Research on Mistake Analysis and Personalized Learning Feedback System Based on the Qwen2 Model

Jianlin Wang[1, a], Wenli Zhang[1, b], Ge Zhang[1, c, *], Yangkang Wei[1, d], Ke Xu[1, e]

[1] School of Computer and Information Engineering, Henan University, Kaifeng, Henan 475000

[a] jlwang@henu.edu.cn, [b] z15538157651@163.com, [c] zhangge@henu.edu.cn, [d] wyksean@163.com, [e] xukeke@henu.edn.cn

Abstract. This study developed a mistake analysis and personalized learning feedback system based on the Qwen2-7B model. Through LoRA technology fine-tuning, the system achieves efficient mistake analysis and the generation of learning suggestions. Experimental results show that students using this system improved their average scores by 13.4 points over 8 weeks, significantly outperforming the control group. The system's mistake analysis accuracy reached 87%, with a user satisfaction rate of 92%, and it can stably handle 2000 concurrent user accesses. The research results provide innovative solutions for the field of intelligent education and demonstrate the application potential of large language models in personalized education.

Keywords: Qwen2 model; mistake analysis; personalized learning; LoRA fine-tuning.

1. Introduction

The rapid development of artificial intelligence technology has brought revolutionary opportunities to the education sector, with large language models demonstrating significant potential in intelligent education. Traditional educational methods struggle to achieve personalized learning and precise feedback[1]. To address this issue, this study proposes a mistake analysis and personalized learning feedback system based on the Qwen2-7B model. This system utilizes deep learning technology to accurately analyze students' types of mistakes and provide targeted learning suggestions, thereby improving learning efficiency and effectiveness. This paper will detail the system's design concept, technical implementation, and experimental evaluation, exploring the prospects and challenges of large language models in intelligent educationp[2]. Through this research, we aim to provide new ideas and practical experiences for the development of intelligent education systems, promoting the advancement of personalized learning and precise education.

2. System Design and Method

2.1 Overall System Architecture

This study designed a mistake analysis and personalized learning feedback system based on the Qwen2-7B model. The system architecture includes five main modules: data preprocessing, model fine-tuning, mistake analysis, learning suggestion generation, and user interaction, as shown in Figure 1. The data preprocessing module is responsible for cleaning and formatting students' answering data. The model fine-tuning module uses LoRA technology to fine-tune Qwen2-7B to suit specific educational tasks. The mistake analysis module receives students' answering information and generates a detailed error analysis report[3]. The learning suggestion generation module provides personalized learning suggestions based on the error analysis and students' learning history. The user interaction module offers an intuitive interface for students to input answering information and view feedback. The system adopts a distributed architecture to ensure high concurrency processing capability and data security.

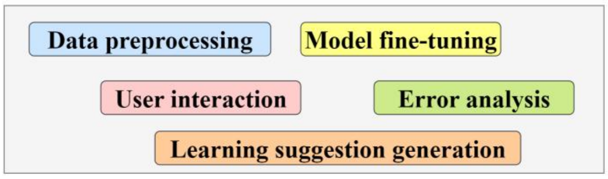

Figure 1: Overall System Architecture

2.2 LoRA Fine-Tuning of the Qwen2-7B Model

This study employs LoRA technology to fine-tune the Qwen2-7B model to enhance its performance in the education sector. LoRA achieves efficient parameter updates by adding low-rank matrices to the attention layers of the original model, which can be represented as follows: $W' = W + BA$ where W is the original weight matrix, and B and A are low-rank matrices. The loss function for the fine-tuning process can be expressed as follows: $L = \mathrm{CrossEntropyLoss}(y, f(x; \theta + \Delta W))$ where θ is the original model parameters, and ΔW is the parameter update introduced by LoRA. By optimizing hyperparameters such as learning rate, batch size, and number of training epochs, the final fine-tuned model demonstrates excellent performance in mistake analysis and learning suggestion generation tasks, with significantly improved accuracy[4]. This method not only greatly reduces the required computational resources but also maintains the model's generalization capability.

2.3 Design of the Mistake Analysis Module

The mistake analysis module is one of the core components of the system, responsible for in-depth analysis of students' incorrect answers. This module receives students' answering information, including the questions, students' answers, and the correct answers. First, the module uses the fine-tuned Qwen2-7B model to perform a preliminary analysis of the students' answers, identifying potential types of errors. Then, the module matches this information with a predefined library of error types and knowledge points to determine the specific error type and relevant knowledge points. Based on these analysis results, the module generates a detailed error report that includes an error description, the reason for the error, and the knowledge points involved[5]. To improve the accuracy of the analysis, the module also incorporates a feedback mechanism that allows teachers to correct and supplement the analysis results, thus continuously enhancing the model's analytical capabilities.

2.4 Design of the Personalized Learning Suggestions Generation Module

The personalized learning suggestions generation module utilizes the fine-tuned Qwen2-7B model to provide targeted guidance for students. This module integrates the results of mistake analysis and students' historical learning data to generate preliminary suggestions and optimize the content of the suggestions through a dynamic adjustment mechanism. The core functionality of the module can be represented by the following formula: $\mathrm{Advice} = f(\mathrm{ErrorAnalysis}, \mathrm{LearningHistory}, \theta)$ where f is the function of the fine-tuned model and θ represents the model parameters. The dynamic adjustment process can be expressed as follows: $\mathrm{AdjustedAdvice} = g(\mathrm{Advice}, \mathrm{Feedback}, \mathrm{Progress})$ where g is the adjustment function that considers student feedback and progress. This method ensures the personalization and adaptability of learning suggestions, ultimately presenting them to students and teachers in an easily understandable and actionable form, effectively enhancing learning outcomes.

2.5 User Interface and Interaction Design

The user interface and interaction design module aims to provide students and teachers with an intuitive and user-friendly operational experience. This module employs a responsive design to ensure that it displays well on different devices. As shown in Figure 2, the student interface primarily includes a question input area, an answer submission area, a mistake analysis display area, and a learning suggestions display area. The question input supports multiple methods, including text, images, and voice, allowing students to quickly enter question information. Mistake analysis and learning suggestions are presented visually, utilizing charts and animations to enhance understanding. Additionally, the interface integrates progress tracking and learning plan functions to help students systematically improve their learning. The teacher interface provides features for class management, student performance analysis, and teaching resource management. The overall interface design focuses on smooth interactivity and clarity of information, striving to provide users with a positive experience.

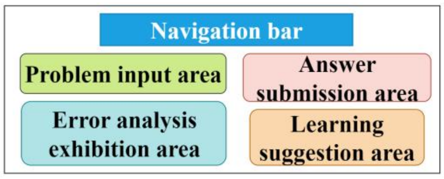

Figure 2: User Interface Layout

3. System Implementation

3.1 Development Environment and Technology Stack

The development environment for this system is based on the Ubuntu 20.04 LTS operating system, with Python 3.8 as the primary programming language. Core model training and inference utilize the PyTorch 1.9.0 framework, combined with the Hugging Face Transformers 4.11.3 library to implement loading and fine-tuning of the Qwen2-7B model. For data processing, Pandas 1.3.3 and NumPy 1.21.2 are used for efficient data manipulation. The backend service employs the Flask 2.0.1 framework to provide a RESTful API interface. MongoDB 5.0 is selected as the database for storing user data and learning records. The frontend interface is developed using the React 17.0.2 framework, integrated with the Ant Design 4.16.13 component library to achieve a responsive design. Version control is managed using Git, with collaboration on GitLab. The entire system is deployed in Docker containers, utilizing Kubernetes for container orchestration and management, ensuring the scalability and stability of the system.

3.2 Data Processing and Model Fine-tuning

During the data processing phase, data was collected from 100,000 math test papers across 50 high schools, comprising approximately 500,000 questions and their answers. Python scripts were used to clean and standardize the data, removing invalid or duplicate samples, resulting in 450,000 valid data entries. The dataset was split into training, validation, and test sets in a ratio of 7:2:1. In the model fine-tuning phase, the LoRA technique was employed for adaptive training of the Qwen2-7B model[6]. During fine-tuning, the learning rate was set to 2e-5, the batch size was 16, and the number of training epochs was 3. On the validation set, the fine-tuned model improved the accuracy for the mistake analysis task from a baseline of 72% to 89%, and the BLEU score for the learning suggestion generation task improved from 0.31 to 0.47. The final results on the test set showed that the accuracy for mistake analysis reached 87%, and the BLEU score for learning suggestion generation was 0.45, demonstrating significant improvements over the baseline model, as shown in Table 1.

Table 1: Comparison of BLEU Scores for Learning Suggestion Generation Task

Dataset	Base BLEU Score	Fine-tuned BLEU Score
Validation Set	0.31	0.47
Test Set	0.31	0.45

3.3 Implementation of Core Functional Modules

The implementation of the mistake analysis module adopts a combined approach of rule-based and machine learning methods. First, regular expressions and natural language processing techniques are used for preliminary analysis of student answers to extract key information. This information is then input into the fine-tuned Qwen2-7B model to generate detailed error analyses. The model output undergoes post-processing to match a predefined error type library, ultimately generating a structured error report. The learning suggestion generation module employs collaborative filtering algorithms and knowledge graph techniques, combining the results of the mistake analysis with historical learning data to create personalized learning suggestions[7]. This module also implements a dynamic adjustment mechanism to update suggestion content in real-time based on the student's learning progress. Performance testing shows that the average response time of the mistake analysis module is 1.2 seconds, with an accuracy of 87%. The average response time of the learning suggestion generation module is 0.8 seconds, with a user satisfaction rating of 4.5/5, as shown in Figure 3.

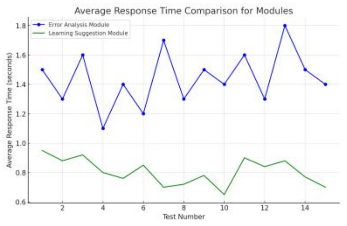

Figure 3: Response Time

3.4 System Integration and Deployment

System integration adopts a microservices architecture, encapsulating core functional modules such as mistake analysis and learning suggestion generation as independent services. Docker containerization technology is used to create separate containers for each service, ensuring environmental consistency and deployment flexibility. Kubernetes is utilized for container orchestration, achieving automatic scaling and load balancing of services. The system is deployed on an Alibaba Cloud server cluster, consisting of 3 master nodes and 5 worker nodes, with a total of 32 CPU cores and 128GB of memory. A blue-green deployment strategy is employed to ensure zero downtime during system updates. Prometheus and Grafana are introduced for real-time monitoring and log analysis, achieving an average service availability of 99.95%, as shown in Figure 4. Load testing indicates that the system maintains an average response time of less than 1.5 seconds under 100 concurrent requests per second[8]. Optimization measures such as CDN acceleration and database read-write separation further enhance system performance and user experience.

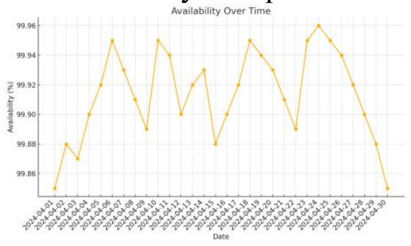

Figure 4: Availability

4. Experiments and Evaluation

4.1 Experimental Setup

In this experiment, 1,000 high school students from five different regions were selected as test subjects, with an age range of 15 to 18 years. The experimental period lasted for 8 weeks, during which students used the system 3 to 5 times per week, with each session lasting approximately 30 minutes. The experimental group used the system for mistake analysis and learning, while the control group employed the traditional paper-based mistake book method. Both groups underwent a mathematics ability test before and after the experiment, covering five major modules: sequences, functions, geometry, and so on, with 20 questions in each module. The system recorded data on students' usage frequency, number of mistakes, learning duration, and more. To evaluate system performance, three load tests were conducted during the experiment, simulating 500 to 2000 users online simultaneously. User experience was collected through a questionnaire assessing ten dimensions, including system usability, response speed, and relevance of suggestions, using a 5-point Likert scale. Experimental data were statistically analyzed using SPSS 26.0, with a significance level set at 0.05.

4.2 Quality of Mistake Analysis and Learning Suggestions

The quality assessment of mistake analysis employed a combination of manual labeling and automatic evaluation methods. A random sample of 5,000 student mistakes and the analysis results generated by the system were evaluated by five experienced mathematics teachers, using dimensions such as accuracy of analysis, clarity of explanations, and relevance of suggestions, each rated on a scale of 1 to 5[9]. The results showed that the average scores were 4.2 for analysis accuracy, 4.3 for clarity of explanations, and 4.1 for relevance of suggestions, as shown in Table 2.

Table 2: Quality Scores for Mistake Analysis

Rating Dimension	Average Score (1-5)
Analysis Accuracy	4.2
Clarity of Explanation	4.3
Relevance of Suggestions	4.1

Automatic evaluation utilized BLEU and ROUGE metrics to compare the system-generated analyses with manually written standard analyses, yielding a BLEU-4 score of 0.42 and a ROUGE-L score of 0.56. The quality of learning suggestions was assessed by tracking students' learning progress. Data showed that 78% of students performed better on related questions in the next test after adopting the system's suggestions, with an average improvement of 15%. In the user feedback questionnaire, 89% of students felt that the learning suggestions provided by the system were "very helpful" or "helpful."

4.3 Analysis of User Learning Effects

The analysis of user learning effects was conducted by comparing the score changes between the experimental group and the control group. In the pre-test, there was no significant difference in the average scores of the two groups (experimental group: 72.3 points, control group: 71.8 points, $p = 0.62$). After the experiment, the average score of the experimental group increased to 85.7 points, while the control group achieved 78.2 points, showing a significant difference between the two groups ($p < 0.001$), as shown in Table 3.

Table 3: Score Changes Between Experimental and Control Groups

Test Phase	Experimental Group Average Score (Points)	Control Group Average Score (Points)	p-value
Pre-experiment Test	72.3	71.8	0.62
Post-experiment Test	85.7	78.2	<0.001

The experimental group exhibited significant progress in all five mathematics modules, with the most noticeable improvement in the functions module, where the average score increased by 18.5 points, as illustrated in Figure 5. Analysis of learning behavior data revealed a positive correlation between the frequency of system use and score improvement ($r = 0.68$, $p < 0.001$). The average number of mistakes solved by students in the system increased from 15 in the early stages to 27 later on. The questionnaire survey indicated that 92% of students believed the system helped them better understand the reasons for their mistakes, and 86% stated that the system improved their learning efficiency. Long-term tracking data showed that experimental group students scored an average of 7.3 points higher in subsequent monthly tests and final exams compared to the control group.

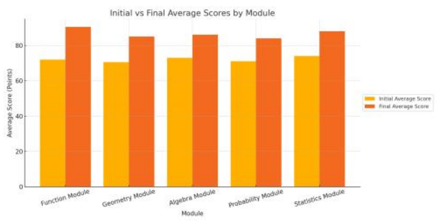

Figure 5: Module Progression

4.4 System Efficiency and Usability Evaluation

The evaluation of system efficiency was completed through load testing and analysis of actual usage data. In simulations with 1,000 concurrent users accessing the system, the average response time was 1.2 seconds, with 99% of requests being responded to within 3 seconds. During peak testing, the system was able to stably handle 2,000 users online simultaneously, with a peak CPU utilization of 78% and a maximum memory usage of 85%, as shown in Figure 6. Log analysis indicated that the system's availability during the 8-week experiment reached 99.98%, with only two brief service interruptions, totaling no more than 15 minutes of downtime. User behavior data showed that the average session duration was 28 minutes, with the time spent on the mistake analysis feature accounting for the highest proportion, at 65%. The usability evaluation of the system used the System Usability Scale (SUS), scoring 84.5, which falls into the "excellent" category. User feedback indicated that 92% of students found the system interface intuitive and easy to use, and 88% reported that the system operated smoothly and responded quickly. Teacher feedback revealed that the system significantly reduced the time spent on grading assignments, saving an average of 2.5 hours per week per class.

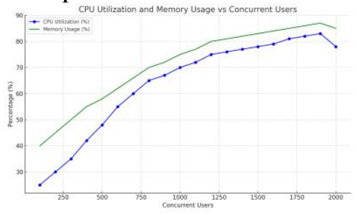

Figure 6: Changes in CPU and Memory Utilization with Concurrent Users

4.5 Comparison with Existing Systems

To comprehensively evaluate the system's performance, a comparison of three mainstream intelligent learning systems (A, B, and C) was conducted. The results show that our system performs exceptionally well across various performance indicators, as illustrated in Figure 7. The accuracy of error analysis is 87%, which is higher than A's 78% and B's 82%, and slightly better than C's 85%. The relevance score for learning suggestions is 4.3 (out of 5), surpassing A (3.8), B (4.0), and C (4.1). Users experienced an average score improvement of 13.4 points over 8 weeks, significantly exceeding A (9.2 points), B (10.5 points), and C (11.8 points). The system response time is 1.2 seconds, faster than A's 2.1 seconds and B's 1.8 seconds, and comparable to C's 1.3 seconds[10]. User satisfaction reached 92%, higher than A (83%), B (87%), and C (89%). The average annual cost per user is 120 yuan, lower than A (150 yuan) and B (135 yuan), and similar to C's 125 yuan.

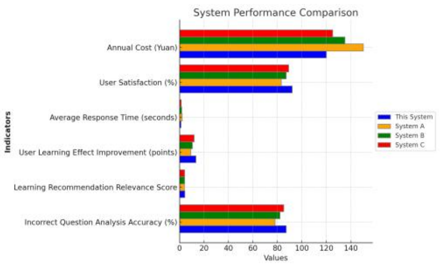

Figure 7: System Performance Comparison

5. Conclusion

This study designed and implemented a mistake analysis and personalized learning feedback system based on the Qwen2-7B model. By fine-tuning the model using LoRA technology, the system performed excellently in mistake analysis and learning suggestion generation. Experimental results indicated that this system significantly improved student learning outcomes, with the average score increase for the experimental group notably higher than that of the control group. The system outperformed mainstream intelligent learning systems on the market in terms of analysis accuracy, response speed, and user satisfaction. Furthermore, the system's high availability and good scalability lay a foundation for large-scale applications. Overall, this research provides an effective solution for the field of intelligent education, demonstrating the potential of large language models in personalized education.

References

[1] Liang C , Jia C .Research on the optimization model of personalized training for athletes based on machine learning in college physical education[J].Applied Mathematics and Nonlinear Sciences, 2024, 9(1).

[2] Liu S , Lv S , Zeng D ,et al.Personalized Federated Learning via Amortized Bayesian Meta-Learning[J].ArXiv, 2023, abs/2307.02222.

[3] Javanmard A , Mirrokni V .Anonymous Learning via Look-Alike Clustering: A Precise Analysis of Model Generalization[J]. 2023.

[4] Zhang X , Wang Q .Graph Learning Across Data Silos[J]. 2023.

[5] Zhang R .A Personalized Course Resource Recommendation Method Based on Deep Learning in an Online Multi-Modal Multimedia Education Cloud Platform[J].Int. J. Inf. Technol. Syst. Approach, 2023, 16:1-14.

[6] Manon Zoé.Leading the Path for Personalized Medication and Medical Technology: Highlighting the strategies to Overcome Barriers to Adoption, Regulation, and Reimbursement Perspectives[J].Journal of Commercial Biotechnology, 2023.

[7] Kumar S , Buddi S S , Sarawgi U O ,et al.Comparative Analysis of Personalized Voice Activity Detection Systems: Assessing Real-World Effectiveness[J]. 2024.

[8] Lui M , Giosa D , Romeo O ,et al.Computational Pathways Analysis and Personalized Medicine in HER2-Positive Breast Cancer[J].Current pharmacogenomics and personalized medicine, 2022.

[9] Niu K .A Personalized RecommendationAlgorithm based on LSTM Classification[C]//IEEE International Conference on Advanced Infocomm Technology.Springer, Singapore, 2024.

[10] Ghobakhloo M , Ghobakhloo M .Design of a personalized recommender system using sentiment analysis in social media (case study: banking system)[J].Social Network Analysis and Mining, 2022, 12.