

# Generating Plausible Distractors for Multiple-Choice Questions via Student Choice Prediction

Yooseop Lee<sup>1,2\*</sup> Suin Kim<sup>2</sup> Yohan Jo<sup>1†</sup>

<sup>1</sup>Graduate School of Data Science, Seoul National University <sup>2</sup>Elice  
{lyooseop, yohan.jo}@snu.ac.kr suin@elicer.com

## Abstract

In designing multiple-choice questions (MCQs) in education, creating *plausible distractors* is crucial for identifying students' misconceptions and gaps in knowledge and accurately assessing their understanding. However, prior studies on distractor generation have not paid sufficient attention to enhancing the difficulty of distractors, resulting in reduced effectiveness of MCQs. This study presents a pipeline for training a model to generate distractors that are more likely to be selected by students. First, we train a *pairwise ranker* to reason about students' misconceptions and assess the relative plausibility of two distractors. Using this model, we create a dataset of pairwise distractor ranks and then train a *distractor generator* via Direct Preference Optimization (DPO) to generate more plausible distractors. Experiments on computer science subjects (Python, DB, MLDL) demonstrate that our pairwise ranker effectively identifies students' potential misunderstandings and achieves ranking accuracy comparable to human experts. Furthermore, our distractor generator outperforms several baselines in generating plausible distractors and produces questions with a higher item discrimination index (DI).<sup>1</sup>

## 1 Introduction

Multiple-Choice Questions (MCQs) hold significant educational value as they provide a useful tool for assessing students' knowledge. Among the most critical elements in MCQs are *distractors*—the incorrect answer options. While the growing demand for education has amplified the need for numerous MCQs, manually creating distractors is time-consuming and costly, even for experts (Luo

\*This work was conducted while the first author was a graduate student at Seoul National University.



†Corresponding author.

<sup>1</sup>Our code and a subset of our dataset are available at <https://github.com/holi-lab/distractor-generator>

### Code Type Question

```
[Question] Look at the following code and choose the correct code to replace (blank).  
my_list = (blank)  
print(my_list)  
Result: [5, 15, 25, 35]  
[Answer][num * 10 - 5 for num in range(1, 5)]
```

Distractor Generation Plausibility Rank

		Plausibility Rank
GPT	<ul style="list-style-type: none"><li>• my_list = [10, 30, 50, 70]</li><li>• my_list = [5, 10, 15, 20]</li></ul>	
Ours	<ul style="list-style-type: none"><li>• [num * 10 - 5 for num in range(1, 6)]</li><li>• [num * 10 for num in range(1, 5)]</li></ul>	

### Statement Type Question

```
[Question] Which of the following is not correct?  
[Answer] sort() can only be applied to strings.
```

Distractor Generation Plausibility Rank



		Plausibility Rank
GPT	<ul style="list-style-type: none"><li>• sort() can be used to sort lists.</li><li>• sort() can be used to sort arrays.</li></ul>	
Ours	<ul style="list-style-type: none"><li>• When using remove(), if the element to be removed is duplicated, only the first occurrence will be removed.</li><li>• The pop() method removes and returns the last item in a list.</li></ul>	

Figure 1: Examples of distractor generation. A question and a correct answer are provided as input, and the output is a set of generated distractors. The plausibility rank metric indicates how likely students are to select the distractors.

et al., 2024). Consequently, the automation of distractor generation has emerged as a promising solution (Doughty et al., 2024).

However, prior research has focused primarily on generating distractors similar to human-authored ones (Fernandez et al., 2024; Wang et al., 2023), with insufficient emphasis on enhancing their plausibility. Plausible distractors are crucial as they encourage students to deliberate longer over their answers, and high-quality MCQs must possess an appropriate level of difficulty to differentiate among levels of achievement (Baek, 2019). By contrast, overly simplistic distractors are eas-

ily dismissed, failing to adequately assess student proficiency and reducing the educational value of the assessment. Therefore, creating plausible distractors that target students' common mistakes or misconceptions is essential for developing highly discriminative MCQs (Shin et al., 2019).

Based on these needs, this study presents a model training pipeline for distractor generation. Figure 1 illustrates example distractors generated by GPT and our model. Our main idea is to assign relative ranks to distractors according to their likelihood of being selected by students, and use this information to train a model to generate plausible distractors. To achieve this, the process involves three steps (Figure 2). First, we train a *pairwise ranker* to predict which distractors are more plausible and likely to confuse students (Step 1). Next, we create a synthetic *student choice dataset* that includes pairwise ranking information among distractors (Step 2). Finally, leveraging this dataset, we train a *distractor generator* by applying Direct Preference Optimization (DPO, Rafailov et al., 2024) (Step 3).

According to evaluation on computer science (CS) subjects (Python, DB, MLDL), our pairwise ranker effectively identifies students' common misconceptions, achieving ranking accuracy comparable to human experts. In addition, the distractor generator surpasses several baselines in generating plausible distractors in both automated metrics and human studies. Notably, the distractors generated by our model exhibit a high discrimination index (DI), an essential educational metric that measures a question's ability to distinguish high-performing students from low-performing ones.

The key contributions of our study are threefold.

- We build a pairwise ranker that reasons through students' misconceptions and predicts which distractor they are more likely to choose.
- We construct a student choice dataset with plausibility rankings among distractors and use it to train a plausible distractor generator.
- We apply our method to MCQs in CS subjects (Python, DB, MLDL) and demonstrate the generator's capability of generating distractors with high plausibility and DI.

## 2 Related Works

### 2.1 Distractor Generation

Previous studies on distractor generation can be categorized according to the question format and

domain.

**Passage-Based Format** This format is used for exams that evaluate accurate knowledge based on provided textual content, with datasets such as RACE (Lai et al., 2017), DREAM (Sun et al., 2019), SciQ (Welbl et al., 2017), and Wikipedia commonly used to generate MCQs (Le Berre et al., 2022). As a distractor generation model for this format, Offerijns et al. (2020) fine-tuned GPT-2 and ensured the validity of MCQs through an external QA filtering step. Qiu et al. (2020) proposed the *EDGE* framework, which reformulates passages and questions through attention mechanisms to generate distractors. Qu et al. (2024) introduced a dual-task training approach in which separate training was conducted using passages and questions as input to generate both answers and distractors.

However, since our study focuses on MCQs in the CS domain without relying on passages, these prior works are not directly comparable to ours.

**Cloze-Style Format** This format is commonly used in literacy tests and science quizzes, where test-takers fill in blanks with appropriate words (Chiang et al., 2022; Ren and Q. Zhu, 2021). Wang et al. (2023) proposed a *pseudo Kullback-Leibler Divergence* method to regulate distractor generation by considering item discrimination factors. Yu et al. (2024) used a knowledge graph to generate distractors by retrieving relevant triplets and selecting those most aligned with the QA context.

Our framework is not limited to cloze-style questions, which are relatively rare in our dataset, and instead supports a broader range of question types.

**Math** Scarlatos et al. (2024) improved the process of generating distractors for math problems by dividing it into two main steps: *overgenerate-and-rank*. In the overgeneration phase, they used a large language model (LLM) to generate  $n$  distractors, while in the ranking phase, a ranker was employed to select the top- $k$  distractors most likely to be chosen by students. Feng et al. (2024) explored a kNN-based approach to retrieve in-context examples similar to the target question and used them to generate distractors. Fernandez et al. (2024) proposed the *DiVERT*, which generates distractors based on learned error representations in math MCQs. Hang et al. (2024) utilize retrieval-augmented generation (RAG) and chain-of-thought (CoT) for generating relevant and challenging MCQs.

We use the methods by Scarlatos et al. (2024),

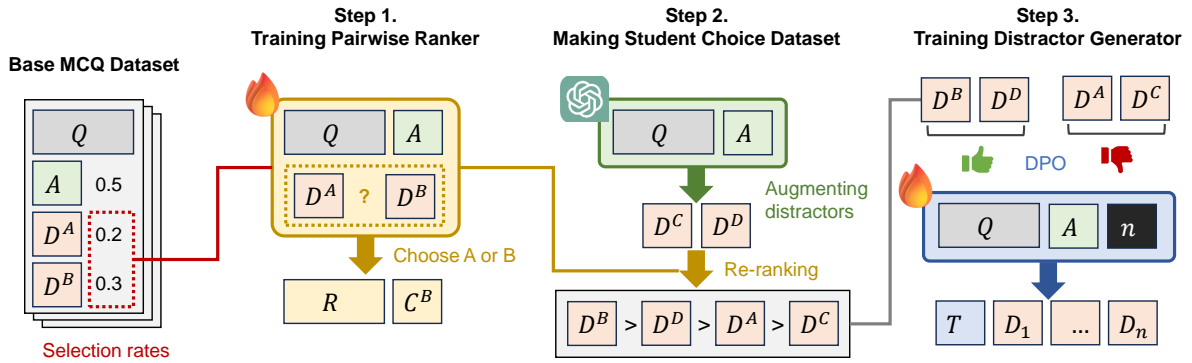


Figure 2: Training pipeline for the distractor generation.

Feng et al. (2024) and Hang et al. (2024) as baselines for comparison with our model. We cannot compare with Fernandez et al. (2024) since their method requires error explanations for each distractor.

**Other Domains** Luo et al. (2024) proposed *Chain-of-Exemplar Reasoning*, a method to sequentially generate distractors for multimodal questions requiring image interpretation, enhancing quality by leveraging contextually similar examples.

Meanwhile, research on distractor generation in the CS domain remains limited. While Doughty et al. (2024) developed a pipeline for generating MCQs aligned with *learning objectives* for programming education using GPT-4, our study emphasizes the plausibility of distractors by leveraging a smaller language model.

## 2.2 Pairwise Ranker

Our study aims to assign plausibility ranks among distractors using an LLM (Figure 2, Step 1 and 2). This approach is motivated by prior findings demonstrating that LLMs exhibit strong inferential abilities, closely aligning with human performance in many evaluation tasks (Sun et al., 2023; Liu et al., 2023). Moreover, distilling these abilities from LLMs into smaller models, such as Prometheus 2 (Kim et al., 2024) fine-tuned from Mistral (Jiang et al., 2023), and ListT5 (Yoon et al., 2024), has achieved comparable performance to LLMs while offering faster inference and reduced positional biases.

However, the reasoning abilities of LLMs to rank plausible distractors in the education domain remain underexplored. A related study by Scarlatos et al. (2024) proposed an approach that trains a pairwise ranker for distractors using data on the actual selection rates of distractors by students. They

further applied DPO to prioritize more plausible distractors. However, their model neither examines nor leverages LLMs’ reasoning abilities, and the trained model lacks interpretability. In contrast, our study extensively evaluates LLMs’ reasoning abilities by comparing various prompting approaches that are broadly applicable across diverse subjects. Additionally, our ranker generates reasoning behind its choices, enhancing its interpretability.

## 3 Methods

In this study, we propose a training pipeline to build a model capable of automatically generating more plausible distractors (as shown in Figure 2). Below, we first describe the base MCQ dataset used for training (§3.1), then introduce the modeling methods for the pairwise ranker (§3.2), student choice dataset (§3.3), and distractor generator (§3.4).

### 3.1 Base MCQ Dataset

To train both the pairwise ranker and the distractor generator, we use an MCQ dataset created by educators on a nationwide online learning platform in South Korea. The MCQs in this dataset have been provided to K12 institutions, large corporations, and government agencies, and contain a variety of CS-related questions and student answers. We retained only those related to Python, DB (SQL), and Machine Learning & Deep Learning (MLDL). We target two categories of MCQs—coding and statement (see Figure 1). The statistics of this dataset are described in Table 1.

A key feature of this dataset is that it includes information on how many students answered each question and the *selection rate* for each distractor. This allows us to determine which distractors were more confusing and plausible to students. Since each question was solved by hundreds of students

Subject	# of questions in train/test set	Avg. correctness rate per question	Avg. # of distractors per question	Avg. # of students per question
Python	264/52	70.7%	3.1	636
DB	54/13	65.6%	2.9	399
MLDL	126/32	61.8%	3.2	1,075

Table 1: Statistics of the base MCQ dataset. The correctness rate refers to the percentage of students who answered the question correctly.

from diverse sectors, the selection rate information is considered reliable. This information will play a key role in training the pairwise ranker and distractor generator, as discussed later. A subset of this dataset without licensing issues is available on our project website (the footnotes on the first page).

### 3.2 Pairwise Ranker

The pairwise ranker ( $M^{Rank}$ ) is designed to take a question ( $Q$ ), its correct answer ( $A$ ), and two distractors ( $D^A, D^B$ ) as input (Figure 2, Step 1), and determine which distractor is more likely to be selected by students.

$$M^{Rank}(Q, A, D^A, D^B) \rightarrow \{R, C^{A \text{ or } B}\} \quad (1)$$

The model outputs two main components:

**(1) Reasoning ( $R$ )** To enhance the interpretability and accuracy of ranking results, we utilize the reasoning abilities of LLMs through a structured prompt. Specifically, we instruct the model to generate reasoning about (1) the knowledge being tested (e.g., “When students approach this problem, they first need to understand ...”) based on the question and the correct answer, and (2) why each of the two given distractors might appear plausible to students (e.g., “Distractor A might confuse students who misunderstand the syntax ...”).

**(2) Choice ( $C^{A \text{ or } B}$ )** The model outputs the result of the reasoning process as a single token (either A or B), indicating which distractor is more likely to be selected by students.

To train a relatively small LM to perform as a ranker, we prepare some training data of reasoning for supervised fine-tuning (SFT). Specifically, for each question in the training set of the base MCQ dataset, we prompt GPT-4o with a distractor pair and the indicator of which one was more frequently selected by students, and instruct it to generate reasoning about the two distractors that concludes

Subject	Avg. # of new distractors in top-3	# of distractor comb. for SFT	# of chosen/rejected sets for DPO
All	1.45	18,899	7,613

Table 2: Statistics of the student choice dataset. Columns 2 and 3 show the number of training samples used for SFT and DPO, respectively.

in favor of the more frequently chosen one. This reasoning ( $R$ ) and the more plausible distractor ( $C^{A \text{ or } B}$ ) form the training data for small LMs.

However, the SFT model exhibited suboptimal accuracy and became more erroneous as the reasoning grew longer. To address this, we use DPO to further train the model’s reasoning process. After inference on the training set using the SFT model, samples diverging from the ground-truth choice were labeled as *rejected*, while the original training samples were set as *chosen*. DPO is then applied to ensure the model generates correct reasoning and choices. Examples of the model’s prompts are provided in Appendix A.1.

### 3.3 Student Choice Dataset

The student choice dataset is created to build training data for the distractor generator (Figure 2, Step 2). For each question in the base MCQ dataset, GPT-4o is used to generate three new distractors distinct from the human-authored ones (Appendix D). These new distractors, along with the original ones, are scored using the pairwise ranker. At this stage, the relative rankings of the original distractors are preserved, while rankings between the original and new distractors, as well as among the new distractors, are determined by our pairwise ranker. Each question ultimately has approximately six distractors ranked in plausible order. This dataset serves for training the distractor generator for both SFT and DPO (§3.4).

Table 2 presents key statistics. Column 1 of Table 2 shows that, on average, 1.45 newly added distractors are ranked among the top 3 for each question, indicating that the newly added distractors are as plausible as the human-authored ones.

### 3.4 Distractor Generator

The distractor generator ( $M^{Gen}$ ) takes as input a question ( $Q$ ), its correct answer ( $A$ ), and a hyperparameter  $n$ , which specifies the number of distractors to generate (Figure 2, Step 3). The model first determines the type ( $T$ ) of distractor (e.g., Cor-



rect/Incorrect knowledge) it will generate, and then outputs  $n$  distractors ( $D_i$ ).

$$M^{Gen}(Q, A, n) \rightarrow \{T, D_1 \dots D_n\} \quad (2)$$

We ensure that the model produces distractors that are both valid and plausible as follows.

**(1) Enhancing Validity** Before generating distractors, the model first determines the type ( $T$ ) of distractor.  $T$  specifies whether the question requires selecting a correct or incorrect statement. This step is critical for questions involving negation (e.g., “Select the *incorrect* statement ...”) as the model has a strong tendency to generate incorrect statements as distractors, even in such cases (see Appendix B.5 for validity evaluation).

**(2) Improving Plausibility** To enhance the plausibility of distractors, we train the model through two stages: SFT and DPO.

**SFT:** We use the student choice dataset to create training data  $\{(Q, A, T, n, D_1, \dots, D_n)\}$  ( $n$  ranges from 1 to the maximum number of distractors available for each question). The trained model learns the basic ability to generate distractors for a given question with varying  $n$ , but without prioritizing more plausible ones.

**DPO:** To enhance the model to generate more plausible distractors, we apply DPO using the student choice dataset. Specifically, for each question, we construct all possible pairs between the top- $n$  distractors and the bottom- $n$  distractors, labeling the distractor from the top- $n$  as *chosen* and the one from the bottom- $n$  as *rejected* in each pair. This allows the model to adjust its generation process to prioritize more plausible distractors that are more likely to challenge students. An example of the model’s prompt is provided in Appendix B.1. We also explored an alternative pairing method for increasing the combinations (Appendix B.2), but its performance was inferior.

## 4 Experiment Settings

In this section, we describe the model training setup (§4.1) and introduce the metrics used to evaluate each model (§4.2 and §4.3).

### 4.1 Model Training

For all experiments, both the pairwise ranker and the distractor generator are fine-tuned by applying

LoRA (Hu et al., 2021) to the Mistral-7B-Instruct-v0.2. The numbers of training and test data are described in Table 1 and Table 2. The detailed settings for SFT and DPO are provided in Appendix A.2 and B.2.

### 4.2 Pairwise Ranker

**Baselines** To assess the performance of the proposed pairwise ranker, we compare it against the following baseline models (the prompts for each baseline are included in Appendix A.1):

- **GPT-3.5-turbo and GPT-4o:** We instruct these GPT models to predict the ranking between two distractors in a zero-shot manner. To examine the impact of different prompt formats, we experiment with four approaches: (1) **Reasoning:** the reasoning-based prompt format described in §3.2, (2) **Rubric:** scoring based on evaluation criteria for assessing plausibility, (3) **G-Eval:** adapting the prompt proposed by Liu et al. (2023) for our specific task, and (4) **Discussion:** simulating a collaborative learning scenario where two teacher agents discuss while observing students’ problem-solving processes.
- **Scarlatos et al. (2024):** We follow the pairwise ranker prompt and training/inference method proposed in this paper, replacing their data with ours.

**Training Data** We use two distinct settings for training data (Table 1):

- **Separate (Sep.):** Models trained separately with data for each subject—Python, DB, and MLDL.
- **Combined (Comb.):** A model trained with data from all subjects combined.

**Distractor Order** One known limitation of LLM-based pairwise ranking is *positional bias*, where the output may vary depending on whether two choices, A and B, are presented in the input prompt as AB or BA (Yoon et al., 2024). To address this, we set the temperature to 0.5 and repeat the reasoning process with both AB and BA input sequences until consistent outputs are achieved, or randomly select a result after 10 attempts.

**Evaluation Metrics** The evaluation metrics for the pairwise ranker are as follows:

- **Rank Accuracy** measures how often the ranker correctly identifies the distractor with the higher student selection rate in the test set.

- **Human Evaluation** aims to compare the model’s performance with human experts. First, two professors in data science perform the pairwise ranking task on 60 test samples (20 per subject), and their results are compared with our model’s rank accuracy. Second, three Master’s students majoring in data science assess the quality of model-generated reasoning and ranking results. For this, 30 samples (10 per subject) of reasoning and choices generated by our pairwise ranker (‘DPO, Comb.’ in Table 3) are randomly selected from the test set. The survey form and the rubric are in Appendix A.6.
- **Consistency in Rank Prediction** tracks the number of iterations required for the model to predict the same choice for both AB and BA inputs. Fewer iterations indicate lower positional bias.

### 4.3 Distractor Generator

The performance of our distractor generator is evaluated using the following metrics:

(1) **Plausibility** We compare the plausibility of distractors generated by our model, GPT models, a kNN approach (Feng et al., 2024), a CoT prompting approach (Hang et al., 2024), and human experts (from the base MCQ dataset) as measured by our pairwise ranker (‘DPO, Comb.’ in Table 3). Win/tie/lose counts are calculated per question/distractor in two settings:

- **Setting A:** For each test question, three distractors are generated by each model ( $n = 3$ ), and only valid ones are retained. These are then compared pairwise between two models, with one point awarded to the winner. Identical distractors are excluded from comparisons.
- **Setting B:** To account for cases where models generate fewer than three valid distractors, each model’s temperature is increased to generate up to five valid distractors per model. After excluding identical distractors between the models, the top-3 are selected for pairwise comparison.

(2) **Human Evaluation** We conduct a human evaluation where actual students assess the difficulty of distractors generated by our method. The test comprises 40 MCQs (Python: 20, DB: 10, MLDL: 10). Each question was sampled from the test set of the base MCQ dataset and paired with four distractors, one from each model (SFT, DPO, GPT-3.5-turbo, and GPT-4o), along with a ‘None of the above’ option. The test is taken by

	Rank Accuracy $\uparrow$			
	Python	DB	MLDL	Avg.
GPT-3.5 (Reasoning)	0.633	0.523	0.606	0.587
GPT-4o (Reasoning)	0.686	0.664	0.570	0.640
GPT-4o (Reasoning, 3-shot)	0.674	<b>0.673</b>	0.600	0.649
GPT-4o (Rubric)	0.686	0.500	0.624	0.603
GPT-4o (G-Eval)	0.632	0.550	0.543	0.575
GPT-4o (Discussion)	0.549	0.482	0.487	0.506
Scarlatos et al. (2024)	0.532	0.386	0.545	0.488
Ours (SFT, Sep.)	0.677	0.491	0.594	0.587
Ours (SFT, Comb.)	0.642	0.650	<b>0.677</b>	0.657
Ours (DPO, Comb.)	<b>0.712</b>	0.659	0.655	<b>0.675</b>
Ours (SFT w/o Reasoning)	0.659	0.523	0.521	0.567

Table 3: Evaluation results on pairwise rankers. The results were averaged over five generations for each model.

15 college students enrolled in AI courses at our university<sup>2</sup>. Based on the selection counts for each distractor, we calculate the plausibility and discrimination index for each model. The discrimination index indicates the ability of each item to differentiate between high- and low-performing students and is calculated as  $DI = (U - L)/N$ , where  $U$  and  $L$  denote the number of students in the upper ( $U$ ) and lower ( $L$ ) groups who answered the item correctly, and  $N$  is the number of students in each group.

We also evaluated the *clarity* (i.e., whether each distractor is clearly written without ambiguity) and *answerability* (i.e., whether a student with relevant knowledge can reasonably answer the question, as defined by Moon et al. (2022)) of MCQs composed solely of the distractors generated by our DPO model with 11 Master’s students in data science. More details about the human evaluation are provided in Appendix B.4.

## 5 Experiment Results

In this section, we present the experimental results for the pairwise ranker (§5.1) and the distractor generator (§5.2).

### 5.1 Pairwise Ranker

(1) **Rank Accuracy** As shown in Table 3, in terms of accuracy, our DPO model achieved an accuracy of 67.5% (row 10), outperforming GPT-3.5-turbo (58.7%, row 1) and GPT-4o (64.0%, row

<sup>2</sup>The sample size is larger than the one tested on three individuals in Luo et al. (2024).

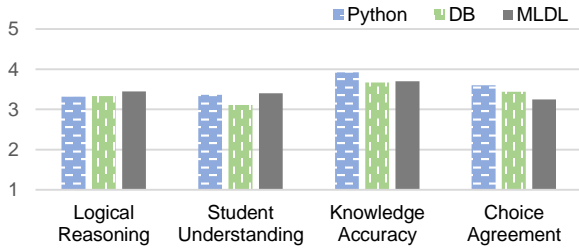


Figure 3: Human evaluation on our pairwise ranker. The results from participants were averaged.

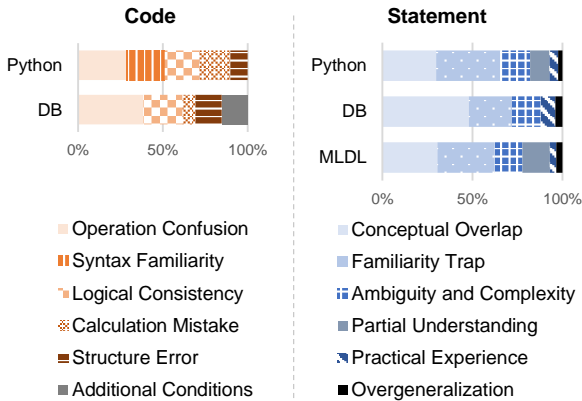


Figure 4: Plausibility factors in our pairwise ranker's reasoning.

2) on average. This result is somewhat surprising because our model was trained on reasoning generated by GPT-4o. Moreover, the DPO model significantly outperformed the SFT models (58.7%–65.7%, rows 8–9), particularly in Python, showing the effectiveness of DPO in enhancing the reasoning capability of the model. While Scarlatos et al. (2024)'s method achieved strong performance on math questions in their original work, it exhibited lower accuracy on the CS subjects (48.8%, row 7).

**(2) Human Evaluation** Human experts (two professors) tasked with choosing the more plausible distractor for 60 questions achieved an accuracy of 71.7%, compared to 70% achieved by our DPO model on the same task. This result suggests that the task is challenging even for experts and that GPT-like LLMs trained on large data can predict the confusion experienced by students at a level comparable to human performance.

Figure 3 presents survey results from three Master's students evaluating the reasoning quality of the DPO model on a 5-point Likert scale. These results provide mild to moderate evidence supporting the model's ability to infer students' misconceptions through logical reasoning and accurate knowledge.

**(3) Plausibility Factors** We analyzed main factors revealed in the model's reasoning to determine plausibility. We selected reasoning outputs where the DPO model predicted correct choices, and categorized plausibility factors in collaboration with GPT-4o. Figure 4 visualizes the proportion of each category. In the code type questions (e.g., determining the output of a code snippet or filling in blanks), factors such as incorrect assumptions about function outputs or operations were the most common, while in the statement type questions (e.g., selecting statements about concepts), factors like conceptual overlap with other similar terms appeared most frequently. Definitions for each category can be found in Appendix A.5.

**(4) Reasoning Methods** We conducted an ablation study to examine the effectiveness of our reasoning method for rank accuracy. As shown in Table 3, for GPT-4o, using our reasoning structure (row 2) substantially outperformed other reasoning methods (rows 4–6), leading us to adopt the current reasoning format for the trained models. Training the model without the reasoning process (row 11) significantly reduced ranking accuracy, highlighting the importance of our reasoning method.

**(5) Consistency in Rank Prediction** We evaluated the consistency of predictions when input order was altered and found that our model exhibits lower positional bias compared to GPT-3.5-turbo. The experimental results are provided in Appendix A.4.

**(6) Error Analysis** Upon analyzing cases where our pairwise ranker produced incorrect reasoning, we identified several types of error, such as misjudging implausible errors as plausible and struggling with reasoning for unfamiliar questions that were underrepresented in the training data. A detailed analysis and suggestions for future work can be found in Appendix A.8.

## 5.2 Distractor Generator

**(1) Plausibility** Table 4 summarizes the win/lose counts of our distractor generators against GPT models, Feng et al. (2024), Hang et al. (2024) and human-authored distractors, as evaluated by our pairwise ranker (DPO-based). Our DPO model generated more plausible distractors than baseline models in most cases. Compared to human-authored distractors, our DPO model excelled in Python but underperformed in DB and MLDL. This discrep-

		per Distractor (Win $\uparrow$ /Lose $\downarrow$ )			
		Setting A		Setting B	
		Ours (SFT)	Ours (DPO)	Ours (SFT)	Ours (DPO)
Python	GPT-3.5	127/ <b>129</b>	145/101	190/ <b>198</b>	198/190
	GPT-4o	158/ <b>199</b>	184/156	178/ <b>212</b>	200/185
	Feng et al. (2024)	153/ <b>157</b>	164/131	176/ <b>223</b>	196/ <b>199</b>
	Hang et al. (2024)	110/ <b>167</b>	137/120	179/ <b>218</b>	202/198
	Human-Authored	191/ <b>199</b>	207/159	220/175	217/178
DB	GPT-3.5	29/ <b>32</b>	28/26	37/ <b>48</b>	34/42
	GPT-4o	40/ <b>55</b>	50/41	38/ <b>47</b>	47/39
	Feng et al. (2024)	35/ <b>47</b>	41/36	45/41	45/33
	Hang et al. (2024)	34/ <b>43</b>	43/25	37/ <b>49</b>	39/ <b>52</b>
	Human-Authored	25/ <b>71</b>	35/ <b>54</b>	24/ <b>53</b>	31/ <b>51</b>
MLDL	GPT-3.5	72/ <b>73</b>	68/65	128/115	150/89
	GPT-4o	104/ <b>110</b>	104/91	135/134	167/99
	Feng et al. (2024)	81/ <b>90</b>	84/68	129/123	150/110
	Hang et al. (2024)	57/ <b>106</b>	62/ <b>83</b>	109/ <b>145</b>	139/113
	Human-Authored	86/ <b>130</b>	81/ <b>107</b>	111/ <b>141</b>	127/119

Table 4: Plausibility evaluation on distractor generators. Win/lose counts of our models (columns) against baselines (rows), averaged over two evaluations.

ancy may be due to the underrepresentation of these subjects in our dataset, leading to limited exposure during training.

We assessed the benefit of augmenting the base MCQ dataset with synthetic distractors and automated ranking (i.e., the student choice dataset). Using only the base MCQ dataset for SFT and DPO led to a significant performance drop compared to using the whole student choice dataset, and no significant difference was observed between SFT and DPO (Appendix B.6). This highlights the importance of incorporating diverse chosen-rejected samples and sufficient distractors during training. Overall, the results demonstrate that our approach of creating the student choice dataset and employing DPO using this data effectively enhances distractor plausibility.

We further examined the models’ performance based on question type (i.e., code vs. statement). Our model outperformed GPT-3.5-turbo in generating plausible distractors for code type questions but was slightly less effective for statement type questions in Python and DB. In contrast, compared to GPT-4o, our model tended to perform better in statement type questions. Detailed results are in Appendix B.3.

**(2) Human Evaluation** Table 5 compares the frequency of distractors selected by students, showing that our DPO model generated more plausible distractors than GPT-4o across all subjects and out-

	# of Selected Distractors $\uparrow$					DI $\uparrow$
	Python	DB	MLDL	Top 50%	Low 50%	Avg.
GPT-3.5	42	<b>18</b>	22	38	44	0.162
GPT-4o	14	5	26	22	23	0.119
Ours (SFT)	40	10	24	32	42	0.194
Ours (DPO)	<b>45</b>	14	<b>27</b>	<b>39</b>	<b>47</b>	<b>0.212</b>

Table 5: Human evaluation on distractor generators.

performed GPT-3.5-turbo in all but one subject. To evaluate whether the distractors have differing impacts based on students’ proficiency levels, we divided the students into two groups—Top 50% and Low 50%—based on their average scores. The distractors generated by the DPO model were most frequently chosen by both groups. These findings suggest that our model may effectively identify areas of confusion across varying proficiency levels as a versatile tool for a wide range of students.

Our DPO model achieved the highest discrimination index (DI) of 0.212, falling within the acceptable range of discrimination (0.21–0.24) (Kumar et al., 2021). This indicates that the distractors generated by our model are better at differentiating between high-performing students and low-performing ones than the baseline models. This is desirable because MCQs with a high DI can identify misconceptions and gaps in students’ knowledge, and challenging MCQs can promote deeper learning.

Expert evaluation of our DPO model on *clarity* and *answerability* using a 5-point Likert scale showed that all metrics scored above 4, confirming that most distractors were clear enough to answer the question. Detailed results are provided in Appendix B.4.

**(3) Additional Evaluations** We additionally evaluated the similarity between model-generated distractors and human-authored ones, as well as their validity. Our DPO model showed greater text similarity to human-authored distractors than GPT-3.5-turbo and GPT-4o. It also demonstrated higher validity compared to GPT-3.5-turbo, particularly excelling in questions that ask for incorrect statements.

Furthermore, we examined the similarity between model-generated distractors and the correct answer to assess the potential issue of distractors being too similar to the correct answer. Our analysis found no evidence that our models pose a par-



		per Distractor (Win↑/Lose↓)			
		Setting A		Setting B	
		Ours (SFT)	Ours (DPO)	Ours (SFT)	Ours (DPO)
Python	GPT-3.5	<b>71/65</b>	<b>81/56</b>	<b>100/74</b>	<b>105/59</b>
	GPT-4o	<b>57/78</b>	<b>74/60</b>	<b>87/53</b>	<b>95/43</b>
DB	GPT-3.5	<b>89/67</b>	<b>88/51</b>	<b>95/64</b>	<b>107/54</b>
	GPT-4o	<b>93/67</b>	<b>90/53</b>	<b>103/47</b>	<b>120/33</b>
MLDL	GPT-3.5	59/ <b>93</b>	59/ <b>82</b>	71/ <b>108</b>	83/ <b>96</b>
	GPT-4o	57/ <b>100</b>	67/ <b>80</b>	77/ <b>98</b>	<b>90/90</b>
English	GPT-3.5	<b>44/44</b>	41/ <b>42</b>	<b>78/65</b>	<b>83/61</b>
	GPT-4o	44/ <b>46</b>	39/ <b>46</b>	<b>80/64</b>	<b>84/59</b>

Table 6: Plausibility evaluation of distractor generators on four publicly available datasets (GPT-generated CS questions and a Korean high school English exam). For the English questions, plausibility was evaluated using GPT-4o due to its higher performance. Win/lose counts of our models (columns) against baselines (rows), averaged over two evaluations.

	Rank Accuracy ↑		
	GPT-3.5	GPT-4o	Ours (SFT)
English	0.483	<b>0.608</b>	0.573

Table 7: Evaluation results on pairwise rankers for English questions. The results were averaged over five generations for each model.

ticularly high risk to students because of this issue. Detailed analyses can be found in Appendix B.5.

**(4) Error Analysis** We analyzed the suboptimal distractors generated by our model and identified several types of issues. For code type questions, the distractors lacked variation in format, while for statement type questions, they were overly similar to the correct answers and failed to incorporate broader conceptual differences. Examples of each type and future improvement strategies are detailed in Appendix B.7.

**(5) Generalizability** To verify the generalizability of our approach beyond the base MCQ dataset and the CS domain, we conducted additional experiments on two publicly available datasets: (1) newly generated CS questions created using GPT-4o and (2) high school English exam questions.

For CS questions, we generated 100 MCQs per subject using GPT-4o and built a new student choice dataset to train a distractor generator. The results in Table 6, evaluated using our pairwise ranker, are consistent with those from the base MCQ dataset, reaffirming that plausibility im-

proves with DPO over SFT.

For English questions, we used 88 questions from a South Korean high school exam<sup>3</sup> to train a pairwise ranker and a distractor generator. In Table 7, our pairwise ranker, despite limited training data, outperformed GPT-3.5-turbo and closely approached GPT-4o. Similarly, Table 6 shows that in Setting B, where more distractors were compared, our DPO model achieved higher plausibility than GPT models, reflecting the trends observed in CS subjects.

## 6 Conclusion

In this study, we proposed a pipeline for training a model to generate more plausible distractors for MCQs and demonstrated its effectiveness across computer science subjects. We trained the pairwise ranker to evaluate the relative plausibility of distractors, and used this to create the student choice dataset where distractors for each question are ranked by plausibility. From this dataset, we created chosen-rejected pairs of distractors to train the distractor generator using DPO. Our models outperformed GPT and other baseline models and performed comparably to humans in various metrics, including pairwise rank accuracy and distractor plausibility. We believe that our work can advance automated educational tools, contributing to more adaptive and effective learning environments.

## Limitations

The models presented in this study have the following limitations. First, the pairwise ranker’s method of comparing distractors pairwise significantly increases the number of combinations and requires substantial computing resources due to the need for generating reasoning. A listwise approach using an encoder-decoder structure could be explored as a solution (Yoon et al., 2024).

Second, the distractor generator occasionally produces invalid distractors, necessitating review by human experts or high-performing LLMs (e.g., GPT-4o) to accurately evaluate students’ knowledge. To address this limitation, future work could include an additional supervision phase, such as integrating feedback loops with other models or applying constraints like *Counterfactual Contrastive Decoding* (Qu et al., 2024).

<sup>3</sup>These MCQs are from the latest CSAT (Korean SAT), and the distractor selection rates were obtained from an online education platform specializing in CSAT preparation.

Finally, our method focuses on generating difficult distractors, but there are instances where adjusting the difficulty level of MCQs to suit the needs of the target students is necessary. While our pairwise ranker can be utilized to select distractors with varying degrees of plausibility, future work could explore more direct approaches, such as incorporating student knowledge tracing or adaptive decoding, to address this challenge (Cui and Sachan, 2023).

## Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) (RS-2024-00333484, RS-2024-00414981). It was also supported by Elice, Inc., which also provided the proprietary datasets.

## References

- Sun-Geun Baek. 2019. *Theory and Practice of Educational Evaluation*. Kyoyookbook, Paju.
- Shang-Hsuan Chiang, Ssu-Cheng Wang, and Yao-Chung Fan. 2022. [CDGP: Automatic cloze distractor generation based on pre-trained language model](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5835–5840, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Peng Cui and Mrinmaya Sachan. 2023. [Adaptive and personalized exercise generation for online language learning](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10184–10198, Toronto, Canada. Association for Computational Linguistics.
- Jacob Doughty, Zipiao Wan, Anishka Bompelli, Jubahed Qayum, Taozhi Wang, Juran Zhang, Yujia Zheng, Aidan Doyle, Pragnya Sridhar, Arav Agarwal, Christopher Bogart, Eric Keylor, Can Kultur, Jaromir Savelka, and Majd Sakr. 2024. [A comparative study of ai-generated \(gpt-4\) and human-crafted mcqs in programming education](#). In *Proceedings of the 26th Australasian Computing Education Conference, ACE '24*, page 114–123, New York, NY, USA. Association for Computing Machinery.
- Wanyong Feng, Jaewook Lee, Hunter McNichols, Alexander Scarlatos, Digory Smith, Simon Woodhead, Nancy Ornelas, and Andrew Lan. 2024. Exploring automated distractor generation for math multiple-choice questions via large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3067–3082.
- Nigel Fernandez, Alexander Scarlatos, Wanyong Feng, Simon Woodhead, and Andrew Lan. 2024. [DiVERT: Distractor generation with variational errors represented as text for math multiple-choice questions](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9063–9081, Miami, Florida, USA. Association for Computational Linguistics.
- Ching Nam Hang, Chee Wei Tan, and Pei-Duo Yu. 2024. [Mccqgen: A large language model-driven mcq generator for personalized learning](#). *IEEE Access*, 12:102261–102273.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. [Lora: Low-rank adaptation of large language models](#). *Preprint*, arXiv:2106.09685.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. [Mistral 7b](#). *arXiv preprint arXiv:2310.06825*.
- Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. [Prometheus 2: An open source language model specialized in evaluating other language models](#). *arXiv preprint arXiv:2405.01535*.
- Dharmendra Kumar, Raksha Jaipurkar, Atul Shekhar, Gaurav Sikri, and V Srinivas. 2021. Item analysis of multiple choice questions: A quality assurance test for an assessment tool. *Medical Journal Armed Forces India*, 77:S85–S89.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. [RACE: Large-scale ReAding comprehension dataset from examinations](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.
- Guillaume Le Berre, Christophe Cerisara, Philippe Langlais, and Guy Lapalme. 2022. [Unsupervised multiple-choice question generation for out-of-domain Q&A fine-tuning](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 732–738, Dublin, Ireland. Association for Computational Linguistics.
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. [G-eval: NLG evaluation using gpt-4 with better human alignment](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.
- Haohao Luo, Yang Deng, Ying Shen, See-Kiong Ng, and Tat-Seng Chua. 2024. [Chain-of-exemplar: Enhancing distractor generation for multimodal educational question generation](#). In *Proceedings of the*

- 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7978–7993, Bangkok, Thailand. Association for Computational Linguistics.
- Wajiha Mahjabeen, Saeed Alam, Usman Hassan, Tahira Zafar, Rubab Butt, Sadaf Konain, and Myedah Rizvi. 2017. Difficulty index, discrimination index and distractor efficiency in multiple choice questions. *Annals of PIMS-Shaheed Zulfiqar Ali Bhutto Medical University*, 13(4):310–315.
- Hyeongdon Moon, Yoonseok Yang, Hangeol Yu, Seunghyun Lee, Myeongho Jeong, Juneyoung Park, Jamin Shin, Minsam Kim, and Seungtaek Choi. 2022. [Evaluating the knowledge dependency of questions](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10512–10526, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Jeroen Offerijns, Suzan Verberne, and Tessa Verhoef. 2020. [Better distractions: Transformer-based distractor generation and multiple choice question filtering](#). Preprint, arXiv:2010.09598.
- Zhaopeng Qiu, Xian Wu, and Wei Fan. 2020. [Automatic distractor generation for multiple choice questions in standard tests](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2096–2106, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Fanyi Qu, Hao Sun, and Yunfang Wu. 2024. [Unsupervised distractor generation via large language model distilling and counterfactual contrastive decoding](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 827–838, Bangkok, Thailand. Association for Computational Linguistics.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.
- Nils Reimers and Iryna Gurevych. 2019. [Sentence-BERT: Sentence embeddings using Siamese BERT-networks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.
- Siyu Ren and Kenny Q. Zhu. 2021. [Knowledge-driven distractor generation for cloze-style multiple choice questions](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4339–4347.
- Assad Ali Rezigalla, Ali Mohammed Elhassan Seid Ahmed Eleragi, Amar Babikir Elhussein, Jaber Alfaifi, Mushabab A ALGhamdi, Ahmed Y Al Ameer, Amar Ibrahim Omer Yahia, Osama A Mohammed, and Masoud Ishag Elkhalifa Adam. 2024. Item analysis: the impact of distractor efficiency on the difficulty index and discrimination power of multiple-choice items. *BMC Medical Education*, 24(1):445.
- Alexander Scarlatos, Wanyong Feng, Digory Smith, Simon Woodhead, and Andrew Lan. 2024. [Improving automated distractor generation for math multiple-choice questions with overgenerate-and-rank](#). In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, pages 222–231, Mexico City, Mexico. Association for Computational Linguistics.
- Jinnie Shin, Qi Guo, and Mark J Gierl. 2019. Multiple-choice item distractor development using topic modeling approaches. *Frontiers in psychology*, 10:825.
- Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. [DREAM: A challenge data set and models for dialogue-based reading comprehension](#). *Transactions of the Association for Computational Linguistics*, 7:217–231.
- Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. [Is ChatGPT good at search? investigating large language models as re-ranking agents](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, Singapore. Association for Computational Linguistics.
- Neeraj Varshney, Satyam Raj, Venkatesh Mishra, Agneet Chatterjee, Ritika Sarkar, Amir Saeidi, and Chitta Baral. 2024. [Investigating and addressing hallucinations of llms in tasks involving negation](#). Preprint, arXiv:2406.05494.
- Hui-Juan Wang, Kai-Yu Hsieh, Han-Cheng Yu, Jui-Ching Tsou, Yu An Shih, Chen-Hua Huang, and Yao-Chung Fan. 2023. [Distractor generation based on Text2Text language models with pseudo Kullback-Leibler divergence regulation](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12477–12491, Toronto, Canada. Association for Computational Linguistics.
- Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. [Crowdsourcing multiple choice science questions](#). In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.
- Soyoung Yoon, Eunbi Choi, Jiyeon Kim, Hyeonju Yun, Yireun Kim, and Seung-won Hwang. 2024. [ListT5: Listwise reranking with fusion-in-decoder improves zero-shot retrieval](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2287–2308, Bangkok, Thailand. Association for Computational Linguistics.
- Han Cheng Yu, Yu An Shih, Kin Man Law, KaiYu Hsieh, Yu Chen Cheng, Hsin Chih Ho, Zih An Lin, Wen-Chuan Hsu, and Yao-Chung Fan. 2024. [Enhancing distractor generation for multiple-choice questions](#)



with retrieval augmented pretraining and knowledge graph integration. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11019–11029, Bangkok, Thailand. Association for Computational Linguistics.

## A Pairwise Ranker

### A.1 Prompt for Pairwise Ranker

The instruction prompts for the pairwise ranker are in Table 17 (Reasoning), 18 (Rubric), 19 (G-Eval), 20 (Discussion) and 21 (Scarlatos et al., 2024). We used the same prompt (Reasoning) with GPT models and ours (SFT, DPO).

### A.2 SFT and DPO Settings for Pairwise Ranker

The pairwise ranker model was trained using Mistral-7B-Instruct-v0.2<sup>4</sup> with 4-bit quantization and fine-tuned using LoRA. For SFT, the learning rate was set to  $2e-4$  and the model was trained for 5 epochs. For DPO, the learning rate was set to  $1e-6$ , also trained for 5 epochs. These hyperparameters were selected as they allowed stable training without overfitting while preserving the quality of the DPO output. SFT took approximately 2 hours, and DPO took about 1 hour on an NVIDIA A6000 GPU. Scarlatos et al. (2024) model was reproduced for baseline comparison using the same model and DPO settings as above.

### A.3 Prompt for Generating Pairwise Ranker Training Data

The instruction prompt for generating pairwise ranker training data is in Table 22. To enhance the diversity of expressions and reasoning used in the samples, two reasoning examples are generated for each pair—one with temperature set to 0 and the other to 1—using GPT-4o.

### A.4 Consistency in Rank Prediction

Table 8 demonstrates that our pairwise ranker exhibits relatively robust to positional bias. In comparison to GPT-3.5-turbo, which required an average of more than two attempts to produce consistent results when the input order was altered, our DPO model was able to achieve consistent results with significantly fewer attempts. Additionally, our DPO model slightly outperformed GPT-4o by requiring fewer average generation attempts.

<sup>4</sup>This model is distributed under the Apache 2.0 license. <https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2>

	Generation Attempts per Question ↓			
	Python	DB	MLDL	Avg.
GPT-3.5 (Reasoning)	2.491	2.482	2.427	2.467
GPT-4o (Reasoning)	1.753	<b>1.699</b>	<b>1.699</b>	1.717
GPT-4o (Rubric)	2.306	2.316	2.212	2.278
GPT-4o (G-Eval)	5.303	5.193	5.150	5.215
Ours (SFT, Sep.)	1.708	1.718	1.771	1.732
Ours (SFT, Comb.)	1.685	1.715	1.740	1.713
Ours (DPO, Comb.)	<b>1.650</b>	1.725	1.740	<b>1.705</b>
Ours (SFT w/o Reasoning)	2.013	2.034	2.036	2.028

Table 8: Consistency evaluation results on pairwise rankers. The results were averaged over five generations for each model.

### A.5 Plausibility Factors

We used GPT-4o to summarize and categorize reasoning samples where our pairwise ranker accurately predicted the rankings on the test set, and selected six representative examples per question type. Definitions for each category are in Table 23 (Code Type) and 24 (Statement Type).

### A.6 Human Evaluation

**Recruitment** We conducted a survey with three Master’s degree students who voluntarily expressed their willingness to participate in this experiment. The survey was designed to begin only after they agreed to provide their results for research purposes and acknowledged the precautions via an online form. The experiment lasted approximately 90 minutes, and participants were compensated above the standard hourly wage for the time they participated. The entire process of human evaluation was conducted following procedures approved by the IRB committee of our university.

**Survey Form** The reasoning quality of our pairwise ranker was evaluated on a 5-point Likert scale based on the following criteria:

- **Logical Reasoning:** Whether the reasoning process is logical.
- **Student Understanding:** Whether the reasoning effectively understands students’ misconceptions or problem-solving processes.
- **Knowledge Accuracy:** Whether the reasoning is based on accurate and error-free knowledge.
- **Choice Agreement:** Whether the evaluator agrees with the model’s final choice.

An example of the survey form is presented in Table 25.



## A.7 Ablation Study

The instruction prompt used for the ablation study (w/o Reasoning) is in Table 26, and the training settings are identical to those of our pairwise ranker training setup (Appendix A.2).

## A.8 Error Analysis

Our pairwise ranker exhibited the following three types of errors:

First, our model tended to incorrectly judge implausible mistakes as plausible—errors that real students would not typically make. For example, in the process of calculating the output of Python code, the model incorrectly deemed ‘unrealistic reasoning’ or ‘mistakes in obvious calculations’ as plausible, even though such errors would be unlikely for actual students to make based on common sense.

Second, our model struggled with reasoning when encountering unfamiliar questions that were insufficiently represented in the training data. This issue was particularly evident in subjects like DB and MLDL, where the training set was relatively small and shared few similar concepts or questions with the test set.

Lastly, in questions requiring the selection of an *incorrect* option, there were cases where our model’s final ranking was correct, but its reasoning was flawed. Instead of identifying why each option seemed more incorrect to the students, the model mistakenly focused on determining which option was more correct.

To improve the pairwise ranker, future work should focus on enabling the model to learn *common student misconceptions* for better reasoning and prediction and enhancing the inference process to clearly recognize *question requirements*.

## B Distractor Generator

### B.1 Prompt for Distractor Generator

The instruction prompt for our distractor generator is in Table 27. We used the same prompt for both GPT models and ours. However, we instructed the GPT models to generate outputs in *JSON* format for stability reasons.

The instruction prompt for the kNN approach proposed by Feng et al. (2024) is presented in Table 28. Following the method outlined in the paper, the target question and answer were encoded using the SBERT encoder (Reimers and Gurevych,

		per Question (Win↑/Tie/Lose↓)		per Distractor (Win↑/Lose↓)	
		Setting A	Setting B	Setting A	Setting B
		Ours (DPO, window)	Ours (DPO, window)	Ours (DPO, window)	Ours (DPO, window)
Python	GPT-3.5	21/13/18	30/1/20	140.5/110.5	216/173
	GPT-4o	21/9/22	22/11/18	186/162	200/187
DB	GPT-3.5	4/6/3	6/1/4	33.5/25.5	42.5/33.5
	GPT-4o	6/2/5	5/4/2	48.5/44.5	50.5/40.5
MLDL	GPT-3.5	7/12/12	13/2/15	60.5/72.5	136.5/107.5
	GPT-4o	12/6/14	21/1/10	95.5/102.5	165/107

Table 9: Plausibility evaluation on the distractor generator, DPO with sliding window setting (Appendix B.2).

2019), MPNet<sup>5</sup>, and the top-3 most similar items based on cosine similarity were extracted from the question pool (training set) and used as in-context examples.

### B.2 SFT and DPO Settings for Distractor Generator

The distractor generator model was trained using Mistral-7B-Instruct-v0.2 with 4-bit quantization and fine-tuned using LoRA. For SFT, the learning rate was set to  $2e-4$  and the model was trained for 2 epochs. For DPO, the learning rate was set to  $1e-5$ , trained for 3 epochs. These hyperparameters were determined as a result of finding a setup that avoids overfitting while ensuring no issues with the quality of the DPO output. SFT and DPO took approximately 3 hours on an NVIDIA A6000 GPU.

As briefly mentioned in §3.4, in addition to the chosen-rejected sample pairing method described in the main text, another setting employs a method similar to a *sliding window* for pairing. In this setting, all distractor candidates are sorted in descending order and grouped into non-overlapping windows of size  $n$ . For example, if there are six candidates and  $n$  is 2, a total of three windows are created. Pairwise combinations between these windows are then used to create chosen-rejected samples. A model trained with DPO using these samples showed no significant performance difference compared to the model described in the main text. The plausibility evaluation results for this model are provided in Table 9.

### B.3 Plausibility Evaluation

**per Question** The results of the plausibility evaluation analyzed from a per-question perspective

<sup>5</sup><https://huggingface.co/sentence-transformers/all-mpnet-base-v2>

		per Question (Win↑/Tie/Lose↓)			
		Setting A		Setting B	
		Ours (SFT)	Ours (DPO)	Ours (SFT)	Ours (DPO)
Python	GPT-3.5	17/13/22	23/15/14	21/7/22	26/1/23
	GPT-4o	12/15/25	24/12/16	19/6/26	24/4/22
	Feng et al. (2024)	14/21/17	21/15/15	16/5/28	25/2/21
	Human-Authored	21/11/20	27/10/15	31/5/15	26/4/20
DB	GPT-3.5	3/3/7	6/2/5	4/1/5	6/1/4
	GPT-4o	4/1/8	4/3/6	5/0/6	6/0/5
	Feng et al. (2024)	3/3/7	6/3/4	6/1/3	6/0/4
	Human-Authored	1/1/11	4/0/9	5/1/7	4/2/7
MLDL	GPT-3.5	9/12/10	12/13/6	12/1/17	18/2/10
	GPT-4o	11/4/17	12/7/12	13/4/15	19/7/5
	Feng et al. (2024)	11/6/14	15/6/9	17/0/15	18/3/10
	Human-Authored	11/4/17	9/7/16	11/3/18	15/2/4

Table 10: Plausibility evaluation on distractor generators.

are presented in Table 10 (compare with Table 4).

**Case Study** The analysis of plausibility results based on question types (Code/Statement) is provided in Table 11. A summary of the case study results is as follows:

First, our model generates more plausible distractors for code type questions compared to GPT-3.5-turbo. The distractors generated by the latter were either significantly different from the correct answer or included code syntax that does not actually exist. On the other hand, for the statement type questions, GPT-3.5-turbo demonstrated higher plausibility only in the cases of Python and DB. This was because its distractors included more diverse knowledge or additional conditions, while our model seemed to construct distractors with relatively limited scope of knowledge, possibly due to the small training dataset.

Next, our model exhibited higher plausibility in the statement type compared to GPT-4o. When compared with the validity results in Appendix B.5 ('Statement'), it can be seen that GPT-4o generated more obvious statements, resulting in a lower risk of invalid distractors but making the difficulty level lower. For the code type, both models generated distractors that were not far from the correct answer. However, in the case of Python, the distractors generated by our model were slightly less plausible than those of GPT-4o, likely because the latter made better use of partial errors in the code.

## B.4 Human Evaluation

**Recruitment** We conducted the evaluation with 15 college students who voluntarily agreed to par-

		per Question (Win↑/Lose↓)		per Distractor (Win↑/Lose↓)	
		Code	State.	Code	State.
		Python	GPT-3.5	12/5	14/18
GPT-4o	6/11		18/11	58.5/70.5	142/115
DB	GPT-3.5	4/0	2/4	22.5/8.5	11.5/33.5
	GPT-4o	2/2	4/3	20/16	27/23
MLDL	GPT-3.5	-	18/10	-	150/89
	GPT-4o	-	19/5	-	167/99

Table 11: Plausibility evaluation on the distractor generator, categorized by question type. This table further details the results from Table 4 and 10, Setting B, Ours (DPO).

	5-point Likert Scale ↑		
	Python	DB	MLDL
Clairity	4.418	4.282	4.373
Answerability	4.414	4.264	4.382

Table 12: Evaluation results on the clarity and answerability of distractors generated by our DPO model.

icipate. The test was conducted online, and participants were allowed to begin the test only after agreeing to the instruction stating that their results would be provided for research purposes. The experiment took approximately 60 minutes, and participants were compensated with a reward above the standard hourly wage for their time. The entire human evaluation process was conducted in accordance with the procedures approved by the IRB committee of our university.

**Test Form** Each question allows for multiple selections (e.g., Select all the correct/incorrect ...) and includes one distractor generated by each model, along with 'None of the above' as the final option. To mitigate unintended effects on the selection rate of distractors when the actual correct answer is included, two versions of each question were created: one with the correct answer included and one without. These versions were randomly distributed. For analysis, the results from both versions were integrated.

**DI** To analyze the DI of a specific model, it is necessary to assume that each item consists solely of options generated by that model. Therefore, we restructured the test results by treating each distractor generated by a model as a separate test item that determines 'whether the corresponding statement (distractor) is true or false'. In other words, we as-

	sBLEU $\uparrow$			BERTScore $\uparrow$		
	Python	DB	MLDL	Python	DB	MLDL
GPT-3.5	12.572	16.794	10.133	0.879	0.893	0.873
GPT-4o	15.387	24.752	16.120	0.893	<b>0.912</b>	<b>0.882</b>
Mistral	11.192	14.642	10.850	0.859	0.872	0.863
Ours (SFT)	16.859	20.892	14.752	0.894	0.897	0.876
Ours (DPO)	<b>18.313</b>	<b>26.322</b>	<b>16.476</b>	<b>0.896</b>	0.906	0.881

Table 13: Similarity between model-generated and human-authored distractors.

sumed that all students took multiple independent tests, each consisting of items created exclusively with distractors from a single model. When grading, if a student chose the distractor generated by the model, the item was considered incorrect; otherwise, it was considered correct. The cutoff for dividing students into high and low groups was set at the top and bottom 27%, and the DI calculation formula was also in line with previous studies (Mahjabeen et al., 2017; Rezigalla et al., 2024).

**Clarity and Answerability** We concern that excessively ambiguous distractors could hinder educational assessment. To verify whether such issues exist, we asked 11 experts to evaluate the clarity and answerability of distractors generated by our DPO model on a 5-point Likert scale. The results, averaged across a total of 40 questions (Python: 20, DB: 10, MLDL: 10) for each subject, are in Table 12. These results confirm that the potential side effects of high plausibility, which were a concern, do not appear to be present in our model.

## B.5 Additional Evaluations

**Similarity Between Model-Generated and Human-Authored Distractors** Table 13 presents the similarity evaluation results for the distractor generator. sBLEU<sup>6</sup> and BERTScore<sup>7</sup> were used as the text similarity metrics. For sBLEU, the ‘smooth\_method’ was set to ‘exp’, and the default parameters were used for BERTScore. In terms of sBLEU, our model (DPO) generates distractors that are most similar to human-authored ones across the majority of subjects.

**Similarity Between Model-Generated Distractors and Correct Answers** We evaluated the semantic similarity between distractors and correct answers using OpenAI’s *text-embedding-3-small* model. Cosine similarity scores were computed

<sup>6</sup><https://github.com/mjpost/sacrebleu>

<sup>7</sup>[https://github.com/Tiiiger/bert\\_score](https://github.com/Tiiiger/bert_score)

between correct answers and distractors generated by each model.

As shown in Table 14, similarity scores are largely consistent across models, with no evidence that our SFT or DPO models generate distractors that are excessively similar to correct answers. For instance, the average similarity score of human-authored distractors (0.53) is comparable to that of our DPO model (0.59) for Python.

- **Question:** "Choose the incorrect statement."
- **Correct Answer:** "If an if statement’s condition is False, its content executes before the else statement."
- **Distractor 1 (similarity = 0.59):** "An if-else statement can include multiple conditions using elif in some languages."
- **Distractor 2 (similarity = 0.53):** "An if statement can exist without an else statement in most languages."

A 0.06 difference in similarity score does not indicate a meaningful impact on quality, as the first distractor is not noticeably more similar to the correct answer than the second.

**Validity** Validity refers to whether the distractors are indeed incorrect options for the question. We assessed validity by categorizing questions as either Correct/Incorrect or Code/Statement. ‘Correct’ and ‘Incorrect’ refer to question types where the task is to select the correct or incorrect statement, respectively. ‘Code’ type questions involve cases where the answer (and distractors) take the form of filling in blanks or matching outputs in code. ‘Statement’ refers to questions composed of explanatory statements about a concept.

Table 15 shows the proportion of valid distractors generated by each model according to the type of question. Our models demonstrate stable validity across various question types, significantly outperforming GPT-3.5-turbo and pre-trained Mistral. This highlights the importance of the proposed methodology—first generating the type ( $T$ ) such as ‘Correct/Incorrect knowledge’—in enhancing validity.

Studies have shown that LLMs perform poorly on tasks involving negation (Varshney et al., 2024), and in a similar vein, GPT-3.5-turbo and Mistral show significantly lower validity when generating distractors for question types that require selecting an incorrect option (in ‘Incorrect’ type, the distractors should actually represent correct knowl-

	Similarity						
	Human	GPT-3.5	GPT-4o	Feng et al. (2024)	Hang et al. (2024)	Ours (SFT)	Ours (DPO)
Python	0.53 (0.04)	0.54 (0.05)	0.60 (0.05)	0.58 (0.04)	0.59 (0.05)	0.57 (0.06)	0.59 (0.06)
DB	0.55 (0.05)	0.55 (0.05)	0.60 (0.05)	0.58 (0.04)	0.59 (0.05)	0.59 (0.06)	0.61 (0.06)
MLDL	0.53 (0.04)	0.47 (0.05)	0.51 (0.05)	0.52 (0.04)	0.52 (0.05)	0.50 (0.05)	0.53 (0.05)

Table 14: Similarity between model-generated distractors and correct answers. Numbers in parentheses represent variance.

	Validity $\uparrow$											
	Correct			Incorrect			Code			Statement		
	Python	DB	MLDL	Python	DB	MLDL	Python	DB	MLDL	Python	DB	MLDL
GPT-3.5	0.883	0.571	0.938	0.400	<b>1.000</b>	0.426	0.877	0.630	-	0.588	0.630	0.684
GPT-4o	<b>0.938</b>	<b>1.000</b>	0.902	<b>0.967</b>	0.917	<b>0.956</b>	<b>0.912</b>	0.917	-	<b>0.970</b>	<b>0.963</b>	<b>0.927</b>
Mistral (w/o $T$ )	0.839	0.227	0.820	0.233	0.917	0.370	0.815	0.917	-	0.485	0.357	0.604
Mistral (w/ $T$ )	0.903	0.364	0.822	0.265	0.750	0.378	0.891	0.750	-	0.500	0.464	0.600
Ours (SFT)	0.874	0.905	0.765	0.902	<b>1.000</b>	0.844	0.842	<b>1.000</b>	-	0.909	0.926	0.802
Ours (DPO)	0.839	0.905	0.627	0.850	0.917	0.800	0.875	0.917	-	0.825	0.889	0.708
Ablation (SFT)	0.783	0.778	0.608	0.733	0.810	0.822	0.717	0.833	-	0.788	0.778	0.708
Ablation (DPO)	0.848	0.722	0.627	0.717	0.905	0.733	0.811	0.750	-	0.788	0.852	0.677

Table 15: Validity evaluation on distractor generators. Mistral is a model that has not been fine-tuned, w/o  $T$  is the result of using a prompt that generates distractors directly without specifying the distractor type, and w/  $T$  is the result using the same prompt as Ours.

		per Question (Win $\uparrow$ /Tie/Lose $\downarrow$ )		per Distractor (Win $\uparrow$ /Lose $\downarrow$ )	
		Ablation (SFT)	Ablation (DPO)	Ablation (SFT)	Ablation (DPO)
Python	GPT-3.5	15/14/ <b>23</b>	20/13/19	98.5/ <b>121.5</b>	115.5/ <b>122</b>
	GPT-4o	14/11/ <b>27</b>	18/9/ <b>25</b>	122/ <b>179</b>	135.5/ <b>184.5</b>
DB	GPT-3.5	<b>5/5/3</b>	3/4/ <b>6</b>	<b>29/21</b>	25.5/ <b>26.5</b>
	GPT-4o	7/1/5	4/4/5	<b>46.5/36</b>	42/ <b>44</b>
MLDL	GPT-3.5	8/10/ <b>13</b>	9/9/ <b>13</b>	55/ <b>70.5</b>	52/ <b>66</b>
	GPT-4o	<b>13/7/12</b>	<b>15/8/9</b>	87.5/ <b>104.5</b>	<b>95.5/82</b>

Table 16: Ablation study on our distractor generator. The evaluation setup is the same as Setting A in Table 4.

edge, but these models mostly generated distractors with incorrect knowledge). However, after going through SFT and DPO, the proportion of valid distractors generated for such types greatly increases, indicating that the proposed methodology in this study (first generating types such as ‘Correct/Incorrect knowledge’) plays an important role in improving validity. Meanwhile, there is a slight decrease in validity after DPO compared to SFT, which appears to be a trade-off arising from the process of creating more confusing distractors.

## B.6 Ablation Study

The training settings used for the ablation study are identical to those of our distractor generator training setup (Appendix B.2), except that the base MCQ dataset was used as the training data instead of the student choice dataset. Table 16 presents the results of the ablation study (compare with Table 4 and 10).

## B.7 Error Analysis

Analyzing the low-quality samples generated by our distractor generator revealed the following types of errors:

First, the model sometimes failed to produce the specified number of distractors based on the input parameter  $n$ , or it created duplicate distractors among the outputs.

Next, for code type questions, the generated distractors lacked diversity in output formats and often made minimal changes, such as altering only one or two variables, resulting in repetitive and insufficiently varied distractors.

Meanwhile, for statement type questions, the model overly mimicked the correct answer, creating distractors based on only one or two concepts,



while failing to effectively incorporate other related concepts.

Future work to improve the distractor generator could involve explicitly providing the model with information on *similar concepts* or *common errors* that students are likely to confuse.

### **B.8 Prompt for Checking Distractor Validity**

The instruction prompt for checking the validity of distractors is in Table 29. If the output is ‘invalid’ (as it is an incorrect option for the question), it is considered a distractor.

## **C Base MCQ Dataset**

We were provided with an MCQ dataset by an online learning platform for educational research purposes and processed it for use within the scope of the provided purpose. The questions and options, originally in Korean, were translated into English for experimental purposes. The provided MCQ data does not contain any personally identifiable information about the individuals who answered the questions, and we manually checked to confirm that the text does not include any offensive content.

### **D Prompt for Augmenting Distractors in the Base MCQ Dataset**

The instruction prompt for augmenting distractors in the base MCQ dataset is in Table 30. Through this prompt, the student choice dataset was constructed only when at least one newly generated distractor by GPT-4o was valid and did not overlap with the original.

## **E Potential Issues**

MCQs serve as a tool for assessing students’ knowledge, so the options must be based on accurate information (i.e., both the correct answer and distractors must be valid). As mentioned earlier in the limitations, distractors generated by the model may not be actual incorrect options to the question. To proactively address the potential issue, we explored methodologies to ensure the validity of the distractors generated by the model. As part of these efforts, we implemented instruction prompts and output formats for the model to classify the type ( $T$ ) of distractors, thereby mitigating this issue.

We used selection rate data from questions answered by hundreds of students to ensure the reliability of common misconception information for

training the pairwise ranker. However, since misconceptions can vary by learning level or educational environment, the model’s reasoning may not generalize to other populations. To make accurate predictions for a target population, selection rates specific to that group should be used.

---

**Pairwise Ranker Prompt (Reasoning)**

---

[INST] You are a teacher analyzing which distractor in a given Multiple Choice Question is more confusing for students and why. Your review should include the following content in one paragraph:

- Describe a realistic process of solving the problem from a student's perspective as you look at each distractor.
  - Consider why it might be plausible as the correct/incorrect statement, based on students' misconceptions, mistakes, intuition, etc., from various angles.
- Output your choice as a single token, either A or B, that students are more likely to choose.

[Question] {*question*}

[Answer] {*answer*}

[Distractor A] {*distractor*}

[Distractor B] {*distractor*}

Generate in the following format:

### Review:

### Choice: [/INST]

---

Table 17: Instruction prompt (Reasoning) for pairwise ranker.

---

**Pairwise Ranker Prompt (Rubric)**

---

Analyze which side of the given Multiple Choice Question distractor pair is more confusing and plausible to students based on the given rubric.

[Question] {*question*}

[Answer] {*answer*}

[Distractor A] {*distractor*}

[Distractor B] {*distractor*}

Evaluation Rubric:

[1]. Conceptual Misunderstandings: Evaluate if the distractor addresses into specific misconceptions or partial understandings related to the question.

[2]. Similarity to Correct Answer: Assess how closely the distractor resembles the correct answer, either in structure, terminology, or context.

[3]. Intuitive Appeal: Analyze if the distractor seems logical or intuitively correct based on common language use or student intuition.

Generation Guide:

- [n]: For each evaluation criterion, review in one sentence how each distractor may or may not confuse students.

- [Summary]: Summarize the review, and choose more confusing and plausible distractor.

- [Choice]: Output your choice as a single token, either A or B.

Generate in the following format:

[1]:

[2]:

[3]:

[Summary]:

[Choice]:

---

Table 18: Instruction prompt (Rubric) for pairwise ranker.

---

**Pairwise Ranker Prompt (G-Eval)**

---

You will be given one multiple-choice question (MCQ) and two distractors. Your task is to choose one distractor based on the metric.

Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:

Plausibility: This metric indicates how likely students are to feel that the distractor is the correct answer and choose it. A distractor with high plausibility is similar in form to the correct answer or contains common misconceptions and mistakes, making students more likely to select it.

Evaluation Steps:

1. Read the MCQ carefully and think about the relevant misconceptions or mistakes related to the question from your perspective as a teacher.
2. Judge how plausible and confusing the distractor would be from a student's perspective.
3. Choose one distractor based on Evaluation Criteria. Output your choice as a single token, either A or B.

[Question] { *question* }

[Answer] { *answer* }

[Distractor A] { *distractor* }

[Distractor B] { *distractor* }

Evaluation Form (A or B ONLY):

- Choice:

---

Table 19: Instruction prompt (G-Eval) for pairwise ranker.

---

**Pairwise Ranker Prompt (Discussion)**

---

**<Prompt - Student>**

Play the role of students with three different levels of proficiency: A is low, B is medium, and C is high.

A lower proficiency level indicates more confusion about the concept, while a higher proficiency level indicates a better understanding of the related knowledge.

- In a cooperative learning situation, three students with different levels of proficiency are discussing and solving a given problem together.
- For each option in the MCQ, share your thoughts according to each proficiency level. Discuss similar concepts and any confused or mistaken knowledge, ask for help, give advice, and interact actively.
- Having a high proficiency level does not mean knowing the correct answer. However, they have better problem-solving skills through reasonable inference.
- Take turns speaking equally among the low, medium, and high proficiency students. Use natural transitions like 'Shall we talk about this option next?' to keep the discussion flowing smoothly. End the discussion after discussing all the options.

[Question] { *question* }

[Options] { *distractors* }

Output the result in the following format:

[A]: "..."

[B]: "..."

[C]: "..."

**<Prompt - Teacher>**

Act as teachers discussing and judging the plausibility (whether it would confuse students) score of each distractor in a given MCQ.

- First, analyze the collaborative learning records of three students. Then, as the first teacher, choose between distractors A and B, deciding which one is more likely to confuse students or be frequently selected by them.
- As the second teacher, share your thoughts and provide reasonable counterarguments. Use the collaborative learning records of the three students as supporting evidence for your scoring. The second teacher should always question the initial score, challenge generalized assumptions, and argue which distractor is more plausible.
- Take turns discussing and adjusting the choice.
- The utterances must be clear and concise.

[Question] { *question* }

[Answer] { *answer* }

[Cooperative Learning Records] { *cooperative learning records* }

[Distractor A] { *distractor* }

[Distractor B] { *distractor* }

[Discussion History] { *history* }

A conclusion must be reached within a maximum of 5 utterances, taking into account both [T1] and [T2] combined.

Once you both agree on the final choice, output ### Choice: A or ### Choice: B.

Generate the next utterance in the discussion based on the discussion history:

[T1 or T2]: "..."

---

Table 20: Instruction prompt (Discussion) for pairwise ranker.

---

**Pairwise Ranker Prompt (Scarlatos et al., 2024)**

---

[INST] A teacher assigns the following programming question to the students.

Question: { *question* }

Correct answer: { *answer* }

Generate a distractor for this question that targets some student misconception.

Distractor: [/INST] { *distractor* }

---

Table 21: Instruction prompt (Scarlatos et al., 2024) for pairwise ranker.



---

**Prompt for Generating Pairwise Ranker Training Data**

---

You are a teacher analyzing which distractor in a given Multiple Choice Question is more confusing for students and why. Your review should include the following content in one paragraph:

- Describe a realistic process of solving the problem from a student's perspective as you look at each distractor. Consider why it might be plausible as the correct/incorrect statement, based on students' misconceptions, mistakes, intuition, etc., from various angles.
- Output your choice as a single token, either A or B, that students are more likely to choose.

[Question] {*question*}

[Answer] {*answer*}

[Distractor A] {*distractor a*}

[Distractor B] {*distractor b*}

Distractor chosen more frequently by actual students: {*a or b*}

Make sure your choice matches the distractor most frequently chosen by actual students. However, you must not mention this information as if you originally knew it.

Generate in the following format:

### Review:

### Choice:

---

Table 22: Instruction prompt for generating pairwise ranker training data.

Category	Definition
Operation Confusion	Distractors that involve misunderstanding of specific operations, such as incorrect assumptions about function outputs or operation precedence.
Structure Error	Distractors reflecting improper syntax or structural misunderstandings.
Calculation Mistake	Distractors that exploit errors in arithmetic, index calculations, or logical evaluations, leading to incorrect results.
Syntax Familiarity	Distractors that align with common syntax conventions or structures from Python or other programming languages, leading to confusion due to familiarity.
Logical Consistency	Distractors that maintain a consistent or plausible logic or pattern, even if incorrect, which can mislead students who are not fully confident in their understanding.
Additional Conditions	Distractors that introduce extra conditions or columns, which may lead students to misinterpret the problem as requiring more complex logic, thus creating confusion.

Table 23: Definitions of plausibility factors of code type question.

Category	Definition
Ambiguity and Complexity	Distractors that introduce nuanced or ambiguous details, leading to confusion and misinterpretation due to their complexity or lack of clarity.
Conceptual Overlap	Distractors that involve concepts or operations that overlap with other similar terms, causing students to conflate them and mistakenly believe they are correct.
Familiarity Traps	Distractors that use familiar terms or straightforward statements, making them seem correct at first glance and less likely to be critically analyzed by students.
Partial Understanding	Distractors built on incomplete knowledge, leading students to make errors due to gaps in conceptual clarity.
Overgeneralization	Distractors that appear plausible by relying on students' tendency to apply learned concepts too broadly without verifying their validity in specific contexts.
Practical Experience	Distractors that leverage students' familiarity with common tasks, such as data manipulation or querying, creating false confidence in their correctness.

Table 24: Definitions of plausibility factors of statement type question.

---

**Human Evaluation Survey Form**

---

**<Guideline>**

The following provides a programming multiple-choice question, along with an analysis (review) that predicts which of the two incorrect options is more challenging for students (i.e., more likely to be chosen). You are tasked with evaluating the quality of the analysis from the perspective of an education expert and stating whether you agree with the analysis.

Provided Items:

[Question]: The question

[Answer]: The correct answer

[Distractor A and B]: The two incorrect options, A and B

[Review]: An analysis of which incorrect option (A or B) would be more confusing (more likely to be chosen) by students, along with the final selection

Evaluation Criteria:

- Logical Reasoning: Whether the reasoning process is logical.
- Student Understanding: Whether the reasoning effectively understands students' misconceptions or problem-solving processes.
- Knowledge Accuracy: Whether the reasoning is based on accurate and error-free knowledge.
- Choice Agreement: Whether the evaluator agrees with the model's final choice.

**<Item>**

[Question] { *question* }

[Answer] { *answer* }

[Distractor A] { *distractor* }

[Distractor B] { *distractor* }

[Review] { *model's reasoning* }

- The reasoning process in the review is logical.  
| 1. Strongly Disagree | 2. Disagree | 3. Neutral | 4. Agree | 5. Strongly Agree |
  - The review demonstrates a good understanding of actual student misconceptions or problem-solving processes.  
| 1. Strongly Disagree | 2. Disagree | 3. Neutral | 4. Agree | 5. Strongly Agree |
  - The review is based on accurate and error-free knowledge.  
| 1. Strongly Disagree | 2. Disagree | 3. Neutral | 4. Agree | 5. Strongly Agree |
  - I agree with the final choice in the review.  
| 1. Strongly Disagree | 2. Disagree | 3. Neutral | 4. Agree | 5. Strongly Agree |
- 

Table 25: Survey form for human evaluation on the pairwise ranker. The original guideline in Korean has been translated into English.

---

**Pairwise Ranker Prompt (Ablation Study, w/o Reasoning)**

---

[INST] You are a teacher analyzing which distractor in a given Multiple Choice Question is more confusing for students. Output your choice as a single token, either A or B, that students are more likely to choose.

[Question] { *question* }

[Answer] { *answer* }

[Distractor A] { *distractor* }

[Distractor B] { *distractor* }

Generate in the following format:

### Choice: [/INST]

---

Table 26: Instruction prompt for ablation study on the pairwise ranker.

---

**Distractor Generator Prompt (Ours)**

---

[INST] You are a teacher tasked with creating distractors (plausible wrong options) for a given Multiple Choice Question. Generate distractors according to the guide below:

1) Distractor type:

- Analyze whether the question asks for a 'correct' or 'incorrect' option.
- If the question asks for a correct option, the distractor type should be "Incorrect knowledge"; if it asks for an incorrect option, the distractor type should be "Correct knowledge".

2) Distractors:

- The distractor should be well-formatted so that it fits naturally when presented together with the question and answer.
- If the distractor type is "Incorrect knowledge", the distractor must be an actually incorrect statement; if the distractor type is "Correct knowledge", the distractor must be an actually correct statement.

[Question] { *question* }

[Answer] { *answer* }

Generate { *n* } distractor(s) in the following format:

### Type:

### Distractor n: [/INST]

---

Table 27: Instruction prompt for distractor generator (Ours).

---

**Distractor Generator Prompt (kNN approach by Feng et al. (2024))**

---

Question: { *in-context question* }

Answer: { *in-context answer* }

Distractor1: { *in-context distractor* }

Distractor2: { *in-context distractor* }

Distractor3: { *in-context distractor* }

Question: { *in-context question* }

Answer: { *in-context answer* }

Distractor1: { *in-context distractor* }

Distractor2: { *in-context distractor* }

Distractor3: { *in-context distractor* }

Question: { *in-context question* }

Answer: { *in-context answer* }

Distractor1: { *in-context distractor* }

Distractor2: { *in-context distractor* }

Distractor3: { *in-context distractor* }

Referencing the above samples, generate 3 distractors.

Question: { *question* }

Answer: { *answer* }

Distractor1:

Distractor2:

Distractor3:

---

Table 28: Instruction prompt for distractor generator (kNN approach).

---

**Prompt for Checking Distractor Validity**

---

Check if the given option is the correct choice in a multiple-choice question (MCQ).

1. Check whether the question asks for a 'correct' or 'incorrect' option. If the question asks for a correct option, label "type" as "asking correct option." If the question asks for an incorrect option, label "type" as "asking incorrect option."
2. Insert the given option into the question and analyze whether it is the correct choice.
3. Based on the analysis, if the option is the correct answer to the question, label it as "valid." If it is not the correct answer, label it as "invalid."

[Question] {*question*}

[Option] {*distractor*}

Output according to the following JSON format:

```
{  
  "type": "asking correct option" or "asking incorrect option",  
  "analysis": "your analysis in one sentence",  
  "validity": "valid" or "invalid"  
}
```

---

Table 29: Instruction prompt for checking the validity of distractors.

---

**Prompt for Augmenting Distractors in the Base MCQ Dataset**

---

You are a teacher tasked with creating distractors (plausible wrong options) for a given Multiple Choice Question.

Generate distractors according to the guide below:

1) Distractor type:

- Analyze whether the question asks for a 'correct' or 'incorrect' option.
- If the question asks for a correct option, the distractor type should be "Incorrect knowledge"; if it asks for an incorrect option, the distractor type should be "Correct knowledge".

2) Distractors:

- The distractor should be well-formatted so that it fits naturally when presented together with the question and answer.
- If the distractor type is "Incorrect knowledge", the distractor must be an actually incorrect statement; if the distractor type is "Correct knowledge", the distractor must be an actually correct statement.
- Refer to the original distractors provided.

[Question] {*question*}

[Answer] {*answer*}

[Original Distractors] {*distractors*}

Generate 3 new distractor(s) in the following JSON format:

```
{  
  "type": "Incorrect knowledge" or "Correct knowledge",  
  "distractor_n": "n-th distractor in string type",  
  ...  
}
```

---

Table 30: Instruction prompt for augmenting distractors in the base MCQ dataset.