Proceedings of CONF-SEML 2026 Symposium: Computational Analysis and Modeling in Complex Intelligent Systems
DOI: 10.54254/2755-2721/2026.GU32662

Application Strategies of LLMs in Computer Programming Teaching

Zihan Lu

International Department, High School Attached to HIT, Harbin, China
alicehs@sina.com

Abstract. With the development of artificial intelligence, Large Language Models (LLMs) has some potential for natural language processing and education. Based on the practical experience of applying LLMs to computer programming, the paper develops a closed loop assistance system, "learning-training-tutoring-feedback" system, that can overcome the problems of traditional teaching (e.g., numerous knowledge points, vague concepts, insufficient training, poor personalized support, single feedback). In the learning stage, it helps students to grasp programming notions and logic by generating code examples, knowledge analysis frameworks and content explanations. In the training stage, the system adjusts the difficulty of exercises due to students' learning records and gives code hints and suggestions to construct algorithms. In TUT, it serves as the debugging partner for correcting errors and suggesting repair suggestions, which improves the debugging and solving abilities of students. In a feedback stage, the paper analyzes learning behavior records to learn knowledge of the knowledge to implement personalized learning adjustment and suggestions. This can improve the current situation, low learning efficiency and poor independent thinking ability of traditional programming teaching and increase the programming skills and independent thinking abilities of the students. The introduction of LLMs into programming can make teaching more personalized, dynamic, systematic and perform implementation for further improving the students' computational thinking and practical abilities.

Keywords: Large language models, Education programming learning, Feedback

1. Introduction

Artificial intelligence has been developed in recent years. As a major part of Natural Language Processing (NLP), large language models (LLMs) are widely used in various industries and applications. By developing a deep learning and transformer model, LLMs are trained on a large amount of text, which is easy to understand, generate and process natural language information. They have such basic roles as text generation, language translation, dialogue generation, sentiment analysis and information extraction. LLMs has received attention due to its high language processing capabilities, cross-scenario flexibility and their ability to handle multiple language tasks, produce relatively complete logic and clear semantics, and provide relatively natural human-computer interaction. LLMs also has cross-task and cross-domain transferability, and can be

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.GU32662

optimized to work for such field as healthcare, finance and law. At the same time, a high amount of repetitive work can also be replaced or assisted by the model with significant efficiency advantages in copywriting, data analysis, and customer service. LLMs may be known to deal with a great amount of cross- domain text in the model training process, and even can provide good knowledge transfer capabilities to support different scenarios [1]. As technologies develop, the generation quality and understanding depth of large-Language models also keep improving, and also there are many possibilities to further improve application settings [2]. The development of artificial intelligence has made programming a basic task, like code writing by AI tools. Learning programming is no longer about writing code but rather learning computational thinking, problem analysis, algorithm design, and problem solving [3].

Coupled with the problems of traditional programming learning, such as multiple knowledge points, unclear concepts, no training, guidance and timely feedback, for most students (especially self-learners) the difficulty will also grow. To address these concerns, the paper will research computer programming learning using a LLMs. First, the paper analyzes the main problems of traditionally programming learning and use a large- language model to solve these problems by constructing a closed loop of "learning-training-tutoring-feedback" to improve the efficiency of programming learning while improving the development of algorithms.

2. Analysis of traditional computer programming teaching

Based on practical observations from the School of Big Data and Internet at Shenzhen University of Technology [4], students generally encounter various difficulties during the learning process of programming courses, as illustrated in Figure 1. These difficulties can be broadly summarized into the following aspects.



Figure 1. Difficulties in programming learning (picture credit: original)

2.1. The abundance of knowledge points leads to learning difficulties

There is a lot of knowledge from the point of view of computer programming. For beginners this is not so obvious. It is a basic syntax, data structures, algorithms, debugging, different programming architectures, etc. It cannot be expressed to a practical user. It gets very abstract and abstract and cannot be translated to the real world. Students are not enough time and techniques to learn every knowledge point. Classroom teaching only covers some basic concepts and analysis and real life guidance for complex problems cannot be given. Students cannot flexibly use knowledge they have learned in the actual coding, which affects the effect.

DOI: 10.54254/2755-2721/2026.GU32662

2.2. Low efficiency in problem solving

One of the main reasons of programming is to make the students analyze and solve problems. On certain programming tasks students are not rational thinking and methodical, so there is low efficiency. Furthermore, when there is a problem, students are usually unable to quickly figure out the root of the problem and often don't decide which of several solutions is the best. Further, many students copy and past code or even imitate examples from programming and are still unable to analyze themselves and reason logical. Furthermore they are not completely trained in thinking about code errors during the debugging, so it is easy for them to quickly spot code mistakes and thus reduce efficiency of the problems.

2.3. Over-reliance on network resources

With increasingly abundant Internet resources, more and more students rely on online resources for programming learning. For example, programming forums, technical communities etc. Online resources may be helpful in a timely manner, but improper use can easily drive students into a dependency mindset and lead to weaker independent thinking. Furthermore, students are not screened and judged when using online resources, and easily disturbed by unverified information. Long-term reliance on ready-to-use solutions is not only hindering their ability to complete tasks independently, but also hinders the understanding of problems. Students who need to complete course tasks will copy online code without taking an idea of logic and principles into account, which can affect their depth of learning.

2.4. Unbalanced teaching resources and limited individual guidance

There is also a problem with the allocation of resources in traditional programming teaching. Students with a different foundation differ in their level of learning, ways and understanding. The paper cannot allocate teaching resources (teachers, experimental equipment, learning materials) according to individual needs, either because the course is going too quickly (making it difficult to keep up or because the study is too slow (for large classes, teacher may not take into account the true situation of each student). In Programming courses, people will be very interested in practice and experimentation. With limited lab time, they may not have enough practice time. When poor or poor foundations are involved, they are very limited in their learning if they do not have proper personal guidance.

3. Application strategies of LLMs in computer programming teaching

To address the issues present in the traditional programming learning process, this article designs a learning system consisting of "personalized learning - personalized training - programming tutoring - dynamic feedback", as illustrated in Figure 2.

DOI: 10.54254/2755-2721/2026.GU32662

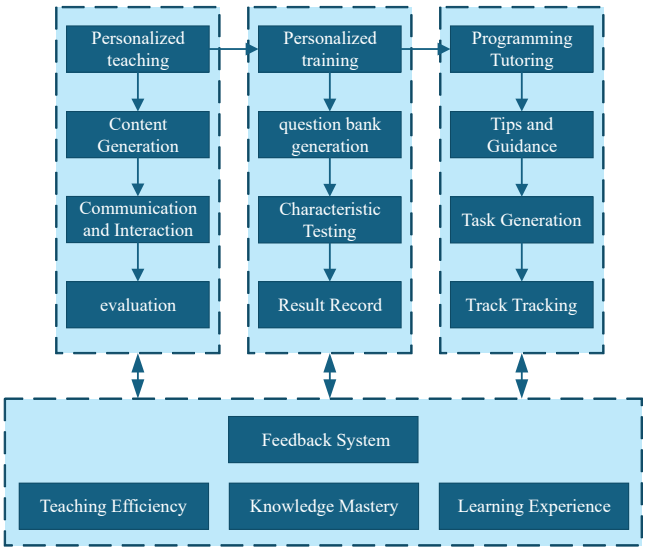


Figure 2. Application strategies of LLMs in programming learning (picture credit: original)

3.1. Personalized teaching

Traditional teaching is quite rigid, the teaching and evaluation is fixed, and students do not take things seriously. For example, students who like to learn by demonstration and examples may use better learned subjects to learn through practice, etc. Only by learning through practice will knowledge become better understood. Complex programming concepts are not explained in a way that students can take. Most students are hard to learn algorithm, especially for students who have no prior programming knowledge. They may not understand programming syntax or abstract concepts. And many students even think that teachers are not helpful and help is insufficient and they try hard to understand many points of knowledge. And course content update. In technology, especially in artificial intelligence and data science, the development speed is very fast, but teaching syllabus and course content are still behind, so the teaching content is not good compared to current standards and best practices.

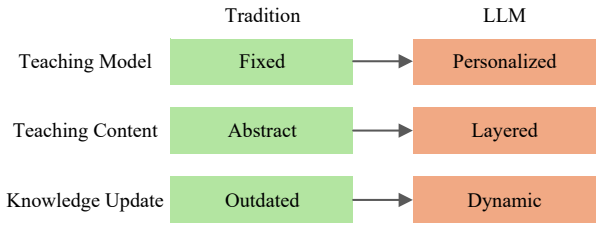


Figure 3. Comparison between traditional teaching and LLM teaching (picture credit: original)

Based on the learning methods selected by students, and the learning records of students, personalized knowledge learning content is generated. Based on learning performance, learning may be accelerated at the learning level according to the knowledge points at stake [5]. Students can start with simple problems. Students are best to learn with problems, and students can gradually learn basic programming syntax using learning problems. At the same time, based on knowledge points in problems, or actual knowledge content used by students for students, students have expressed problems raised by students summarized, analyzed, compared to students to further gain experience. In the case of the solving of simple programming problems, students can analyze and solve

DOI: 10.54254/2755-2721/2026.GU32662

problems based on object-oriented thinking, using prompts. In practical basic application projects, students are provided with file reading/writing, visualization, etc. based on simple real-life situations. In detail application projects are made with code and code snippets to help students better concentrate on the solution. As illustrated in Fig. 3, the teaching mode, teaching content and knowledge update speed are significant differences between teaching modes and LLMs. Compared with traditional mode, LLMs mode can provide personalized teaching methods, clearer hierarchical progressive teaching content, and more timely knowledge updates.

3.2. Personalized training

Traditional training: Students often search for relevant questions based on test results or learned material. This can cause several challenges. For example, students may struggle to figure out weak spots, and what are they training based on their test results. Training questions found by students also tend to be very random and often there are very unknown information in depth and breadth of knowledge. A small mistake can lead to a very difficult question and result in high students' confidence and enthusiasm, resulting in low learning efficiency. In addition, the paper often lack the information from one test result. The paper can not achieve high training efficiency with only one knowledge point. Web: the data is lacking, so what are the online training resources, unclear questions, incorrect answers, bad test data, incomplete code examples, or poor usability [6]. Training resource alone: Students need to search and select resources. The train required more than one knowledge points and the resources. The paper could not train the students by looking for resources for each knowledge point and a lot of time, as in practice, students could lose interest in learning due to repeated setbacks. Testing resources also requires very a lot to be done, and after a long period of time the performance of the students will be low. A vicious cycle can be induced by searching resources.

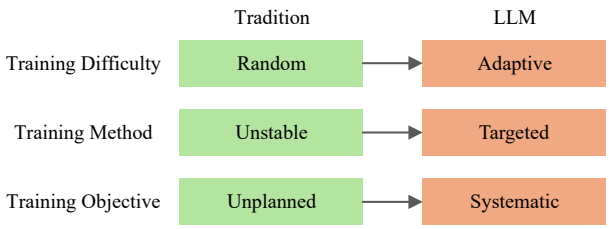


Figure 4. Comparison between traditional training and LLM training (picture credit: original)

In contrast to traditional training, LLMs might have set up challenge-based questions based upon learning records, train students' programming skills, and adapt difficulty and content of questions based of students' feedback and abilities. Also, large models may also help students with problems by making reasonable suggestions when students encounter problems, provide them with new ideas, solve problems, and construct algorithms [7]. Training experience: With real programming tasks, large model may also generate code or code snippets with prompts, that help them to understand the role of each part of the code. This helps to train students in writing code and modify code with prompts. Large model might also record students' behavior and training efforts during training, and adjust and adapt training projects based on the records [8]. As shown by Fig. 4, there are significant differences in training difficulty, training methods and training plans compared with LLMs. Compared to LLMs mode, this mode can provide more appropriate training methods, more focused training methods as well as more systematic training. Problem-guidance-practice-feedback mode.

DOI: 10.54254/2755-2721/2026.GU32662

3.3. Auxiliary programming tips

Traditionally, students of the traditional programming type come with several problems. A grammar problem is a grammar problem. Students do not learn grammar rules correctly enough and will miss spellings, incorrect parentheses, or mis-written keywords. While these problems might be encountered in compilers or interpreters, if they happen, they will do not improve learning, and logical errors are the hardest problem, because the program has correct syntax but some results are not acceptable [9]. There are problems that cannot be identified directly from the error report and usually the result will be debugging or outputting intermediate variable to get to the root cause, where students are stuck "knowing the error, but not knowing what it is."

There are difficulties in the algorithm debugging. Many algorithms have complicated data structures and many layers of loops, that they easily ignore when implementing them due to boundary condition or special conditions, so that they have abnormal program operation or false output results; and, many algorithms are too easily ignore, because if the data is large they run slowly and/or timeout, requiring students to understand the notions of time and space complexity; yet, debugging tools are also difficult. Some beginners are not very good at using breakpoint, log output or debuggers to find problems, so troubleshooting is very slow. There is also a difficulty in learning because the errors often occur continuously, and this affects learning interest and confidence.

Code writing and code debugging: The LLMs can help students solve code writing and codes debugging. When the students encounter some problem structure, functionality and code specifications at code writing, the big language should provide them with functional code frameworks or fragments as an interaction, and explain the logic and nature of generated code line by line to help students understand the program design ideas. The LLMs could also suggest style optimization suggestions based on the students' code (e.g., variable name, function splitting, loop structure optimization, etc.) to encourage good code habits and improve readability and maintainability.

After code debugging, students deal with grammar error, logic error, and incorrect boundary conditions. The LLMs may be a "debugger partner", analyze problems, generate test data, diagnose errors, help students gradually find the source of the problem and propose repair/improper solutions [10].

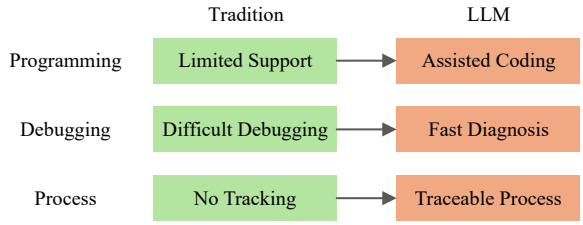


Figure 5. Comparison between traditional programming and LLM programming (picture credit: original)

On the one hand, it can speed up error troubleshooting, and on the other hand it can teach students to use their logical thinking and independent thinking ability to gradually learn how to handle a problem in an interactive way.

On the other hands, the large language framework can keep track of students' operations, errors, and solution paths in the code writing process and produce structured data which can be used to

DOI: 10.54254/2755-2721/2026.GU32662

learn and analyze, thus revealing their weaknesses and better understanding level and continue on to learn; It can even organize more specifically and encourage teachers to "teach with students' aptitude". From Fig 5 it can be noted that there are major differences between traditional methods and LLMs practices in writing and debugging. Through the help of the whole process, the huge language model could help students learn skills and better understand problems and help programming education to develop more intelligent ways.

3.4. Feedback mechanism

Traditional learning feedback is mainly based on periodic tests or grades. Although such measures indicate whether a student has mastered a certain level of knowledge, they do not clearly reveal thinking errors, practical difficulties, or misconceptions. A student may complete a task functionally while still writing logically flawed or non-standard code, or may not truly understand the core algorithm [11]. Traditional evaluation also tends to lag behind the learning process, because students often receive feedback only after homework or exams have been completed. In large classes, teachers find it difficult to provide timely individualized guidance, and students' long-term learning trajectories are rarely tracked in a systematic way.

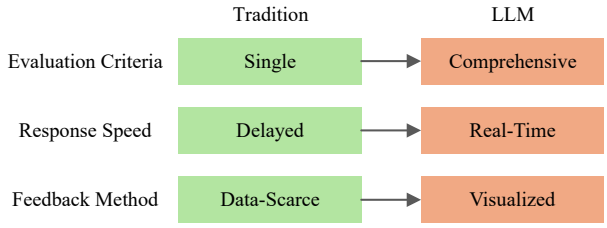


Figure 6. Comparison between traditional feedback mechanisms and LLM feedback mechanisms (picture credit: original)

As shown in Figure 6, LLM-based feedback mechanisms offer more comprehensive evaluation criteria, faster response speed, and more visualized forms of feedback. By analyzing how students learn concepts, write code, debug, and solve problems, LLMs can identify weak links, logical mistakes, and knowledge gaps, and then provide more specific feedback than a simple score. They can also adjust content difficulty dynamically and generate personalized follow-up tasks.

Another important advantage of LLMs is their immediacy and persistence. Whenever students write code, debug, or complete training tasks, the model can provide instant analysis and suggestions, helping them correct mistakes early and reduce error accumulation. Visual learning reports can further summarize students' learning conditions, abilities, and existing problems over time, providing useful design suggestions for teachers and supporting more personalized and systematic programming education.

4. Conclusion

This article focuses on the application of the LLMs in programming learning. In response to the problems of excessive knowledge, insufficient training, and slow feedback in traditional programming education, a " learning-training-tutoring-feedback" closed-loop system is proposed to explore intelligent programming learning methods. In traditional programming learning, students generally have problems with multiple knowledge points, abstract concepts, and strong logic. In addition to grammar, data structures, algorithm design, and debugging, programming learning also

DOI: 10.54254/2755-2721/2026.GU32662

requires students to comprehensively apply multiple knowledge to solve problems. Beginners often develop a fear of difficulty. At the same time, traditional teaching methods are difficult to teach and cannot be dynamically adjusted according to students' learning differences, which affects the effectiveness of personalized learning. In response to the above issues, an auxiliary learning method based on the big language model is proposed: providing knowledge introduction, examples, and analysis during the learning stage; During the training phase, adjust the difficulty dynamically based on the learning situation and guide students to think through interactive prompts; Provide incorrect positioning and correction suggestions during the coaching phase; Provide the learning process and generate personalized analysis during the feedback stage to promote the personalized, dynamic, and systematic development of programming learning, and improve students' programming and problem-solving abilities.

At present, this article is still at the application level of research, and further verification of the impact of LLMs on programming learning is needed in real teaching scenarios. The practical path will be continuously improved, and an AI assisted programming learning system will be gradually built.

References

[1] Song, T., Zhang, H., Xiao, Y. (2024). A high-quality generation approach for educational programming projects using LLM. IEEE Transactions on Learning Technologies, 17: 2296-2309.
[2] Bernik, A., Radosevic, D., Cep, A. (2025). A comparative study of large language models in programming education: accuracy, efficiency, and feedback in student assignment grading. Applied Sciences-Basel, 15(18).
[3] Fu, Q., Zhao, Y., Jia, Z., et al. (2025). Large language models (LLMs) in programming learning: the current research state and agenda. IEEE Transactions on Learning Technologies, 18: 942-961.
[4] Hu, L. S. (2025). Reform and exploration of program-based curriculum teaching based on large language models. Computer Knowledge and Technology, 21(30).
[5] Lubin, L. (2025). Leveraging large language models to support individualized learning. Preprint Book.
[6] Choi, S., Kim, H. (2025). The impact of a large language model-based programming learning environment on students' motivation and programming ability. Education and Information Technologies, 30(6): 8109-8138.
[7] Kumamoto, K., Cong, Y., Masuhara, H. (2025). An interactive learning environment for program design//Henz M, Hermans F, Patterson D. Proceedings of the 2025 ACM SIGPLAN International Symposium on SPLASH-E, SPLASH-E 2025. Singapore: Association for Computing Machinery, 66-74.
[8] Shin, J., Cruz-Castro, L., Yang, Z., et al. (2024). Understanding optimal interactions between students and a chatbot during a programming task//2024 Winter Simulation Conference, WSC. Orlando, FL: IEEE, 3106-3117.
[9] Liu, Z., Xing, W., Jiao, X., et al. (2025). What are the differences between student- and ChatGPT-generated pseudocode? Detecting AI-generated pseudocode in high school programming using explainable machine learning. Education and Information Technologies, 30(11): 14853-14892.
[10] Garcia, M. B. (2025). Teaching and learning computer programming using ChatGPT: a rapid review of literature amid the rise of generative AI technologies. Education and Information Technologies, 30(12): 16721-16745.
[11] Koutcheme, C., Hellas, A. (2024). Propagating large language models programming feedback//Proceedings of the Eleventh ACM Conference on Learning@Scale, L@S 2024. Atlanta, GA: Association for Computing Machinery, 366-370.