an Associate Professor in Chengdu University of Information Technology with expertise in Cyberspace Governance and Cyber Situational Awareness. His research focuses on understanding and addressing the complex dynamics of digital environments and their security implications.