ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202607_9(7).0008

Teaching Practice of Explainable Feedback in Programming Empowered by Large Language Models

Xinzhi Li, Yixin Zhang, Jingjing Xie*, Rui Zhang, Mengmeng Liu, Jing Zhao

School of Emergency Technology and Management, University of Emergency Management, Langfang 065201, Hebei Province, China

* Corresponding author

Abstract

Against the background of artificial intelligence and large language models, undergraduate natural language processing programming tasks often face delayed feedback, interrupted debugging processes, and insufficient result interpretation. This paper analyzes the scaffolding role of generative artificial intelligence in error diagnosis, step-by-step prompting, and autonomous revision, and proposes an explainable feedback approach characterized by "AI-assisted diagnosis, teacher-guided rule constraints, student-led revision, and process-based reflection". Taking named entity recognition and sentiment analysis as practical tasks, this paper designs an AI-supported feedback process for programming ability development to improve feedback timeliness, strengthen problem-locating and debugging abilities, and enhance the problem-solving experience in complex programming tasks.

Keywords

Large language models; generative artificial intelligence; programming teaching; explainable feedback; natural language processing; cognitive scaffolding.

1. Introduction

The development of generative artificial intelligence and large language model provides new technical support for the teaching reform of higher education. Its application has gradually extended from data generation and intelligent question answering to learning feedback, process guidance and teaching assistance [1-3]. For programming courses, students not only need to understand algorithm principles and grammatical structures, but also need to complete comprehensive practical activities such as code implementation, error localization, program debugging and result analysis in real tasks [4, 5]. Especially in artificial intelligence related courses, programming tasks often involve model structure understanding, parameter setting, data processing and result interpretation at the same time. Students are prone to problems such as ' running code, not positioning errors ', ' getting results, not explaining reasons '. This paper takes the undergraduate natural language processing programming task as the practice scenario, and focuses on how the big model can support students to complete error diagnosis, debugging correction and reflection improvement in complex programming tasks.

Traditional programming teaching usually adopts the feedback method of ' classroom teaching-after-class submission-teacher correction ', which can complete the result evaluation, but it is difficult to intervene in the students ' debugging process in time. When students encounter difficulties in model construction, parameter setting or semantic analysis, they often can only rely on repeated trial and error, network retrieval or peer communication to solve problems, which can easily cause learning interruption and ineffective cognitive burden [6, 7]. Generative artificial intelligence can generate explanatory feedback based on error messages, code snippets, and problem descriptions submitted by students, providing students with error

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202607_9(7).0008

localization, cause analysis, and modification directions [8]. However, if it is simply used as a code generation or answer generation tool, it may weaken students ' ability of independent thinking, program understanding and problem solving. Therefore, the key to the application of large language models in programming teaching is not to let them replace students to complete code writing, but to transform them into an explanatory feedback tool for the learning process, so that they can provide step-by-step support around error localization, cause interpretation, and modification paths, and help students form program analysis and correction capabilities in independent debugging [9, 11].

Based on the above teaching practice, this paper takes the undergraduate natural language processing programming task as an example, and designs and practices a large model-enabled program design explanatory feedback process around two typical scenarios of named entity recognition and sentiment analysis. This process emphasizes the teaching path of ' students submit questions-AI step-by-step diagnosis-teachers' rule constraints-students 'self-correction-process record reflection ',and guides students to retain independent judgment, code modification and result verification process while obtaining instant feedback. This paper focuses on task design, feedback implementation, typical cases and teaching reflection, in order to provide reference for the teaching reform of programming courses under the background of artificial intelligence and large language model.

2. Instructional Task Design

In order to avoid large models being simply used as ' code generator ' or ' answer generator ' in programming teaching, this paper embeds them into the practical teaching of programming courses, and takes undergraduate natural language processing programming tasks as the specific carrier, focusing on serving students ' error positioning, cause analysis and autonomous correction in complex programming tasks. The teaching task design follows the principle of ' task reality, problem typical, feedback interpretable, correction traceable '.Two representative tasks of named entity recognition and sentiment analysis are selected as practice carriers, which correspond to the training needs of code debugging ability and semantic reasoning ability respectively. The overall teaching design framework is shown in Figure 1.

The teaching design of this paper consists of four levels : curriculum practice, task ability, AI feedback and student correction. Firstly, the teaching object focuses on the complex programming practice tasks in the programming courses, and takes the undergraduate natural language processing task as the application scenario. Secondly, two typical tasks are set around named entity recognition and sentiment analysis. The former focuses on model structure understanding and code debugging, while the latter focuses on semantic reasoning and result interpretation. Thirdly, according to the characteristics of different tasks, differentiated AI feedback methods are designed, including error location, structural interpretation, semantic disassembly and logical prompt. Finally, students complete self-modification, result verification and problem reflection with the help of prompts. The design emphasizes the auxiliary and scaffolding of the large model, and highlights its process feedback value in the training of programming ability.

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202607_9(7).0008

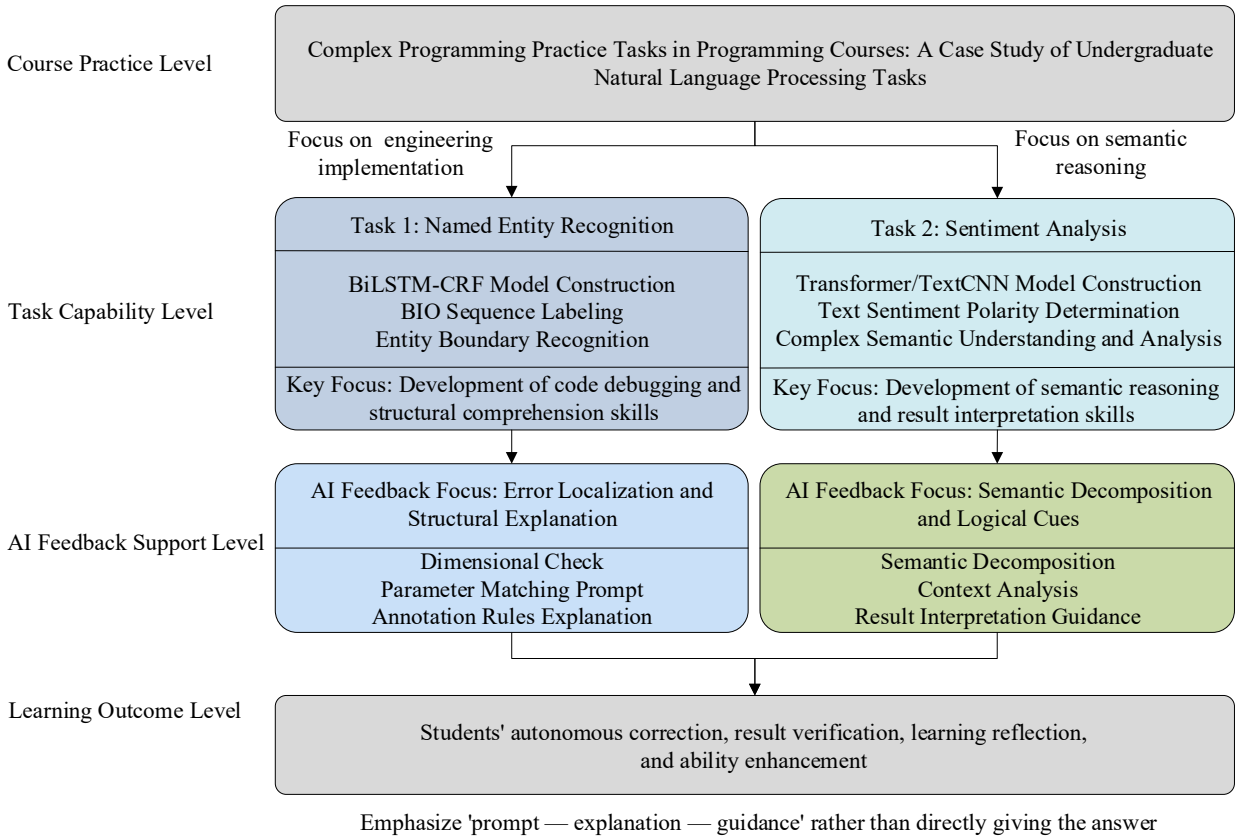

Figure 1. Framework for Program Design Explanatory Feedback Teaching Empowered by Large Models

2.1. Task Selection and Competency Goals

The task of named entity recognition is mainly used to cultivate students' ability of engineering implementation and code debugging. In teaching practice, students need to build a BiLSTM-CRF model based on PyTorch to complete the recognition of entity information such as person name, place name and organization name in the text. The task involves knowledge points such as sequence labeling, word vector representation, recurrent neural network, conditional random field and BIO labeling rules. The common problems in the implementation process of students include the confusion of input sequence, hidden state and label sequence dimension relationship, the wrong number of CRF layer labels, the mismatch of transfer matrix dimension, and the inaccurate recognition of entity boundary. Therefore, this task is suitable for training students ' ability to understand model structure, analyze error information and complete code correction.

The task of sentiment analysis is mainly used to cultivate students ' ability of semantic reasoning and result interpretation. In teaching practice, students need to use Transformer or TextCNN model to complete text sentiment polarity classification. Compared with named entity recognition, this task not only requires students to complete model construction and training, but also requires them to understand how the model judges emotional tendencies based on context semantics. Students ' common difficulties include relying on a single emotional word for judgment, ignoring negative and transitional structures, difficulty in analyzing ironic expressions, and inability to explain the basis of model classification. Therefore, this task is suitable for training students to shift from surface word judgment to context semantic analysis, and to improve their ability to interpret the output of the model.

In order to further clarify the ability orientation and feedback focus of the two types of tasks, this paper designs the teaching tasks, students ' common difficulties and AI feedback methods, as shown in Table 1.

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202607_9(7).0008

Table 1. The Correspondence Between Teaching Tasks and AI Feedback Priorities

Teaching Task	Core Knowledge Points	Common Difficulties Faced by Students	Key Points of AI Feedback
Named Entity Recognition	Sequence labeling, BiLSTM-CRF, BIO tagging	Tensor dimension errors, label boundary confusion, incorrect CRF layer parameter settings	Error localization, explanation of dimensional relationships, parameter matching prompts, and annotation rule explanations
Sentiment Analysis	Text classification, Transformer, TextCNN, semantic understanding	Relying on keywords for judgment, ignoring contextual semantics, and difficulty in explaining the basis for classification	Semantic decomposition, logical reasoning prompts, context analysis, and guidance for result interpretation

Based on Table 1, this paper further designs differentiated feedback methods around the two types of tasks, so that the big model mainly plays the role of prompting, explaining and guiding, rather than directly replacing students to complete code writing or result judgment.

2.2. Key Design of Task Feedback

Based on the difference of ability objectives between the two types of tasks, this paper designs the feedback content of generative artificial intelligence differently. The basic principle is not to directly output the complete answer, but to provide comprehensible, executable and traceable tips around the students ' current problems, so that the students can complete the independent judgment and correction with the support of feedback. For the named entity recognition task, the feedback focuses on ' error location-structure interpretation-code correction ' ; for the sentiment analysis task, the feedback focuses on ' semantic disassembly-logical prompt-result explanation '.

In the task of named entity recognition, students ' common difficulties mainly focus on model structure understanding and code implementation details. Therefore, generative artificial intelligence mainly undertakes the functions of error location and structural interpretation. When students submit error information or key code fragments, AI does not directly give the complete program, but guides them to check the key links such as input tensor shape, BiLSTM output dimension, linear mapping layer output dimension, CRF label number and BIO labeling rules in turn. For example, when there is a tensor dimension mismatch problem, AI can prompt students to judge whether the error comes from batch dimension setting, inconsistent label category number or CRF layer parameter definition error by comparing the input and output relationship of each layer of the model. In this way, students can transform abstract error reporting into specific checking steps, so as to understand the internal structure of the model while correcting the code.

In the task of sentiment analysis, students ' common difficulties mainly focus on semantic understanding and result interpretation. Therefore, generative artificial intelligence mainly undertakes the functions of semantic disassembly and logical prompt. When students have questions about the classification results, AI can guide them to focus on emotional words, negative words, adversative relations, modified objects and contextual semantics, rather than directly giving judgment conclusions. For example, for texts containing expressions such as ' not bad ' and ' hardly enjoyable ', AI can prompt students to analyze the scope of negation, the object of evaluation and the overall context, so that they can gradually form an analysis path from keyword matching to semantic relationship judgment. In this way, AI feedback not only

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202607_9(7).0008

serves the interpretation of results, but also helps students understand the analysis methods of complex semantic phenomena in natural language processing.

In the above design, teachers assume the role of feedback rule making and teaching goal checking. On the one hand, teachers require students to submit error information, key code, personal preliminary judgment and modify records when using AI, so as to avoid directly requiring AI to generate complete answers. On the other hand, teachers supplement the explanation according to the problems that students focus on, and connect AI feedback with course knowledge points, classroom teaching objectives and homework evaluation requirements. Therefore, generative artificial intelligence is limited to the teaching role of ' prompt, explanation and guidance '.Students still need to complete code modification, result verification and problem reflection, so as to realize the transformation from result correction to process support.

3. Feedback Implementation and Teaching Reflection

3.1. Implementation Process of Explanatory Feedback

In the process of teaching implementation, this paper embeds the big model into the key link of the programming practice task, and takes the natural language processing programming task as the specific application scenario to form an explanatory feedback process of ' problem submission-stepwise diagnosis-independent correction-result verification-teacher supplement '. When students complete the task of named entity recognition or sentiment analysis, they need to submit error information, key code fragments, running results and personal preliminary judgment to AI if they encounter code errors, abnormal model training or difficult result interpretation. AI generates step-by-step prompts based on the input content to help students judge the type of problem, prompt possible causes and check the path, and guide them to modify the code or improve the analysis process. After the students complete the correction, they need to rerun the program, verify the output results, and record the changes before and after the modification.

In order to avoid students simply using big models as ' code generator ' or ' answer generator ', teachers set clear rules in teaching : AI should not be directly required to generate complete code, and AI output content should not be copied without understanding ; when submitting homework or classroom reports, it is necessary to explain the source of the problem, the content of the AI prompt, the individual modification process and the final verification result. Teachers supplement the explanation according to the problems that students focus on, summarize the common errors, and strengthen the connection between AI feedback and course knowledge points.

3.2. Typical Teaching Cases

Taking the construction of a BiLSTM-CRF model in named entity recognition tasks as an example, students often encounter the problem of a mismatch between the input dimensions of the CRF layer and the number of labels during implementation. Under the traditional feedback mode, students usually can only repeatedly modify the code based on error messages, which can easily lead to a blind trial-and-error state and makes it difficult to truly understand the data flow relationship between layers of the model.

In this practice, when students submit relevant error reporting information and part of the code, the big model does not directly give a completely modified program, but provides prompts in the order of ' locating errors-checking dimensions-checking labels-re-verifying '. Firstly, AI prompts students to view the shape of BiLSTM output tensor to confirm whether it is consistent with the input of subsequent linear mapping layer. Secondly, guide students to check whether the output dimension of the linear layer is equal to the number of label categories; thirdly,

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202607_9(7).0008

remind students to check whether the settings of entity labels, non-entity labels and special labels in the BIO labeling system are consistent. Finally, students are asked to re-run the model and observe whether the loss function and prediction results are back to normal.

Through this process, students need to take the initiative to complete code inspection and modification, rather than passively accept ready-made answers. The case shows that the value of the big model in programming teaching is mainly reflected in the decomposition of debugging ideas and the prompt of key checkpoints, which can help students transform fuzzy error reporting into specific analysis steps, and then understand the relationship between model structure, tensor dimension and label constraints.

3.3. Observation of Teaching Effectiveness

In order to avoid the teaching practice staying at the level of experience description, this paper observes the implementation of the explanatory feedback process from three dimensions: classroom participation, homework correction and learning reflection. The classroom participation dimension mainly focuses on whether students can continue to ask, supplement error information and try different modification schemes based on AI tips; the revision dimension mainly focuses on whether students can explain the source of errors, the adjustment process and the verification results in the revision record. The learning reflection dimension mainly focuses on whether students can distinguish between ' tips given by AI ' and ' judgments completed by individuals ', and explain the model structure, code logic or semantic analysis process.

From the implementation point of view, the feedback process has improved the passive waiting state of students in the face of complex programming tasks to a certain extent. When students encounter model errors, abnormal code operation or difficulty in interpreting results, they can quickly obtain preliminary diagnostic clues, and carry out code inspection, semantic analysis and result verification around AI tips. Especially in the BiLSTM-CRF dimension check, BIO label setting and sentiment analysis semantic interpretation, step-by-step prompts can help students transform the original fuzzy problem into a clearer check path and reduce purposeless trial and error.

At the same time, the practice also shows that the effectiveness of big model feedback does not depend on whether the tool itself can give an immediate answer, but on whether students can complete re-analysis, re-modification and re-verification based on prompts. Teachers still play a key role in it, and they need to guide students to transform AI output into a personal understanding process through task requirements, classroom norms and evaluation criteria, rather than simply copying and generating content. Therefore, the feedback process is more suitable as a procedural scaffold in complex programming tasks, rather than an automated tool to replace teacher evaluation or student thinking.

3.4. Teaching Reflection and Directions for Improvement

From the perspective of teaching practice, the large model is suitable for the process feedback function in complex programming tasks, especially for error location, reason explanation, step prompt and result analysis. Compared with the traditional correction method, the advantage of is that it can provide instant support when students encounter problems and reduce the learning interruption caused by waiting for feedback; the step-by-step prompt also helps students to disassemble complex problems into several operational small problems, reducing the invalid burden caused by blind trial and error.

However, the application of large models in programming teaching must set clear boundaries. In the absence of teacher guidance, students may simply understand it as a tool for automatically generating code or answers, thus weakening their ability to think, understand and debug code. Therefore, teachers need to design task requirements and usage specifications

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202607_9(7).0008

in advance, and guide students to record and reflect on ' what problems have been encountered, what have been tried, what AI prompts, and how to finally modify '. At the same time, AI output content may still be inaccurate, incomplete or inconsistent with curriculum requirements, and teachers cannot completely withdraw from the feedback process [5]. The more reasonable way is that AI is responsible for the high-frequency, real-time and explanatory process support, and teachers are responsible for the control of teaching objectives, the induction of common problems and the explanation of key knowledge points, so as to form a collaborative mechanism of ' AI auxiliary feedback-students ' self-correction-teachers ' teaching adjustment '.

4. Conclusion

This paper focuses on the process feedback requirements of complex programming tasks in programming courses. Taking the undergraduate natural language processing programming tasks as the practice carrier, this paper designs a large model-enabled explanatory feedback teaching process, and applies it to typical tasks such as named entity recognition and sentiment analysis. This process limits AI feedback to error location, reason explanation and step prompt, and emphasizes that students complete self-correction, result verification and learning reflection under the support of prompt, which is helpful to promote the transformation of programming teaching from result correction to process support. The follow-up research can further combine the data of learning log, homework performance, classroom observation and student feedback to verify the teaching effect, applicable boundary and promotion conditions of this method more systematically.

Acknowledgements

University-level Education and Teaching Research Project of the University of Emergency Management, “Research on the Design of a Closed-Loop Personalized Learning Feedback System Supported by Generative Artificial Intelligence” (JY2025B09);

Category A Intramural Science and Technology Fund Project of the University of Emergency Management, “Research on Online Intelligent Anti-Jamming Technology for Radar in Complex Electromagnetic Environments in Mines” (3142024009).

References

[1] Ling, J., Zhang, L. L., Chen, H. G., et al. (2025). Application of generative artificial intelligence in computer experiment teaching. *Computer Education*, (5), 28–32. https://doi.org/10.16512/j.cnki.jsjjy.2025.05.002. (In Chinese)

[2] Yu, H. Y. (2025). Teaching reform and practice of Python programming data analysis empowered by generative AI. *Journal of Computer Technology and Education*, 13(7), 78–86. https://doi.org/10.12427/jcte2325-0208.20251212. (In Chinese)

[3] Mittal, U., Sai, S., Chamola, V., & Sangwan, D. (2024). A comprehensive review of generative AI for education. *IEEE Access*, 12, 142733–142759. https://doi.org/10.1109/ACCESS.2024.3468368.

[4] Yao, D. H., Zhong, X. F., & Du, M. (2025). Research on intelligent adaptive teaching model of programming course based on deep learning. *Journal of Computer Technology and Education*, 13(5), 8–13. https://doi.org/10.12427/jcte2325-0208.20251002. (In Chinese)

[5] Estévez-Ayres, I., Callejo, P., Hombrados-Herrera, M. Á., et al. (2025). Evaluation of LLM tools for feedback generation in a course on concurrent programming. *International Journal of Artificial Intelligence in Education*, 35(2), 774–790.

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202607_9(7).0008

[6] Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2), 257–285.

[7] Si, G. D., Peng, L. M., & Zhang, Y. Q. (2023). Design of data structure experiment question bank from the perspective of cognitive load. *Computer Education*, (4), 173–176. https://doi.org/10.16512/j.cnki.jsjjy.2023.04.044. (In Chinese)

[8] Wang, W. J., Shi, N. F., & Lü, Z. G. (2024). Research on the construction of teaching practice model of "Algorithm Design and Analysis" course based on feedback mode. *Computer Application Digest*, 40(24), 16–18, 21. (In Chinese)

[9] Li, H., Lin, S., & Ouyang, Y. (2021). Exploration and practice of natural language processing course teaching based on deep learning. *Computer Education*, (11), 147–151. https://doi.org/10.16512/j.cnki.jsjjy.2021.11.034. (In Chinese)

[10] Ding, X. J. (2024). Teaching reform and exploration of data structure course based on artificial intelligence. In *2024 International Conference on Artificial Intelligence and Digital Libraries (AIDL)* (pp. 18–21). Nanjing, China. https://doi.org/10.1109/AIDL66202.2024.00011.

[11] Liu, J., Gao, Y., Yang, L., & Wang, Z. W. (2025). Exploration of computer network practical teaching model empowered by artificial intelligence. *Journal of Computer Technology and Education*, 13(5), 105–110. https://doi.org/10.12427/jcte2325-0208.20251018. (In Chinese)