CDS: Knowledge Component-Driven Data Synthesis Guided by Cognitive Diagnosis Theory

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have achieved significant advancements, but the increasing complexity of tasks and higher performance demands highlight the need for continuous improvement. Some approaches utilize synthetic data generated by advanced LLMs based on evaluation results to train models. However, conventional evaluation methods fail to provide detailed, fine-grained profiles of LLMs, limiting their guidance for data synthesis. In this paper, we introduce the **Cognitive Diagnostic Synthesis** (CDS) method, which incorporates a diagnostic process inspired by **Cognitive Diagnosis Theory** (CDT) to refine evaluation results and characterize model profiles at the knowledge component level. Based on these diagnostics, we propose two diagnosis-synthesis strategies for weakness-targeted data synthesis. Additionally, we present an enhanced data augmentation and selection pipeline to improve the quality and diversity of synthesized data. Our experiments with several open-source models show significant improvements across multiple benchmarks, achieving up to 6.00% improvement in code generation, 13.10% in mathematical reasoning, and 5.43% in academic exams. Code and data are available on GitHub [1].

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across diverse tasks. However, the increasing complexity of emerging tasks and the limitations revealed in real-world applications highlight the critical need for continuous improvement of LLM performance.

To achieve continuous improvement, researchers typically analyze the model's evaluation metrics, then refine or supplement the corpora accordingly for subsequent iterations (Lee et al., 2024; Zhao et al., 2024). For example, if LLMs are found to perform poorly in mathematical tasks, more math data will be deliberately integrated into the dataset for the next training cycle. In this process, advanced LLMs (*e.g.*, GPT-4) are increasingly utilized as data synthesizers to automate training data generation and augmentation (Dai et al., 2023; Liu et al., 2023; Sun et al., 2023), thereby reducing reliance on costly manual annotation.

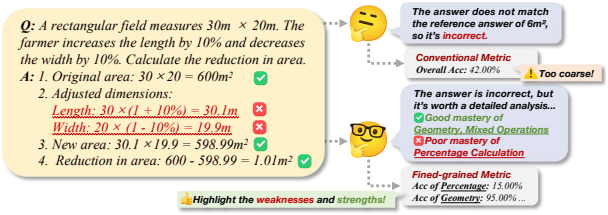

Figure 1: The math problem assesses Geometry, Mixed Operations, and Percentage Calculations. The model performs well overall but makes an error in percentage calculation. Conventional metrics lack the granularity to capture these deficiencies, whereas fine-grained metrics can identify specific strengths and weaknesses at the knowledge level.

However, there are two limitations in this process: 1) **Coarse-Grained Evaluation.** Conventional metrics, such as overall accuracy, focus solely on binary (correct/incorrect) outcomes for each test sample, providing a summary of model performance at the dataset level. Figure 1 highlights the limitation of these metrics: when a model performs well on most of the knowledge assessed but makes a mistake in percentage calculation, leading to an incorrect final result, conventional metrics would simply classify such cases as "incorrect," failing to identify the weakness in percentage calculation. As a result, the lack of granular evaluation limits the attribution of errors to specific sub-skills or competencies, which are referred to as **Knowledge Components** (KCs) in educational theory (Moore et al., 2024), thus hindering the precise identification of weaknesses. 2) The **coarse-grained evaluation limits the guidance for subsequent data synthesis,** leading to gener-

[1] https://anonymous.4open.science/r/cds-04D1

ated data being general and insufficiently targeted at the specific weaknesses of the model. Recent studies attempt to use erroneous questions as seed data for synthesis to align the generated data with the model's weaknesses (Lee et al., 2024; Ying et al., 2024), but they still treat each error in isolation, failing to map and summarize observed mistakes to underlying capability deficiencies. Thus, these methods may correct superficial errors but fail to address fundamental weaknesses at KC level.

To address these limitations, we draw inspiration from **Cognitive Diagnosis Theory** (CDT)—an educational framework that uses "diagnosis" to systematically map assessments to mastery of KCs, identifying specific strengths and weaknesses in students' abilities. In this paper, the LLMs to be enhanced are treated as "student." We apply this diagnostic approach to summarize their performance in evaluations, profiling their capabilities at the KC level. This profiling then guides advanced LLMs to synthesize data aimed at improving the weak KCs in the next training cycle.

Specifically, we introduce the **Cognitive Diagnostic Synthesis** (CDS) method. First, we propose two diagnosis-synthesis strategies from different diagnostic perspectives, using advanced LLMs as data synthesizers: **1) Global Strategy:** Diagnosis at the dataset level with fine-grained metrics such as KC accuracy. These metrics quantify mastery of each KC, helping identify weak mastery KCs and generating tailored training data. **2) Fine-grained Strategy:** Diagnosis at the question level, leveraging the analytical capabilities of advanced LLMs (Bai et al., 2023b; Dai et al., 2023). We use advanced LLMs to perform cognitive diagnosis on specific erroneous cases, identifying KCs requiring remediation. These analyses generated during the diagnostic process are then integrated into the synthesis prompt to expand the length of the chain-of-thought (CoT), as long CoTs enhance generation quality (Jin et al., 2024; Wang et al., 2024).

These synthetic data will undergo augmentation through data rewriting and fusion to enhance their diversity and comprehensiveness. Following this, we propose a **two-stage data selection** process to ensure data quality. In Stage 1, an advanced LLM is used to filter out erroneous data. In Stage 2, a novelty score, $\mathbf{CDS_{score}}$, is designed, which references global diagnosis outcomes to select high-quality, weakness-relevant data.

Our contributions are as follows:

- We introduce the diagnosis process of CDT to refine conventional evaluation, using fine-grained knowledge components to characterize model capabilities.
- We propose two diagnosis-synthesis strategies from different diagnostic perspectives to achieve targeted data synthesis.
- We propose an improved data augmentation and selection pipeline to enhance the quality and diversity of synthesized data. Specifically, we introduce a novelty score, CDS_score, enabling efficient selection of high-quality and relevant data.
- We conduct extensive experiments spanning multiple benchmarks and diverse domains, demonstrating the dominant effectiveness and applicability of CDS.

2 Related Work

2.1 Cognitive Diagnosis Theory

Cognitive Diagnosis Theory (CDT) provides fine-grained assessments by diagnosing an individual's mastery of specific knowledge points, offering actionable insights for targeted interventions (Junker and Sijtsma, 2001; Rupp et al., 2010). CDT focuses on identifying strengths and weaknesses through models such as DINA (De La Torre, 2009) and G-DINA (de la Torre, 2011). These models leverage Q-Matrix Theory (Tatsuoka, 1983) to link test items with underlying knowledge points and provide probabilistic mastery estimates. While CDT integrated with AI has been widely applied in educational assessments (Minn, 2022; Wang et al., 2019; Liu, 2021), its application in data synthesis and model improvement is highly underexplored.

2.2 Synthetic Data for Improving Model

Leveraging advanced LLMs to generate training data has become a widely adopted strategy for improving open-source models (Dai et al., 2023; Xu et al., 2023; Mitra et al., 2024; Wang et al., 2023; Ivison et al., 2023; Chen et al., 2023b; Mitra et al., 2023; Fu et al., 2023; Kumar et al., 2020; Li et al., 2024a, 2023). Concurrently, researchers have investigated generating corrective data through error analysis of target models (An et al., 2023; Lee et al., 2024) and enhancing learning via comparative analysis of positive and negative examples (Ying et al., 2024). Zhang et al. (2024) optimized prompts by extracting reasoning principles from errors, while Liao et al. (2024) analyzed errors in

smaller LMs, storing derived knowledge and summaries in specialized knowledge bases to enhance reasoning performance.

Some studies begin with knowledge-based synthesis, generating knowledge concepts from online course platforms (Huang et al., 2024b), GPT-4 (Li et al., 2024b), and seed instruction analysis and clustering (Huang et al., 2024a), thereby guiding advanced LLMs in data synthesis. However, these approaches have several limitations: simple nominal concepts are inadequate for producing high-quality and diverse synthetic data and may significantly deviate from real-world distributions. Moreover, these methods focus solely on synthesis and overlook the potential of knowledge points to evaluate model weaknesses, thereby limiting the targeting and effectiveness of data synthesis.

3 CDS Method

In the framework shown in Figure 2, we annotate the test samples in the benchmark with their KCs to evaluate the model and diagnose the evaluation results to identify weak KCs, which serve as targets to guide subsequent data synthesis. These synthetic data will undergo augmentation and selection processes to ensure quality, and then be used for supplementary training of the student model. The details are provided as follows.

3.1 Model Evaluation

KC Annotation. For a benchmark $\mathcal{D} = \{d \mid d = (q, a_{\text{ref}})\}$, where each sample d consists of a question q and a reference answer a_{ref}, we allocate the training data to the target dataset, $\mathcal{D}_{\text{target}}$, and reserve the test data for the evaluation dataset, $\mathcal{D}_{\text{eval}}$. An advanced model $\mathcal{M}_{\text{a}}$ is then used to annotate each sample in $\mathcal{D}_{\text{target}}$ with relevant KCs. The annotation process is carried out in two stages.

Stage 1: We use $\mathcal{M}$a to perform **coarse annotations**, with the prompt shown in Figure 3. To ensure an appropriate level of granularity, we sample chapter titles from digital learning platforms, such as MOOCs, to provide examples of KCs like Probability. Indeed, subsequent experiments will demonstrate the flexibility of KC annotations. We aggregate the KC tags from each sample to build an initial set. This set is then refined by $\mathcal{M}_{\text{a}}$, with optional expert involvement, to eliminate redundancies and ensure that the KCs are mutually exclusive, collectively exhaustive, and appropriately granular. The refined set of KCs is denoted as $\mathcal{K}$.

Stage 2: We use $\mathcal{M}_{\text{a}}$ to perform **constrained annotations** to ensure that the tagged KCs originate from $\mathcal{K}$, yielding the tagged benchmark $\mathcal{D}^*_{\text{target}} = \{d^* | d^* = (q, a_{\text{ref}}, \mathcal{K}_{\text{q}}), \mathcal{K}_{\text{q}} \subsetneq \mathcal{K}\}$. Thereby, we build the Question-Knowledge Component (Q-KC) matrix, represented as:

$$Q\text{-}KC \in \{0,1\}^{|\mathcal{D}_{\text{target}}| \times |\mathcal{K}|} \tag{1}$$

$$Q\text{-}KC[i,j] = \begin{cases} 1 & \text{if } kc_j \in \mathcal{K}_{\text{q}_i}, kc_j \in \mathcal{K}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

where $|\mathcal{D}_{\text{target}}|$ is the number of benchmark questions, $|\mathcal{K}|$ is the cardinality of the KC set, and $\mathcal{K}_{\text{q}_i}$ denotes the KC set tagged to question q_i, which is a subset of $\mathcal{K}$.

Model Evaluation and Result Collection. We evaluate the student model $\mathcal{M}_{\text{s}}$ using the tagged benchmark $\mathcal{D}^*_{\text{target}}$. To better reveal the model's KC deficiencies, we collect **erroneous cases**, denoted as $\mathcal{D}_{\text{err}} = \{(q, r_{\text{err}}, \mathcal{K}_{\text{q}})\}$, for subsequent diagnosis, where r_{err} is the response from $\mathcal{M}_{\text{s}}$ to question q that does not match the reference answer a_{ref}.

3.2 Diagnosis and Data Synthesis

Diagnosis is defined as the process of analyzing KC mastery profiles derived from evaluation results. To holistically assess the model's proficiency in KCs, we propose two diagnosis-synthesis strategies from different diagnostic perspectives.

Global Strategy. We diagnose the model's performance from a global perspective using aggregated KC metrics at the dataset level. Specifically, we follow the *Deterministic Input, Noise And* (DINA) cognitive diagnosis framework (De La Torre, 2009) that posits a binary mastery assumption:

- A correct response to question q_i implies mastery of **all** associated KCs
- An incorrect response implies **no mastery** of **any** associated KCs

Under these assumptions, we compute KC-specific accuracy and frequency metrics:

$$\text{Acc}(kc_j) = \frac{\sum_{i=1}^{|\mathcal{D}_{\text{target}}|} \mathbb{I}(q_i) \cdot Q\text{-}KC[i,j]}{\sum_{i=1}^{|\mathcal{D}_{\text{target}}|} Q\text{-}KC[i,j]} \tag{3}$$

$$\text{Freq}(kc_j) = \frac{\sum_{i=1}^{|\mathcal{D}_{\text{target}}|} Q\text{-}KC[i,j]}{|\mathcal{D}_{\text{target}}|} \tag{4}$$

where $\mathbb{I}(q_i) \in \{0,1\}$ indicates the correctness of $\mathcal{M}_{\text{s}}$'s response to q_i.

Based on these metrics, we construct a global KC diagnostic profile for $\mathcal{M}_{\text{s}}$, which consists of the accuracy and frequency of each KC, identifying

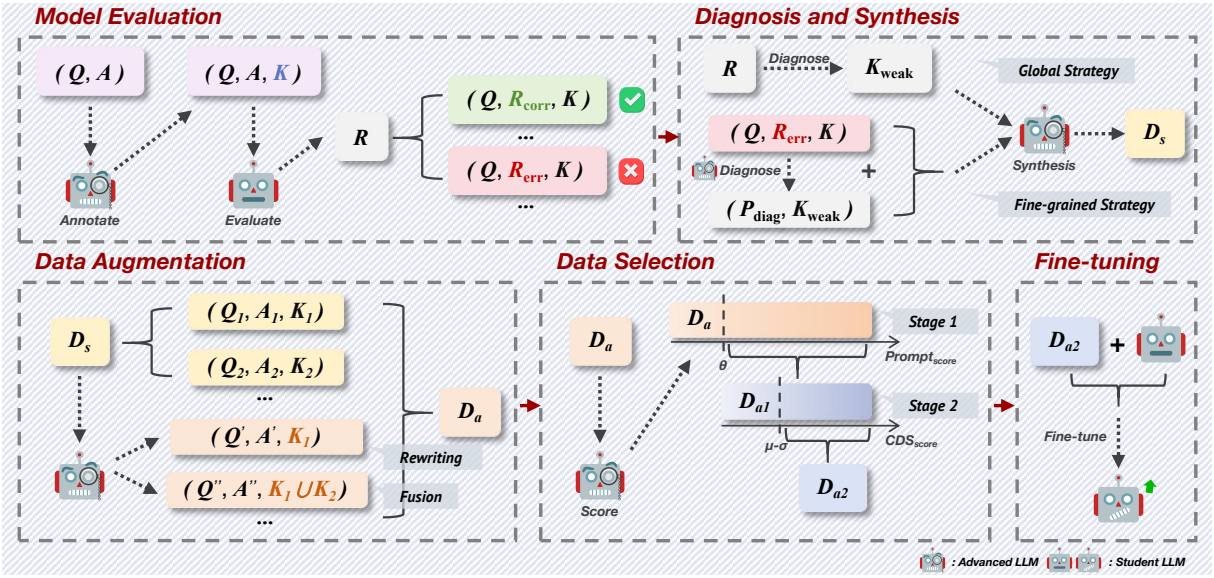

Figure 2: The pipeline of CDS method.

weakly mastered KCs as those with low accuracy or low frequency, denoted as $\mathcal{K}_{\text{w}}$. To address these weaknesses, we use $\mathcal{M}_{\text{a}}$ to generate data targeted at these weaknesses, based on $\mathcal{K}_{\text{w}}$. The process is detailed in Algorithm 1.

Algorithm 1 Global Strategy

Require: A set of KCs $\mathcal{K}$, an advanced model $\mathcal{M}_{\text{a}}$, accuracy threshold δ_{a}, frequency threshold δ_{f}
Ensure: A synthesized dataset $\mathcal{D}_{\text{global}}$

$\mathcal{K}_{\text{w}} \leftarrow \emptyset$
for each $kc \in \mathcal{K}$ **do**
  if $\text{Acc}(kc) \leq \delta_{\text{a}}$ **or** $\text{Freq}(kc) \leq \delta_{\text{f}}$ **then**
    $\mathcal{K}_{\text{w}} \leftarrow \mathcal{K}_{\text{w}} \cup \{kc\}$
  end if
end for
$\mathcal{D}_{\text{global}} \leftarrow \emptyset$
for each $kc \in \mathcal{K}_{\text{w}}$ **do**
  $(q,\ a,\ \mathcal{K}_q = \{kc\}) \leftarrow \text{Generate}(\mathcal{M}_{\text{a}},\ kc)$
  $\mathcal{D}_{\text{global}} \leftarrow \mathcal{D}_{\text{global}} \cup \{(q,\ a,\ \mathcal{K}_q)\}$
end for
return $\mathcal{D}_{\text{global}}$

This strategy operates exclusively at the KC level, avoiding the introduction of original questions into synthetic prompts. By summarizing specific questions into KCs, we address a key limitation of traditional example-question-based synthesis methods: when prompted with original questions, the model tends to unconsciously rewrite or rephrase them. This ensures that the generated data is both novel and independent, free from overfitting to the original dataset. Detailed prompts and case studies are provided in the Appendix.

Fine-grained Strategy. We diagnose the model's **erroneous cases** at the question level from a fine-grained perspective. We use $\mathcal{M}_{\text{a}}$ as a diagnoser to analyze the model's problem-solving process in the current case, thereby identifying the underlying weak KCs exposed by this case, which are also denoted as $\mathcal{K}_{\text{w}}$. These analytical processes generated during the diagnosis are denoted as p_{diag}, and are integrated with the original questions and erroneous responses, forming long CoTs within the prompt for the advanced LLM to stimulate deeper reasoning, leading to higher-quality generation results. The process is detailed in Algorithm 2.

Algorithm 2 Fine-grained Strategy

Require: A set of erroneous responses $\mathcal{D}_{\text{err}} = \{\, d_{\text{err}} \mid d_{\text{err}} = (q, r_{\text{err}}, \mathcal{K}_q)\}$, an advanced model $\mathcal{M}_{\text{a}}$
Ensure: A synthesized dataset $\mathcal{D}_{\text{fine-grained}}$

$\mathcal{O}_{\text{diag}} \leftarrow \emptyset$ {$\mathcal{O}_{\text{diag}}$ stores erroneous cases and their corresponding diagnostic outputs.}
for each $d_{\text{err}} \in \mathcal{D}_{\text{err}}$ **do**
  $(p_{\text{diag}}, \mathcal{K}_{\text{w}}) \leftarrow \text{Generate}\big(\mathcal{M}_{\text{a}},\ d_{\text{err}}\big)$
  $\mathcal{O}_{\text{diag}} \leftarrow \mathcal{O}_{\text{diag}} \cup \big\{(d_{\text{err}},\ p_{\text{diag}},\ \mathcal{K}_{\text{w}})\big\}$
end for
$\mathcal{D}_{\text{fine-grained}} \leftarrow \emptyset$
for each $(d_{\text{err}},\ p_{\text{diag}},\ \mathcal{K}_{\text{w}}) \in \mathcal{O}_{\text{diag}}$ **do**
  $(q',\ a',\ \mathcal{K}_{\text{w}}) \leftarrow \text{Generate}\big(\mathcal{M}_{\text{a}},\ d_{\text{err}},\ p_{\text{diag}},\ \mathcal{K}_{\text{w}}\big)$
  $\mathcal{D}_{\text{fine-grained}} \leftarrow \mathcal{D}_{\text{fine-grained}} \cup \{(q', a', \mathcal{K}_{\text{w}})\}$
end for
return $\mathcal{D}_{\text{fine-grained}}$

This diagnosis-synthesis paradigm leverages LLMs' analytical capabilities beyond data generation (Bai et al., 2023b; Dai et al., 2023). Recent studies show that long CoTs guide advanced LLMs to deeper reasoning, generating higher-quality outputs (Jin et al., 2024; Wang et al., 2024). By integrating diagnostic processes into data synthesis, our strategy produces more targeted and higher-quality data than direct synthesis methods. Detailed

prompts and case studies are in the Appendix.

Data Augmentation. We concatenate the data generated by two synthesis methods and employ the following augmentation strategies to further increase data diversity and volume:

- **KC-Constrained Rewriting:** Adapting traditional data rewriting methods (Dai et al., 2023; Sun et al., 2023), we add a constraint: the rewritten data should contain the same KCs as the originals, avoiding deviation from targeted weaknesses while enhancing diversity.
- **Multi-KC Fusion:** We pair data from the synthetic dataset and prompt the advanced LLM to generate new data containing KCs from both, thereby increasing data complexity and comprehensiveness.

We sample a small proportion of the data for augmentation and reintegrate the augmented samples into the dataset. Additionally, we limit the max number of KCs per data to prevent the generation of overly complex or ambiguous samples.

3.3 Data Selection

We implement a two-stage data selection process to refine augmented synthetic dataset $\mathcal{D}_\text{a}$, eliminating subpar samples and retaining those that meet high-quality and high-relevance standards.

Stage 1: $\mathcal{M}_\text{a}$ assigns scores to the data based on multiple criteria such as correctness and KC relevance, filtering out samples below a threshold.

Stage 2: Simply scoring data individually with the model does not leverage global KC diagnostic profiles to select weakness-targeted data. Thus, we introduce a novel metric, CDS_score. We hypothesize that data with more KCs have higher complexity and comprehensiveness, while data containing low-frequency and low-accuracy KCs are more effective for targeting weaknesses. Based on these assumptions, specifically, for $d_\text{a} = (q, a, \mathcal{K}_q) \in \mathcal{D}_\text{a}$, the CDS_score is calculated as follows:

$$\mathcal{V}(kc_j) = w_1 \log(\text{Acc}(kc_j) + \epsilon) + w_2 \log(\text{Freq}_a(kc_j) + \epsilon) \tag{5}$$

$$CDS_\text{score}(d_a) = \sum_{kc_j \in \mathcal{K}_q} \mathcal{V}(kc_j) \tag{6}$$

where $\mathcal{V}(\cdot)$ represents the significance of KC, $\text{Acc}(kc_j)$ denotes the student LLM's initial accuracy on kc_j, $\text{Freq}_s(kc_j)$ is the frequency of kc_j in $\mathcal{D}_\text{a}$, w_1 and w_2 are balancing weights, and ϵ is a small constant to avoid division by zero.

We apply a 1-σ principle, retaining samples with $\text{CDS}_\text{score}(d_i) > \mu - \sigma$, to construct the final training set for fine-tuning the student LLM.

4 Experimental Setup

4.1 Datasets and Models

Datasets. We evaluate three primary tasks: mathematical reasoning, coding, and academic examination. To validate the effectiveness of CDS, we select GSM8k (Cobbe et al., 2021), MBPP (Austin et al., 2021), and GAOKAO-Bench (Zhang et al., 2023) for each respective task. As described in Section 3.1, these benchmarks are split into $\mathcal{D}_\text{target}$ and $\mathcal{D}_\text{eval}$. Due to the similar distribution of In-Domain(ID) training and test sets, we also incorporate an Out-of-Domain(OOD) dataset in $\mathcal{D}_\text{eval}$ to better assess the generalization of CDS. The OOD datasets include GSM8k-PLUS (Li et al., 2024c), HumanEval (Chen et al., 2021), and GAOKAO-Bench-Updates[2].

For the academic examination benchmark's KC annotation, we use chapter titles from the GAOKAO syllabus[3] as KCs, offering an alternative method distinct from Section 3.1.

Models. We use Llama3-8B-Instruct(AI@Meta, 2024) and Qwen1.5-7B-Chat(Bai et al., 2023a) as the student LLMs, with Qwen2-72B-Instruct(Yang et al., 2024) serving as the advanced LLM.

4.2 Setups

Training Setup. We train the models on 1 NVIDIA A800 GPU using ZeRO Stage 1 (Rajbhandari et al., 2020) and AdamW (Kingma and Ba, 2015) as the optimizer, with LoRA (Hu et al., 2021) (rank $r = 8$). The batch size is 32, the maximum sequence length is 2,048, and training runs for 1 epoch.

Inference Setup. For code generation, mathematical reasoning, and academic exams, we use greedy decoding with a maximum output of 512 tokens. For data generation, we use a temperature of 0.5, top-p of 0.8, and a maximum output of 4096 tokens. All inference is conducted in a 0-shot setting. See Appendix D for more details.

4.3 Baselines

Main Experiments. We consider several baselines for comparison with our method as follows:

[2]https://github.com/OpenLMLab/GAOKAO-Bench-Updates

[3]https://gaokao.neea.edu.cn/html1/category/1509/6212-1.htm

Method	Coding		Math		Examination		Avg
	MBPP	H-Eval	GSM8k	GSMPlus	GAOKAO	GAOKAO_{U}	–
	P@1	P@1	Acc	Acc	Acc	Acc	
Qwen1.5-7B-Chat							
Prompt(vanilla)	32.00	40.24	54.00	33.92	60.60	48.87	44.94
IFT	30.80	39.02	52.42	33.72	61.40	47.96	44.22
LEC	32.40	39.63	52.20	33.76	**64.80**	**53.39**	46.03
AugGPT	**34.80**	40.85	46.38	28.86	63.40	52.49	44.46
LLM2LLM	34.40	**43.90**	53.82	33.76	64.40	**53.39**	**47.28**
MUSTARD	35.40	39.02	**57.42**	**35.96**	62.40	52.04	47.04
CDS(our)	**38.00**	**44.51**	**64.54**	**43.86**	**64.60**	**53.85**	**51.56**
Llama3-8B-Instruct							
Prompt(vanilla)	40.80	54.88	62.02	42.50	41.20	33.03	45.74
IFT	41.60	**55.49**	61.64	**42.78**	41.00	31.67	45.70
LEC	**42.80**	54.88	55.62	42.48	**41.60**	32.58	44.99
AugGPT	40.00	51.83	47.32	36.00	**42.20**	**35.28**	42.10
LLM2LLM	**42.80**	**56.10**	55.76	41.46	**41.60**	31.67	44.90
MUSTARD	41.40	54.88	**62.16**	42.44	41.20	33.03	**45.85**
CDS(our)	**42.80**	**55.49**	**73.14**	**55.60**	41.40	**38.46**	**51.15**

Table 1: The main experimental results of our methods and baseline approaches across various tasks are presented. Experiments are conducted using two different LLMs: Qwen1.5-7B-Chat and Llama3-8B-Instruct. The top two performances are highlighted in **red bold** and **black bold**, respectively.

(1) Prompt: Direct prompting for answers. *(2) IFT*: Fine-tuning with in-domain training data. *(3) LEC* (Ying et al., 2024): Embedding erroneous cases with SentenceBERT (Reimers and Gurevych, 2019), selecting similar positive examples via L2 distance, and synthesizing with both positive and negative cases. *(4) AugGPT* (Dai et al., 2023): Sampling unused instructions from the in-domain training set for synthetic data generation with the advanced LLM. *(5) LLM2LLM* (Lee et al., 2024): Generating additional data from incorrect examples using the advanced LLM. *(6) MUSTARD* (Huang et al., 2024b): Generating questions from seed concepts, followed by advanced LLM-generated answers and correctness filtering. Consistent data quantity is maintained across baselines: 2k for mathematics, 0.5k for code generation, and 0.5k for academic examination.

Data Selection Experiments. We evaluate our data selection algorithm against several baselines as follows: *(1) CBS* (Chen et al., 2023a): Instructions are embedded using SentenceBERT, clustered with HDBSCAN (Campello et al., 2013), and selected using the K-Center-Greedy algorithm. *(2) Coreset* (Sener and Savarese, 2017): Similar to CBS, with instructions embedded using SentenceBERT and selected using K-Center-Greedy. *(3) Diversity* (Wang et al., 2022): For each data, the ROUGE score is computed against a subset of n samples, and the k with the lowest ROUGE scores are selected. *(4) Length*: Samples are selected based on data length, focusing on the longest instances ($\text{Length}_{\text{long}}$). *(5) Perplexity* (Marion et al., 2023): Samples are selected based on low per-token perplexity, indicating model certainty and fluency. *(6) AlpaGasus* (Chen et al., 2024): Instances are scored by an advanced LLM like ChatGPT on dimensions such as helpfulness and accuracy, and low-scoring instances are filtered out. *(7) Random*: Instances are randomly selected from the dataset.

5 Experiments

5.1 Main Results

The main experimental results of our methods and baseline approaches across various tasks are presented in Table 1. Our observations are summarized as follows:

Dominant effectiveness and applicability of CDS. CDS demonstrates significant improvements across different models and tasks. For example, on the GSM8k task, Qwen1.5-7B improves by

Method	Coding							Math						
	MBPP			H-Eval			Avg	GSM8k			GSMPlus			Avg
	0.1k	0.2k	0.3k	0.1k	0.2k	0.3k	–	0.4k	0.8k	1.6k	0.4k	0.8k	1.6k	–
CBS	25.60	33.80	33.40	**43.29**	**38.41**	37.20	35.28	41.80	**62.00**	60.94	27.08	**40.40**	40.24	45.41
CoreSet	25.60	33.40	35.60	40.85	34.15	39.63	34.87	50.76	59.70	60.06	34.10	38.14	38.66	46.90
Diversity	26.60	34.40	35.40	**42.68**	32.32	**42.07**	35.58	43.28	56.14	61.64	28.54	35.08	39.80	44.08
$\text{Length}_{\text{long}}$	23.80	33.20	33.20	42.07	34.15	40.85	34.55	36.96	59.52	61.02	24.10	38.40	39.46	43.24
Perplexity	**31.80**	**35.00**	35.00	**42.68**	**37.20**	41.46	**37.19**	**53.04**	60.80	**62.52**	**35.32**	40.18	**41.56**	**48.90**
AlpaGasus	30.60	34.20	35.40	40.24	35.37	40.85	36.11	49.44	**62.34**	56.82	30.70	**40.20**	37.32	46.14
Random	24.60	34.00	35.20	40.24	34.15	39.63	34.64	45.68	58.38	59.32	33.76	38.70	38.86	45.78
$\text{CDS}_{\text{score}}$(our)	**31.60**	**34.60**	**36.20**	40.85	37.20	**43.29**	**37.29**	**53.80**	60.80	**62.56**	**36.02**	39.96	**41.24**	**49.06**

Table 2: The experimental results of our data selection strategy and baseline approaches across various tasks are presented. Experiments are conducted using Qwen1.5-7B-Chat. The top two performances are highlighted in **red bold** and **black bold**, respectively.

10.54%, and Llama3-8B by 11.12%. Additionally, CDS consistently produces optimal results in coding and examination tasks(including subjects such as biology, chemistry, geography, history, mathematics, and physics), highlighting its effectiveness and broad applicability to tasks that can be decomposed into well-defined KCs.

Strong generalization. Although CDS utilizes synthetic data generated based on ID tasks, it generalizes effectively to OOD tasks. For example, Qwen1.5-7B shows a 4.27% improvement on the Humaneval Bench, while Llama3-8B improves by 5.43% on GAOKAO-Bench-Updates. In contrast, methods like AugGPT and LEC fail to consistently improve performance on OOD tasks and may even lead to degradation.

Flexibility in KC Annotation. Section 4.1 explains that GAOKAO-Bench constructs the KC set for annotation by summarizing chapter titles, while the MBPP and GSM8k benchmarks use the annotation approach described in Section 3.1. Despite these different annotation methods, all benchmarks show consistent performance improvements with CDS.

CDS's data selection improves robustness. Unfiltered use of synthetic data may lead to model degradation. For instance, both LEC and AugGPT experienced performance declines on two mathematical benchmarks, with AugGPT showing a 7.62% decrease on GSM8k and 5.06% on GSMPlus. This aligns with prior research, which suggests that unchecked, low-quality instructional data can impair model performance (Zhou et al., 2023; Chen et al., 2023b). Table 5 shows an erroneous sample generated by AugGPT, which can be filtered out during CDS's selection Stage 1.

5.2 Evaluation of Selection Strategies

Experimental results comparing different data selection strategies are presented in Table 2, where we fine-tuned Qwen1.5-7B-Chat using samples selected from an augmented synthetic dataset (prescreened by Qwen2-72B-Instruct for basic correctness). To evaluate the effectiveness of the CDS score, we retained only Stage 2 data selection and excluded Stage 1, which is similar to AlpaGasus. Our key findings are:

High-quality data selection and broad task applicability. $\text{CDS}_{\text{score}}$ achieves the best average metrics on both math and coding tasks. Specifically, in 12 tests corresponding to three different sample sizes across four datasets, 8 tests ranked in the top 2. $\text{CDS}_{\text{score}}$ consistently outperforms the Random method in all scenarios, with average improvements of 2.65% and 3.28% for the two tasks, respectively. This demonstrates that $\text{CDS}_{\text{score}}$ enhances training data quality through selection and exhibits broad task applicability.

Strong stability. $\text{CDS}_{\text{score}}$ shows consistent performance across datasets and sample sizes, while some baselines exhibit fluctuating performance. For example, CBS's performance varied significantly with sample size, and Coreset performed well on math tasks but struggled with coding tasks.

Scalability. $\text{CDS}_{\text{score}}$ expands its advantage as sample sizes increase, demonstrating optimal accuracy with 0.3k samples for MBPP and H-Eval, and 1.6k samples for GSM8K. Notably, at 0.3k samples on Humaneval, it outperformed the second-best

Strategy	Coding		Math		Examination		Avg
	MBPP	H-Eval	GSM8k	GSMPlus	GAOKAO	GAOKAO_{U}	–
	P@1	P@1	Acc	Acc	Acc	Acc	
Prompt(vanilla)	32.00	40.24	54.00	33.92	60.60	48.87	44.94
Synthesis Strategy							
+ Global	32.00	**40.85**	57.96	36.22	**62.00**	**52.04**	46.85
+ Fine-grained	**34.00**	39.63	**60.96**	**37.62**	60.80	49.77	**47.13**
+ Global & Fine-grained	**35.40**	**40.24**	**61.84**	**38.60**	**61.40**	**52.49**	**48.33**
Augmentation Strategy							
+ Rewrite	**35.40**	41.46	**61.52**	**41.38**	62.60	51.13	48.91
+ Fusion	35.20	41.46	61.00	40.42	**63.40**	**52.49**	**48.99**
+ Rewrite & Fusion	**35.40**	**43.29**	**63.78**	**42.04**	**64.60**	**53.85**	**50.49**

Table 3: Ablation results of data synthesis and augmentation strategies. Experiments were conducted using Qwen1.5-7B-Chat. Data augmentation was applied to the synthetic data generated by the '+ Global & Fine-grained' approach. The top two performances are highlighted in **red bold** and **black bold**, respectively. The sample sizes for the synthesis strategy are 300, 300, and 600 for coding; 300, 1000, and 1300 for math; and 300, 300, and 600 for exams. The augmentation proportions are 0.5, 0.5, 0.25 & 0.25.

methods, Diversity and Perplexity, by 1.22% and 1.83%, respectively.

Computational Efficiency. As shown in Table 6, compared to the suboptimal Perplexity method when selecting 2,000 samples from a 3,000-sample dataset, Perplexity requires 307.78 seconds, while $\text{CDS}_{\text{score}}$ takes negligible time. This efficiency results from the elimination of computationally intensive tasks such as embedding generation and clustering, and without the need for GPU.

5.3 Ablation Study

To evaluate the effectiveness of each component of the CDS method, we conducted an ablation study using various combinations of data synthesis and augmentation strategies. A basic correctness check was performed using Qwen-72B-Instruct to filter out obviously incorrect data. The results are shown in Table 3. Our key findings are:

Dual Strategies Outperform Single Strategies. Dual strategies generally outperform single strategies in both the data synthesis and augmentation stages. This effect is particularly noticeable in augmentation, where combining Rewriting and Fusion strategies resulted in optimal performance across all six datasets.

Data Augmentation Effect. Combining synthesis and augmentation typically improves performance. After applying augmentation, the average performance exceeded previous results. However, exceptions exist, such as with Rewriting, where performance on GAOKAO-UPDATES deteriorated by 1.36%. This may be due to overfitting to similar synthesis data. Such degradation was not observed with dual strategy augmentation, suggesting that combining multiple strategies to increase data complexity and diversity improves robustness.

6 Conclusion

In this paper, we introduce the **Cognitive Diagnostic Synthesis** (CDS) method, inspired by Cognitive Diagnosis Theory, which refines evaluation results and characterizes model profiles at the knowledge component level. Building on the Knowledge Component diagnostic construct, we optimize conventional evaluation metrics, design dual diagnosis-synthesis strategies, and propose a novel data selection method. Leveraging advanced LLMs, we automate the generation of weakness-targeted, high-quality instructional data. We applied the synthesized data to small LLMs, such as Qwen1.5-7B-Chat and Llama3-8B-Instruct, achieving significant improvements in code generation, mathematical reasoning, and academic testing. Notably, CDS achieves these improvements without relying on expensive closed-source LLMs like GPT-4, instead using only the open-source Qwen2-72B-Chat for automated diagnosis and synthesis.

7 Limitations

In this paper, (1) due to cost limitations, our advanced LLM selection is restricted to the open-source Qwen-72B-Instruct, which is not the most cutting-edge model available. Given the current

limitations in both the model's analytical and generative capabilities, the full potential of CDS remains to be explored. In future work, we plan to experiment with more advanced models, such as GPT-4(Achiam et al., 2023) and DeepSeek-V3(DeepSeek-AI et al., 2024).

(2) The identification and annotation of KCs still involve significant randomness and subjectivity. In future work, we aim to refine our KC annotation strategy, including exploring the use of pseudo-labels as a substitute for explicit labels.

References

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haim ing Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Made laine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Benjamin Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Sim'on Posada Fishman, Juston Forte, Is abella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Raphael Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Jo hannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Lukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Ryan Kiros, Matthew Knight, Daniel Kokotajlo, Lukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Ma teusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel P. Mossing, Tong Mu, Mira Murati, Oleg Murk, David M'ely, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Ouyang Long, Cullen O'Keefe, Jakub W. Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alexandre Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Pondé de Oliveira Pinto, Michael Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack W. Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario D. Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas A. Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cer'on Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll L. Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qim ing Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. Gpt-4 technical report.

AI@Meta. 2024. Llama 3 model card.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2023. Learning from mistakes makes llm better reasoner. *ArXiv*, abs/2310.20689.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021. Program synthesis with large language models. *ArXiv*, abs/2108.07732.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenhang Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, K. Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Yu Bowen, Hongyi Yuan, Zheng Yuan, Jianwei

Zhang, Xing Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023a. Qwen technical report. *ArXiv*, abs/2309.16609.

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, Jiayin Zhang, Juanzi Li, and Lei Hou. 2023b. Benchmarking foundation models with language-model-as-an-examiner. *ArXiv*, abs/2306.04181.

Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Haowen Chen, Yiming Zhang, Qi Zhang, Hantao Yang, Xiaomeng Hu, Xuetao Ma, Yifan YangGong, and Junbo Jake Zhao. 2023a. Maybe only 0.5% data is needed: A preliminary exploration of low training data instruction tuning. *ArXiv*, abs/2305.09246.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2024. Alpagasus: Training a better alpaca with fewer data. *Preprint*, arXiv:2307.08701.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. 2023b. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, Suchir Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168.

Haixing Dai, Zheng Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, W. Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. 2023. Auggpt: Leveraging chatgpt for text data augmentation.

Jimmy De La Torre. 2009. Dina model and parameter estimation: A didactic. *Journal of educational and behavioral statistics*, 34(1):115–130.

Jimmy de la Torre. 2011. The generalized dina model framework. *Psychometrika*, 76:179–199.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Yao Fu, Hao-Chun Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller lan-

guage models towards multi-step reasoning. *ArXiv*, abs/2301.12726.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Yiming Huang, Xiao Liu, Yeyun Gong, Zhibin Gou, Yelong Shen, Nan Duan, and Weizhu Chen. 2024a. Key-point-driven data synthesis with its enhancement on mathematical reasoning. *Preprint*, arXiv:2403.02333.

Yinya Huang, Xiaohan Lin, Zhengying Liu, Qingxing Cao, Huajian Xin, Haiming Wang, Zhenguo Li, Linqi Song, and Xiaodan Liang. 2024b. Mustard: Mastering uniform synthesis of theorem and proof data. *ArXiv*, abs/2402.08957.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew E. Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hanna Hajishirzi. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *ArXiv*, abs/2311.10702.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. *ArXiv*, abs/2401.04925.

Brian W. Junker and Klaas Sijtsma. 2001. Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement*, 25:258 – 272.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. *ArXiv*, abs/2003.02245.

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Krishna Anumanchipalli, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2024. Llm2llm: Boosting llms with novel iterative data enhancement. In *Annual Meeting of the Association for Computational Linguistics*.

Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, Han Hu, Zheng Zhang, and Houwen Peng. 2024a. Common 7b language models already possess strong math capabilities. *ArXiv*, abs/2403.04706.

Haoran Li, Qingxiu Dong, Zhengyang Tang, Chaojun Wang, Xingxing Zhang, Haoyang Huang, Shaohan Huang, Xiaolong Huang, Zeqiang Huang, Dongdong Zhang, Yuxian Gu, Xin Cheng, Xun Wang, Si-Qing Chen, Li Dong, Wei Lu, Zhifang Sui, Benyou Wang, Wai Lam, and Furu Wei. 2024b. Synthetic data (almost) from scratch: Generalized instruction tuning for language models. *ArXiv*, abs/2402.13064.

Haoran Li, Yiran Liu, Xingxing Zhang, Wei Lu, and Furu Wei. 2023. Tuna: Instruction tuning using feedback from large language models. In *Conference on Empirical Methods in Natural Language Processing*.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024c. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. In *Annual Meeting of the Association for Computational Linguistics*.

Huanxuan Liao, Shizhu He, Yao Xu, Yuanzhe Zhang, Kang Liu, and Jun Zhao. 2024. Neural-symbolic collaborative distillation: Advancing small language models for complex reasoning tasks. *ArXiv*, abs/2409.13203.

Hanmeng Liu, Zhiyang Teng, Leyang Cui, Chaoli Zhang, Qiji Zhou, and Yue Zhang. 2023. Logicot: Logical chain-of-thought instruction-tuning data collection with gpt-4. In *Conference on Empirical Methods in Natural Language Processing*.

Qi Liu. 2021. Towards a new generation of cognitive diagnosis. In *IJCAI*, pages 4961–4964.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When less is more: Investigating data pruning for pretraining llms at scale. *Preprint*, arXiv:2309.04564.

Sein Minn. 2022. Ai-assisted knowledge assessment techniques for adaptive learning environments. *Comput. Educ. Artif. Intell.*, 3:100050.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agrawal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. Orca 2: Teaching small language models how to reason. *ArXiv*, abs/2311.11045.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. Orca-math: Unlocking the potential of slms in grade school math. *ArXiv*, abs/2402.14830.

Steven Moore, Robin Schmucker, Tom Mitchell, and John C. Stamper. 2024. Automated generation and tagging of knowledge components from multiple-choice questions. *ArXiv*, abs/2405.20526.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Conference on Empirical Methods in Natural Language Processing*.

Andre A. Rupp, Jonathan L Templin, and Robert Henson. 2010. Diagnostic measurement: Theory, methods, and applications.

Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv: Machine Learning*.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David D. Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. *ArXiv*, abs/2305.03047.

Kikumi K. Tatsuoka. 1983. Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, 20:345–354.

Fei Wang, Qi Liu, Enhong Chen, Zhenya Huang, Yuying Chen, Yu Yin, Zai Huang, and Shijin Wang. 2019. Neural cognitive diagnosis for intelligent education systems. In *AAAI Conference on Artificial Intelligence*.

Jiaan Wang, Fandong Meng, Yunlong Liang, and Jie Zhou. 2024. Drt-o1: Optimized deep reasoning translation via long chain-of-thought. *ArXiv*, abs/2412.17498.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *ArXiv*, abs/2306.04751.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. In *Annual Meeting of the Association for Computational Linguistics*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *ArXiv*, abs/2304.12244.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. 2024. Qwen2 technical report. *ArXiv*, abs/2407.10671.

Jiahao Ying, Mingbao Lin, Yixin Cao, Wei Tang, Bo Wang, Qianru Sun, Xuanjing Huang, and Shuicheng Yan. 2024. Llms-as-instructors: Learning from errors toward automating model improvement. *ArXiv*, abs/2407.00497.

Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. 2024. In-context principle learning from mistakes. *ArXiv*, abs/2402.05403.

Xiaotian Zhang, Chun yan Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. Evaluating the performance of large language models on gaokao benchmark. *ArXiv*, abs/2305.12474.

Haokun Zhao, Haixia Han, Jie Shi, Chengyu Du, Jiaqing Liang, and Yanghua Xiao. 2024. Cem: A data-efficient method for large language models to continue evolving from mistakes.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, L. Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. *ArXiv*, abs/2305.11206.

A Knowledge Component Details

The following knowledge components (KCs) shown in table 4 are used for various tasks:

B Case Study

C Computational Cost of Data Selection Strategies

As shown in Table 6, we compared the time efficiency of different data selection methods when selecting 2,000 samples from a pool of 3,000. The key parameters and computational requirements are as follows:

- **Model**: For embedding generation, we used the Qwen1.5-7B-Chat.
- **GPU Requirements**:
  - AlpaGasus requires 4 A800 GPUs and uses Qwen2-72B-Chat for inference.
  - Other methods, including CBS, CoreSet, Diversity, and Perplexity, only require 1 A800 GPU for processing.
- **Clustering Parameters**:

Task	Knowledge Components
Coding	Basic Data Types, Bitwise Operations, Boolean Logic, Class Definitions, Comparison Operators, Conditional Statements, Copying and Deep Copying, Dictionary Operations, Dynamic Programming, Exception Handling, Finding Min and Max, Heap Operations, Importing Libraries and Modules, Indexing and Slicing, Lambda Functions, List and Array Operations, Looping, Map Function, Recursion, Regular Expressions, Search Algorithms, Sorting Algorithms, Stacks and Queues, String Operations, Summation, Tree Structures, Tuple Operations, Type Checking and Conversion
Math	Basic Arithmetic Operations, Decimal and Fraction Operations, Mixed Operations, Prime and Composite Numbers, Factors and Multiples, GCD and LCM, Algebraic Expressions, Equations, Inequalities, Basic Geometry, Area, Perimeter, Volume, Angles, Coordinates, Mean, Median, Mode, Probability, Permutations, Combinations, Financial Calculations, Unit Conversion, Time and Date Calculations, Speed, Distance, and Time, Measurement, Money, Ratio and Proportion, Bar Graphs, Line Graphs, Number Sequences, Word Problems, Linear Equations, Simple Algebra, Pattern Recognition, Mathematical Logic, Shapes and Spatial Understanding, Symmetry, Congruence, Units of Measurement, Temperature, Length, Mass, Capacity
Exams	**Biology**: Protein and Nucleic Acid Structure, Sugar and Lipid Types and Functions, Water and Inorganic Salts, Cell Theory, Prokaryotic and Eukaryotic Cells, Cell Membrane Structure, Organelles Structure and Function, Nucleus Structure and Function, Substance Transport Across Cell Membrane, Enzyme Role in Metabolism, ATP Metabolism, Photosynthesis Process, Environmental Impact on Photosynthesis, Cellular Respiration, Cell Growth and Division, Cell Differentiation, Cell Aging and Apoptosis, Cancer Cells and Prevention, Meiotic Division, DNA Structure and Replication, Gene Transcription and Translation, Mendel's Laws, Sex-linked Inheritance, Gene Mutation, Transgenic Food Safety, Human Genetic Diseases, Evolution Theory, Plant Hormones, Nervous and Hormonal Regulation, Nerve Impulse Transmission, Homeostasis, Immune System Role, Population and Community, Ecosystem Structure and Function, Ecosystem Stability, Biodiversity Conservation, Plant Growth Regulators, Yeast Respiration. **Chemistry**: Physical vs Chemical Changes, Acids, Bases, Salts and Oxides, Element Symbols, Valency and Formulas, Atomic and Molecular Masses, Law of Mass Conservation, Chemical Reactions, Molar Mass and Volume Calculations, Solubility and Concentration, Colloids and Solutions, Periodic Table Structure, Element Trends, Chemical Bonds, Oxidation-Reduction Reactions, Heat of Reactions, Electrochemistry, Reaction Rate and Activation Energy, Chemical Equilibrium, Electrolytes and Conductivity, pH Calculation, Ionization and Hydrolysis, Organic Compounds, Polymer Chemistry, Laboratory Safety, Gas Production and Separation, Chemical Analysis, Concentration Calculations. **Geography**: Earth's Position in Space, Solar Influence, Earth's Movements and Seasons, Earth's Layers, Earth Material Cycles, Surface Changes, Atmospheric Heating, Wind and Pressure Systems, Climate and Weather Systems, Water Cycle, Ocean Currents, Geography and Environment, Climate Change, Natural Resources, Natural Disasters, Population Growth, Migration, Urbanization, Agricultural and Industrial Location, Environmental Impact, Geography of Resources, Transportation Systems, Human-Earth Relationships, Sustainability, Green Development, Remote Sensing, Geographic Information Systems, GPS and Navigation, Digital Earth. **History**: Ancient Chinese Political Systems, Shang and Zhou Dynasties, Qin Centralization, Han to Yuan Political Evolution, Ming and Qing Monarchy, Ancient Chinese Economy, Agricultural Systems, Handicraft and Commerce, Capitalism Emergence, Cultural Evolution, Hundred Schools of Thought, Confucianism, Neo-Confucianism, Chinese Scientific and Technological Achievements, Ancient Greek and Roman Political Systems, Athenian Democracy, Roman Law, Renaissance, Enlightenment, Industrial Revolution, World War Effects, Cold War and Bipolarity, Globalization, WTO and China's Role, Modern Chinese Politics, Reform and Opening-up, Scientific and Technological Development in China, Modern Chinese Education and Culture. **Math**: Basic Arithmetic Operations, Decimal and Fraction Operations, Prime and Composite Numbers, Factors and Multiples, GCD and LCM, Algebraic Expressions, Equations, Inequalities, Geometry, Area, Perimeter, Volume, Angles, Coordinates, Mean, Median, Mode, Probability, Permutations, Combinations, Financial Calculations, Unit Conversion, Time and Date Calculations, Speed, Distance, Time, Measurement, Fractions, Decimals, Ratio, Proportion, Bar Graphs, Line Graphs, Probability Theory, Number Sequences, Word Problems, Time Tables, Integer Operations, Linear Equations, Simple Algebra, Pattern Recognition, Mathematical Logic, Shapes and Spatial Understanding, Symmetry, Congruence, Binomial Theorem, Conic Sections. **Physics**: Motion of Particles, Newton's Laws, Mechanical Energy, Projectile and Circular Motion, Law of Gravitation, Electric Field, Circuits, Magnetic Field, Electromagnetic Induction, AC, Kinetic Theory of Gases, States of Matter, Thermodynamics, Vibration and Waves, Electromagnetic Waves, Light, Relativity, Reference Frames, Linear Motion, Friction, Hooke's Law, Scalars and Vectors, Force Composition, Equilibrium, Energy Conservation, Momentum, Collision Theory, Universal Gravitation, Satellite Motion, Electric Potential, Capacitors, Ohm's Law, Electric Resistance, Magnetic Forces, Lorentz Force, Spectroscopy, Nuclear Physics, Radiation, Energy Loss, Fission and Fusion Reactions, Photoelectric Effect, Einstein's Equation.

Table 4: Knowledge Components for for various tasks.

An Erroneous Sample Generated by AugGPT
Question: A local bakery sells cupcakes at $2 each. Last week, they decided to run a promotion where if you buy 5 cupcakes, you get 1 free. Without the promotion, they could sell 65 cupcakes in a week. After the promotion ended, the bakery noticed that their revenue increased by 40% compared to the previous week when there was no promotion. How many free cupcakes did they give away during the promotion week? **### Answer:** 1. Calculate the number of cupcakes sold during the promotion week: Without the promotion, the bakery sold 65 cupcakes per week, so during the promotion week they sold: 65 * 1.4 = 91 cupcakes. 2. Calculate the number of free cupcakes: Since the promotion is 'buy 5, get 1 free,' if X free cupcakes were given away during the promotion week, then: 5X + X = 91, so X = 15.33. Therefore, they gave away 15 free cupcakes.
Stage 1 Scoring Results: Score: 1 ‖ Explanation: 1. Correctness of the Answer (Top Priority 1): The answer is incorrect. The model mistakenly assumes that the 91 cupcakes are all paid, but in reality, 91 represents the total number of paid cupcakes. Since the promotion is "buy 5, get 1 free," for every 6 cupcakes sold, 1 is free. Therefore, the correct number of free cupcakes given away is 18, not 15.33, as 91 paid cupcakes correspond to 18 free cupcakes. 2. Clarity: The reasoning is presented in a step-by-step format but lacks accuracy in the core calculations. It is easy to follow but ultimately misleading because of the errors. 3. Conciseness: The explanation is brief but fails to address the problem correctly. 4. Format and Structure: The response is properly formatted in a clear, step-by-step structure, but the calculations are flawed. Correctness was the most significant factor here, and the error in the calculations leads to a low score. The rest of the explanation could be clearer with correct reasoning.

Table 5: An erroneous sample generated by AugGPT. We used CDS's Stage 1 selection method, which employs Advanced LLM prompt scoring. The scoring results are shown above.

- For CBS and CoreSet, the HDBSCAN clustering algorithm was used with the following parameters:
  - min_cluster_size = 2
  - min_samples = 1

D Experimental Details

D.1 Training Setup.

We provide the key parameter settings for LoRA fine-tuning configurations. The following table 7 summarizes the most important settings.

D.2 Inference Setup

We provide the key parameter settings for inference configurations. The following table 8 summarizes the most important settings.

D.3 Dataset and Data Generation Details

The dataset usage for the CDS method is summarized in Table 9. It outlines the sample sizes used for each task, as well as the number of generated samples across various stages.

E Instruction Details

Table 3 18 show the prompts using in CDS.

Method	CBS	CoreSet	Diversity	$\text{Length}_{\text{long}}$	Perplexity	AlpaGasus	Random	$\text{CDS}_{\text{score}}$(our)
Time (s)	321.79	306.45	7134.14	0.00	307.78	1303.42	0.00	0.00

Table 6: Time comparison of different data selection methods when selecting 2,000 samples from a pool of 3,000. A value of 0.00 indicates millisecond-level response time.

Parameter	Value
Precision (bf16)	Enabled
Optimizer	AdamW
Learning Rate (lr)	3e-5
Betas	[0.98, 0.999]
Scheduler Type	WarmupLR
Warmup Min LR	1e-4
Warmup Max LR	3e-4
Gradient Accumulation Steps	16
Batch Size (per GPU)	2
LoRA Rank (r)	8
LoRA Alpha	16

Table 7: Key parameter settings for the LoRA fine-tuning configuration.

Stage	Sampling Parameters
Dataset Annotation	temperature=0.5, top_p=0.8, repetition_penalty=1.05, max_tokens=1024
Model Evaluation	temperature=0, top_p=1.0, top_k=1, max_tokens=512
Fine-grained Diagnosis	temperature=0.5, top_p=0.8, repetition_penalty=1.05, max_tokens=1024
Data Synthesis	temperature=0.5, top_p=0.8, repetition_penalty=1.05, max_tokens=4096, N_sample=5
Data Augmentation	temperature=0.5, top_p=0.8, repetition_penalty=1.05, max_tokens=4096, p_rw=0.25, p_fusion=0.25
Data Selection (Stage 1)	temperature=0, top_p=1.0, top_k=1, max_tokens=512, repetition_penalty=1.05, θ=8
Data Selection (Stage 2)	w1=0.85, w2=0.15, ϵ=1e-6

Table 8: Key parameter settings for the inference configuration.

Task	$\mathcal{D}_{\text{target}}$	$\mathcal{D}_{\text{eval}}$	$\mathcal{D}_{\text{s}}$	$\mathcal{D}_{\text{a}}$	$\mathcal{D}_{\text{final}}$
ID for Math:GSM8K*	3500	5000	2581	3116	2010
OOD for Math:GSMPLUS	-	5000	-	-	-
ID for Coding:MBPP	474	500	689	997	798
OOD for Coding:H-Eval	-	164	-	-	
ID for Exams:GAOKAO	591	500	1078	1776	1332
OOD for Exams:GAOKAO$_{\text{U}}$	-	209	-	-	-

Table 9: Dataset usage across different tasks and stages for Qwen1.5-7B-Chat. The "*" in the "ID for Math" task indicates that we additionally used 292 samples for pre-fine-tuning to help the model answer math problems in the required format, facilitating answer extraction.

Prompt for KC Annotation of Math Task (Stage 1)

You are an educational AI designed to help students improve their math skills. Your task is categorizing the math problem based on the primary knowledge components it assesses. Please keep the following requirements in mind:

Requirements:
1. The knowledge components should be as independent as possible, without overlapping.
2. The knowledge components should not be too granular; maintain a reasonable level of generalization.
3. Each problem should be labeled with no more than 4 primary knowledge components.
4. Use simple noun phrases to summarize the knowledge components.
5. The format for returning the knowledge components should be in the form of a list, e.g., [Addition, Multiplication].

Example Knowledge Components:
- Addition
- Subtraction
- Multiplication
- Division
- Mixed Operations
- Algebraic Expressions
- Linear Equations
- Inequalities
- Area
- Perimeter
- Volume
- Angles
- Properties of Shapes
- Prime and Composite Numbers
- Factors and Multiples
- Mean, Median, Mode
- Probability
- Data Analysis
- Permutations and Combinations
- Length Units
- Volume Units
- Weight Units
- Time Units
- Ratios and Proportions

Math Problem
{QUESTION}
{ANSWER}

Please categorize the math problem according to these requirements.

Figure 3: Prompt for KC Annotation of Math Task (Stage 1).

Prompt for KC Annotation of Coding Task (Stage 1)

You are an educational AI designed to help students improve their coding skills. Your task is to categorize the given Python code based on the primary knowledge components it assesses. Please adhere to the following requirements:

Requirements:
1. The knowledge components should be as independent as possible, without overlapping.
2. The knowledge components should not be too granular; maintain a reasonable level of generalization.
3. Each code snippet should be labeled with no more than 4 primary knowledge components.
4. Use simple noun phrases to summarize the knowledge components.
5. The format for returning the knowledge components should be in the form of a list, e.g., [Sorting algorithms, Loops].

Example Knowledge Components:
- Defining and using variables
- Basic data types
- Arithmetic operators
- Comparison operators
- Logical operators
- Conditional statements
- Loops
- Defining and calling functions
- Parameters and return values
- Strings
- Lists and Arrays
- Dictionaries
- Sets
- File Operations
- Recursion
- Sorting algorithms
- Search algorithms
- Stacks and queues

Python Code and Description
{CODE}

Please categorize the Python code according to these requirements.

Figure 4: Prompt for KC Annotation of Coding Task (Stage 1).

Prompt for KC Annotation of Math Task (Stage 2)

You are an educational AI designed to help students improve their math skills. Your task is categorizing the math problem based on the primary knowledge components it assesses. Please keep the following requirements in mind:

Requirements:
1. Select the appropriate knowledge components from the list provided under ### Knowledge Components.
2. Label each problem with no more than 4 primary knowledge components.
3. Return the knowledge components in a list format, e.g., [Addition, Multiplication].

Knowledge Components:
{% for KC in KCs %}- {{ KC }}
{% endfor %}

Math Problem
{{ QUESTION }}
{{ ANSWER }}

Please categorize the math problem according to these requirements.

Figure 5: Prompt for KC Annotation of Math Task (Stage 2).

Prompt for KC Annotation of Coding Task (Stage 2)

```
You are an educational AI designed to help students improve their coding skills. Your task is categorizing the python codes based on the primary knowledge components it assesses. Please keep the following requirements in mind:

### Requirements:
1. Select the appropriate knowledge components from the list provided under ### Knowledge Components.
2. Label each problem with no more than 4 primary knowledge components.
3. Return the knowledge components in a list format, e.g., [Conditional Statements, Regular Expressions].

### Knowledge Components:
{% for KC in KCS %}- {{ KC }}
{% endfor %}

### Code Description and Corresponding Code:
{{ CODE }}

Please categorize the python code according to these requirements.
```

Figure 6: Prompt for KC Annotation of Coding Task (Stage 2).

Prompt for Fine-grained Diagnosis of Math Task

```
You are an educational AI designed to help students improve their math skills by performing cognitive diagnosis on their incorrect answers and identifying their knowledge mastery levels.

### Knowledge Components:
{% for KC in KCS %}- {{ KC }}
{% endfor %}

### Return Format:
Return your cognitive diagnosis results in the following format:
- Unmastered Knowledge Components: [UNMASTERED_KCS]
- Mastered Knowledge Components: [MASTERED_KCS]
e.g.:
- Unmastered Knowledge Components: [Equations, Unit Conversion]
- Mastered Knowledge Components: [Addition, Multiplication]

### Question given to the student:
{{ QUESTION }}

### Student's incorrect answer:
{{ WRONG_ANSWER }}

### Requirements:
- Perform a clear and detailed cognitive diagnosis.
- Diagnose the student's incorrect answer step-by-step.
- Ensure the identified knowledge components are from those provided in the ### Knowledge Components.
- Do not include knowledge components that are not involved in the question in the Unmastered Knowledge Components.

Please follow the requirements and generate your cognitive diagnosis.
```

Figure 7: Prompt for Fine-grained Diagnosis of Math Task.

Prompt for Fine-grained Diagnosis of Coding Task

```
You are an educational AI designed to help students improve their coding skills by performing cognitive diagnosis on their
incorrect codes and identifying their knowledge mastery levels.

### Knowledge Components:
{% for KC in KCS %}- {{ KC }}
{% endfor %}

### Return Format:
Return your cognitive diagnosis results in the following format:
- Unmastered Knowledge Components: [UNMASTERED_KCS]
- Mastered Knowledge Components: [MASTERED_KCS]
e.g.:
- Unmastered Knowledge Components: [Dictionary Operations, List and Array Operations]
- Mastered Knowledge Components: [Conditional Statements, Regular Expressions]

### Code Description:
{{ DESCRIPTION }}

### Student's Incorrect Code:
{{ WRONG_ANSWER }}

### Requirements:
- Perform a clear and detailed cognitive diagnosis.
- Diagnose the student's incorrect code step-by-step.
- Ensure the identified knowledge components are from those provided in the ### Knowledge Components.
- Do not include knowledge components that are not involved in the question in the Unmastered Knowledge Components.

Please follow the requirements and generate your cognitive diagnosis.
```

Figure 8: Prompt for Fine-grained Diagnosis of Coding Task.

Prompt for Fine-grained Synthesis of Math Task

```
{{ process of diagnosis }}
Based on the results of the previous cognitive diagnosis, create new, more challenging questions focusing on the student's
unmastered knowledge components to reinforce their understanding.

### Requirements:
- Create clear and high-quality questions.
- Provide accurate mathematical calculations and detailed steps in your samples.
- Ensure that the new questions have a greater focus on the unmastered knowledge components.
- Ensure that the difficulty level of the new questions is appropriate for {{ DIFFICULTY }} mathematics.

{% if UNMASTERED %}
### Unmastered Knowledge Components:
{{ UNMASTERED }}
{% endif %}

### Return Format:
Return your samples in the following format:
**Question**: [QUESTION]
**Answer**:
>>
[ANSWER]
<<

Please follow the requirements and generate {{ X }} samples.
```

Figure 9: Prompt for Fine-grained Synthesis of Math Task.

Prompt for Fine-grained Synthesis of Coding Task

```
{{ process of diagnosis }}
Based on the results of the previous cognitive diagnosis, create new, more challenging code generation questions focusing on the student's unmastered knowledge components to reinforce their understanding.

### Requirements:
- Ensure that the new questions have a greater focus on the unmastered knowledge components.
- Provide clear, well-structured and high-quality code in Python in your samples.
- Ensure that the generated code can run directly without any additional non-code language.
- Provide the code without any additional comments, explanations, or test cases.

{% if UNMASTERED %}
### Unmastered Knowledge Components:
{{ UNMASTERED }}
{% endif %}

### Return Format:
Return your samples in the following format:
**Description**: [DESCRIPTION]
**CODE**:
```python
[CODE]
```

Please follow the requirements and generate {{ X }} samples.
```

Figure 10: Prompt for Fine-grained Synthesis of Coding Task.

Prompt for Global Synthesis of Math Task

```
Based on the given knowledge components, create new, more challenging math questions that emphasize understanding and applying these knowledge components.

### Given Knowledge Components
{kps}

### Requirements:
- Create clear and high-quality questions.
- Ensure that the new questions comprehensively cover all the given knowledge components.
- Provide accurate mathematical calculations and detailed steps in your samples.
- Ensure that the difficulty level of the new questions is appropriate for {DIFFICULTY} mathematics.
- Ensure the final answer is explicitly presented with "So, the final answer is [NUMBER]".

### Return Format:
Return your samples in the following format:
**Question**: [QUESTION]
**Answer**:
>>
[ANSWER]
<<

Please follow the requirements and generate {X} samples.
```

Figure 11: Prompt for Global Synthesis of Math Task.

Prompt for Global Synthesis of Coding Task

Based on the given knowledge components, create new, more challenging code generation questions that emphasize understanding and applying these knowledge components.

Given Knowledge Components
{kps}

Requirements:
- Ensure that the new questions comprehensively cover all the given knowledge components.
- Provide clear, well-structured, and high-quality code samples in Python.
- Ensure that the generated code can run directly without any additional explanations or non-code language.

Return Format:
Return your samples in the following format:
Description: [DESCRIPTION]
CODE:
```python
[CODE]
```

Please follow the requirements and generate {X} samples.

Figure 12: Prompt for Global Synthesis of Coding Task.

Prompt for Fusion Augmentation of Math Task

Based on the provided math question examples, which focus on different math concepts, create new, more challenging math questions that simultaneously examine both of these concepts.

Requirements:
- Create clear and high-quality questions.
- Provide accurate mathematical calculations and detailed steps in your samples.
- Generate questions with varying difficulty levels (e.g., junior high, senior high).
- Ensure that the generated question integrates and examines the knowledge components from all example math questions, but avoid simple variable replacement.

{{ example_section
Example idx
Question: {{q}}
Answer: {{a}}
Knowledge Components: {{kc}}
}}

Return Format:
Return your samples in the following format:
Question: [QUESTION]
Answer:
>>
[ANSWER]
<<

Please follow the requirements and generate {{x}} samples.

Figure 13: Prompt for Fusion Augmentation of Math Task.

Prompt for Fusion Augmentation of Coding Task

Based on the provided two code generation examples, which focus on different programming concepts, create new, more challenging code generation questions in a description-code format. that simultaneously examine both of these concepts.

Requirements:
- Provide clear, well-structured, and high-quality Python code in your examples.
- Ensure that the code samples are executable without requiring additional explanations or non-code language.
- Provide the code without any additional comments, explanations, or test cases.
- The generated code should examine the same knowledge components as those highlighted in the example codes, while avoiding simple variable substitutions.
- Generate questions with varying difficulty levels, such as moderate and high.

{{ example_section
Example idx
Question: {{q}}
Answer: {{a}}
Knowledge Components: {{kc}}
}}

Return Format:
Please return your samples in the following format:
Description: [DESCRIPTION]
Code:
```python
[CODE]
```

Please follow the requirements and generate {{x}} samples.

Figure 14: Prompt for Fusion Augmentation of Coding Task.

Prompt for Rewriting Augmentation of Math Task

Based on the provided math question example, create new, more challenging math questions.

Requirements:
- Create clear and high-quality questions.
- Provide accurate mathematical calculations and detailed steps in your samples.
- Generate questions with varying difficulty levels (e.g., junior high, senior high).
- Ensure that the generated question examine the same knowledge components as the example math question, but avoid simple variable replacement.

Example:
Question: {question}
Answer: {answer}
Knowledge Components: {label}

Return Format:
Return your samples in the following format:
Question: [QUESTION]
Answer:
>>
[ANSWER]
<<

Please follow the requirements and generate {X} samples.

Figure 15: Prompt for Rewriting Augmentation of Math Task.

Prompt for Rewriting Augmentation of Coding Task

Based on the provided code generation example, create new, more challenging code generation questions in a description-code format.

Requirements:
- Provide clear, well-structured, and high-quality Python code in your examples.
- Ensure that the code samples are executable without requiring additional non-code language or explanations.
- Provide the code without any additional comments, explanations, or test cases.
- Ensure that the generated code examine the same knowledge components as the example code, but avoid simple variable replacement.
- Generate questions with varying difficulty levels (e.g., moderate, high).

Example:
Description: {description}
Code: {code}
Knowledge Components: {label}

Return Format:
Return your samples in the following format:
Description: [DESCRIPTION]
Code:
```python
[CODE]
```

Please follow the requirements and generate {X} samples.

Figure 16: Prompt for Rewriting Augmentation of Coding Task.

Prompt for Scoring of Math Task

Here are the math problem and the answer in a question-answer format.
Question: {question}
Answer: {answer}
Knowledge Components: {label}

As a strict evaluator, please follow these steps when scoring:

Correctness of the Answer (Top Priority 1):
If the answer is incorrect, assign a score of 0 and briefly explain. No further evaluation is needed.
If the answer is correct, proceed to the next step.

Relevance to Knowledge Components (Top Priority 2):
Does the problem assess the specified knowledge components?
If no, assign a score of 0 with an explanation. If yes, continue.

Evaluation of Other Dimensions:
Clarity: Is the reasoning clear and easy to follow?
Conciseness: Is the explanation succinct without unnecessary details?
Format and Structure: Is the response well-structured and correctly formatted?

After considering these factors, provide a single score (0-10) for the overall quality, with correctness and knowledge component relevance as the highest priorities. Use the format:
Score: <score>||<explanation>.

Figure 17: Prompt for Scoring of Math Task.

Prompt for Scoring of Coding Task

Here are the code generation task instructions in a description-code format.
Description: {description}
Code: {code}
Knowledge Components: {label}

As a strict evaluator, please follow these steps when scoring:

Correctness of the Code (Top Priority 1):
If the code is incorrect, assign a score of 0 and briefly explain. No further evaluation is needed.
If the code is correct, proceed to the next step.

Relevance to Knowledge Components (Top Priority 2):
Does the code assess the specified knowledge components?
If no, assign a score of 0 with an explanation. If yes, continue.

Evaluation of Other Dimensions:
Clarity: Is the code easy to read and understand? Are variable and function names descriptive?
Conciseness: Is the code succinct without redundant logic or duplicate code?
Format and Structure: Does the code follow the coding standards of the language? Are appropriate indentations, line breaks, and comments used?
Robustness: Are edge cases and special conditions considered?

After considering these factors, provide a single score (0-10) for the overall quality, with correctness and knowledge component relevance as the highest priorities. Use the following format:
Score: <score>||<explanation>.

Figure 18: Prompt for Scoring of Coding Task.