Cognitive Diagnosis for Programming Domains

Yongfeng Huang[✉] and Kaiyuan Wang

School of Computer Science and Technology, Donghua University, Shanghai, China
yfhuang@dhu.edu.cn

Abstract. Cognitive diagnosis plays an important role in intelligent educational scenarios as a method that can reveal students' knowledge mastery. Existing cognitive diagnostic methods are mainly applicable to objective questions in core subject areas (e.g., mathematics); however, in the field of programming, where the questions are subjective, no research has been conducted to apply cognitive diagnostics to analyze and assess the impact of students' subjective answers on students' knowledge acquisition. Therefore, we explored how to design a cognitive diagnostic approach that can be applied in the programming domain. In this paper, we design a neural network-based cognitive diagnostic model, PECDM, where our approach not only exploits the interactions between the student factor and the exercise factor, but also includes the student answer source code and the difficulty of the exercise among the diagnostic factors, further considering that the student's knowledge mastery can be captured from the student's subjective answers (source code), and finally uses multiple fully connected layers to interactions are modeled, resulting in more accurate and interpretable diagnostics. We conduct relevant experiments on the codeforces dataset, and the experimental results show that the accuracy of our designed PECDM model is about 95%, which is better in terms of accuracy, rationality, and interpretability when compared with other cognitive diagnostic models.

Keywords: cognitive diagnosis · programming · knowledge mastery

1 Introduction

As a result of the rapid expansion of artificial intelligence, several areas and fields have undergone new developments and modifications. Traditional assessments in the field of education are predominantly manual, time-consuming, and subjective [1]. Cognitive diagnosis has emerged in this case.

As cognitive diagnostic technologies continue to evolve and innovate, more and more intelligent tools and platforms are being developed. These tools and platforms not only help professionals such as teachers to quickly and accurately assess and diagnose, but also provide personalized and customized learning and treatment programs for students. For example, in the field of smart education, various smart learning platforms and applications have been widely used, such as Coursera and Khan Academy. These platforms and applications not only provide a large number of online courses and educational resources, but also can automatically generate course contents and exercise questions suitable for students' learning according to their learning situations.

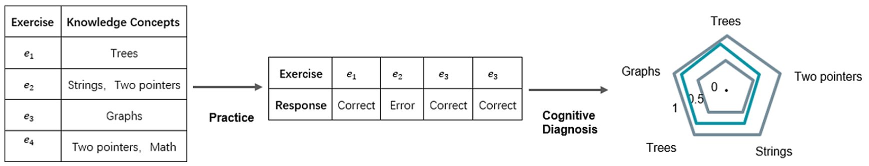


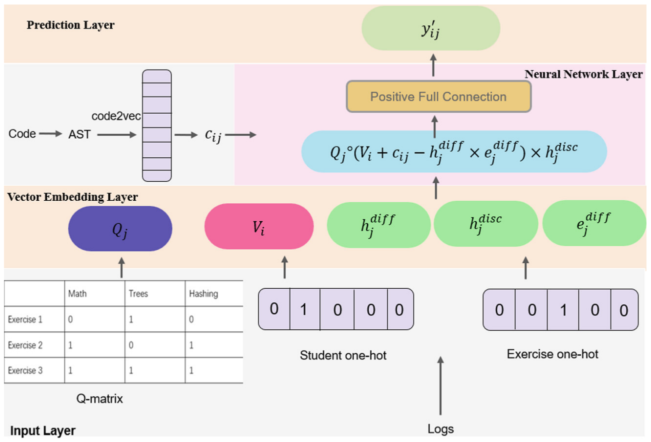
Fig. 1. Shows an illustration of a cognitive diagnostic. By selecting which tasks to complete, students receive grades. The degree to which each knowledge topic was mastered by the students was assessed using the cognitive diagnostic technique.

The goal of cognitive diagnosis in the context of intelligent education is to determine students' level of knowledge mastery based on their answer history [8]. Consider Fig. 1 as an illustration, after a series of exercises and based on the correct or incorrect status of the exercises, the student's knowledge mastery level is obtained through the process of cognitive diagnosis. In practice, these diagnostic evaluations are crucial and serve as the foundation for providing students with exercise recommendations and individualized instruction [9].

Many experts have conducted extensive research in the field of cognitive diagnosis, such as item response theory (IRT) [2] and its multidimensional extension (MIRT) [3], the noisy gate model (DINA) [4] and its variants, which are classic cognitive diagnosis models in educational psychology. Despite the fact that these models have produced some outcomes, their interaction functions are manually created, and most of them are solved through parameter estimation, where they are linear combinations of student features and exercise features [10]. These might not be adequate to capture the nuanced dynamics of how students interact with activities in intelligent education. As a result, numerous researchers are working hard to increase the precision of cognitive diagnostics. To address this issue, many researchers have designed neural network-based cognitive diagnosis frameworks, such as the Neural Cognitive Diagnosis Model (NeuralCDM) [5] to incorporate more contextual data and preexisting linkages between knowledge areas, or to better respond to the complex interaction between students and activities [6].

In contrast, we have found that traditional cognitive diagnosis methods are mainly used for core subjects (such as mathematics) and mainly analyze objective questions. However, in programming exercises that are mostly subjective, we have found that using traditional cognitive diagnosis methods will have significant limitations, such as poor diagnostic effectiveness and insufficient use of information. Programming exercise is a special type of subjective question, and the student's answer (source code) contains rich information. When students write source code, it can be considered as the process of using multiple knowledge points to solve problems, including programming languages, algorithms, and data structures. Therefore, if traditional cognitive diagnosis is directly applied to programming exercises, it is equivalent to simply diagnosing it as an objective question, ignoring the information contained in the subjective answers of programming exercises.

Therefore, to address the above issues, we propose a cognitive diagnostic model for the field of program design (PECDM). The PECDM model incorporates student responses to program design exercises and the difficulty level of the exercises into traditional cognitive diagnostic factors, allowing for the capture of rich information from



	Math	Trees	Hashing
Exercise 1	0	1	0
Exercise 2	1	0	1
Exercise 3	1	1	1

Fig. 2. Description of PECDM Model Diagnosis Process

student subjective answers in source code to improve diagnostic accuracy and adapt cognitive diagnosis to the field of program design. Our main contributions are as follows:

- In order to accurately and effectively apply cognitive diagnosis to the programming domain, we propose a cognitive diagnosis model applicable to the programming domain.
- We crawled the information of more than 100 students doing questions, 7152 programming exercises, and 37 types of knowledge from the codeforces website, pre-processed the data, and organized them into a dataset.
- We propose the PECDM model with the overall structure shown in Fig. 2. Our model's precision, adaptability, and interpretability are validated by comparing it to other models using a custom-created dataset.

The remaining sections of the paper are organized as follows. Section 2 explains the development and historical context of cognitive diagnostics, followed by a relevant task specification in Sect. 3, the composition of our model in Sect. 4, experimental results in Sect. 5, and conclusions in Sect. 6.

2 Related Work

Programming. Programming is the process of programming on a computer to achieve some predetermined function. With the rapid development of the Internet, programming is not only becoming more and more popular, but also becoming one of the essential skills for people in the digital economy. Programming skills have brought a lot of convenience to people's daily work and life, providing more job opportunities not only for computer professionals but also for non-computer professionals. Nowadays, there is a growing demand for learning programming and more and more people are learning programming. To meet the needs of students, more and more programming learning websites are widely used, such as Codeforces, LeetCode and other websites that provide a large number of programming exercises to help students improve their programming skills. However, the difficulty of learning programming is that students not only need to master the correct syntax structure, but also need to apply relevant algorithms and knowledge points to solve problems according to practical needs. Therefore, programming learning requires a lot of effort and time, but with continuous learning, one can gradually master this important skill and play an important role in various fields.

Cognitive Diagnosis. Currently, in the context of intelligent education, researchers have proposed many cognitive diagnosis models. IRT [2] and DINA [4] are the most fundamental and traditional approaches for cognitive diagnostics. IRT mainly uses a one-dimensional continuous latent feature to describe students and exercises, such as students' latent potential, exercise difficulty, and exercise discrimination, and then predicts the probability that students will correctly answer exercises through a normal distribution logistic function [6]. In order to reflect students' abilities in multiple dimensions, Multidimensional item response theory was proposed by Reckase et al. (MIRT) [3]. DINA not only considers students' ability levels but also considers their mastery of different attributes. In the DINA model, each item has several attributes, and students may or may not have mastery of each attribute. The model introduces an attribute matrix that describes which attributes each item is related to. If a student masters all the attributes related to an item, he can answer the item correctly. In this way, the DINA model can identify students' mastery of different attributes and use it for diagnostic classification. In recent years, a number of researchers have created cognitive diagnosis models based on neural networks, continually seeking to enhance their precision and interpretability. NeuralCDM, for instance, employs a MIRT-inspired shadow layer and two fully connected layers to replicate the interaction between students and activities. The connection-driven cognitive diagnostic RCD [11] models the prior structural relationship between students' exercise interaction and knowledge points using a relation graph with multiple layers. CDGK [12] transforms knowledge concepts into a graph and only considers the leaf nodes of the knowledge concept tree to collect knowledge concepts and decrease the dimensions of the model.

3 Problem Statement

3.1 Definition of Cognitive Diagnosis

The learning system can consist of a series of students, exercise records and knowledge points, each exercise is marked with the relevant knowledge point, and students can pick which ones they want to complete. Assume that the learning system has N students, M exercises, and K knowledge points., which can be denoted as S = $\{s_1, s_2, s_3 \ldots\ldots, s_N\}$, E = $\{e_1, e_2, e_3 \ldots\ldots, e_M\}$ and K = $\{k_1, k_2, k_3 \ldots\ldots, k_k\}$. Where each student has an exercise record, i.e., a response log R, represented as a tuple (s, e, r, c) with s belonging to S, e to E, and r to {0, 1} representing the score of students on exercise e. r = 1 means that student s answered exercise e correctly and r = 0 means that he did not answer correctly. c represents the source code of the student's response on exercise e. And there is also the Q matrix, Q = $\{Q_{ij}\}_{M\times K}$ represents the Q-matrix, it shows how the practice and the knowledge point are related. $Q_{ij} = 1$ represents the exercise e_i contains the knowledge point k_j, otherwise $Q_{ij} = 0$. The problem is defined as given the student's response log R and Q matrix, our goal is to uncover the student's knowledge mastery through the student performance prediction process.

3.2 Components Need for the System

Several considerations that should be taken into account when using the cognitive diagnosis approach for programming are introduced in this section.

Exercise Records. The exercise records of students in the learning system contain important information. By analyzing these records, we can determine the students' level of mastery of the knowledge and predict their accuracy in answering questions.

Knowledge Composition. Referring to the Q-matrix, which explains how exercises and knowledge points relate to one another.

Knowledge Difficulty and Exercise Discriminability. Knowledge difficulty h^{diff} and exercise discriminability h^{disc} are extended from the IRT model and DINA model. h^{diff} Represents the difficulty of the knowledge point, while h^{disc} indicates the exercise's capacity to identify between students with various levels of expertise.

Exercise Difficulty. The difficulty value corresponding to the programming exercises.

Exercise Code. The code submitted by students contains semantic and syntactic information, which reflects their mastery of programming languages and related algorithmic knowledge points. Therefore, we can extract the students' level of knowledge from the code.

4 The Proposed PECDM Framework

Figure 2 shows the overall framework of PECDM. Specifically, the input layer prepares the information needed for cognitive diagnosis, such as student exercise logs and the Q matrix. In the embedding layer, according to the data provided by the input layer, relevant embedding vectors are created, mapping students, exercises, and knowledge points to a hidden space, obtaining some embedding representations. The layer of the neural network, the cognitive diagnosis factors from the embedding layer are first combined and then undergo non-linear transformations through the fully connected layer. The final output layer is used to predict students' exercise performance.

4.1 The Input Layer

In this layer, we use the information of student practice logs (information of student exercises and scores generated during practice) and the Q matrix of exercises and knowledge points. During training, the student and exercise sets are read, and each element is converted into a one-hot vector, $x^s \in \{0, 1\}^{1\times N}$, $x^e \in \{0, 1\}^{1\times M}$.

4.2 The Vector Embedding Layer

This layer's primary function is to create matching embedding vectors for use in the following neural network layer depending on the data provided by the input layer. To acquire the corresponding initialized embedding vectors, we multiply the one-hot vectors of the students, exercises, and knowledge points with trainable matrices after first encoding them into a hidden space, as shown in Eq. 1.

$$V_i = \sigma(x_i^S \times A)$$

$$h_j^{diff} = \sigma(x_j^e \times B)$$

$$h_j^{disc} = \sigma(x_j^e \times C)$$

$$e_j^{diff} = \sigma(x_j^e \times D)$$

$$Q_j = x_j^e \times Q \tag{1}$$

Which A $\in R^{N\times K}$, $B \in R^{M\times K}$, $C \in R^{M\times 1}$, and $D \in R^{M\times 1}$ are trainable matrices and Q is a pre-defined Q matrix. The vector $V_i \in (0, 1)^{1\times K}$ represents the mastery level of knowledge, $h_j^{diff} \in (0, 1)^{1\times K}$ represents the difficulty of each knowledge point in exercise e_j, $h_j^{disc} \in (0, 1)$ represents the ability of exercise e_j to differentiate between students of different skill levels, and $e_j^{diff} \in (0, 1)$ represents the difficulty value of exercise e_j, $Q_j \in \{0, 1\}^{1\times k}$ represents the relationship between exercise e_j and knowledge points. The σ function here represents the activation function, and the sigmoid function is used in this case.

The vector c_{ij} represents the proficiency of the knowledge points demonstrated by student s_i in solving problem j. The code2vec technique is mainly used to represent the student's practice code, which contains rich information that can be effectively utilized in the subsequent neural network layer to enhance the diagnostic effect. First, the student's practice code is converted into an AST syntax tree, and different paths are extracted. The embedding representations are obtained for each token and path of the code snippet, which are the starting token x_s, path p_j, and ending token x_e. Then, a comprehensive vector is obtained by combining the starting and ending tokens with the path, followed by a fully connected layer to merge the various parts of the previous comprehensive vector. Since different contexts have different semantics and structures, their importance should also be different. Therefore, the corresponding attention α_i is calculated for each context, and the final overall vector v is obtained by weighted sum. Then, the related knowledge points are constructed as a label matrix, and the proficiency vector of the knowledge points is obtained by passing through a fully connected layer and a softmax function, representing the student's mastery level in the corresponding knowledge points.

$$c_i = embedding(< x_s, p_j, x_e >) \tag{2}$$

$$c_i^{'} = \tanh(w \cdot c_i) \tag{3}$$

$$\alpha_i = \frac{\exp(c_i^{'} \cdot \alpha)}{\sum_{j=1}^{n} \exp(c_i^{'} \cdot \alpha)} \tag{4}$$

$$v = \sum_{i=1}^{n} \alpha_i \cdot c_i^{'} \tag{5}$$

4.3 The Neural Network Layer

Inspired by the NCDM model, we use some interpretable diagnostic factors obtained from the vector embedding layer connected to predict students' practice performance through the student practice interaction function. The details are as follows:

$$x_{ij} = Q_j^e \circ \left(V_i + c_{ij} - h_j^{diff} \times e_j^{diff} \right) \times h_j^{disc} \tag{6}$$

The fully connected layer is then designed as follows:

$$f_{ij} = \varphi\left(\omega \times x_{ij} + b\right) \tag{7}$$

Here, f_{ij} represents the probability that student s_i answers question j correctly, and φ is a non-linear interactive function. ω and b are trainable parameters. The neural network layer uses two fully connected layers and one output layer. A technique is utilized to constrain that each in the layer is positive in order to satisfy the monotonicity assumption.

4.4 The Prediction Layer

A student's final knowledge proficiency vector V_i, can be found by estimating how well they would execute an exercise. The loss function is defined as the cross-entropy between the true score y_{ij} and the predicted score$y_{ij}^{'}$, defined as follows:

$$loss = -\sum_i \sum_j (y_{ij} \log y_{ij}^{'} + (1 - y_{ij} \log(1 - y_{ij}^{'}))) \tag{8}$$

5 Experiments

To validate the efficacy and interpretability of the proposed model, comparative tests were done using a number of baseline models. This section begins with a brief introduction to the dataset, followed by an analysis and discussion of the experimental outcomes.

5.1 Overview of Datasets

From Codeforces, we crawled the practice records of over 100 students and 7152 programming exercises encompassing 37 knowledge points for our research. To guarantee that each student has sufficient data for diagnosis, we chose students with at least 15 workout recordings. In addition, we divided the data into three distinct sets: 70% for training, 10% for validation, and 20% for testing. The training set was used to train the model, the validation set was used to assess the quality of the model and update parameters, and the test set was used to evaluate the model.

5.2 Baselines

We compared the PECDM model on the same data set with the following baseline models:

- Item Response Theory (IRT)
- Multidimensional Item Response Theory (MIRT)
- Deterministic Input Noise and Gate Model (DINA)
- Neural Cognitive Diagnosis Model (NCDM)

5.3 Experimental Results

Analysis of Prediction Accuracy. It is challenging to do a direct assessment because the findings of cognitive diagnostic take the shape of a vector, where each element denotes the student's mastery of a particular knowledge point. We test our model by predicting how well students will perform on issues they haven't seen before because the outcomes of cognitive diagnostic are typically positively connected with the likelihood that students would properly answer exercises. In predictive accuracy analysis, a variety of metrics can be used to assess the performance of cognitive diagnostic models. Commonly used metrics include accuracy, recall, F1 score, AUC, and RMSE. We selected two metrics, accuracy and AUC, for performance evaluation.

Table 1 summarizes the predictive performance of the evaluation metrics, including accuracy and AUC. The best outcome is shown in the table's boldface. it can be seen from the table that our model outperforms the NCDM model by about 17% and 12% in terms of improvement in prediction, which significantly indicates the effectiveness of our model.

Ablation Experiments. As our cognitive diagnosis model incorporates factors such as exercise difficulty and student practice code, to understand their impact on model performance, we conducted related experiments, and Table 2 shows the results. The table clearly shows that each optimization reduces model accuracy when it is removed, showing that each factor is appropriate.

Interpretability Analysis. The likelihood that a student would successfully complete an activity depends, intuitively, on how well they understand a certain knowledge idea. For a certain knowledge point k, for instance, if student a has a higher level of mastery than student b, it means that student an is more likely to respond correctly to exercise

Table 1. Shows the outcomes of experiments on real datasets to predict student grades.

Method	Accuracy	AUC
DINA	0.634	0.714
IRT	0.739	0.821
MIRT	0.837	0.922
NCDM	0.776	0.859
PECDM	**0.949**	**0.983**

Table 2. Results of an ablative experiment. The components we eliminated for our trials are shown by the horizontal lines in the first column. P is the exercise code, while E is the exercise difficulty.

Method	Accuracy	AUC
~~P~~ECDM	0.781	0.875
P~~E~~CDM	0.942	0.970
PECDM	0.949	0.983

questions containing knowledge point k. As a measurement of model interpretability, consistency (DOA) is used. The following equation represents DOA(k) for knowledge point k:

$$DOA(k) = \frac{1}{Z}\sum_{a=1}^{N}\sum_{b=1}^{N}\delta(F_{ak}^{s}, F_{bk}^{s})\sum_{j=1}^{M}\frac{I_{jk}\bigwedge J(j,a,b)\bigwedge \delta(r_{aj}, r_{bj})}{I_{jk}\bigwedge J(j,a,b)} \tag{9}$$

where $Z = \sum_{a=1}^{N}\sum_{b=1}^{N}\delta(F_{ak}^{s}, F_{bk}^{s})$, F_{ak}^{s} indicates student's level of expertise for knowledge topic k. If x $>$ y, then $\delta(x, y) = 1$, otherwise $\delta(x, y) = 0$. If exercise j contains knowledge concept k, then $I_{jk} = 1$, otherwise $I_{jk} = 0$. If both student a and student b have practiced exercise j, then $J(j, a, b) = 1$, otherwise $J(j, a, b) = 0$. r_{aj} shows student's performance on exercise j.

We compared the PECDM model with DINA and NCDM because there isn't a clear correlation between the exercises and knowledge points in conventional cognitive diagnostic models like IRT, MIRT, and PMF. The PECDM model is superior to the other baselines in terms of interpretability, according to the findings displayed in Fig. 3. Additionally, we notice that the DOA value of NCDM is lower than that of PECDM, demonstrating the greater explanatory power of our model.

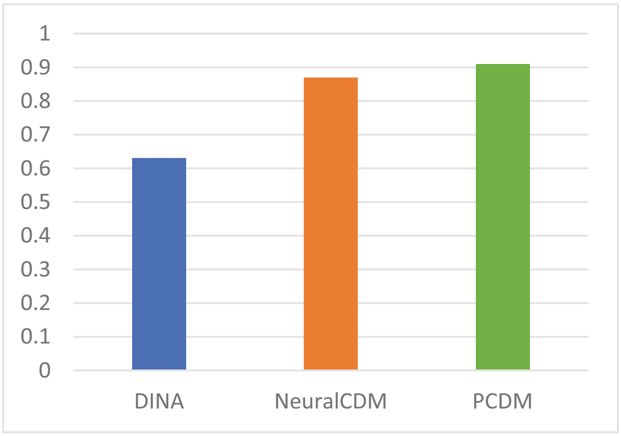
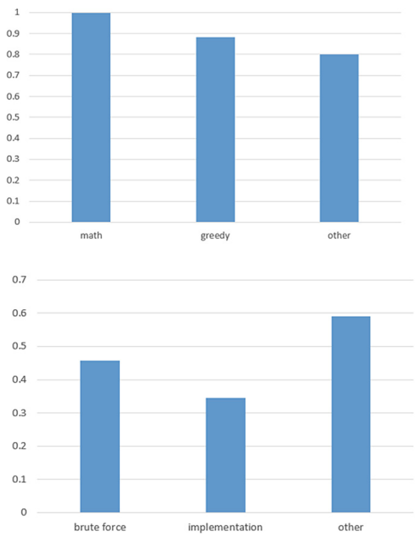
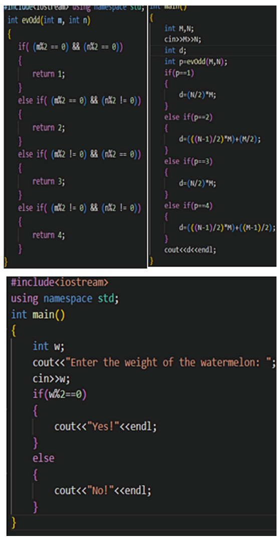


Fig. 3. Shows the interpretability of DOA results.





```
#include<iostream> using namespace std;
int evOdd(int m, int n)
{
    if( (m%2 == 0) && (n%2 == 0))
    {
        return 1;
    }
    else if( (m%2 == 0) && (n%2 != 0))
    {
        return 2;
    }
    else if( (m%2 != 0) && (n%2 == 0))
    {
        return 3;
    }
    else if( (m%2 != 0) && (n%2 != 0))
    {
        return 4;
    }
}
```

```
int main()
{
    int M,N;
    cin>>M>>N;
    int d;
    int p=evOdd(M,N);
    if(p==1)
    {
        d=(N/2)*M;
    }
    else if(p==2)
    {
        d=(((N-1)/2)*M)+(M/2);
    }
    else if(p==3)
    {
        d=(N/2)*M;
    }
    else if(p==4)
    {
        d=(((N-1)/2)*M)+((M-1)/2);
    }
    cout<<d<<endl;
}
```

```
#include<iostream>
using namespace std;
int main()
{
    int w;
    cout<<"Enter the weight of the watermelon: ";
    cin>>w;
    if(w%2==0)
    {
        cout<<"Yes!"<<endl;
    }
    else
    {
        cout<<"No!"<<endl;
    }
}
```

Fig. 4. Representation of students' application of knowledge points in code representation.

Case Study. In order to demonstrate that the level of student mastery of relevant knowledge points can be extracted from subjective answers to exercise codes, we selected two examples. As shown in Fig. 4, the bar chart at the top represents the student's code proficiency, and it can be seen from the code that the student used the corresponding knowledge points and answered the question correctly. The two bars corresponding to the math and greedy knowledge points in the histogram are significantly higher, indicating a certain degree of interpretability. In another case, considering the brute force and execution knowledge points, the student answered the question incorrectly, and there was no obvious use of the relevant knowledge points or incorrect usage in the code. The corresponding knowledge points were found to be relatively low compared to others in the histogram. Both of these cases illustrate to some extent that the level of student application of relevant knowledge points can be captured from their code.

6 Conclusion

This research presents a novel programming-specific cognitive diagnosis paradigm based on neural networks. In particular, the model incorporates the source code of student answers to exercises and the difficulty of the tasks, so exploiting the subjective information provided by students and enhancing the diagnostic efficacy. Experiment results on real-world datasets demonstrate that our model is more precise and understandable. In addition, a sensitivity analysis was performed to validate the added parameters.

References

1. Brown, G. T. (2017). Assessment of student achievement. Routledge.

2. Lord, F. (1952). A theory of test scores. Psychometric monographs.
3. Reckase, M. D., & Reckase, M. D. (2009). Multidimensional item response theory models (pp. 79–112). Springer New York.
4. Haertel, E. H. (1989). Using restricted latent class models to map the skill structure of achievement items. Journal of Educational Measurement, 26(4), 301–321.
5. Wang, F., Liu, Q., Chen, E., Huang, Z., Chen, Y., Yin, Y., ... & Wang, S. (2020, April). Neural cognitive diagnosis for intelligent education systems. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 04, pp. 6153–6161).
6. Ma, H., Li, M., Wu, L., Zhang, H., Cao, Y., Zhang, X., & Zhao, X. (2022, October). Knowledge-Sensed Cognitive Diagnosis for Intelligent Education Platforms. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management (pp. 1451–1460).
7. Baker, F. B. (2001). The basics of item response theory. For full text: http://ericae.net/irt/baker.
8. Liu, Q., Wu, R., Chen, E., Xu, G., Su, Y., Chen, Z., & Hu, G. (2018). Fuzzy cognitive diagnosis for modelling examinee performance. ACM Transactions on Intelligent Systems and Technology (TIST), 9(4), 1–26.
9. Kuh, G. D., Kinzie, J., Buckley, J. A., Bridges, B. K., & Hayek, J. C. (2011). Piecing together the student success puzzle: Research, propositions, and recommendations: ASHE higher education report (Vol. 116). John Wiley & Sons.
10. Li, S., Guan, Q., Fang, L., Xiao, F., He, Z., He, Y., & Luo, W. (2022, October). Cognitive Diagnosis Focusing on Knowledge Concepts. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management (pp. 3272–3281).
11. Gao, W., Liu, Q., Huang, Z., Yin, Y., Bi, H., Wang, M. C., ... & Su, Y. (2021, July). Rcd: Relation map driven cognitive diagnosis for intelligent education systems. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 501–510).
12. Wang, X., Huang, C., Cai, J., & Chen, L. (2021, October). Using knowledge concept aggregation towards accurate cognitive diagnosis. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management (pp. 2010–2019).