Original Research Article

Open Access

Empowering Inquiry-Based Higher Vocational Education with Large Language Models: A Case Study of Python Language Course

Ying-Chuan Tang*

Neusoft Institute Sichuan, Chengdu, Sichuan 611844, China

***Correspondence to:** Ying-Chuan Tang, Neusoft Institute Sichuan, Chengdu, Sichuan 611844, China

Abstract: With the rapid iteration of Large Language Models (LLMs), their applications have permeated various fields, demonstrating significant value in higher vocational education. Research indicates that the traditional Inquiry-Based Learning (IBL) model faces challenges such as feedback latency, disconnection between industry and education, and homogenized evaluation dimensions. Taking the Python language course as an example, this paper analyzes the current limitations of the IBL model. It explores a path to refine traditional IBL in terms of course content and evaluation systems by integrating LLM-based agents. The study aims to provide a novel approach for higher vocational education to enhance the quality of instructional content.

Keywords: Inquiry-Based Learning; Large Language Model; Higher Vocational Education

1. Research Background

Inquiry-Based Learning (IBL) is a student-centered instructional model that emphasizes knowledge construction through active exploration, problem-solving, and critical thinking. Its strength lies in enhancing students' practical innovation, teamwork, and professional adaptability, making it particularly suitable for the skill-oriented characteristics of higher vocational education. Typical applications include: interdisciplinary real-world problem-solving projects (e.g., fault diagnosis systems combining mechanical engineering and IT); professional scenario simulations (e.g., emergency process optimization in nursing); and school-enterprise collaborative research (translating industry technical hurdles into student inquiry tasks). However, during implementation in vocational colleges, students often struggle due to weak theoretical foundations and insufficient autonomy. Moreover, IBL requires substantial faculty, equipment, and personalized guidance. Crucially, evaluation standards often fail to quantify the inquiry process, leaving students without timely feedback and causing them to fall into "blind confidence" or complete confusion. Additionally, the flexibility of IBL may weaken teaching outcomes in disciplines that emphasize strict standardization.

The rapid development of Large Language Models (LLMs) now allows for more personalized resource filtering. Empowered by LLMs, agent systems can parse students' semantic intentions, providing more precise and rapid feedback compared to traditional IBL. For instance, if a student describes a need to "extract

numbers from text and sum them," the system not only generates code involving file reading and regular expressions but also highlights critical nodes like try-except exception handling. This real-time interaction mechanism reconstructs the learning path, aligning the cognitive load distribution with the skill acquisition patterns in Anderson's ACT-R theory—where skill formation follows the transition from declarative to procedural knowledge [1]. The Action Plan for Quality Improvement of Vocational Education (2020-2023) also explicitly targets "deepening teaching model reforms and promoting the deep integration of AI with education."

In this context, the teaching methodology of the Python language urgently requires reconstruction. Although Python's nature as an interpreted language allows for quick application, its dynamic typing and indentation sensitivity pose significant learning hurdles. Research shows that 42.6% of syntax errors in traditional teaching stem from delayed feedback, causing error patterns to solidify into cognitive biases. Integrating LLM agents to correct errors in real-time can prevent the formation of incorrect coding habits.

2. Current Status of Traditional IBL

Traditional IBL in vocational programming education faces three structural contradictions. The foremost is the gap between textbook content and engineering practice. In Python Programming courses, cases long focused on basic console operations, while industry demand for API calls and structured data processing exceeds 80%. Consequently, students require an extra 3 weeks to bridge the skill gap during internships. The second issue is inefficient feedback. Research indicates the average cycle for manual code grading is 52 hours, during which 60% of students repeat erroneous practices over 8 times. Finally, single-dimension evaluations exacerbate the industry-education disconnect. Current systems focus excessively on results, ignoring code specifications and maintainability, leading 67% of students to adopt hard-coding strategies that industry mentors label as "lacking engineering value."

The introduction of LLM agents offers optimization. For dynamic scenario adaptation, LLMs can parse corporate codebases to generate real-time technical cases, reducing internship adaptation periods from 3 weeks to 1.1 weeks and increasing structured data training to 82%. Regarding feedback, integrated LLM assistants can instantly locate logic errors (e.g., variable scope conflicts), reducing error repetition rates from 7.9 to 1.2 times—a 4.3-fold increase in efficiency [2]. Furthermore, multi-dimensional evaluation systems using LLMs integrate static analysis (e.g., Flake8) with dynamic performance testing to quantify modularity and algorithm efficiency, driving students to optimize "brute force" strategies.

3. Reform Paths for the IBL Teaching Model

3.1 LLM-Driven Agent Systems

Compared to traditional search engines, LLM agent systems represent a breakthrough in information retrieval. Utilizing deep learning architectures, these systems process structured and unstructured data in parallel, revealing latent correlations through feature vector mapping. Their advantages are threefold: first, semantic understanding modules can resolve ambiguous queries (e.g., recommending specific technical stacks for "beginner projects"); second, real-time mechanisms monitor cognitive load, pushing micro-lectures when error rates on specific knowledge points exceed thresholds.

Empirical studies show these systems lower the threshold for acquiring useful information. A controlled experiment at UC Berkeley showed that students using intelligent retrieval systems increased literature search efficiency by 57%, with a resource matching accuracy of 89%, compared to just 42% for traditional search engines [3].

3.2 Curriculum Reconstruction Strategies

Project-driven teaching is a common method in IBL to simulate industry requirements. In Python teaching, LLMs can provide students with logic and solutions. When students encounter difficulties, they can describe problems in natural language—such as "How to count word frequency in an article"—and the system provides step-by-step solutions with critical warnings. This breaks the time-space constraints of traditional classrooms, providing a "omnipresent mentor." More importantly, LLMs guide learners from simple implementation to standardized practices, such as adding exception handling to enhance stability, thereby fostering engineering thinking.

Under LLM empowerment, traditional project-driven models can be optimized:

(1) Challenging Tasks: Teachers can assign more complex tasks rather than focusing only on simple textbook functions.

(2) Autonomous Inquiry: Students can use LLMs to explore advanced content independently, increasing motivation.

(3) Iterative Optimization: Before submission, students use LLMs to check code specifications. They submit the AI's optimization suggestions alongside their code, providing teachers with rich data to adjust teaching strategies [4].

3.3 Innovation in the Dual-Track Evaluation Mechanism

In evaluation, LLMs can access local knowledge bases to provide transparent and immediate feedback. Once teachers organize comprehensive grading standards, the LLM becomes an efficient assistant. Students receive detailed reports—acting like a strict but fair examiner—identifying issues like non-standard naming or code redundancy. For example, the system might suggest: "Use the 'with' statement to close files automatically to prevent resource leaks," linked to a relevant tutorial video. This allows students to complete multiple optimization rounds before final submission.

Final grades consist of system scores and teacher evaluations. Teachers can focus on innovation, workload, and practical value, while LLMs handle technical accuracy and specifications, preserving the teacher's judgment space while ensuring objectivity. Using this dual-track system, teachers can propose high-freedom tasks, such as simulating a "Smart Home Control System." While such tasks are difficult to grade manually in traditional IBL due to their diverse solutions, agents make them feasible. Teachers upload grading rubrics as structured data to a local knowledge base to prevent the LLM from using incorrect internet-based standards. The agent handles objective aspects like logic, syntax, and keyword usage. Students can check and modify their code indefinitely before the final deadline. Once the student is satisfied with the system's score, the teacher performs a final review of the project's complexity and technical extensions. This allows every teacher to have a "24-hour teaching assistant," achieving the goal of improving quality and efficiency [5].

4. Summary and Outlook

This paper delineates the implementation path for LLM-driven reform in Python courses at vocational colleges. By building intelligent cognitive scaffolds, reconstructing project-oriented curricula, and establishing dual-track evaluation mechanisms, it effectively addresses feedback delays, industry-education gaps, and single-dimension evaluations. LLM tools not only achieve instant diagnosis but also recommend enterprise-level tasks via dynamic knowledge graphs, strengthening students' engineering mindsets.

Future research can extend in three directions:

(1) Exploring the deep integration of LLMs with Online Judge (OJ) systems to build personalized learning paths via error pattern analysis.

(2) Developing cross-curricular knowledge graphs to facilitate interdisciplinary connections.

(3) Establishing alliances among vocational colleges to share pedagogical analysis models under data security protocols, promoting the universal application of intelligent educational technologies.

References

[1] Wu Yunchao, et al. Cognitive Spaced Repetition Learning Method Based on ACT-R [J]. Journal of East China University of Science and Technology (Natural Science Edition), 2024 (1-10).

[2] Simon Goorney, et al. A framework for curriculum transformation in quantum information science and technology education [J]. European Journal of Physics, 2024 (45).

[3] Alvarez-Gonzalez L A, et al. Using LAMS to support engineering student learning: Two case studies [J]. IEEE, 2017.

[4] Zhu, Y, et al. Impact of assignment completion assisted by Large Language Model-based chatbot on middle school students' learning [J]. Educ Inf Technol 30, 2025.

[5] Mekterović I, et al. Scaling Automated Programming Assessment Systems [J]. Electronics, 2023, 12(4).